



Money Mobiliser API Reference

Sybase Mobiliser Platform 5.1

SP03

DOCUMENT ID: DC01980-01-0513-01

LAST REVISED: August 2013

Copyright © 2013 by Sybase, Inc. All rights reserved.

This publication pertains to Sybase software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement.

Upgrades are provided only at regularly scheduled software release dates. No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc.

Sybase trademarks can be viewed at the Sybase trademarks page at <http://www.sybase.com/detail?id=1011207>. Sybase and the marks listed are trademarks of Sybase, Inc. ® indicates registration in the United States of America.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world.

Java and all Java-based marks are trademarks or registered trademarks of Oracle and/or its affiliates in the U.S. and other countries.

Unicode and the Unicode Logo are registered trademarks of Unicode, Inc.

All other company and product names mentioned may be trademarks of the respective companies with which they are associated.

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

com.sybase365.mobiliser.money package	1
ams package	1
dao package	1
export package	50
businesslogic package	59
audit package	60
authentication package	68
billpayment package	76
binary package	83
bulkprocessing package	93
customer package	110
fee package	157
invoice package	166
normalizer package	174
oauth2 package	178
payment package	179
system package	218
transaction package	258
umgr package	424
umgrroles package	435
util package	438
wallet package	491
coupon package	507
businesslogic package	507
dao package	537
flows package	590
bulkfileprocessing package	590
customer package	595
wallet package	605
jobs package	609
alerts package	609

Contents

event package	634
tasks package	659
odc package	664
businesslogic package	664
dao package	674
persistence package	700
dao package	700
model package	826
svcpkg package	1603
businesslogic package	1603
dao package	1616

com.sybase365.mobiliser.money package

Members

All public members of the money package.

- **ams package** –
- **businesslogic package** –
- **coupon package** –
- **flows package** –
- **jobs package** –
- **odc package** –
- **persistence package** –
- **svcpkg package** –

ams package

Members

All public members of the ams package.

- **dao package** –
- **export package** –

dao package

Members

All public members of the dao package.

- **api package** –
- **model package** –

api package

Members

All public members of the api package.

- **BookingDao interface** – Interface BookingDao for accessing the Booking entity.
- **ClearingConfigDao interface** – Interface ClearingConfigDao for accessing the ClearingConfig entity.

- **CurrencyExchangeDao interface** – Interface CurrencyExchangeDao for accessing the CurrencyExchange entity.
- **DocDao interface** – Interface DocDao for accessing the Doc entity.
- **PaymentDao interface** – Interface PaymentDao for accessing the Payment entity.
- **PiBalanceDao interface** – Interface PiBalanceDao for accessing the PiBalance entity.
- **ReversalDao interface** – Interface ReversalDao for accessing the Reversal entity.

BookingDao interface

Interface BookingDao for accessing the Booking entity.

Syntax

```
public interface BookingDao
```

Usage

2013 SAP® AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

ClearingConfigDao interface

Interface ClearingConfigDao for accessing the ClearingConfig entity.

Syntax

```
public interface ClearingConfigDao
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CurrencyExchangeDao interface

Interface CurrencyExchangeDao for accessing the CurrencyExchange entity.

Syntax

```
public interface CurrencyExchangeDao
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

DocDao interface

Interface DocDao for accessing the Doc entity.

Syntax

```
public interface DocDao
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

PaymentDao interface

Interface PaymentDao for accessing the Payment entity.

Syntax

```
public interface PaymentDao
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

PiBalanceDao interface

Interface PiBalanceDao for accessing the PiBalance entity.

Syntax

```
public interface PiBalanceDao
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

ReversalDao interface

Interface ReversalDao for accessing the Reversal entity.

Syntax

```
public interface ReversalDao
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

model package

Members

All public members of the model package.

- **Booking class** – Booking Entity bean.
- **ClearingConfig class** – ClearingConfig.
- **CurrencyExchange class** – CurrencyExchange.
- **Doc class** – Doc.
- **Payment class** – Payment.
- **PiBalance class** – PiBalance.
- **Reversal class** – Reversal.

Booking class

Booking Entity bean.

Syntax

```
public class Booking
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

BookingPk class

Syntax

```
public class BookingPk
```

Usage

This is the embedded primary key, comprising the according document this booking belongs to and the sequence of this booking within the document.

© 2000-2009 by paybox solutions AG

Revision:1.2Name:not supported by cvs2svn

*BookingPk() constructor***Syntax**`BookingPk ()`**Usage**

Creates a new instance of

`BookingPk`

.

*BookingPk(Long, int) constructor***Syntax**`BookingPk ( Long docId ,   int sequence )`**Parameters**

- **doc** – the according document this booking belongs to
- **sequence** – the sequence of this booking within the document

Usage

Creates a new instance of

`BookingPk`

.

*compareTo(BookingPk) method***Syntax**`int compareTo ( BookingPk o )`*equals(Object) method***Syntax**`boolean equals ( Object arg0 )`*getDocId() method***Syntax**`Long getDocId ()`

Returns

the docId

Usage

the docId

getSequence() method

Syntax

int getSequence ()

Returns

the sequence of this booking within the document

Usage

the sequence of this booking within the document

hashCode() method

Syntax

int hashCode ()

setDocId(Long) method

Syntax

void setDocId (Long *docId*)

Parameters

- **docId** – the docId to set

setSequence(int) method

Syntax

void setSequence (int *sequence*)

Parameters

- **sequence** – the sequence of this booking within the document

toString() method

Syntax

`String toString ()`

Booking() constructor

Syntax

`Booking ()`

Usage

Creates a new instance of

`TcmDtinFile`

.

Booking(BookingPk) constructor

Syntax

`Booking ( BookingPk id )`

Parameters

- **id** – the PK

Usage

Creates a new instance of

`Booking`

.

getAmount() method

Syntax

`BigDecimal getAmount ()`

Returns

the amount of the instance

Usage

the amount of the instance

getAmountBalance() method

Syntax

BigDecimal getAmountBalance ()

Returns

the amountBalance

Usage

the amountBalance

getBookingType() method

Syntax

String getBookingType ()

Returns

the bookingType

Usage

the bookingType

getClearingModeId() method

Syntax

String getClearingModeId ()

Returns

the clearingModeId

Usage

the clearingModeId

getClearingNumber() method

Syntax

Long getClearingNumber ()

Returns

the clearingNumber

Usage

the clearingNumber

getClearingPild() method

Syntax

Long getClearingPiId ()

Returns

the clearingPiId

Usage

the clearingPiId

getCustomerId() method

Syntax

Long getCustomerId ()

Returns

the customerId

Usage

the customerId

getDbOp() method

Syntax

Character getDbOp ()

Returns

the dbOp

Usage

the dbOp

getFeeTypeId() method

Syntax

Integer getFeeTypeId ()

Returns

the feeTypeId

Usage

the feeTypeId

getOriginalPiId() method

Syntax

Long getOriginalPiId ()

Returns

the originalPiId

Usage

the originalPiId

getPiId() method

Syntax

long getPiId ()

Returns

the piId

Usage

the piId

getStatus() method

Syntax

int getStatus ()

Returns

the status

Usage

the status

getTxnDate() method

Syntax

Date getTxnDate ()

Returns

the posting date of the instance

Usage

the posting date of the instance

isCredit() method

Syntax

boolean isCredit ()

Returns

true if this instance is a credit booking; false otherwise

Usage

true if this instance is a credit booking; false otherwise

isSetAmount() method

Syntax

boolean isSetAmount ()

isSetAmountBalance() method

Syntax

boolean isSetAmountBalance ()

isSetBookingType() method

Syntax

boolean isSetBookingType ()

isSetClearingModelId() method

Syntax

boolean isSetClearingModeId ()

Returns

the clearingModeId

Usage

the clearingModeId

isSetClearingNumber() method

Syntax

boolean isSetClearingNumber ()

isSetClearingPild() method

Syntax

boolean isSetClearingPiId ()

Returns

the clearingPiId

Usage

the clearingPiId

isSetCredit() method

Syntax

boolean isSetCredit ()

isSetCustomerId() method

Syntax

boolean isSetCustomerId ()

Returns

the customerId

Usage

the customerId

isSetDbOp() method

Syntax

```
boolean isSetDbOp ()
```

isSetFeeTypeId() method

Syntax

```
boolean isSetFeeTypeId ()
```

Returns

the feeTypeId

Usage

the feeTypeId

isSetOriginalPiId() method

Syntax

```
boolean isSetOriginalPiId ()
```

Returns

the originalPiId

Usage

the originalPiId

isSetPiId() method

Syntax

```
boolean isSetPiId ()
```

Returns

the piId

Usage

the piId

isSetStatus() method

Syntax

boolean isSetStatus ()

isSetTxnDate() method

Syntax

boolean isSetTxnDate ()

setAmount(BigDecimal) method

Syntax

void setAmount (BigDecimal *amount*)

Parameters

- **amount** – the amount to set

setAmountBalance(BigDecimal) method

Syntax

void setAmountBalance (BigDecimal *amountBalance*)

Parameters

- **amountBalance** – the amountBalance to set

setBookingType(String) method

Syntax

void setBookingType (String *bookingType*)

Parameters

- **bookingType** – the bookingType to set

setClearingModeId(String) method

Syntax

void setClearingModeId (String *clearingModeId*)

Parameters

- **clearingModeId** – the clearingModeId to set

setClearingNumber(Long) method

Syntax

```
void setClearingNumber ( Long clearingNumber )
```

Parameters

- **clearingNumber** – the clearingNumber to set

setClearingPiId(Long) method

Syntax

```
void setClearingPiId ( Long clearingPiId )
```

Parameters

- **clearingPiId** – the clearingPiId to set

setCredit(boolean) method

Syntax

```
void setCredit ( boolean credit )
```

Parameters

- **credit** – the credit value to set

setCustomerId(Long) method

Syntax

```
void setCustomerId ( Long customerId )
```

Parameters

- **customerId** – the customerId to set

setDbOp(Character) method

Syntax

```
void setDbOp ( Character dbOp )
```

Parameters

- **dbOp** – the dbOp to set

setFeeTypeId(Integer) method

Syntax

void setFeeTypeId (Integer *feeTypeId*)

Parameters

- **feeTypeId** – the feeTypeId to set

setOriginalPiId(Long) method

Syntax

void setOriginalPiId (Long *originalPiId*)

Parameters

- **originalPiId** – the originalPiId to set

setPiId(long) method

Syntax

void setPiId (long *piId*)

Parameters

- **piId** – the piId to set

setStatus(int) method

Syntax

void setStatus (int *status*)

Parameters

- **status** – the status to set

setTxnDate(Date) method

Syntax

void setTxnDate (Date *txnDate*)

Parameters

- **txnDate** – the posting date to set

ClearingConfig class

ClearingConfig.

Syntax

```
public class ClearingConfig
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

ClearingConfigPk class**Syntax**

```
public class ClearingConfigPk
```

Usage

This is the embedded primary key, comprising the according document this ClearingConfig belongs to.

© 2000-2009 by paybox solutions AG

Revision:1.2Name:not supported by cvs2svn

ClearingConfigPk() constructor**Syntax**

```
ClearingConfigPk ()
```

Usage

Creates a new instance of

BookingPk

.

ClearingConfigPk(String, String, String, Currency) constructor**Syntax**

```
ClearingConfigPk (String orgunitId , String settlementModeId , String  
clearingModeId , Currency currency)
```

Usage

Creates a new instance of

BookingPk

.

compareTo(ClearingConfigPk) method

Syntax

int compareTo (ClearingConfigPk o)

equals(Object) method

Syntax

boolean equals (Object arg0)

getClearingModeId() method

Syntax

String getClearingModeId ()

Returns

the clearingModeId

Usage

the clearingModeId

getCurrency() method

Syntax

Currency getCurrency ()

Returns

the currency of the instance

Usage

the currency of the instance

getOrgunitId() method

Syntax

String getOrgunitId ()

getSettlementModeId() method

Syntax

String getSettlementModeId ()

Returns

the settlementModeId

Usage

the settlementModeId

hashCode() method

Syntax

int hashCode ()

setClearingModeId(String) method

Syntax

void setClearingModeId (String *clearingModeId*)

Parameters

- **clearingModeId** – the clearingModeId to set

setCurrency(Currency) method

Syntax

void setCurrency (Currency *currency*)

Parameters

- **the** – currency to set

setOrgunitId(String) method

Syntax

void setOrgunitId (String *orgunitId*)

setSettlementModeId(String) method

Syntax

void setSettlementModeId (String *settlementModeId*)

Parameters

- **settlementModeId** – the settlementModeId to set

toString() method

Syntax

String toString ()

ClearingConfig() constructor

Syntax

ClearingConfig ()

Usage

Creates a new instance of

Product

.

ClearingConfig(ClearingConfigPk) constructor

Syntax

ClearingConfig (ClearingConfigPk *id*)

Usage

Creates a new instance of

Product

.

getClearingPi() method

Syntax

PaymentInstrument getClearingPi ()

Returns

the clearingPi

Usage

the clearingPi

setClearingPi(PaymentInstrument) method

Syntax

void setClearingPi (PaymentInstrument *clearingPi*)

Parameters

- **clearingPi** – the clearingPi to set

CurrencyExchange class

CurrencyExchange.

Syntax

```
public class CurrencyExchange
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CurrencyExchangePk class

Syntax

```
public class CurrencyExchangePk
```

Usage

This is the embedded primary key, comprising the according document this CurrencyExchange belongs to.

© 2000-2009 by paybox solutions AG

Revision:1.2Name:not supported by cvs2svn

CurrencyExchangePk() constructor

Syntax

CurrencyExchangePk ()

Usage

Creates a new instance of

BookingPk

.

CurrencyExchangePk(String, String) constructor

Syntax

CurrencyExchangePk (String *currencyFrom* , String *currencyTo*)

Parameters

- **currencyFrom** –
- **currencyTo** –

Usage

Creates a new instance of

CurrencyExchangePk

.

compareTo(CurrencyExchangePk) method

Syntax

int compareTo (CurrencyExchangePk *o*)

equals(Object) method

Syntax

boolean equals (Object *arg0*)

getCurrencyFrom() method

Syntax

Currency getCurrencyFrom ()

Returns

the transaction's Currency

Usage

Returns the currency this transaction is executed with.

the transaction's Currency

getCurrencyTo() method

Syntax

Currency getCurrencyTo ()

Returns

the transaction's Currency

Usage

Returns the currency this transaction is executed with.

the transaction's Currency

hashCode() method

Syntax

int hashCode ()

setCurrencyFrom(Currency) method

Syntax

void setCurrencyFrom (Currency *dbCurrencyFrom*)

Parameters

- **currency** – the currency value to set

Usage

Sets the

currency

of this instance.

setCurrencyTo(Currency) method

Syntax

`void setCurrencyTo ( Currency dbCurrencyTo )`

Parameters

- **currency** – the `currency` value to set

Usage

Sets the

currency

of this instance.

toString() method

Syntax

`String toString ()`

CurrencyExchange() constructor

Syntax

`CurrencyExchange ()`

Usage

Creates a new instance of

`CurrencyExchange`

.

CurrencyExchange(CurrencyExchangePk) constructor

Syntax

`CurrencyExchange ( CurrencyExchangePk id )`

Parameters

- **id** –

Usage

Creates a new instance of
`CurrencyExchange`

.

getPaymentInstrumentId() method

Syntax

```
long getPaymentInstrumentId ()
```

Returns

the `paymentInstrumentId`

Usage

the `paymentInstrumentId`

isSetPaymentInstrumentId() method

Syntax

```
boolean isSetPaymentInstrumentId ()
```

setPaymentInstrumentId(long) method

Syntax

```
void setPaymentInstrumentId ( long paymentInstrumentId )
```

Parameters

- **paymentInstrumentId** – the `paymentInstrumentId` to set

Doc class

`Doc`.

Syntax

```
public class Doc
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

Doc() constructor

Syntax

`Doc ()`

Usage

Creates a new instance of

`TcmDtinFile`

.

getStatus() method

Syntax

`int getStatus ()`

Returns

the status of the instance

Usage

the status of the instance

getSubTxnId() method

Syntax

`Long getSubTxnId ()`

Returns

the subTxnId

Usage

the subTxnId

setStatus(int) method

Syntax

`void setStatus ( int status )`

Parameters

- **status** – the status to set

setSubTxnId(Long) method

Syntax

```
void setSubTxnId ( Long subTxnId )
```

Parameters

- **subTxnId** – the subTxnId to set

Payment class

Payment.

Syntax

```
public class Payment
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Payment() constructor

Syntax

```
Payment ()
```

Usage

Creates a new instance of

```
TcmDtinFile
```

.

getAmount() method

Syntax

```
BigDecimal getAmount ()
```

Returns

the amount of the instance

Usage

the amount of the instance

getBankRef() method

Syntax

String getBankRef ()

Returns

the bank ref. of the instance

Usage

the bank ref. of the instance

getBatchRunId() method

Syntax

Long getBatchRunId ()

Returns

the batchRunId

Usage

the batchRunId

getClearingNumber() method

Syntax

Long getClearingNumber ()

Returns

the clearing number of the instance

Usage

the clearing number of the instance

getClearingPiId() method

Syntax

Long getClearingPiId ()

Returns

the clearingPiId

Usage

the clearingPiId

getCurrency() method

Syntax

Currency getCurrency ()

Returns

the currency of the instance

Usage

the currency of the instance

getDocId() method

Syntax

Long getDocId ()

Returns

the docId

Usage

the docId

getFee() method

Syntax

BigDecimal getFee ()

Returns

the fee amount of the instance

Usage

the fee amount of the instance

getFile() method

Syntax

File getFile ()

Returns

the file of the instance

Usage

the file of the instance

getFrom() method

Syntax

Date getFrom ()

Returns

the from

Usage

the from

getPild() method

Syntax

Long getPild ()

Returns

the pild

Usage

the pild

getPostingDate() method

Syntax

Date getPostingDate ()

Returns

the posting date of the instance

Usage

the posting date of the instance

getReversalDate() method

Syntax

Date getReversalDate ()

Returns

the reversal date of the instance

Usage

the reversal date of the instance

getSendDate() method

Syntax

Date getSendDate ()

Returns

the send date of the instance

Usage

the send date of the instance

getSettlementMode() method

Syntax

String getSettlementMode ()

Returns

the settlement mode of the instance

Usage

the settlement mode of the instance

getStatus() method

Syntax

int getStatus ()

Returns

the status of the instance

Usage

the status of the instance

getSubTxnId() method

Syntax

Long getSubTxnId ()

getTo() method

Syntax

Date getTo ()

Returns

the to

Usage

the to

getTxnAmount() method

Syntax

BigDecimal getTxnAmount ()

Returns

the transaction amount of the instance

Usage

the transaction amount of the instance

getTxnCount() method

Syntax

Long getTxnCount ()

Returns

the number of transactions for this instance

Usage

the number of transactions for this instance

getVat() method

Syntax

```
BigDecimal getVat ()
```

Returns

the vat amount of the instance

Usage

the vat amount of the instance

isCredit() method

Syntax

```
boolean isCredit ()
```

Returns

true if this instance is a credit payment; false otherwise

Usage

true if this instance is a credit payment; false otherwise

setAmount(BigDecimal) method

Syntax

```
void setAmount ( BigDecimal amount )
```

Parameters

- **amount** – the amount to set

setBankRef(String) method

Syntax

```
void setBankRef ( String bankRef )
```

Parameters

- **bankRef** – the bank ref. to set

setBatchRunId(Long) method

Syntax

`void setBatchRunId ( Long batchRunId )`

Parameters

- **batchRunId** – the batchRunId to set

setClearingNumber(Long) method

Syntax

`void setClearingNumber ( Long clearingNumber )`

Parameters

- **clearingNumber** – the clearing number to set

setClearingPiId(Long) method

Syntax

`void setClearingPiId ( Long clearingPiId )`

Parameters

- **clearingPiId** – the clearingPiId to set

setCredit(boolean) method

Syntax

`void setCredit ( boolean credit )`

Parameters

- **credit** – the credit value to set

setCurrency(Currency) method

Syntax

`void setCurrency ( Currency currency )`

Parameters

- **the** – currency to set

setCurrency(String) method

Syntax

```
void setCurrency ( String currency )
```

Parameters

- **the** – currency to set

setDocId(Long) method

Syntax

```
void setDocId ( Long docId )
```

Parameters

- **docId** – the docId to set

setFee(BigDecimal) method

Syntax

```
void setFee ( BigDecimal fee )
```

Parameters

- **fee** – the fee amount to set

setFile(File) method

Syntax

```
void setFile ( File file )
```

Parameters

- **file** – the file to set

setFrom(Date) method

Syntax

```
void setFrom ( Date from )
```

Parameters

- **from** – the from to set

setPild(Long) method

Syntax

void setPiId (Long *piId*)

Parameters

- **piId** – the piId to set

setPostingDate(Date) method

Syntax

void setPostingDate (Date *postingDate*)

Parameters

- **postingDate** – the posting date to set

setReversalDate(Date) method

Syntax

void setReversalDate (Date *reversalDate*)

Parameters

- **reversalDate** – the reversal date to set

setSendDate(Date) method

Syntax

void setSendDate (Date *sendDate*)

Parameters

- **sendDate** – the send date to set

setSettlementMode(String) method

Syntax

void setSettlementMode (String *settlementMode*)

Parameters

- **settlementMode** – the settlement mode to set

setStatus(int) method

Syntax

```
void setStatus ( int status )
```

Parameters

- **status** – the status to set

setSubTxnId(Long) method

Syntax

```
void setSubTxnId ( Long subTxnId )
```

setTo(Date) method

Syntax

```
void setTo ( Date to )
```

Parameters

- **to** – the to to set

setTxnAmount(BigDecimal) method

Syntax

```
void setTxnAmount ( BigDecimal txnAmount )
```

Parameters

- **txnAmount** – the transaction amount to set

setTxnCount(Long) method

Syntax

```
void setTxnCount ( Long txnCount )
```

Parameters

- **txnCount** – the number of transactions to set

setVat(BigDecimal) method

Syntax

```
void setVat ( BigDecimal vat )
```

Parameters

- **vat** – the vat amount to set

PiBalance class

PiBalance.

Syntax

```
public class PiBalance
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

PiBalance() constructor

Syntax

```
PiBalance ()
```

Usage

Creates a new instance of

```
PiBalance
```

.

PiBalance(Long) constructor

Syntax

```
PiBalance ( Long id )
```

Parameters

- `id` –

Usage

Creates a new instance of

`PiBalance`

.

`add2Credit(BigDecimal)` method

Syntax

`void add2Credit ( BigDecimal amount )`

`add2Debit(BigDecimal)` method

Syntax

`void add2Debit ( BigDecimal amount )`

`getBalance()` method

Syntax

`BigDecimal getBalance ()`

`getClearingRun()` method

Syntax

`Long getClearingRun ()`

Returns

the clearingRun

Usage

the clearingRun

`getCreditBalance()` method

Syntax

`BigDecimal getCreditBalance ()`

Returns

the creditBalance

Usage

the creditBalance

getDebitBalance() method

Syntax

BigDecimal getDebitBalance ()

Returns

the debitBalance

Usage

the debitBalance

getLastCreditBalance() method

Syntax

BigDecimal getLastCreditBalance ()

Returns

the lastCreditBalance

Usage

the lastCreditBalance

getLastDebitBalance() method

Syntax

BigDecimal getLastDebitBalance ()

Returns

the lastDebitBalance

Usage

the lastDebitBalance

getLastDocId() method

Syntax

Long getLastDocId ()

Returns

the lastDocId

Usage

the lastDocId

getNextClearingNumber() method

Syntax

long getNextClearingNumber ()

setClearingRun(Long) method

Syntax

void setClearingRun (Long *clearingRun*)

Parameters

- **clearingRun** – the clearingRun to set

setCreditBalance(BigDecimal) method

Syntax

void setCreditBalance (BigDecimal *creditBalance*)

Parameters

- **creditBalance** – the creditBalance to set

setDebitBalance(BigDecimal) method

Syntax

void setDebitBalance (BigDecimal *debitBalance*)

Parameters

- **debitBalance** – the debitBalance to set

setLastCreditBalance(BigDecimal) method

Syntax

```
void setLastCreditBalance ( BigDecimal lastCreditBalance )
```

Parameters

- **lastCreditBalance** – the lastCreditBalance to set

setLastDebitBalance(BigDecimal) method

Syntax

```
void setLastDebitBalance ( BigDecimal lastDebitBalance )
```

Parameters

- **lastDebitBalance** – the lastDebitBalance to set

setLastDocId(Long) method

Syntax

```
void setLastDocId ( Long lastDocId )
```

Parameters

- **lastDocId** – the lastDocId to set

Reversal class

Reversal.

Syntax

```
public class Reversal
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Reversal() constructor

Syntax

Reversal ()

Usage

Creates a new instance of

TcmDtinFile

.

getAmount() method

Syntax

BigDecimal getAmount ()

Returns

the amount of the instance

Usage

the amount of the instance

getBankFee() method

Syntax

Long getBankFee ()

Returns

the bank fee of the instance

Usage

the bank fee of the instance

getCurrency() method

Syntax

Currency getCurrency ()

Returns

the currency of the instance

Usage

the currency of the instance

getCustomer() method

Syntax

Customer getCustomer ()

Returns

the customer of the instance

Usage

the customer of the instance

getFile() method

Syntax

File getFile ()

Returns

the file of the instance

Usage

the file of the instance

getForeignReference() method

Syntax

String getForeignReference ()

Returns

the foreign reference of the instance

Usage

the foreign reference of the instance

getOtherFee() method

Syntax

Long getOtherFee ()

Returns

the other fees of the instance

Usage

the other fees of the instance

getPayment() method

Syntax

`Payment getPayment ()`

Returns

the payment of the instance

Usage

the payment of the instance

getPi() method

Syntax

`PaymentInstrument getPi ()`

Returns

the PI of the instance

Usage

the PI of the instance

getReason() method

Syntax

`String getReason ()`

Returns

the reason of the instance

Usage

the reason of the instance

getReasonId() method

Syntax

Integer getReasonId ()

Returns

the reason ID of the instance

Usage

the reason ID of the instance

getStatus() method

Syntax

Integer getStatus ()

Returns

the status of the instance

Usage

the status of the instance

getTxn() method

Syntax

Transaction getTxn ()

Returns

the transaction of the instance

Usage

the transaction of the instance

isCredit() method

Syntax

boolean isCredit ()

Returns

true if this instance is a credit payment; false otherwise

Usage

true if this instance is a credit payment; false otherwise

setAmount(BigDecimal) method

Syntax

void setAmount (BigDecimal *amount*)

Parameters

- **amount** – the amount to set

setBankFee(Long) method

Syntax

void setBankFee (Long *bankFee*)

Parameters

- **bankFee** – the bank fee to set

setCredit(boolean) method

Syntax

void setCredit (boolean *credit*)

Parameters

- **credit** – the credit value to set

setCurrency(Currency) method

Syntax

void setCurrency (Currency *currency*)

Parameters

- **the** – currency to set

setCurrency(String) method

Syntax

void setCurrency (String *currency*)

Parameters

- **the** – currency to set

setCustomer(Customer) method

Syntax

void setCustomer (Customer *customer*)

Parameters

- **customer** – the customer to set

setFile(File) method

Syntax

void setFile (File *file*)

Parameters

- **file** – the file to set

setForeignReference(String) method

Syntax

void setForeignReference (String *foreignReference*)

Parameters

- **foreignReference** – the foreign reference to set

setOtherFee(Long) method

Syntax

void setOtherFee (Long *otherFee*)

Parameters

- **otherFee** – the other fees to set

setPayment(Payment) method

Syntax

void setPayment (Payment *payment*)

Parameters

- **payment** – the payment to set

setPi(PaymentInstrument) method

Syntax

void setPi (PaymentInstrument *pi*)

Parameters

- **pi** – the PI to set

setReason(String) method

Syntax

void setReason (String *reason*)

Parameters

- **reason** – the reason to set

setReasonId(Integer) method

Syntax

void setReasonId (Integer *reasonId*)

Parameters

- **reasonId** – the reason ID to set

setStatus(Integer) method

Syntax

void setStatus (Integer *status*)

Parameters

- **status** – the status to set

setTxn(Transaction) method

Syntax

void setTxn (Transaction *txn*)

Parameters

- **txn** – the transaction to set

export package

Members

All public members of the export package.

- **api package** –

api package

Members

All public members of the api package.

- **FileExporter< RowType, ExportType > class** –
- **IFileExporter interface** –

FileExporter< RowType, ExportType > class

Syntax

```
public class FileExporter< RowType, ExportType >
```

Usage

This abstract class forms a base class for file exporters. It implements the `IFileExporter` interface and adds further abstract methods that need to be implemented. Those methods enhance the `IFileExporter` interface by means to be used within the mobiliser AMS context, which means it adds things like a file type, a batchrun type, etc. This class offers a default implementation of the `export()` method utilizing AMS functionalities such as opening up a new batchrun for the current file. It further drives the usage of spring batch as means of querying the data to be exported, to process the data and to do the actual file export by serializing the processed export data into a XML file.

The most important methods to implement are `doExport()` and `newFileStarted()` to implement the actual file export (how the records are read, processed and written) and how the file exporter should react to a new file (e.g. writing an ams files entry or finishing the previous file).

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

afterPropertiesSet() method

Syntax

void afterPropertiesSet () throws Exception

closeItemReader() method

Syntax

void closeItemReader () throws ItemStreamException

closeItemWriter() method

Syntax

void closeItemWriter () throws ItemStreamException

doExport(ExecutionContext, ClearingEvent) method

Is called from within export() and has to implement the actual export mechanism.

Syntax

abstract List< Long > doExport (ExecutionContext *ctx* ,
ClearingEvent *event*) throws Exception

Parameters

- **ctx** –
- **event** –

Exceptions

- **Exception** –

Usage

The reference implementation uses the item reader to read records from the database, uses the item processor to do some basic checks and uses the item writer to export the processed records.

existPaymentsToExport() method

Perform checks to see wheather there are records to export.

Syntax

abstract boolean existPaymentsToExport () throws Exception

Exceptions

- **Exception** –

export(final ClearingEvent, long) method

Default implementation of the file exporting mechanism.

Syntax

```
synchronized boolean export ( final ClearingEvent event , long  
callerId )
```

Usage

- Try to acquire a new batchrun lock
- Save the batchrun in the local variable `batchRunLock`
- Check if there is any data available to export
- Initialize the spring batch context
- Create a new file
- Initialize the item writer with the newly created file
- Open the item reader
- Open the item writer
- Signal a new file was created (creating a new ams files entry)
- Do the actual export
- Close item reader
- Close item writer
- Update exported records (e.g. set new status)
- Validate the exported files

exportRecord(ExportType) method

Syntax

```
void exportRecord ( ExportType record ) throws Exception
```

getBatchRunLock() method

Syntax

```
IBatchRunLock getBatchRunLock ()
```

getBatchRunMaxAgeInSeconds() method

Returns the max age in seconds of the batchrun.

Syntax

```
abstract int getBatchRunMaxAgeInSeconds ()
```


getBatchRunType() method

Returns the batchrun type.

Syntax`abstract int getBatchRunType ()`*getCallerId() method***Syntax**`Long getCallerId ()`*getCurrentBatchRun() method***Syntax**`BatchRun getCurrentBatchRun ()`*getCurrentFile() method***Syntax**`com.sybase365.mobiliser.money.persistence.model.system.File
getCurrentFile ()`*getFileCreationPath() method***Syntax**`String getFileCreationPath ()`*getFileDao() method***Syntax**`FileDAO getFileDao ()`*getFileType() method*

Returns the ams file type.

Syntax`abstract int getFileType ()`*getItemProcessor() method***Syntax**`ItemProcessor< RowType, ExportType > getItemProcessor ()`

getItemReader() method

Syntax

```
AbstractItemCountingItemStreamItemReader< RowType >  
getItemReader ()
```

getItemWriter() method

Syntax

```
ResourceAwareItemWriterItemStream< ExportType > getItemWriter  
()
```

getJdbcTemplate() method

Syntax

```
final JdbcOperations getJdbcTemplate ()
```

getNewFile(ClearingEvent) method

Is called whenever a new (physical) file has to be created (e.g.

Syntax

```
abstract File getNewFile ( ClearingEvent event )
```

Parameters

- **event** – The clearing finished event initially triggering the export

Usage

the first time it is called from within

```
export ()
```

). The file exporter reference implementation calls this method whenever it finds a switch of the debit/credit flag of the export data.

isItemReaderOpen() method

Syntax

```
boolean isItemReaderOpen ()
```

isItemWriterOpen() method

Syntax

`boolean isItemWriterOpen ()`

newFileStarted(File) method

Syntax

`void newFileStarted ( File file )`

openItemReader(ExecutionContext, boolean) method

Syntax

`void openItemReader ( ExecutionContext ctx, boolean forceClose )` throws
`IllegalArgumentException, ItemStreamException`

openItemWriter(ExecutionContext, boolean) method

Syntax

`void openItemWriter ( ExecutionContext ctx, boolean forceClose )` throws
`IllegalArgumentException, ItemStreamException`

renameFile(File) method

Rename the temporary file.

Syntax

`abstract File renameFile ( File file )`

Parameters

- **file** –

setBatchRunLock(IBatchRunLock) method

Syntax

`void setBatchRunLock ( IBatchRunLock batchRunLock )`

setCurrentFile(com.sybase365.mobiliser.money.persistence.model.system.File)
method

Syntax

```
void setCurrentFile  
( com.sybase365.mobiliser.money.persistence.model.system.File  
currentFile )
```

setFileCreationPath(String) *method*

Syntax

```
void setFileCreationPath ( String fileCreationPath )
```

setFileDao(FileDAO) *method*

Syntax

```
void setFileDao ( FileDAO fileDao )
```

setItemProcessor(ItemProcessor< RowType, ExportType >) *method*

Syntax

```
void setItemProcessor ( ItemProcessor< RowType, ExportType >  
itemProcessor )
```

setItemReader(AbstractItemCountingItemStreamItemReader< RowType >) *method*

Syntax

```
void setItemReader ( AbstractItemCountingItemStreamItemReader<  
RowType > itemReader )
```

setItemWriter(ResourceAwareItemWriterItemStream< ExportType >) *method*

Syntax

```
void setItemWriter ( ResourceAwareItemWriterItemStream<  
ExportType > itemWriter )
```

setJdbcTemplate(JdbcOperations) *method*

Syntax

```
void setJdbcTemplate ( JdbcOperations jdbcTemplate )
```

startNewFile(RowType, RowType) method

Handle the creation of a new file.

Syntax

```
abstract boolean startNewFile ( RowType lastRecord , RowType  
currentRecord )
```

Parameters

- **lastRecord** –
- **currentRecord** –

Usage

The reference implementation creates a new ams files entry.

updateRecords(final List< Long >, final long) method

Syntax

```
void updateRecords ( final List< Long > updatedPaymentIds , final  
long callerId ) throws DataAccessException
```

validateFile(File) method

Validate the written files.

Syntax

```
boolean validateFile ( File file )
```

Parameters

- **file** –

writeFilesEntry(BatchRun, String, Long) method

Syntax

```
long writeFilesEntry ( BatchRun batchRun , String fileName , Long  
callerId )
```

DF variable

Syntax

```
final DateFormat DF
```

exportedRecordsCount variable

Syntax

```
long exportedRecordsCount
```

fileCreationPath variable

Syntax

```
String fileCreationPath
```

fileList variable

Syntax

```
List< File > fileList
```

PAYMENT_STATUS_PROCESSED variable

Syntax

```
final int PAYMENT_STATUS_PROCESSED
```

IFileExporter interface

Syntax

```
public interface IFileExporter
```

Derived classes

- *FileExporter< RowType, ExportType >* on page 50

Usage

This interface defines methods which each ams file exporter has to implement.

Usually an AMS file exporter will be triggered by an event handler listening to "clearing finished events". This event handler will call the `export()` method, which must implement the complete file handling process, e.g. where the data is read from, how the data is processed and how the export file is written.

For a reference implementation see `ReferenceFileExporter` and its abstract base class `FileExporter`.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

export(final ClearingEvent, long) method

Has to do the actual export job.

Syntax

```
boolean export ( final ClearingEvent event , long callerId )
```

Parameters

- **event** –
- **callerId** –

Returns

`true` if no exceptions or unforeseen behavior occurred during processing `false` otherwise

Usage

`true` if no exceptions or unforeseen behavior occurred during processing `false` otherwise

businesslogic package

Members

All public members of the businesslogic package.

- **audit package** –
- **authentication package** –
- **billpayment package** –
- **binary package** –
- **bulkprocessing package** –
- **customer package** –
- **fee package** –
- **invoice package** –
- **normalizer package** –
- **oauth2 package** –
- **payment package** –
- **system package** –
- **transaction package** –
- **umgr package** –
- **umgrroles package** –
- **util package** –

- **wallet package** –

audit package

Members

All public members of the audit package.

- **IAuditLogLogic interface** –
- **INoteLogic interface** –

IAuditLogLogic interface

Syntax

```
public interface IAuditLogLogic
```

Usage

This interface defines CRUD operations on AuditLogs. In addition, a few find operations are defined. All operations depend on an existing database transaction.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

createAuditLog(final AuditLog, final long) method

Creates the passed AuditLog in the database.

Syntax

```
long createAuditLog ( final AuditLog auditLog , final long callerId )  
throws EntityMandatoryException, IllegalDataException
```

Parameters

- **auditLog** – the audit log entry to create.
- **callerId** – the caller id that is persisted as creator of this audit log.

Returns

the id of the newly created audit log.

Exceptions

- **EntityMandatoryException** – if the passed auditLog is null.
- **IllegalDataException** – if the passed auditLog has an id provided.

Usage

The id must be

`null`

, since it will be generated during creation.

the id of the newly created audit log.

deleteAuditLog(final long, final long) method

Deletes the passed AuditLog from the database.

Syntax

`void deleteAuditLog ( final long auditLogId , final long callerId )` throws `EntityNotFoundException`

Parameters

- **auditLogId** – the id of the audit log
- **callerId** – the current caller id

Exceptions

- **EntityNotFoundException** – if the audit log is not found with the passed id

getAuditLog(final long, final long) method

Gets the AuditLog with the passed id from the database.

Syntax

`AuditLog getAuditLog ( final long auditLogId , final long callerId )`
throws `EntityNotFoundException`

Parameters

- **auditLogId** – the id of the audit log
- **callerId** – the current caller id

Returns

the audit log

Exceptions

- **EntityNotFoundException** – if the audit log is not found with the passed id

Usage

the audit log

getAuditLogs(Date, Date, Long, Long, Long, Boolean, List< String >, Integer) method

Loads a list of AuditLogs based on the given search criteria.

Syntax

```
List< AuditLog > getAuditLogs ( Date start , Date end , Long callerId ,  
Long customerId , Long transactionId , Boolean faultyOnly , List< String >  
category , Integer rows )
```

Parameters

- **start** – filters by creation date after this date
- **end** – filters by creation date before this date
- **callerId** – filters by caller id
- **customerId** – filters by customer id
- **transactionId** – filters by transaction id
- **faultyOnly** – is set to `true` only audit logs with `AuditLog#getActionResultCode()` not equal to zero will be returned
- **category** – filters by category
- **rows** – limits the maximum size of the returned list

Usage

All criteria are optional, however this method will always return 1000 records max.

getAuditLogsByCustomer(final long, final long) method

Load AuditLogs created for a specific customer.

Syntax

```
List< AuditLog > getAuditLogsByCustomer ( final long customerId ,  
final long callerId ) throws EntityNotFoundException
```

Parameters

- **customerId** – the customer id to filter for
- **callerId** – the current caller id

Returns

a list of matching audit logs

Exceptions

- **EntityNotFoundException** – if no customer exists for the given customer id

Usage

a list of matching audit logs

getAuditLogsByTransaction(final long, final long) method

Load AuditLogs created for a specific transaction.

Syntax

```
List< AuditLog > getAuditLogsByTransaction ( final long txnId ,  
final long callerId ) throws EntityNotFoundException
```

Parameters

- **txnId** – the transaction id to filter for
- **callerId** – the current caller id

Returns

a list of matching audit logs

Exceptions

- **EntityNotFoundException** – if no transaction exists for the given transaction id

Usage

a list of matching audit logs

updateAuditLog(final AuditLog, final long) method

Updates the passed AuditLog in the database.

Syntax

```
void updateAuditLog ( final AuditLog auditLog , final long callerId )  
throws EntityMandatoryException
```

Parameters

- **auditLog** – the audit log to update
- **callerId** – the caller id that is persisted as updater of this audit log.

Exceptions

- **EntityMandatoryException** – if the passed auditLog or its id are null

Usage

The audit log id must be provided.

INoteLogic interface

Syntax

```
public interface INoteLogic
```

Usage

This interface defines CRUD operations on Notes. In addition, a few find operations are provided. All operations depend on an existing database transaction.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

createNote(final Note, final long) method

Creates the passed Note in the database.

Syntax

```
long createNote ( final Note note , final long callerId ) throws  
EntityMandatoryException, IllegalDataException
```

Parameters

- **note** – the note to create.
- **callerId** – the caller id that is persisted as creator of this note.

Returns

the id of the newly created note.

Exceptions

- **EntityMandatoryException** – if the passed note is null.
- **IllegalDataException** – if the passed note has an id provided.

Usage

The id must be

null

, since it will be generated during creation.

the id of the newly created note.

deleteNote(final long, final long) method

Deletes the passed Note from the database.

Syntax

```
void deleteNote ( final long noteId , final long callerId ) throws
EntityNotFoundException
```

Parameters

- **noteId** – the id of the note
- **callerId** – the current caller id

Exceptions

- **EntityNotFoundException** – if the note is not found with the passed id

getCountedNotes(final Long, final Integer, final Integer, final Date, final Date, final String, final String, final Integer, final int, final long) method

Searches for Notes based on the given search criteria.

Syntax

```
List< Note > getCountedNotes ( final Long customerId , final
Integer noteType , final Integer status , final Date fromDate , final
Date toDate , final String subject , final String creator , final
Integer creatorIdentificationType , final int maxRecords , final long
callerId ) throws EntityNotFoundException
```

Parameters

- **customerId** – filter by customer id
- **noteType** – filter by note type
- **status** – filter by status
- **fromDate** – filter by creation date after this date
- **toDate** – filter by creation date before this date
- **subject** – filter by subject
- **creator** – filter by creator; in this case you must also define the identification type, otherwise the method does not filter by creator
- **creatorIdentificationType** – identification type of the passed creator identification

- **maxRecords** – limits the maximum size of the result list
- **callerId** – filter by caller id

Returns

the list of matching Notes

Exceptions

- **EntityNotFoundException** – if filtering by a creator that does not exist

Usage

the list of matching Notes

getNote(final long, final long) method

Gets the Note with the passed id from the database.

Syntax

Note `getNote ( final long noteId , final long callerId )` throws
EntityNotFoundException

Parameters

- **noteId** – the id of the note
- **callerId** – the current caller id

Returns

the note

Exceptions

- **EntityNotFoundException** – if the note is not found with the passed id

Usage

the note

getNotesByCustomer(final long, final Integer, final Integer, final Date, final Date, final String, final String, final Integer, final long) method

Searches for Notes based on the given search criteria.

Syntax

List< Note > `getNotesByCustomer ( final long customerId , final Integer noteType , final Integer status , final Date fromDate , final Date toDate , final String subject , final String creator , final`

Integer *creatorIdentificationType* , final long *callerId*) throws
EntityNotFoundException

Parameters

- **customerId** – filter by customer id
- **noteType** – filter by note type
- **status** – filter by status
- **fromDate** – filter by creation date after this date
- **toDate** – filter by creation date before this date
- **subject** – filter by subject
- **creator** – filter by creator; in this case you must also define the identification type, otherwise the method does not filter by creator
- **creatorIdentificationType** – identification type of the passed creator identification
- **callerId** – filter by caller id

Returns

the list of matching Notes

Exceptions

- **EntityNotFoundException** – if filtering by a creator that does not exist

Usage

the list of matching Notes

updateNote(final Note, final long) method

Updates the passed Note in the database.

Syntax

void updateNote (final Note *note* , final long *callerId*) throws
EntityMandatoryException

Parameters

- **note** – the note to update
- **callerId** – the caller id that is persisted as updater of this note.

Exceptions

- **EntityMandatoryException** – if the passed note or it's id are null

Usage

The note id must be provided.

authentication package

Members

All public members of the authentication package.

- **api package** –

api package

Members

All public members of the api package.

- **IAuthenticationBroker interface** – An IAuthenticationBroker provides a way to retrieve the IAuthenticationHandler needed for a configured authentication method.
- **IAuthenticationHandler interface** –
- **IAuthenticationMethodService interface** – Implementations of this can compute the authentication methods required for a given SubTransaction.
- **IConsumerAuthenticator interface** – Abstraction of a consumer authentication for a transaction.
- **IResetableAuthHandler interface** – The Resetable Authentication Handler Interface is implemented by Authentication methods that can reset/restart the authentication process.

IAuthenticationBroker interface

An IAuthenticationBroker provides a way to retrieve the IAuthenticationHandler needed for a configured authentication method.

Syntax

```
public interface IAuthenticationBroker
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getAuthenticationHandler(final Integer) method

Returns the the authentication handler for the given method.

Syntax

```
IAuthenticationHandler getAuthenticationHandler ( final Integer  
authenticationMethod )
```


Parameters

- **authenticationMethod** – the authentication method

Returns

the handler or null if no handler is configured for the given method

Usage

If no match is found, null is returned.

the handler or null if no handler is configured for the given method

getResetableAuthHandler(final Integer) method

Like.

Syntax

```
IResetableAuthHandler getResetableAuthHandler ( final Integer  
authenticationMethod )
```

Parameters

- **authenticaionMethod** – the authentication method

Returns

the handler or null if no IResetableAuthHandler is found

Usage

the handler or null if no IResetableAuthHandler is found

IAuthenticationHandler interface

Syntax

```
public interface IAuthenticationHandler
```

Derived classes

- *IResetableAuthHandler* on page 75

Usage

The IAuthenticationHandler implementation is responsible for authenticating a transaction, i.e. get the user's approval for the transaction. If the authentication returns normally, it is assumed that the transaction was authenticated. Otherwise, an exception should be thrown.

Multiple instances with different coverages can be active parallel and additional implementation of this interface can be added to the OSGi container. As soon as a new instance is registered the IAuthenticationBroker service is notified and the service can be used. In order to add a new AuthenticationHandler, a new entry in MOB_AUTHENTICATION_METHOD table is needed for the new coverage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

authenticate(final SubTransaction, final String, final boolean) method

Syntax

```
void authenticate ( final SubTransaction transaction , final String  
authToken , final boolean payer ) throws AuthenticationException
```

Parameters

- **transaction** – the SubTransaction to authenticate (in general, this should be the reservation)
- **authToken** – the Authentication value
- **authTokenType** – the Authentication Type id to apply
- **payer** – indicates if payer or payee must authenticate for the transaction

Exceptions

- **AuthenticationException** – if authentication fails
- **AuthenticationFailedException** – if the authentication failed but can be retried
- **AuthenticationFailedPermanentlyException** – if the authentication failed and can not be retried

Usage

Authenticates the specified transaction.

getCoverage() method

Syntax

```
Integer[] getCoverage ()
```

Returns

an array of AuthenticationMethods IDs

Usage

Returns the types of authentication methods this handler is responsible for.

an array of AuthenticationMethods IDs

getName() method

Syntax

String getName ()

Usage

Returns the authentication handler name

initAuthentication(final SubTransaction, final boolean) method

Syntax

boolean initAuthentication (final SubTransaction *transaction* ,
final boolean *payer*) throws AuthenticationException

Parameters

- **transaction** – the SubTransaction to authenticate (in general, this should be the reservation)
- **method** – the AuthenticationMethod id to apply
- **payer** – indicates if payer or payee must authenticate for the transaction

Returns

true if the initialisation did not create a new authentication token and the authentication can be continued right after the initialization. On false the transaction processing should break immediately

Exceptions

- **AuthenticationException** – if the initialization fails and the transaction processing can to be picked up later

Usage

Initializes the Authentication for payer or payee. This method can be used by the actual authentication handler to send notifications to the participants, such as one time transaction tokens created on time.

true if the initialisation did not create a new authentication token and the authentication can be continued right after the initialization. On false the transaction processing should break immediately

isDynamicCredentials() method

Syntax

```
boolean isDynamicCredentials ()
```

Usage

Returns true if the authentication handler initializes a new credential in the init call. If it does not an authorise call can contain the credentials already

isInlineAuthentication() method

Syntax

```
boolean isInlineAuthentication ()
```

Usage

Returns true if it is an inline authentication and the transaction is continued/authenticated without the service caller automatically

Example: An SMS AoC authentication handler calls the brand mobiliser and the brand mobilier does the continueAuthentication service call.

IAuthenticationMethodService interface

Implementations of this can compute the authentication methods required for a given SubTransaction.

Syntax

```
public interface IAuthenticationMethodService
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

computeRules(final SubTransaction, final Long) method

Syntax

AuthenticationMethodComputation computeRules (final
SubTransaction *txn* , final Long *actor*) throws
MobiliserServiceException

Parameters

- **txn** – the subtransaction
- **actor** – the actor

Returns

the computed authentication methods

Exceptions

- **MobiliserServiceException** – if computing the rules fails for any business rules related reason

Usage

Computes the authentication rules for the given transaction.

the computed authentication methods

IConsumerAuthenticator interface

Abstraction of a consumer authentication for a transaction.

Syntax

public interface IConsumerAuthenticator

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

consumerAuthentication(final SubTransaction, final Integer, final String, final boolean, final Integer, final String, final boolean) method

Syntax

ConsumerAuthenticationResult consumerAuthentication (final
SubTransaction *txn* , final Integer *payeeAuthenticationMethod* , final

String payeeAuthToken , final boolean resetPayeeAuth , final Integer payerAuthenticationMethod , final String payerAuthToken , final boolean resetPayerAuth) throws MobiliserServiceException

Parameters

- **txn** – the transaction to authenticate
- **payeeAuthenticationMethod** – the auth method for the payee
- **payeeAuthToken** – the auth token for the payee
- **payerAuthenticationMethod** – the auth method for the payer
- **payerAuthToken** – the auth token for the payer

Returns

ConsumerAuthenticationResult the result of the authentication:

- FAILED_BOTH: authentication failed and may be repeated
- FAILED_PAYER: authentication failed for payee and may be repeated
- FAILED_PAYEE: authentication failed for payer and may be repeated

Exceptions

- **MobiliserServiceException** – if computing the authenticating fails

Usage

Performs the appropriate transaction authentication with the affected consumer.

ConsumerAuthenticationResult the result of the authentication:

- FAILED_BOTH: authentication failed and may be repeated
- FAILED_PAYER: authentication failed for payee and may be repeated
- FAILED_PAYEE: authentication failed for payer and may be repeated

initConsumerAuthentication(final SubTransaction, final Integer, final Integer) method

Syntax

ConsumerAuthenticationResult initConsumerAuthentication (final SubTransaction *txn* , final Integer *payeeAuthenticationMethod* , final Integer *payerAuthenticationMethod*) throws MobiliserServiceException

Parameters

- **txn** – the transaction to authenticate
- **payeeAuthenticationMethod** – the auth method for the payee

- **payerAuthenticationMethod** – the auth method for the payer

Returns

`ConsumerAuthenticationResult` the result of the authentication:

- **FAILED_BOTH**: authentication is asynchronous for both participants
- **FAILED_PAYER**: authentication is asynchronous for payee
- **FAILED_PAYEE**: authentication is asynchronous for payer

Exceptions

- **MobiliserServiceException** – if computing the initialisation fails final

Usage

Performs the appropriate transaction authentication initialization with the affected consumers. If the call does not return success, one of the participants authentication is asynchronous and the transaction processing should wait/stop until the transaction processing can be picked up with the new participants auth token in the request

`ConsumerAuthenticationResult` the result of the authentication:

- **FAILED_BOTH**: authentication is asynchronous for both participants
- **FAILED_PAYER**: authentication is asynchronous for payee
- **FAILED_PAYEE**: authentication is asynchronous for payer

IResetableAuthHandler interface

The Resetable Authentication Handler Interface is implemented by Authentication methods that can reset/restart the authentication process.

Syntax

```
public interface IResetableAuthHandler
```

Usage

This can be used to implement the business logic such as resent an authentication token that was generated during the authentication initialization. Reset does not mean that the authentication is in the initial state afterwards. Wrong credentials counts may not be reset to 0.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

resetAuthentication(final SubTransaction, final boolean) method

Reset the transaction authentication.

Syntax

```
void resetAuthentication ( final SubTransaction transaction , final boolean payer ) throws AuthenticationException
```

Parameters

- **transaction** –
- **payer** –

Exceptions

- **AuthenticationException** – like when the transaction can not be reset multiple times or current condition does not allow to reset it

billpayment package

Members

All public members of the billpayment package.

- **api package** –

api package

Members

All public members of the api package.

- **BillPaymentHandler interface** – The BillPaymentHandler interface is implemented by actual bill payment handler.
- **IBillPaymentBroker interface** – IBillPaymentBroker OSGi service interface of the bill payment broker.
- **InvoiceStatusCode class** – Additional status codes / return values used by the invoice bundles.

BillPaymentHandler interface

The BillPaymentHandler interface is implemented by actual bill payment handler.

Syntax

```
public interface BillPaymentHandler
```


Usage

On service startup the actual bill payment handler triggers an registration event and the bill payment broker gets the coverage of it

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

checkInvoice(Invoice) method

This method gets called before the funds get authorized from the bill payer.

Syntax

```
void checkInvoice ( Invoice invoice ) throws  
UnsupportedOperationException, BillPaymentHandlerException
```

Parameters

- **invoice** – to check

Exceptions

- **UnsupportedOperationException** – must be thrown if checking is not possible, the bill payment process will continue as if the check were successful
- **BillPaymentHandlerException** – indicates that the bill cannot be paid, bill payment process must fail

Usage

It is important that all possible checks that could prevent the bill from being paid are executed at this stage. If the bill cannot be paid, a

`BillPaymentHandlerException`

must be thrown. However, the `checkInvoice` is not binding in any way. If the authorization is not possible at a later point no further information is sent to the bill payment handler.

getInvoiceTypeHandlerTypeCoverage() method

Returns the list of invoice types for which this implementation is responsible for.

Syntax

```
int[] getInvoiceTypeHandlerTypeCoverage ()
```

Returns

the list of invoice types for which this implementation is responsible for.

Usage

the list of invoice types for which this implementation is responsible for.

getName() method

Syntax

```
String getName ()
```

Returns

the name.

Usage

Returns the name of the PaymentHandler. Can be the class name or a short description. This is shown by the broker in the log file

the name.

invoiceStatusChangeAdvice(Invoice, InvoiceStatus) method

This service is called when the Invoice status has changed (e.g.

Syntax

```
void invoiceStatusChangeAdvice ( Invoice invoice , InvoiceStatus  
invoiceStatus )
```

Parameters

- **invoice** – where the status has changed
- **newInvoiceStatus** –

Usage

canceled by user, expired, failed, ...).

payInvoice(Invoice, long) method

This method gets called when the invoice is being paid.

Syntax

```
String payInvoice ( Invoice invoice , long txnId ) throws  
BillPaymentHandlerException
```

Parameters

- **invoice** – the invoice to pay

Returns

A reference for the bill payment process, can be `null` if no external reference is generated

Exceptions

- **BillPaymentHandlerException** – if the bill cannot be closed, this will result in the transaction being rolled back.

Usage

The money from the payer is already secured.

A reference for the bill payment process, can be `null` if no external reference is generated

resendInvoiceStatusChange(Invoice, int) method

This method is called when the invoices is pending status on the OnlineCommmandQueue.

Syntax

```
boolean resendInvoiceStatusChange ( Invoice invoice , int counter )
```

Parameters

- **invoice** – where the status needs to be resend
- **counter** – retry counter

Returns

true if success or false.

Usage

true if success or false.

updateAndCreateInvoices(InvoiceType) method

This method is called based on the configure interval.

Syntax

```
void updateAndCreateInvoices ( InvoiceType invoiceType )
```

Parameters

- **invoiceType** – where to update/create an invoice

Usage

It is not connected to the actual bill payment process but rather gives the implementation the option to update existing invoices or create new ones.

updateAndCreateInvoicesForInvoiceConfiguration(InvoiceConfiguration) method

This method is used to query for new or updated invoices based on an existing InvoiceConfiguration.

Syntax

```
void updateAndCreateInvoicesForInvoiceConfiguration  
( InvoiceConfiguration invoiceConfiguration )
```

Parameters

- **invoiceType** – where to update/create the invoice configuration

Usage

The process is usually invoked by the customer.

IBillPaymentBroker interface

IBillPaymentBroker OSGi service interface of the bill payment broker.

Syntax

```
public interface IBillPaymentBroker
```

Usage

The broker is responsible to get the payment handler for a given type

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getBillPaymentHandler(Integer) method

Find the matching BillPaymentHandler.

Syntax

```
BillPaymentHandler getBillPaymentHandler ( Integer type )
```

Parameters

- **type** –

Returns

the matching bill payment handler or null if no matching

Usage

the matching bill payment handler or null if no matching

InvoiceStatusCode class

Additional status codes / return values used by the invoice bundles.

Syntax

```
public class InvoiceStatusCode
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

ERROR_ACTIVE_INVOICES variable**Syntax**

```
final int ERROR_ACTIVE_INVOICES
```

ERROR_CUSTOMER_REFERENCE_UNKNOWN variable**Syntax**

```
final int ERROR_CUSTOMER_REFERENCE_UNKNOWN
```

ERROR_INVOICE_ALREADY_PAID variable**Syntax**

```
final int ERROR_INVOICE_ALREADY_PAID
```

ERROR_INVOICE_AMOUNT_INCORRECT variable**Syntax**

```
final int ERROR_INVOICE_AMOUNT_INCORRECT
```

ERROR_INVOICE_AMOUNT_TOO_HIGH variable**Syntax**

```
final int ERROR_INVOICE_AMOUNT_TOO_HIGH
```

ERROR_INVOICE_AMOUNT_TOO_LOW variable

Syntax

```
final int ERROR_INVOICE_AMOUNT_TOO_LOW
```

ERROR_INVOICE_CANCELED variable

Syntax

```
final int ERROR_INVOICE_CANCELED
```

ERROR_INVOICE_EXPIRED variable

Syntax

```
final int ERROR_INVOICE_EXPIRED
```

ERROR_INVOICE_REFERENCE_UNKNOWN variable

Syntax

```
final int ERROR_INVOICE_REFERENCE_UNKNOWN
```

ERROR_INVOICE_STATUS_INCORRECT variable

Syntax

```
final int ERROR_INVOICE_STATUS_INCORRECT
```

ERROR_INVOICE_TYPE_REFERENCE_UNKNOWN variable

Syntax

```
final int ERROR_INVOICE_TYPE_REFERENCE_UNKNOWN
```

ERROR_ISSUER_TECH_ERROR variable

Syntax

```
final int ERROR_ISSUER_TECH_ERROR
```

ERROR_UNKNOWN_ERROR variable

Syntax

```
final int ERROR_UNKNOWN_ERROR
```

binary package

Members

All public members of the binary package.

- **BinaryData class** – Data container for binary data and some meta information.
- **BinaryDataDestination interface** – A mobiliser abstraction for adding binary data.
- **BinaryDataDownloadException class** – Exception used when data is being uploaded.
- **BinaryDataSource interface** – A mobiliser abstraction for accessing binary data.
- **BinaryDataUploadException class** – Exception used when data is being uploaded.
- **BinaryDataUrlHandler interface** – The interface for creating upload and download urls.

BinaryData class

Data container for binary data and some meta information.

Syntax

```
public class BinaryData
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getContentType() method

Syntax

```
String getContentType ()
```

Returns

the contentType

Usage

the contentType

getData() method

Syntax

```
byte[] getData ()
```

Returns

the data

Usage

the data

getName() method

Syntax

```
String getName ()
```

Returns

the name

Usage

the name

isAddContentDisposition() method

Syntax

```
boolean isAddContentDisposition ()
```

Returns

the addContentDisposition

Usage

the addContentDisposition

setAddContentDisposition(final boolean) method

Syntax

```
void setAddContentDisposition ( final boolean addContentDisposition )
```

Parameters

- **addContentDisposition** – the addContentDisposition to set

setContentType(final String) method

Syntax

```
void setContentType ( final String contentType )
```


Parameters

- **contentType** – the contentType to set

setData(final byte[]) method

Syntax

```
void setData ( final byte[] data )
```

Parameters

- **data** – the data to set

setName(final String) method

Syntax

```
void setName ( final String name )
```

Parameters

- **name** – the name to set

addContentDisposition variable

Syntax

```
boolean addContentDisposition
```

contentType variable

Syntax

```
String contentType
```

data variable

Syntax

```
byte[] data
```

name variable

Syntax

```
String name
```

BinaryDataDestination interface

A mobiliser abstraction for adding binary data.

Syntax

```
public interface BinaryDataDestination
```

Usage

Implementations of this interface will provide access to generic binary data stores.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getType() method

The type name for this binary data source.

Syntax

```
String getType ()
```

Returns

the name

Usage

This should be unique for each data source.

the name

storeData(final Map< String, String >, final BinaryData, final CallerInformation) method

Upload the binary data as requested by the caller.

Syntax

```
Long storeData ( final Map< String, String > parameters , final  
BinaryData data , final CallerInformation caller ) throws  
BinaryDataUploadException
```

Parameters

- **parameters** – the parameters for the binary data.
- **caller** – the id of the caller accessing the binary data
- **data** – the binary data to update

Returns

the id of the newly created data

Exceptions

- **BinaryDataUploadException class** – the if data cannot be uploaded for any reason

Usage

the id of the newly created data

BinaryDataDownloadException class

Exception used when data is being uploaded.

Syntax

```
public class BinaryDataDownloadException
```

Usage

The Reason will give the exact cause of failure.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

BinaryDataDownloadException(Reason, String) constructor

Syntax

```
BinaryDataDownloadException ( Reason reason , String message )
```

Parameters

- **reason** –
- **message** –

BinaryDataDownloadException(Reason, String, final Throwable) constructor

Syntax

```
BinaryDataDownloadException ( Reason reason , String message , final  
Throwable e )
```

Parameters

- **reason** –

- **message** –
- **e** –

getReason() method

Syntax

Reason getReason ()

Returns

the reason

Usage

the reason

Reason() enumeration

Reason the download failed.

Enum Constant Summary

- **ACCESS_DENIED** –
- **TOO_LARGE** –
- **DATA_NOT_FOUND** –
- **INTERNAL_ERROR** –
- **OTHER** –

reason variable

Syntax

Reason reason

BinaryDataSource interface

A mobiliser abstraction for accessing binary data.

Syntax

```
public interface BinaryDataSource
```

Usage

Implementations of this interface will provide access to generic binary data stores.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getData(final Map< String, String >, final CallerInformation) method

Load the binary data as requested by the caller.

Syntax

BinaryData.getData (final Map< String, String > *parameters* , final CallerInformation *caller*) throws BinaryDataDownloadException

Parameters

- **parameters** – the parameters for getting the binary data.
- **caller** – the id of the caller accessing the binary data

Returns

the binary data

Exceptions

- **BinaryDataDownloadException class** – the if data cannot be fetched for any reason

Usage

the binary data

getType() method

The type name for this binary data source.

Syntax

String.getType ()

Returns

the name

Usage

This should be unique for each data source.

the name

BinaryDataUploadException class

Exception used when data is being uploaded.

Syntax

```
public class BinaryDataUploadException
```

Usage

The Reason will give the exact cause of failure.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

BinaryDataUploadException(Reason, String) constructor

Syntax

```
BinaryDataUploadException ( Reason reason , String message )
```

Parameters

- **reason** –
- **message** –

BinaryDataUploadException(Reason, String, Exception) constructor

Syntax

```
BinaryDataUploadException ( Reason reason , String message ,  
Exception e )
```

Parameters

- **reason** –
- **message** –
- **e** –

getReason() method

Syntax

```
Reason getReason ()
```

Returns

the reason

Usage

the reason

Reason() enumeration

Reason the upload failed.

Enum Constant Summary

- **ACCESS_DENIED** –
- **TOO_LARGE** –
- **UNSUPPORTED_CONTENT_TYPE** –
- **INVALID_DATA** –
- **INTERNAL_ERROR** –
- **OTHER** –

*reason variable**Syntax*

Reason reason

BinaryDataUrlHandler interface

The interface for creating upload and download urls.

Syntax

```
public interface BinaryDataUrlHandler
```

Usage

This allows clients to interactive with mobiliser in a native way by posting or getting data from given urls. The typical flow is for a client to request a download or upload url which will be returned with some embedded encrypted parameters. Then the actual HTTP POST or GET will use this returned URL. The types referenced here must match to existing implementations of com.sybase365.mobiliser.money.businesslogic.binary.BinaryDataSources or com.sybase365.mobiliser.money.businesslogic.binary.BinaryDataDestination

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

createDownloadUrl(final String, final Map< String, String >) method

Returns a ready-to-consume download URL.

Syntax

```
String createDownloadUrl ( final String type ,  final Map< String,
String > parameters )
```

Parameters

- **type** – the type or url to create - this maps to a `BinaryDataSource`
- **parameters** – parameters defining the data to be fetched

Returns

the url which can be used to actually download the data.

Usage

The parameters will be encoded (encrypted) into the URL and will be passed to the `BinaryDataProvider` of the given type - once the download is being made.

the url which can be used to actually download the data.

createUploadUrl(final String, final Map< String, String >) method

Returns a ready-to-use upload URL.

Syntax

```
String createUploadUrl ( final String type ,   final Map< String,  
String > parameters )
```

Parameters

- **type** – the type or url to create - this maps to a `BinaryDataDestination`
- **parameters** – parameters defining the data to be uploaded

Returns

the url which can be used to actually upload the data.

Usage

The parameters will be encoded (encrypted) into the URL and will be passed to the `BinaryDataProvider` of the given type along with the binary data - once the upload is being made.

the url which can be used to actually upload the data.

parseUrlParameter(final String) method

Parses the encoded and encrypted URL parameter into a key/value map.

Syntax

```
Map< String, String > parseUrlParameter ( final String parameter )
```


Parameters

- **parameter** – the parameter in the form returned by this handler.

Returns

the parameters parsed from the URL

Usage

the parameters parsed from the URL

bulkprocessing package

Members

All public members of the bulkprocessing package.

- **api package** –
- **handlers package** –

api package

Members

All public members of the api package.

- **IBulkProcessingBroker interface** – The IBulkProcessingBroker is responsible for determining which IBulkProcessingFileHandler to use for processing a binary file of a given file type.
- **IBulkProcessingConfiguration interface** –
- **IBulkProcessingConverter interface** –
- **IBulkProcessingFileHandler interface** –
- **IBulkProcessingLineProcessor interface** –
- **IBulkProcessingLogic interface** –

IBulkProcessingBroker interface

The IBulkProcessingBroker is responsible for determining which IBulkProcessingFileHandler to use for processing a binary file of a given file type.

Syntax

```
public interface IBulkProcessingBroker
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getBulkProcessingFileHandler(int) method

Find the matching IBulkProcessingFileHandler for the passed type.

Syntax

```
IBulkProcessingFileHandler getBulkProcessingFileHandler ( int  
fileType )
```

Parameters

- **fileType** –

Returns

the matching bulk processing handler, or null if no handler is available

Usage

the matching bulk processing handler, or null if no handler is available

IBulkProcessingConfiguration interface

Syntax

```
public interface IBulkProcessingConfiguration
```

Usage

Configuration interface for the IBulkProcessingFileHandler implementations.

A bulk processing file handler implementation uses these methods to configure the thread pool executor queuein and executing worker threads performing the bulk processing operations.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCorePoolSize() method

The executor's core worker thread pool size.

Syntax

```
int getCorePoolSize ()
```

getKeepAliveSeconds() method

The thread pool's keep alive seconds.

Syntax

```
int getKeepAliveSeconds ()
```

getMaxPoolSize() method

The executor's max worker thread pool size.

Syntax

```
int getMaxPoolSize ()
```

getQueueCapacity() method

Size of the queue.

Syntax

```
int getQueueCapacity ()
```

getThreadRejectedRetryLimit() method

Number of times a rejected thread is retried to be queued.

Syntax

```
int getThreadRejectedRetryLimit ()
```

getThreadRejectedWaitTimeMillis() method

Returns the time the executor waits before retrying a failed queue operation.

Syntax

```
long getThreadRejectedWaitTimeMillis ()
```

getWaitUntilShutdownTimeMillis() method

Wait time to finish active workers after shutdown was called on the executor.

Syntax

```
long getWaitUntilShutdownTimeMillis ()
```

IBulkProcessingConverter interface

Syntax

```
public interface IBulkProcessingConverter
```

Usage

Converter interface that can be used by any service to convert bulk upload service beans to persistence beans and vice versa.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

fromContract(BulkFileType) method

Convert BulkFileType service bean into BulkProcessingFileType persistence bean.

Syntax

```
BulkProcessingFileType fromContract ( BulkFileType contract )
```

Parameters

- **contract** –

fromContract(WrkBulk) method

Convert WrkBulk service bean into WrkBulkProcessing persistence bean.

Syntax

```
WrkBulkProcessing fromContract ( WrkBulk contract )
```

Parameters

- **contract** –

toContract(BulkProcessingFileType) method

Convert BulkProcessingFileType persistence bean into BulkFileType service bean.

Syntax

```
BulkFileType toContract ( BulkProcessingFileType model )
```

Parameters

- **contract** –

toContract(WrkBulkProcessing) method

Convert WrkBulkProcessing persistence bean into WrkBulk service bean.

Syntax

```
WrkBulk toContract ( WrkBulkProcessing model )
```

Parameters

- **contract** –

toContract(BulkProcessingFile) method

Convert BulkFileHistoryBean persistence bean into BulkProcessingFile service bean.

Syntax

```
BulkFileHistoryListItem toContract ( BulkProcessingFile model )
```

Parameters

- **contract** –

toContract2(BulkProcessingFile) method

Convert BulkProcessingFile service bean into BulkFileHistoryListItems persistence bean.

Syntax

```
BulkFileHistoryBean toContract2 ( BulkProcessingFile model )
```

Parameters

- **contract** –

IBulkProcessingFileHandler interface*Syntax*

```
public interface IBulkProcessingFileHandler
```

Derived classes

- *AbstractMobiliserServiceBulkProcessor* on page 105

Usage

The IBulkProcessingFileHandler is responsible for parsing and processing files. The interface does not impose any limitations on the file format and the type of processing of file content. However, the intended use is to implement bulk (or batch) processing of information through this interface.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCoverage() method

Returns the list of supported file types, which can be handled by this bulk processing handler.

Syntax

```
int[] getCoverage ()
```

Returns

the list of supported file types, which can be handled by this bulk processing handler

Usage

the list of supported file types, which can be handled by this bulk processing handler

getName() method

Returns the name of the IBulkProcessingFileHandler.

Syntax

```
String getName ()
```

Returns

the name.

Usage

Can be the class name or a short description. This is shown by the broker in the log file the name.

process(final byte[]) method

Processes the provided file content.

Syntax

```
byte[] process ( final byte[] fileContent ) throws  
BulkProcessingFileHandlerException
```

Parameters

- **fileContent** – the binary file content

Returns

the contents of a response file

Exceptions

- **BulkProcessingFileHandlerException** – if the file processing failed

Usage

If the file is processed successfully this method must return silently.
the contents of a response file

IBulkProcessingLineProcessor interface

Syntax

```
public interface IBulkProcessingLineProcessor
```

Derived classes

- *AbstractOXMLLineProcessor* on page 107

Usage

The *IBulkProcessingLineProcessor* is responsible for processing a single line from a (bulk) file. This interface is typically used by a file handler and is called for each line the file handler parsed from the file.

It is the responsibility of the line processor to further parse the line's content (e.g. split it into different fields) and execute whichever operation is intended for the file content provided to the file handler (e.g. service call to create a new customer or do a transaction, etc.).

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

processLine(String) method

Processes a single line from a file.

Syntax

```
String processLine ( String line ) throws  
BulkProcessingLineException
```

Parameters

- **line** –

Returns

A result string (for example a serialized service response)

Exceptions

- **BulkProcessingLineException** – if the line processing failed

Usage

A result string (for example a serialized service response)

IBulkProcessingLogic interface

Syntax

```
public interface IBulkProcessingLogic
```

Usage

The IBulkProcessingLogic interface is responsible for implementing methods to interact with the IBulkProcessingFileHandler and IBulkProcessingFileHandler to trigger the execution of a bulk (or batch) file.

It further defines (CRUD) methods to handle the configuration of bulk file types.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

createBulkFileType(BulkProcessingFileType, long) method

Create a new bulk processing file type.

Syntax

```
void createBulkFileType ( BulkProcessingFileType bulkFileType ,  
long callerId ) throws EntityNotUniqueException
```

Parameters

- **bulkFileType** –
- **callerId** –

Exceptions

- **EntityNotUniqueException** –

createWrkBulk(WrkBulkProcessing, long) method
Create a new maker/checker configuration for a file type.

Syntax

`void createWrkBulk ( WrkBulkProcessing wrkBulk , long callerId )` throws
`EntityNotUniqueException`

Parameters

- **wrkBulk** –
- **callerId** –

Exceptions

- **EntityNotUniqueException** –

deleteBulkFileType(int, long) method
Delete an existing bulk processing file type.

Syntax

`void deleteBulkFileType ( int bulkFileTypeId , long callerId )` throws
`EntityNotFoundException`

Parameters

- **bulkFileTypeId** –
- **callerId** –

Exceptions

- **EntityNotFoundException** –

deleteWrkBulk(int, long) method
Remove the maker/checker configuration of the given file type.

Syntax

`void deleteWrkBulk ( int bulkFileTypeId , long callerId )` throws
`EntityNotFoundException`

Parameters

- **bulkFileTypeId** –
- **callerId** –

Exceptions

- **EntityNotFoundException** –

getBulkFileType(int) method

Return a specific bulk processing file type.

Syntax

```
BulkProcessingFileType getBulkFileType ( int bulkFileTypeId )
```

Parameters

- **bulkFileTypeId** –

getBulkFileTypes() method

Return all bulk processing file types.

Syntax

```
List< BulkProcessingFileType > getBulkFileTypes ()
```

getFinishedBulkProcessingFile(Long) method

Return a single finished file.

Syntax

```
BulkProcessingFile getFinishedBulkProcessingFile ( Long  
bulkFileInId )
```

Parameters

- **bulkFileInId** –

getFinishedBulkProcessingFiles(final Integer, final Integer, final String, final Date, final Date, final Long) method

Return all finished files that match the criteria.

Syntax

```
List< BulkProcessingFile > getFinishedBulkProcessingFiles  
( final Integer bulkFileType , final Integer bulkFileStatus , final  
String fileName , final Date startDate , final Date endDate , final  
Long uploader )
```

Parameters

- **bulkFileType** –
- **bulkFileStatus** –
- **fileName** –
- **startDate** –
- **endDate** –
- **uploader** –

getWrkBulk() method

Return all bulk file type maker/checker configurations.

Syntax

```
List< WrkBulkProcessing > getWrkBulk ()
```

getWrkBulk(int) method

Return the maker/checker configuration for a specific file type.

Syntax

```
WrkBulkProcessing getWrkBulk ( int bulkFileTypeId )
```

Parameters

- **bulkFileTypeId** –

processBulkProcessingFile(final Integer, final String, final String, final byte[], final Boolean, final Long) method

Process a (bulk) file.

Syntax

```
void processBulkProcessingFile ( final Integer fileTypeId , final  
String fileName , final String contentType , final byte[] file , final  
Boolean approve , final Long callerId ) throws IllegalArgumentException
```

Parameters

- **fileTypeId** –
- **fileName** –
- **contentType** –
- **file** –
- **approve** –
- **callerId** –

Exceptions

- **IllegalDataException** –

updateBulkFileType(BulkProcessingFileType, long) method

Update an existing bulk processing file type.

Syntax

`void updateBulkFileType (BulkProcessingFileType bulkFileType ,
long callerId) throws EntityNotFoundException`

Parameters

- **bulkFileType** –
- **callerId** –

Exceptions

- **EntityNotFoundException** –

updateWrkBulk(WrkBulkProcessing, long) method

Update a maker/checker configuration for a file type.

Syntax

`void updateWrkBulk (WrkBulkProcessing wrkBulk , long callerId) throws
EntityNotFoundException`

Parameters

- **wrkBulk** –
- **callerId** –

Exceptions

- **EntityNotFoundException** –

handlers package

Members

All public members of the handlers package.

- **util package** –

util package*Members*

All public members of the util package.

- **AbstractMobiliserServiceBulkProcessor** class –
- **AbstractOXMLLineProcessor** class –
- **LineProcessorRunnable** class –

*AbstractMobiliserServiceBulkProcessor class**Syntax*

```
public class AbstractMobiliserServiceBulkProcessor
```

Usage

The `AbstractMobiliserServiceBulkProcessor` provides an implementation for bulk processes that take a file and generate a Mobiliser request for each single line in that file. This class takes care of threading the execution and run the calls in parallel. Implementing classes must provide a `parse` method for translating a single line into a `MobiliserRequestType` (line processor) and means of configuring the thread pool executor.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

*AbstractMobiliserServiceBulkProcessor() constructor***Syntax**

```
AbstractMobiliserServiceBulkProcessor ()
```

*afterPropertiesSet() method***Syntax**

```
void afterPropertiesSet ()
```

*getCorePoolSize() method***Syntax**

```
abstract int getCorePoolSize ()
```

getKeepAliveSeconds() method

Syntax

```
abstract int getKeepAliveSeconds ()
```

getLineProcessor() method

Syntax

```
abstract IBulkProcessingLineProcessor getLineProcessor ()
```

getMaxPoolSize() method

Syntax

```
abstract int getMaxPoolSize ()
```

getQueueCapacity() method

Syntax

```
abstract int getQueueCapacity ()
```

getThreadpoolExecutor() method

Syntax

```
ThreadPoolTaskExecutor getThreadpoolExecutor ()
```

getThreadRejectedRetryLimit() method

Syntax

```
abstract int getThreadRejectedRetryLimit ()
```

getThreadRejectedWaitTimeMillis() method

Syntax

```
abstract long getThreadRejectedWaitTimeMillis ()
```

getWaitUntilShutdownTimeMillis() method

Syntax

```
abstract long getWaitUntilShutdownTimeMillis ()
```

process(final byte[]) method

Processes the provided file content.

Syntax

```
byte[] process ( final byte[] fileContent ) throws
BulkProcessingFileHandlerException
```

Parameters

- **fileContent** – the binary file content

Returns

the contents of a response file

Exceptions

- **BulkProcessingFileHandlerException** – if the file processing failed

Usage

If the file is processed successfully this method must return silently.

the contents of a response file

*AbstractOXMLLineProcessor class***Syntax**

```
public class AbstractOXMLLineProcessor
```

Usage

The abstract class `AbstractOXMLLineProcessor` defines and offers default implementations for methods common among line processors, which are supposed to handle serialized service request.

Default implementations to marshall and umarshall mobiliser service requests and check a service response for error codes are given. An abstract method to do the actual service call is defined.

A new line processor does not have to extends this class, but it is advised if the line processor is supposed to handle bulk files where each line represents a complete mobiliser service call.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

afterPropertiesSet() method

Syntax

```
void afterPropertiesSet ()
```

checkResponse(T, String) method

Syntax

```
protected< T extends MobiliserResponseType > void  
checkResponse ( T response , String responseString )
```

doServiceCall(MobiliserRequestType) method

Syntax

```
abstract MobiliserResponseType doServiceCall  
( MobiliserRequestType request ) throws BulkProcessingLineException
```

getMarshaller() method

Syntax

```
Marshaller getMarshaller ()
```

getUnmarshaller() method

Syntax

```
Unmarshaller getUnmarshaller ()
```

marshallResponse(T) method

Syntax

```
protected< T > String marshallResponse ( T response ) throws  
BulkProcessingLineException
```

setMarshaller(Marshaller) method

Syntax

```
void setMarshaller ( Marshaller marshaller )
```

setUnmarshaller(Unmarshaller) method

Syntax

```
void setUnmarshaller ( Unmarshaller unmarshaller )
```


*unmarshallRequest(String) method***Syntax**

```
protected< T > T unmarshallRequest ( String line ) throws
BulkProcessingLineException
```

*LineProcessorRunnable class***Syntax**

```
public class LineProcessorRunnable
```

Usage

This class represents a task which is queued to the thread pool executor by AbstractMobiliserServiceBulkProcessor for each line while the bulk file is parsed.

Its run () implementation uses a configurable line processor to process the line parsed from the bulk file. It stores the result of the processed line mapped to the line number for easier consolidation in the mobiliser service bulk processor (put the responses in the right order, given the execution of the line processors is asynchronous). Exceptions are collected in an exception list and it is up to the mobiliser service bulk processor to evaluate them and take actions where required.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

*LineProcessorRunnable(String, long, IBulkProcessingLineProcessor, Map< Long, String >, Map< Long, Throwable >) constructor***Syntax**

```
LineProcessorRunnable ( String line , long lineNo ,
IBulkProcessingLineProcessor lineProcessor , Map< Long, String >
responseMap , Map< Long, Throwable > exceptions )
```

*run() method***Syntax**

```
void run ()
```

customer package

Members

All public members of the customer package.

- **Captcha class** – This class represents the ID of the OTP and an URL that can be used to download the captcha.
- **IAddressLogic interface** –
- **IAttachmentLogic interface** –
- **ICommissionConfigurationLogic interface** –
- **ICustomerAttributeLogic interface** –
- **ICustomerLifecycleHandler interface** –
- **ICustomerLogic interface** –
- **ICustomerNetworksLogic interface** –
- **ICustomerOtpLogic interface** –
- **IIdentificationLogic interface** –
- **IIdentityLogic interface** –
- **INotificationLogic interface** –
- **IOtpLogic interface** –
- **IRemittanceVoucherLogic interface** –

Captcha class

This class represents the ID of the OTP and an URL that can be used to download the captcha.

Syntax

```
public class Captcha
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCaptchaUrl() method

Syntax

```
String getCaptchaUrl ()
```

getOtpId() method

Syntax

```
long getOtpId ()
```

setCaptchaUrl(String) method**Syntax**

```
void setCaptchaUrl ( String captchaUrl )
```

setOtpId(long) method**Syntax**

```
void setOtpId ( long otpId )
```

IAddressLogic interface**Syntax**

```
public interface IAddressLogic
```

Usage

This service provides common helper methods used when managing addresses in the mobiliser system, like crating, updating, deleting and retrieving addresses (not necessarily bound to a customer)

An implementation of this class is exported via the OSGi service registry.

Use this to import the service into your code:

```
<osgi:reference id="addressLogicBean"
interface="com.sybase365.mobiliser.money.businesslogic.custo
mer.IAddressLogic" availability="optional" sticky="false" />
```

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

createAddress(final Address, long) method**Syntax**

```
long createAddress ( final Address address , long callerId )
```

deleteAddress(final long, long) method**Syntax**

```
void deleteAddress ( final long addressId , long callerId )
```

findNextAddresses(BigDecimal, BigDecimal, Long, Integer, Integer, Integer) method

Syntax

```
List< DAddress > findNextAddresses ( BigDecimal latitude ,  
BigDecimal longitude , Long customerId , Integer locationType , Integer  
radius , Integer maxRecords )
```

getAddress(final long, long) method

Syntax

```
Address getAddress ( final long addressId , long callerId )
```

getAddressByCustomer(final long, int, long) method

Syntax

```
Address getAddressByCustomer ( final long customerId , int  
addressType , long callerId )
```

getAllAddressesByCustomer(final long, long) method

Syntax

```
List< Address > getAllAddressesByCustomer ( final long customerId ,  
long callerId )
```

updateAddress(final Address, long) method

Syntax

```
void updateAddress ( final Address address , long callerId )
```

IAttachmentLogic interface

Syntax

```
public interface IAttachmentLogic
```

Usage

Service to handle attachments linked to a specific customer or transaction. An attachment can be provided as additional identification when a customer registers (e.g. passport scan) or a form of consent form when doing transactions (e.g. a direct debit mandate or a cheque balancing a transaction).

An implementation of this class is exported via the OSGi service registry.

Use this to import the service into your code:

```
<osgi:reference id="attachmentLogicBean"
interface="com.sybase365.mobiliser.money.businesslogic.customer.IAttachmentLogic" availability="optional"
sticky="false" />
*
```

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

createAttachment(Attachment, long) method

Create a new attachment.

Syntax

long createAttachment (Attachment *attachment* , long *callerId*) throws EntityMandatoryException

Parameters

- **attachment** –
- **callerId** –

Exceptions

- **EntityMandatoryException** –

Usage

Either the customer or the transaction field have to be set

deleteAttachment(long, long) method

Delete an existing attachment.

Syntax

void deleteAttachment (long *attachmentId* , long *callerId*) throws EntityNotFoundException

Parameters

- **attachmentId** –
- **callerId** –

Exceptions

- **EntityNotFoundException** –

getAttachment(long) method

Retrieve a specifc attachment identified by its ID.

Syntax

```
Attachment getAttachment ( long attachmentId )
```

Parameters

- **attachmentId** –

getAttachmentsByCustomer(long, Integer) method

Retrieve attachments for a specific customer.

Syntax

```
List< Attachment > getAttachmentsByCustomer ( long customerId ,  
Integer attachmentType ) throws EntityNotFoundException
```

Parameters

- **customerId** –
- **attachmentType** – If null, all attachments for the given customer are returned

Exceptions

- **EntityNotFoundException** –

getAttachmentsByTransaction(long, Integer) method

Retrieve all attachments for a specific transaction.

Syntax

```
List< Attachment > getAttachmentsByTransaction ( long transactionId ,  
Integer attachmentType ) throws EntityNotFoundException
```

Parameters

- **transactionId** –
- **attachmentType** – If null, all attachments for the given transaction are returned

Exceptions

- **EntityNotFoundException** –

updateAttachment(Attachment, long) method

Update an existing transaction.

Syntax

```
void updateAttachment ( Attachment attachment , long callerId ) throws
EntityNotFoundException
```

Parameters

- **attachment** –
- **callerId** –

Exceptions

- **EntityNotFoundException** –

ICommissionConfigurationLogic interface**Syntax**

```
public interface ICommissionConfigurationLogic
```

Usage

Service to handle commission configurations. A commission is a special fee type which instead of crediting the configured beneficiary, debits the beneficiary and credits a customer (merchant) configured in such a commission configuration. This interface defines CRUD methods to handle a commission configuration's lifecycle.

An implementation of this class is exported via the OSGi service registry.

Use this to import the service into your code:

```
<osgi:reference id="commissionLogicBean"
interface="com.sybase365.mobiliser.money.businesslogic.custo
mer.ICommissionConfigurationLogic" availability="optional"
sticky="false" />
```

*

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

createCommissionConfiguration(final CommissionConfiguration, long) method

Create a new commission configuration.

Syntax

```
long createCommissionConfiguration ( final  
CommissionConfiguration cc ,  long callerId )
```

Parameters

- **cc** –
- **callerId** –

deleteCommissionConfiguration(final long, long) method

Delete an existing commission configuration.

Syntax

```
void deleteCommissionConfiguration ( final long ccId ,  long  
callerId )
```

Parameters

- **ccId** –
- **callerId** –

getCommissionConfiguration(final long, long) method

Retrieve a specific commission configuration identified by its ID.

Syntax

```
CommissionConfiguration getCommissionConfiguration ( final long  
ccId ,  long callerId )
```

Parameters

- **ccId** –
- **callerId** –

getCommissionConfigurationByCustomer(final long, long) method

retrieve a commission configuration for a specific customer

Syntax

```
CommissionConfiguration getCommissionConfigurationByCustomer  
( final long customerId ,  long callerId )
```


Parameters

- **customerId** –
- **callerId** –

updateCommissionConfiguration(final CommissionConfiguration, long) method

Update an existing commission configuration.

Syntax

```
void updateCommissionConfiguration ( final
CommissionConfiguration cc ,  long callerId )
```

Parameters

- **cc** –
- **callerId** –

ICustomerAttributeLogic interface**Syntax**

```
public interface ICustomerAttributeLogic
```

Usage

Service offering CRUD methods to manage customer attributes. Customer attributes can be used to store additional information not directly reflected on the customer's entity.

An implementation of this class is exported via the OSGi service registry.

Use this to import the service into your code:

```
<osgi:reference id="attributeLogicBean"
interface="com.sybase365.mobiliser.money.businesslogic.custo
mer.ICustomerAttributeLogic" availability="optional"
sticky="false" />
```

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

createAttribute(CustomerAttribute, long) method

Create an attribute linked to a customer.

Syntax

```
void createAttribute ( CustomerAttribute attribute ,  long caller ) throws
EntityNotUniqueException
```

Parameters

- **attribute** –
- **caller** –

Exceptions

- **EntityNotUniqueException** –

deleteAttribute(long, int, long) method

Delete an existing customer attribute.

Syntax

`void deleteAttribute ( long customer , int type , long caller )` throws
`EntityNotFoundException`

Parameters

- **customer** –
- **type** –
- **caller** –

Exceptions

- **EntityNotFoundException** –

getAttribute(long, int, long) method

Retrieve a specific attribute for a given customer and attribute type.

Syntax

`CustomerAttribute getAttribute ( long customer , int type , long caller )` throws `EntityNotFoundException`

Parameters

- **customer** –
- **type** –
- **caller** –

Exceptions

- **EntityNotFoundException** –

updateAttribute(CustomerAttribute, long) method

Update an existing customer attribute.

Syntax

`void updateAttribute ( CustomerAttribute attribute , long caller )` throws `EntityNotFoundException`

Parameters

- **attribute** –
- **caller** –

Exceptions

- **EntityNotFoundException** –

ICustomerLifecycleHandler interface

Syntax

`public interface ICustomerLifecycleHandler`

Usage

The `ICustomerLifecycleHandler` interface defines methods build around the csutomer's lifecycle.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

postDeactivation(Customer, long) method

This method is called right after the customer has been deactivated.

Syntax

`void postDeactivation ( Customer customer , long callerId )`

Parameters

- **customer** –

postRegistration(Customer, long) method

This method gets invoked once the customer has been created in hte DB.

Syntax

```
void postRegistration ( Customer customer , long callerId )
```

Parameters

- **customer** –

Usage

Other features of the customer (address, ...) might not yet be available. This method participates in the same transaction. Every Exception thrown will lead to the full operation getting rolled back.

ICustomerLogic interface

Syntax

```
public interface ICustomerLogic
```

Usage

Service to manage customers and their lifecycle in the mobiliser system. Collection of CRUD services along with typical search operations.

An implementation of this class is exported via the OSGi service registry.

Use this to import the service into your code:

```
<osgi:reference id="customerLogicBean"  
interface="com.sybase365.mobiliser.money.businesslogic.custo  
mer.ICustomerLogic" availability="optional" sticky="false" />
```

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

PersistentLogin class

Holder type for returning the persistent login along with its validity.

Syntax

```
public class PersistentLogin
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

PersistentLogin(CustomerPersistentLogin, int) constructor

Syntax

`PersistentLogin ( CustomerPersistentLogin login , int validity )`

Parameters

- **login** –
- **validity** –

login variable
the login

Syntax
`final CustomerPersistentLogin login`

validity variable
number of seconds the login is still valid.

Syntax
`int validity`

blacklistCustomer(final long, int, long) method
Blacklist an existing customer.

Syntax

`void blacklistCustomer ( final long customerId , int blr , long callerId )`
throws `EntityNotFoundException`, `CustomerStatusException`

Parameters

- **customerId** –
- **blr** –
- **callerId** –

Exceptions

- **EntityNotFoundException** –
- **CustomerStatusException** –

checkAndUpdateLogin(final String, final String, final long) method

This method checks if the supplied login is still valid (not closed and not expired), that the token matches and also update the last activity timestamp.

Syntax

PersistentLogin checkAndUpdateLogin (final String *series* , final String *presentedToken* , final long *callerId*) throws
EntityNotFoundException, PersistentLoginException

Parameters

- **series** – the series identifier
- **presentedToken** – the token supplied by the user
- **callerId** – the callerid

Returns

the login object

Exceptions

- **EntityNotFoundException** –
- **PersistentLoginException** –

Usage

the login object

checkAndUpdateSession(final String, final long) method

This method checks if the supplied session is still valid (not closed and not expired) and also update the last activity timestamp.

Syntax

long checkAndUpdateSession (final String *sessionId* , final long *callerId*) throws EntityNotFoundException, SessionException

Parameters

- **sessionId** –
- **callerId** –

Returns

the customer id associated with the session

Exceptions

- **EntityNotFoundException** –
- **SessionException** –

Usage

the customer id associated with the session

createCustomer(final Customer, long) method

Create a new customer.

Syntax

```
long createCustomer ( final Customer customer , long callerId )
```

Parameters

- **customer** –
- **callerId** –

createPersistentLogin(final long, final long) method

Creates a new persistent login for a customer, based on the attributes defined in the session policy.

Syntax

```
PersistentLogin createPersistentLogin ( final long customerId ,  
final long callerId ) throws SessionException
```

Parameters

- **customerId** –
- **identification** –
- **callerId** –

Returns

the newly created login

Exceptions

- **SessionException** –

Usage

the newly created login

createPersistentLogin(final long, final boolean, final long) method

Creates a new persistent login for a customer, based on the attributes defined in the session policy.

Syntax

```
PersistentLogin createPersistentLogin ( final long customerId ,  
final boolean bindToResponse , final long callerId ) throws  
SessionException
```

Parameters

- **customerId** –
- **bindToResponse** – whether to bind this persistent login to the http response -only works if we are currently processing an http request
- **callerId** –

Returns

the newly created login

Exceptions

- **SessionException** –

Usage

the newly created login

createPersistentLogin(final

com.sybase365.mobiliser.money.persistence.model.customer.Customer, final long)
method

Creates a new persistent login for a customer, based on the attributes defined in the session policy.

Syntax

```
PersistentLogin createPersistentLogin ( final  
com.sybase365.mobiliser.money.persistence.model.customer.Cus  
tomer customer , final long callerId ) throws SessionException
```

Parameters

- **customer** –
- **callerId** –

Returns

the newly created login

Exceptions

- **SessionException** –

Usage

the newly created login

*createPersistentLogin(final
com.sybase365.mobiliser.money.persistence.model.customer.Customer, final
boolean, final long) method*

Creates a new persistent login for a customer, based on the attributes defined in the session policy.

Syntax

```
PersistentLogin createPersistentLogin ( final  
com.sybase365.mobiliser.money.persistence.model.customer.Cus  
tomer customer , final boolean bindToResponse , final long callerId )  
throws SessionException
```

Parameters

- **customer** –
- **bindToResponse** – whether to bind this persistent login to the http response -only works if we are currently processing an http request
- **callerId** –

Returns

the newly created login

Exceptions

- **SessionException** –

Usage

the newly created login

*createSession(final
com.sybase365.mobiliser.money.persistence.model.customer.Customer, final
String, final String, final long) method*

Creates a new session for a customer, based on the attributes defined in the session policy.

Syntax

```
CustomerSession createSession ( final  
com.sybase365.mobiliser.money.persistence.model.customer.Cus  
tomer customer, final String origin, final String identification, final  
long callerId) throws SessionException
```

Parameters

- **customer** –
- **origin** –
- **identification** –
- **callerId** –

Returns

the newly created session

Exceptions

- **SessionException** –

Usage

the newly created session

createWrkCustomer(WrkCustomer, long) method

Creates a new dual approval customer registration configuration.

Syntax

```
void createWrkCustomer ( WrkCustomer wrkCustomer, long callerId) throws  
EntityNotUniqueException
```

Parameters

- **wrkCustomer** –

deactivateCustomer(final long, int, long) method

Deactivate an existing customer.

Syntax

```
void deactivateCustomer ( final long customerId , int  
cancellationReason , long callerId ) throws EntityNotFoundException,  
CustomerStatusException
```

Parameters

- **customerId** –
- **cancellationReason** –
- **callerId** –

Exceptions

- **EntityNotFoundException** –
- **CustomerStatusException** –

deactivateCustomer(final long, int, boolean, long) method

Deactivate an existing customer with the option to obfuscate his data.

Syntax

```
void deactivateCustomer ( final long customerId , int  
cancellationReason , boolean obfuscationNeeded , long callerId ) throws  
EntityNotFoundException, CustomerStatusException
```

Parameters

- **customerId** –
- **cancellationReason** –
- **obfuscationNeeded** –
- **callerId** –

Exceptions

- **EntityNotFoundException** –
- **CustomerStatusException** –

deleteLogin(final CustomerPersistentLogin) method

Delete the given login from the backing store.

Syntax

```
void deleteLogin ( final CustomerPersistentLogin login )
```

Parameters

- **login** – the login to delete

deleteLoginsForUser(final String) method

Delete all logins from the backing store for the given user.

Syntax

```
void deleteLoginsForUser ( final String identification )
```

Parameters

- **identification** – the identification

deleteLoginsOlderThan(final Date) method

Delete all logins from the backing store older than the given date.

Syntax

```
int deleteLoginsOlderThan ( final Date cutoff )
```

Parameters

- **cutoff** – the cutoff date

Returns

the number of logins deleted

Usage

the number of logins deleted

deleteWrkCustomer(int, long) method

Deletes an existing dual approval customer registration configuration.

Syntax

```
void deleteWrkCustomer ( int customerTypeId , long callerId ) throws  
EntityNotFoundException
```

Parameters

- **request** –

destroyPersistentLogin() method

Destroys the persistent login (usually cookie) that is bound to the current HTTP request.

Syntax

```
void destroyPersistentLogin ()
```

Usage

Should / can be called upon successful logout from the business logic.

findCustomer(Address, Identification, Long, String, Integer, int) method**Syntax**

```
List< Customer > findCustomer ( Address address , Identification
identification , Long blacklistReason , String displayName , Integer
customerType , int maxRecords )
```

findCustomer(Address, Identification, boolean, Long, String, Integer, int, Long) method**Syntax**

```
List< Customer > findCustomer ( Address address , Identification
identification , boolean anyIdOrgUnit , Long blacklistReason , String
displayName , Integer customerType , int maxRecords , Long callerId )
```

findCustomer(Address, Identification, boolean, Long, String, Integer, String, int, Long) method

Find customers with a flexible filter on address, identification, black list reason, display name, customer type and org unit.

Syntax

```
List< Customer > findCustomer ( Address address , Identification
identification , boolean anyIdOrgUnit , Long blacklistReason , String
displayName , Integer customerType , String orgUnit , int maxRecords ,
Long callerId )
```

Parameters

- **address** –

- **identification** –
- **anyIdOrgUnit** –
- **blacklistReason** –
- **displayName** –
- **customerType** –
- **orgUnit** –
- **maxRecords** –
- **callerId** –

Usage

All of them are optional

findCustomerIncludeIdentifications(Address, Identification, boolean, Long, String, Integer, String, int, Long) method

Find customers with a flexible filter on address, identification, black list reason, display name, customer type and org unit.

Syntax

```
Map< Customer, List< Identification > >  
findCustomerIncludeIdentifications ( Address address ,  
Identification identification , boolean anyIdOrgUnit , Long blacklistReason ,  
String displayName , Integer customerType , String orgUnit , int  
maxRecords , Long callerId )
```

Usage

All of them are optional Also this service not just returns the customer objects, but the customer's identifications, too

findDeactivatedCustomer(int, Integer, long) method

Retrieve a list of already inactive customers matching the given criteria.

Syntax

```
List< Customer > findDeactivatedCustomer ( int customerType ,  
Integer blacklistReason , long callerId )
```

Parameters

- **customerType** –
- **blacklistReason** –
- **callerId** –

findHierarchicalCustomer(long, Address, Identification, Long, String, Integer, int, Integer, Integer) method

Syntax

```
Map< Customer, Integer > findHierarchicalCustomer ( long
callingAgentId , Address address , Identification identification , Long
blacklistReason , String displayName , Integer customerType , int
maxRecords , Integer minHierarchyLevel , Integer maxHierarchyLevel )
```

findHierarchicalCustomer(long, Address, Identification, boolean, Long, String, Integer, int, Integer, Integer, Long) method

Syntax

```
Map< Customer, Integer > findHierarchicalCustomer ( long
callingAgentId , Address address , Identification identification , boolean
anyIdOrgUnit , Long blacklistReason , String displayName , Integer
customerType , int maxRecords , Integer minHierarchyLevel , Integer
maxHierarchyLevel , Long callerId )
```

findHierarchicalCustomer(long, Address, Identification, boolean, Long, String, Integer, String, int, Integer, Integer, Long) method

Retrieve a hierarchy of customers matching the given criteria.

Syntax

```
Map< Customer, Integer > findHierarchicalCustomer ( long
callingAgentId , Address address , Identification identification , boolean
anyIdOrgUnit , Long blacklistReason , String displayName , Integer
customerType , String orgUnit , int maxRecords , Integer minHierarchyLevel ,
Integer maxHierarchyLevel , Long callerId )
```

Parameters

- **callingAgentId** – The agent performing the search. Is used as the starting point in the hierarchy.
- **address** –
- **identification** –
- **anyIdOrgUnit** –
- **blacklistReason** –
- **displayName** –
- **customerType** –
- **orgUnit** –

- **maxRecords** –
- **minHierarchyLevel** –
- **maxHierarchyLevel** –
- **callerId** –

findHierarchicalCustomerIncludeIdentifications(long, Address, Identification, boolean, Long, String, Integer, String, int, Integer, Integer, Long) method

Hierarchical customer search returning the customer, a list of his identifications and his hierarchy level.

Syntax

```
List< ExtendedCustomerSearchResult >  
findHierarchicalCustomerIncludeIdentifications ( long  
callingAgentId , Address address , Identification identification , boolean  
anyIdOrgUnit , Long blacklistReason , String displayName , Integer  
customerType , String orgUnit , int maxRecords , Integer minHierarchyLevel ,  
Integer maxHierarchyLevel , Long callerId )
```

Parameters

- **callingAgentId** – The agent performing the search. Is used as the starting point in the hierarchy.
- **address** –
- **identification** –
- **anyIdOrgUnit** –
- **blacklistReason** –
- **displayName** –
- **customerType** –
- **orgUnit** –
- **maxRecords** –
- **minHierarchyLevel** –
- **maxHierarchyLevel** –
- **callerId** –

getActiveCustomer(long) method

Retrieve an active customer with the given ID.

Syntax

Customer getActiveCustomer (long *customerId*) throws
EntityNotFoundException, CustomerStatusException

Parameters

- **customerId** –

Exceptions

- **EntityNotFoundException** –
- **CustomerStatusException** –

getCustomer(final long, long) method

Load a specific customer.

Syntax

Customer getCustomer (final long *customerId* , long *callerId*) throws
EntityNotFoundException, CustomerStatusException

Parameters

- **customerId** –
- **callerId** –

Exceptions

- **EntityNotFoundException** –
- **CustomerStatusException** –

getCustomerByIdentification(final String, final int, long) method

Syntax

Customer getCustomerByIdentification (final String *identification* ,
final int *identificationType* , long *callerId*) throws
CustomerStatusException

getCustomerByIdentification(final String, final int, long, String) method

Retrieve a customer based on one of his identifications.

Syntax

Customer getCustomerByIdentification (final String *identification* ,
final int *identificationType* , long *callerId* , String *orgUnit*) throws
CustomerStatusException

Parameters

- **identification** –

- **identificationType** –
- **callerId** –
- **orgUnit** –

Exceptions

- **CustomerStatusException** –

getCustomerHistory(long, Date, Date, Long) method

Retrieve customer history entries.

Syntax

```
List< HistoryEntry > getCustomerHistory ( long customerId , Date  
fromDate , Date toDate , Long creatorId ) throws  
EntityNotFoundException
```

Parameters

- **customerId** –
- **fromDate** –
- **toDate** –
- **creatorId** –

Exceptions

- **EntityNotFoundException** –

getLogin(String) method

Find the login with the given series id or null if nothing found.

Syntax

```
CustomerPersistentLogin getLogin ( String seriesId )
```

Parameters

- **seriesId** – the series id

Returns

the login or null

Usage

the login or null

getSession(final String) method

Retrieve the session from persistent storage.

Syntax

`CustomerSession getSession ( final String sessionId )` throws
`EntityNotFoundException`

Parameters

- **sessionId** – the session id

Returns

the session

Exceptions

- **EntityNotFoundException** –

Usage

the session

getWrkCustomer(int) method

Fetches a dual approval configuration for a specific customer tpy.

Syntax

`WrkCustomer getWrkCustomer ( int customerType )`

Parameters

- **customerType** –

Returns

the dual approval configuration or null ,if no configuration exists

Usage

the dual approval configuration or null ,if no configuration exists

getWrkCustomerByCustomer(long) method

Fetches a dual approval configuration for a specific customer.

Syntax

`WrkCustomer getWrkCustomerByCustomer ( long customerId )`

Parameters

- **customerId** –

Returns

the dual approval configuration or null ,if no configuration exists

Usage

the dual approval configuration or null ,if no configuration exists

getWrkCustomers() method

Fetches all dual approval customer registration configurations.

Syntax

```
List< WrkCustomer > getWrkCustomers ()
```

obfuscateCustomer(final long, long) method

Obfuscate customer data.

Syntax

```
void obfuscateCustomer ( final long customerId , long callerId ) throws  
CustomerObfuscationException
```

Parameters

- **customerId** –
- **callerId** –

Exceptions

- **CustomerObfuscationException** –

updateCustomer(final Customer, long) method

Update an existing customer.

Syntax

```
void updateCustomer ( final Customer customer , long callerId ) throws  
EntityNotFoundException, CustomerStatusException
```

Parameters

- **customer** –
- **callerId** –

Exceptions

- **EntityNotFoundException** –
- **CustomerStatusException** –

updateWrkCustomer(WrkCustomer, long) method

Updates an existing dual approval customer registration configuration.

Syntax

`void updateWrkCustomer (WrkCustomer wrkCustomer, long callerId)` throws `EntityNotFoundException`

Parameters

- **request** –

ICustomerNetworksLogic interface**Syntax**

```
public interface ICustomerNetworksLogic
```

Usage

Service offering CRUD methods to manage the relationship of a network for a customer. `CustomerNetwork` entries connect different customers in the mobiliser system to a network of related persons. An example would be to use these methods to establish a "friends and family" network.

An implementation of this class is exported via the OSGi service registry.

Use this to import the service into your code:

```
<osgi:reference id="networkLogicBean"
interface="com.sybase365.mobiliser.money.businesslogic.custo
mer.ICustomerNetworksLogic" availability="optional"
sticky="false" />
```

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

createCustomerNetwork(CustomerNetwork, long) method

Create a new `CustomerNetwork` entry.

Syntax

`long createCustomerNetwork (CustomerNetwork customerNetwork, long callerId)` throws `IllegalDataException`

Parameters

- **customerNetwork** –
- **callerId** –

Exceptions

- **IllegalDataException** –

deleteCustomerNetwork(long) method

Delete and existing CustomerNetwork

Syntax

`void deleteCustomerNetwork (long customerNetWorkId) throws
EntityNotFoundException`

Parameters

- **customerNetWorkId** –

Exceptions

- **EntityNotFoundException** –

getCustomerNetworks(long, boolean, Integer, Integer) method

Retrieve a list of CustomerNetwork entries representing the complete or partial network of the given customer.

Syntax

`List< CustomerNetwork > getCustomerNetworks (long customerId ,
boolean includeInactive , Integer status , Integer type)`

Parameters

- **customerId** –
- **includeInactive** –
- **status** –
- **type** –

updateCustomerNetwork(CustomerNetwork, long) method

Update an existing CustomerNetwork entry.

Syntax

`void updateCustomerNetwork (CustomerNetwork customerNetwork , long
callerId) throws IllegalDataException`

Parameters

- **customerNetwork** –
- **callerId** –

Exceptions

- **IllegalDataException** –

ICustomerOtpLogic interface***Syntax***

```
public interface ICustomerOtpLogic
```

Usage

Service to create and verify One-Time-Passwords for customers. OTPs can be used for registration to check the entered MSISDN of a customer and are used for example as authentication in a financial transaction. OTPs can also be used for other purposes. New OTP types must first be configured in the database.

An implementation of this class is exported via the OSGi service registry.

Use this to import the service into your code:

```
<osgi:reference id="customerOtpLogicBean"
interface="com.sybase365.mobiliser.money.businesslogic.customer.ICustomerOtpLogic" availability="optional"
sticky="false" />
```

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

createAndSendSmsToken(String, Locale) method

Sends an sms token to the given MSISDN.

Syntax

```
void createAndSendSmsToken ( String msisdn , Locale locale ) throws
OtpValidationException
```

Parameters

- **msisdn** – The MSISDN the token is sent to
- **locale** – The locale for the message template

Exceptions

- **OtpValidationException** –

Usage

In the reference implementation the token is loaded from the preferences and sent to the MSISDN.

createCustomerOtp(long, String, int, long, Map< String, String >, long) method
Creates a new CustomerOtp in the database.

Syntax

`CustomerOtp createCustomerOtp ( long customerId , String otp , int otpType , long referenceId , Map< String, String > msgParams , long callerId ) throws OtpValidationException`

Parameters

- **customerId** – for whom to create the OTP
- **otp** – if this is provided it will be used instead of an automatic generated one (optional). It needs to be numeric in case the OTP type is configured to be numeric, otherwise an `OtpValidationException` is thrown.
- **otpType** – the type of OTP to generate/send, defines length, message template, and send mode (email/SMS) (optional)
- **referenceId** – is stored with the CustomerOtp and can be retrieved when validating the OTP. This can be for example a transaction id in case the OTP is issued for a specific transaction
- **msgParams** – List of message parameters that is appended to the message that is sent out to the customer. Those can be referenced in the message template (which is configured for the OTP type)
- **callerId** – represents the ID of the caller of the operation. Used for tracking/audit purposes.

Returns

the newly created CustomerOtp

Exceptions

- **OtpValidationException** –

Usage

The OTP itself is either automatically generated according to the rules defined with the OTP type or the provided otp is used. The OTP is sent out in an email or SMS to the customer. A list

of parameters can be provided to further customize the message. The key for the otp parameter is "otp". A SecureRandom is used to generate the OTP.

the newly created CustomerOtp

createCustomerOtp(long, int, long, long) method

Variation of createCustomerOtp that does not contain the otp and the list of message parameters in the list of parameters.

Syntax

```
CustomerOtp createCustomerOtp ( long customerId , int otpType , long referenceId , long callerId ) throws OtpValidationException, EntityNotFoundException
```

Parameters

- **customerId** – for whom to create the OTP
- **otpType** – the type of OTP to generate/send, defines length, message template, and send mode (email/SMS) (optional)
- **referenceId** – is stored with the CustomerOtp and can be retrieved when validating the OTP. This can be for example a transaction id in case the OTP is issued for a specific transaction
- **callerId** – represents the ID of the caller of the operation. Used for tracking/audit purposes.

Returns

the newly created CustomerOtp

Exceptions

- **OtpValidationException** –
- **EntityNotFoundException** –

Usage

the newly created CustomerOtp

createCustomerOtp(long, String, int, long, long) method

Variation of createCustomerOtp that does not contain the list of message parameters in the list of parameters.

Syntax

```
CustomerOtp createCustomerOtp ( long customerId , String otp , int otpType , long referenceId , long callerId ) throws OtpValidationException
```

Parameters

- **customerId** – for whom to create the OTP
- **otp** – if this is provided it will be used instead of an automatic generated one (optional). It needs to be numeric in case the OTP type is configured to be numeric, otherwise an `OtpValidationException` is thrown.
- **otpType** – the type of OTP to generate/send, defines length, message template, and send mode (email/SMS) (optional)
- **referenceId** – is stored with the CustomerOtp and can be retrieved when validating the OTP. This can be for example a transaction id in case the OTP is issued for a specific transaction
- **callerId** – represents the ID of the caller of the operation. Used for tracking/audit purposes.

Returns

the newly created CustomerOtp

Exceptions

- **OtpValidationException** –

Usage

the newly created CustomerOtp

createCustomerOtp(long, int, long, Map< String, String >, long) method

Variation of `createCustomerOtp` that does not contain the `otp` in the list of parameters.

Syntax

```
CustomerOtp createCustomerOtp ( long customerId , int otpType , long referenceId , Map< String, String > msgParams , long callerId ) throws OtpValidationException, EntityNotFoundException
```

Parameters

- **customerId** – for whom to create the OTP
- **otpType** – the type of OTP to generate/send, defines length, message template, and send mode (email/SMS) (optional)
- **referenceId** – is stored with the CustomerOtp and can be retrieved when validating the OTP. This can be for example a transaction id in case the OTP is issued for a specific transaction

- **msgParams** – List of message parameters that is appended to the message that is sent out to the customer. Those can be referenced in the message template (which is configured for the OTP type)
- **callerId** – represents the ID of the caller of the operation. Used for tracking/audit purposes.

Returns

the newly created CustomerOtp

Exceptions

- **OtpValidationException** –
- **EntityNotFoundException** –

Usage

the newly created CustomerOtp

createNonPersitentOtp(int) method

Create a new OTP.

Syntax

`String createNonPersitentOtp (int otpType) throws
OtpValidationException`

Parameters

- **otpType** – the type of OTP that is to be generated.

Returns

the generated OTP.

Exceptions

- **OtpValidationException** –

Usage

The OTP is returned as a String. The OTP is not stored and not send out. A SecureRandom is used to generate the OTP.

the generated OTP.

resendCustomerOtp(long, int, long, long) method

Syntax

`void resendCustomerOtp ( long customerId , int otpType , long referenceId , long callerId )` throws `EntityNotFoundException`

Parameters

- **customerId** – The customer who initially received the OTP and to whom the OTP will be resent to.
- **otpType** – The OTP type
- **referenceId** – The reference ID of the OTP
- **callerId** – The caller ID

Exceptions

- **EntityNotFoundException** –
- **OtpValidationException** –

Usage

Resends an OTP to the customer it was previously delivered to.

An

`OtpValidationException`

is thrown if the OTP is already expired, if the configured resent count for the OTP type is exceeded or if no OPT was found for the given OTP type

An

`EntityNotFoundException`

is thrown in case the OTP was never sent out

verifyCustomerOtp(String, long, int, long, long) method

Verifies if the provided OTP matches the one stored in the DB.

Syntax

`boolean verifyCustomerOtp ( String otp , long customerId , int otpType , long referenceId , long callerId )` throws `OtpValidationException`

Parameters

- **otp** – the OTP to validate
- **customerId** – the customer Id for whom to validate hte OTP
- **otpType** – the type of OTP to validate
- **referenceId** – the reference Id as provided when the OTP was created
- **callerId** – the ID of the caller of the operation. Used for tracking/audit purposes.

Returns

`true` when the otp matched, `false` when the otp did not match.

Exceptions

- **OtpValidationException** – is thrown if no OTP was found in the DB, the OTP in the DB has expired, the threshold of failed verifications has been exceeded.

Usage

In case the OTP in the DB already expired, was not found, or the threshold of failed verifications has been exceeded an

`OtpValidationException`

is thrown.

`true` when the otp matched, `false` when the otp did not match.

verifySmsToken(String, String) method

Verifies an token in conjunction with an MSISDN, the token was previously sent to.

Syntax

`boolean verifySmsToken ( String msisdn , String token )` throws `OtpValidationException`

Parameters

- **msisdn** – The MSISDN the token was sent to and against which it will be verified
- **token** – The token to verify

Returns

`true` if the token was valid, `false` otherwise

Exceptions

- **OtpValidationException** –

Usage

true if the token was valid, false otherwise

IIentificationLogic interface

Syntax

```
public interface IIentificationLogic
```

Usage

Service providing common methods used to manage customer identifications, such as creating, updating, deleteing and retrieving identifications.

Identifications as an itegral entity of the mobiliser system allow to uniquely identify a customer and can be used as substitutes for the customer ID (beeing an indentification in its own) in service requests, such as a login or an authorisation.

An implementation of this class is exported via the OSGi service registry.

Use this to import the service into your code:

```
<osgi:reference id="identificationLogicBean"
interface="com.sybase365.mobiliser.money.businesslogic.custo
mer.IIentificationLogic" availability="optional"
sticky="false" />
```

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

createDefaultIdentifications(long, long) method

Syntax

```
void createDefaultIdentifications ( long customerId , long callerId )
```

createDefaultIdentifications(long, long, String) method

Syntax

```
void createDefaultIdentifications ( long customerId , long callerId ,
String orgUnit )
```

createIdentification(Identification, long) method**Syntax**

long createIdentification (Identification *identification* , long *callerId*) throws EntityNotUniqueException, EntityMandatoryException

deleteIdentification(long, long) method**Syntax**

void deleteIdentification (long *identificationId* , long *callerId*) throws EntityNotFoundException

getActiveIdentification(long, int, String) method**Syntax**

Identification getActiveIdentification (long *customerId* , int *identificationType* , String *orgUnit*) throws EntityNotFoundException

getActiveIdentification(long, int) method**Syntax**

Identification getActiveIdentification (long *customerId* , int *identificationType*) throws EntityNotFoundException

getIdentifications(long, Integer, boolean) method**Syntax**

List< Identification > getIdentifications (long *customerId* , Integer *identificationType* , boolean *includeInactive*)

getIdentifications(long, Integer, boolean, String) method**Syntax**

List< Identification > getIdentifications (long *customerId* , Integer *identificationType* , boolean *includeInactive* , String *orgUnit*)

updateIdentification(Identification, long) method**Syntax**

void updateIdentification (Identification *identification* , long *callerId*) throws EntityMandatoryException

IIIdentityLogic interface

Syntax

```
public interface IIIdentityLogic
```

Usage

Service to manage the identities of a customer. Information stored as an identity is usually stored for customer KYC reasons like passport ID and such.

An implementation of this class is exported via the OSGi service registry.

Use this to import the service into your code:

```
<osgi:reference id="identityLogicBean"
interface="com.sybase365.mobiliser.money.businesslogic.customer.IIIdentityLogic"
availability="optional" sticky="false" />
```

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

createIdentity(Identity, long) method

Creates a new identity.

Syntax

```
long createIdentity ( Identity identity , long callerId ) throws
EntityNotUniqueException, EntityMandatoryException
```

Parameters

- **identity** –
- **callerId** –

Exceptions

- **EntityNotUniqueException** –
- **EntityMandatoryException** –

deleteIdentity(long, long) method

Deletes an existing identity.

Syntax

```
void deleteIdentity ( long identityId , long callerId ) throws
EntityNotFoundException
```


Parameters

- **identityId** –
- **callerId** –

Exceptions

- **EntityNotFoundException** –

getActiveIdentity(long, int) method

Retrieves an identity for a specific customer and identity type.

Syntax

`Identity getActiveIdentity ( long customerId , int identityType )` throws
`EntityNotFoundException`

Parameters

- **customerId** –
- **identityType** –

Exceptions

- **EntityNotFoundException** –

getIdentities(long, Integer, boolean) method

Retrieves a list of identities for a specific customer matching the given criteria.

Syntax

`List< Identity > getIdentities (long customerId , Integer
identityType , boolean includeInactive)`

Parameters

- **customerId** –
- **identityType** –
- **includeInactive** –

updateIdentity(Identity, long) method

Updates an existing identity.

Syntax

`void updateIdentity ( Identity identity , long callerId )` throws
`EntityMandatoryException`

Parameters

- **identity** –
- **callerId** –

Exceptions

- **EntityMandatoryException** –

INotificationLogic interface

Syntax

```
public interface INotificationLogic
```

Usage

This service provides common helper methods used when sending notifications to a customer, like retrieving the preferred MSISDN or email address based on the customer configuration.

An implementation of this class is exported via the OSGi service registry.

Use this to import the service into your code:

```
<osgi:reference id="notificationLogic"
interface="com.sybase365.mobiliser.money.businesslogic.customer.INotificationLogic" sticky="false" />
```

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCustomersEmail(Customer, long) method

Retrieve a list of email addresses sorted from the most to least preferred.

Syntax

```
List< String > getCustomersEmail ( Customer customer , long actor )
```

Parameters

- **customer** – for whom to retrieve the email addresses
- **actor** – who has invoked the service (for tracking purposes)

Returns

a list of email addresses sorted from most to least preferred.

Usage

The call returns an empty list if no email address is available. The default implementation only returns one email address which is taken from the Customer's main address

a list of email addresses sorted from most to least preferred.

getCustomersMsisdn(Customer, long) method

Retrieve a list of MSISDNs sorted from the most to least preferred.

Syntax

```
List< String > getCustomersMsisdn ( Customer customer , long actor )
```

Parameters

- **customer** – for whom to retrieve the MSISDNs
- **actor** – who has invoked the service (for tracking purposes)

Returns

a list of MSISDN sorted from most to least preferred.

Usage

The call returns an empty list if no MSISDN is available. The default implementation will return a maximum of one MSISDN. It only returns MSISDNs that are registered as Identifications.

a list of MSISDN sorted from most to least preferred.

getCustomersNotificationEmail(Customer, long) method

Same as getCustomersEmail but will also require that the Notification mode of the customer indicates that he/she wants to receive email notifications.

Syntax

```
List< String > getCustomersNotificationEmail ( Customer customer ,  
long actor )
```

Parameters

- **customer** – for whom to retrieve the email addresses
- **actor** – who has invoked the service (for tracking purposes)

Returns

a list of email addresses sorted from most to least preferred.

Usage

If this is not the case an empty list is returned.

a list of email addresses sorted from most to least preferred.

getCustomersNotificationMsisdn(Customer, long) method

Same as getCustomersMsisdn but will also require that the Notification mode of the customer indicates that he/she wants to receive SMS notifications.

Syntax

```
List< String > getCustomersNotificationMsisdn ( Customer  
customer , long actor )
```

Parameters

- **customer** – for whom to retrieve the MSISDNs
- **actor** – who has invoked the service (for tracking purposes)

Returns

a list of MSISDN sorted from most to least preferred.

Usage

If this is not the case an empty list is returned.

a list of MSISDN sorted from most to least preferred.

getLocale(Customer, long) method

returns the Locale of the provided customer.

Syntax

```
Locale getLocale ( Customer customer , long actor )
```

Parameters

- **customer** –
- **actor** –

getReceiver(Customer, String, long) method

Returns the Receiver object for the provided customer.

Syntax

```
Receiver getReceiver ( Customer customer , String notificationType ,  
long actor ) throws CustomerDataMissingException,  
UnsupportedNotificationTypeException
```

Parameters

- **customer** – the customer to whom to send the message
- **notificationType** – the notification type (email, sms)
- **actor** – who has invoked the service (for tracking purposes)

Returns

the Receiver object

Exceptions

- **CustomerDataMissingException** – is thrown if the customer has no data for the requested notification type
- **UnsupportedNotificationTypeException** – is thrown if the notification type is not supported (at the moment only email and sms are supported)

Usage

the Receiver object

setReceiverAndLocale(Message, Customer, long) method

Sets the correct message receiver base on the notification type (must already be set) and sets the correct locale for the provided user (if the Message is of the sub-class TemplateMessage)

Syntax

```
void setReceiverAndLocale (Message message, Customer receiver, long actor)
```

Parameters

- **message** – the message in which the field will be populated.
- **customer** – who is the receiver of the message
- **actor** – who has invoked the service (for tracking purposes)

IOtpLogic interface

Syntax

```
public interface IOtpLogic
```

Usage

Service to create and verify One-Time-Passwords. OTPs can be used for registration to check the entered MSISDN of a customer and are used for example as authentication in a financial transaction. OTPs can also be used for other purposes. New OTP types must first be configured in the database.

An implementation of this class is exported via the OSGi service registry.

Use this to import the service into your code:

```
<osgi:reference id="otpLogicBean"
interface="com.sybase365.mobiliser.money.businesslogic.customer.IOtpLogic" availability="optional" sticky="false" />
```

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

createCaptchaOtp(int, String, Long, long) method

Similar to createOtp but instead of sending the OTP to the receiver, a URL to a captcha image containing the code is generated.

Syntax

Captcha createCaptchaOtp (int *otpType* , String *referenceId* , Long *customerId* , long *callerId*) throws OtpValidationException

Parameters

- **otpType** – the type of OTP to generate/send, defines length, message template, and send mode (email/SMS) (optional)
- **referenceId** – is stored with the Otp and is returned when validating the OTP. This can be for example a transaction id in case the OTP is issued for a specific transaction.
- **customerId** – This is completely optional and is only used to attach the outgoing message to the customer for viewing in the message history. In general those kind of OTPs are not linked to customers.
- **callerId** – represents the ID of the caller of the operation. Used for tracking/audit purposes.

Exceptions

- **OtpValidationException** –

createOtp(int, String, Locale, String, Map< String, String >, Long, long) method

Creates a new CustomerOtp in the database.

Syntax

long createOtp (int *otpType* , String *receiver* , Locale *locale* , String *referenceId* , Map< String, String > *msgParams* , Long *customerId* , long *callerId*) throws OtpValidationException

Parameters

- **otpType** – the type of OTP to generate/send, defines length, message template, and send mode (email/SMS) (optional)
- **receiver** – the receiver information. Probably an email address or mobile phone number. The Otp-types send mode has to correspond with the type of receiver provided here. Otherwise the service sends an SMS to an email address, which will not work...
- **referenceId** – is stored with the Otp and is returned when validating the OTP. This can be for example a transaction id in case the OTP is issued for a specific transaction.
- **msgParams** – List of message parameters that is appended to the message that is sent out to the customer. Those can be referenced in the message template (which is configured for the OTP type)
- **customerId** – This is completely optional and is only used to attach the outgoing message to the customer for viewing in the message history. In general those kind of OTPs are not linked to customers.
- **callerId** – represents the ID of the caller of the operation. Used for tracking/audit purposes.

Returns

the ID of the newly created Otp

Exceptions

- **OtpValidationException** –

Usage

The OTP itself is either automatically generated according to the rules defined with the OTP type or the provided otp is used. The OTP is sent out in an email or SMS to the customer. A list of parameters can be provided to further customize the message. The key for the otp parameter is "otp". A SecureRandom is used to generate the OTP.

the ID of the newly created Otp

resendOtp(long, long) method

Sends out an existing OTP again.

Syntax

```
void resendOtp ( long otpId ,   long callerId ) throws
EntityNotFoundException, OtpValidationException
```

Parameters

- **otpId** –

- **callerId** –

Exceptions

- **EntityNotFoundException** –
- **OtpValidationException** –

verifyOtp(String, long, long) method

Syntax

Otp verifyOtp (String *otp* , long *otpId* , long *callerId*) throws
OtpValidationException

Parameters

- **otp** – the value that is compared against the value that is stored in the DB and was sent out
- **otpId** – the internal ID of the OTP that was returned when the OTP was created
- **callerId** – the caller ID of the service caller

Returns

upon success the Otp with the receiver and the refernceId that were stored when the OTP was created are returned. Other fields of the Otp are not guaranteed to be set.

Exceptions

- **OtpValidationException** – is thrown in case the validation failed

Usage

upon success the Otp with the receiver and the refernceId that were stored when the OTP was created are returned. Other fields of the Otp are not guaranteed to be set.

IRemittanceVoucherLogic interface

Syntax

```
public interface IRemittanceVoucherLogic
```

Usage

Service offering a find method to search for remittance vouchers.

An implementation of this class is exported via the OSGi service registry.

Use this to import the service into your code:

```
<osgi:reference id="remittanceVoucherLogicBean"  
interface="com.sybase365.mobiliser.money.businesslogic.custo
```



```
mer.IRemittanceVoucherLogic" availability="optional"
sticky="false" />
```

*

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

findRemittanceVoucher(long, String) method

Syntax

```
FindInternalVoucherResultBean findRemittanceVoucher ( long
customerId , String pickupCode )
```

fee package

Members

All public members of the fee package.

- **IFeeAccess interface** – The IFeeAccess interface provides methods to simplify the access to the applicable Fee Scale and Fee Step for a given Customer and FeeType.
- **IFeeCalculation interface** – Business logic pertaining to fees.
- **IFeeTypeManagementLogic interface** – CRUD type methods to work with Fee Types and the directly related entities.

IFeeAccess interface

The IFeeAccess interface provides methods to simplify the access to the applicable Fee Scale and Fee Step for a given Customer and FeeType.

Syntax

```
public interface IFeeAccess
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getFeeScale(final Customer, final FeeType) method

Determines the Scale to use for the provided Customer and FeeType.

Syntax

```
Scale getFeeScale ( final Customer cus , final FeeType feeType )
```

Parameters

- **cus** – The Customer for whom to retrieve the Scale.
- **feeType** – The FeeType for which to retrieve the Scale.

Returns

The Scale that matches best the input parameters or null if no Scale was found.

Usage

The Scale is defined in a FeeSet which is attached to the Customer, the type of the Customer, or the OrgUnit of the Customer.

The Scale that matches best the input parameters or null if no Scale was found.

getFeeStep(final Scale, final long, final Currency, final Date) method

Returns the Fee Step that best matches the input parameters.

Syntax

```
Step getFeeStep ( final Scale scale , final long amount , final  
Currency currency , final Date transactionDate )
```

Parameters

- **scale** – The Scale for which to retrieve the Step
- **amount** – The amount used to select the right step based on the threshold
- **currency** – The currency matching the amount
- **transactionDate** – The date the transaction occurred - in most cases the current date.

Returns

The Fee Step that matches all the input parameters or null in case no step was found. This is mostly due to mis-configuration (e.g. no configuration for the currency or no step with threshold of 0) but should be handled anyway.

Usage

The Fee Step that matches all the input parameters or null in case no step was found. This is mostly due to mis-configuration (e.g. no configuration for the currency or no step with threshold of 0) but should be handled anyway.

IFeeCalculation interface

Business logic pertaining to fees.

Syntax

```
public interface IFeeCalculation
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

calculateFees(final SubTransaction, Authorisation) method

Syntax

```
List< Fee > calculateFees ( final SubTransaction txn ,  
Authorisation auth )
```

Parameters

- **txn** – the SubTransaction to calculate the fees for
- **auth** – the authorisation, which can be used to copy the fees from (if configured). This parameter is optional, the implementation must search the successful authorisation from the DB if not provided.

Returns

the calculated fees

Usage

Calculates the online fees that are to be considered for the specified transaction. The fees are stored into the database and available via

```
SubTransaction#getFees()
```

```
.
```

the calculated fees

copyFees(final SubTransaction) method

Syntax

```
List< Fee > copyFees ( final SubTransaction source )
```

Parameters

- **source** – the SubTransaction whose fees should be copied

Returns

the list of fee (they have not been inserted into the database)

Usage

Creates copies of the source transaction's fees.

the list of fee (they have not been inserted into the database)

getFees(final SubTransaction, final boolean) method

Syntax

```
long getFees ( final SubTransaction txn ,   final boolean payee )
```

Parameters

- **txn** – the SubTransaction to examine
- **payee** – indicates if the affected Customer is payee or payer

Returns

the fee sum

Usage

Calculates the fee amounts relevant for the specified customer and transaction.

the fee sum

getFeesForLimit(final SubTransaction, final boolean) method

Syntax

```
long getFeesForLimit ( final SubTransaction txn ,   final boolean  
payee )
```

Parameters

- **txn** – the SubTransaction to examine
- **payee** – indicates if the affected Customer is payee or payer

Returns

the fee sum

Usage

Calculates the fee amounts relevant for the specified customer and transaction limit.

the fee sum

IFeeTypeManagementLogic interface

CRUD type methods to work with Fee Types and the directly related entities.

Syntax

```
public interface IFeeTypeManagementLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

createFeeSet(FeeSet, long) method

Creates a new Fee Set.

Syntax

```
long createFeeSet ( FeeSet feeSet , long callerId ) throws  
EntityMandatoryException
```

Parameters

- **feeSet** –
- **callerId** –

Exceptions

- **EntityMandatoryException** –

createFeeType(FeeType, long) method

Creates a new Fee Type.

Syntax

```
int createFeeType ( FeeType feeType , long callerId ) throws  
EntityMandatoryException
```

Parameters

- **feeType** –
- **callerId** –

Exceptions

- **EntityMandatoryException** –

createIndividualFeeSet(long, long) method

Create a new Fee Set that is marked as being "individual" (= only for a single customer).

Syntax

long createIndividualFeeSet (long *customerId* , long *callerId*) throws
EntityNotFoundException

Parameters

- **customerId** –
- **callerId** –

Returns

the ID of the generated Fee Set

Exceptions

- **EntityNotFoundException** –

Usage

the ID of the generated Fee Set

createUseCaseFeeType(UseCaseFee, long) method

Creates a new Use Case to Fee Type association.

Syntax

long createUseCaseFeeType (UseCaseFee *useCaseFee* , long *callerId*)
throws EntityMandatoryException, IllegalDataException

Parameters

- **useCaseFee** –
- **callerId** –

Returns

The ID of the newly created UseCaseFee

Exceptions

- **EntityMandatoryException** –
- **IllegalDataException** –

Usage

The ID of the newly created UseCaseFee

deleteFeeSet(long, long) method

Deletes an existing Fee Set.

Syntax

```
void deleteFeeSet ( long feeSetId , long callerId ) throws  
EntityNotFoundException
```

Parameters

- **feeSetId** –
- **callerId** –

Exceptions

- **EntityNotFoundException** –

deleteFeeType(int, long) method

Deletes an existing Fee Type.

Syntax

```
void deleteFeeType ( int feeTypeId , long callerId ) throws  
EntityNotFoundException
```

Parameters

- **feeTypeId** –
- **callerId** –

Exceptions

- **EntityNotFoundException** –

deleteUseCaseFeeType(long, long) method

Deletes an existing Use Case to Fee Type association by its primary key.

Syntax

```
void deleteUseCaseFeeType ( long useCaseFeeTypeId , long callerId ) throws  
EntityNotFoundException
```

Parameters

- **useCaseFeeTypeId** –
- **callerId** –

Exceptions

- **EntityNotFoundException** –

getFeeSets(Boolean) method

Returns a list of available Fee Sets.

Syntax

```
List< FeeSet > getFeeSets ( Boolean fetchIndividual )
```

Parameters

- **fetchIndividual** – if this is set to true it returns only the individual Fee Sets. If it is set ot false it only returns the global Fee Sets. If null, all Fee Sets are returned.

Usage

Individual ones (that are only used for a single customer) can be filtered out.

getFeeTypes() method

To get all Fee Types.

Syntax

```
List< FeeType > getFeeTypes ( )
```

Returns

all Fee Types

Usage

all Fee Types

getFeeTypesByFeeSet(long) method

To get all Fee Types for a given Fee Set.

Syntax

```
List< FeeType > getFeeTypesByFeeSet ( long feeSetId )
```


Parameters

- **feeSetId** – The Fee Set ID for which to retrieve the Fee Type

getScaleStepsByFeeTypeAndFeeSet(long, Integer, String) method

Retrieves the Scale steps for the provided Fee Set.

Syntax

```
Map< FeeType, Map< String, List< Scale.Step > > >
getScaleStepsByFeeTypeAndFeeSet ( long feeSetId , Integer feeTypeId ,
String currency )
```

Parameters

- **feeSetId** – the mandatory Fee Set ID
- **feeTypeId** – optional Fee Type ID
- **currency** – Optional currency

Usage

Fee Type and Currency are optional in which case Scale Steps for all Fee Types and currencies are returned.

getUseCasesFeeTypes() method

Returns all existing Use Case Fee Types associations.

Syntax

```
List< UseCaseFee > getUseCasesFeeTypes ()
```

setScaleStepsByFeeTypeAndFeeSet(int, long, Map< String, List< Scale.Step >>, long) method

Create new Fee Scale Period and steps which are valid from the time of creation and replaces any previous steps for the given currency.

Syntax

```
void setScaleStepsByFeeTypeAndFeeSet ( int feeTypeId , long feeSetId ,
Map< String, List< Scale.Step >> currencyScaleSteps , long callerId )
throws EntityMandatoryException, EntityNotFoundException
```

Parameters

- **feeTypeId** –
- **feeSetId** –

- **currencyScaleSteps** –
- **callerId** –

Exceptions

- **EntityMandatoryException** –
- **EntityNotFoundException** –

updateFeeSet(FeeSet, long) method

Updates an existing Fee Set.

Syntax

void updateFeeSet (FeeSet *feeSet* , long *callerId*) throws
EntityMandatoryException

Parameters

- **feeSet** –
- **callerId** –

Exceptions

- **EntityMandatoryException** –

updateUseCaseFeeType(UseCaseFee, long) method

Updates an existing use Case to Fee Type association.

Syntax

void updateUseCaseFeeType (UseCaseFee *useCaseFee* , long *callerId*)
throws EntityMandatoryException

Parameters

- **useCaseFee** –
- **callerId** –

Exceptions

- **EntityMandatoryException** –

invoice package

Members

All public members of the invoice package.

- **IInvoiceLogic interface** – The class `IInvoiceLogic` contains basic CRUD type operation to deal with Invoices, Invoice Types, and Invoice Configurations.
- **InvoiceException class** – `InvoiceException`.

IInvoiceLogic interface

The class `IInvoiceLogic` contains basic CRUD type operation to deal with Invoices, Invoice Types, and Invoice Configurations.

Syntax

```
public interface IInvoiceLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

createInvoice(Invoice, long) method

Creates a new Invoice Configuration.

Syntax

```
long createInvoice ( Invoice ic , long callerId )
```

Parameters

- **ic** –
- **callerId** – The ID of the customer invoking the method

Returns

The ID (primary key) of the generated Invoice.

Usage

The ID (primary key) of the generated Invoice.

createInvoiceConfiguration(InvoiceConfiguration, long) method

Creates a new Invoice Configuration.

Syntax

```
long createInvoiceConfiguration ( InvoiceConfiguration ic , long callerId )
```

Parameters

- **ic** – All the data for the Invoice Configuration
- **callerId** – The ID of the customer invoking the method

createInvoiceForInvoiceType(long, long, Long, String, String, String, Long, Duration, Duration, Long, Map< Integer, String >, Duration, Date, Date, long) method

Creates a new Invoice for an Invoice Type.

Syntax

```
long createInvoiceForInvoiceType ( long customerId , long invoiceTypeId , Long amount , String currency , String reference , String alias , Long authorisationThreshold , Duration autoPayDelay , Duration billingCycle , Long defaultPi , Map< Integer, String > attributes , Duration maxDelay , Date startDate , Date endDate , long callerId )
```

Parameters

- **customerId** – refers to the customer for whom to create the Invoice
- **invoiceTypeId** –
- **amount** – is optional, if not provided the default amount from the Invoice Type is used
- **currency** – is optional, if not provided the default currency from the Invoice Type is used
- **callerId** – The ID of the customer invoking the method

Returns

the Invoice id that was created.

Usage

In case the required InvoiceConfiguration in between is missing, it is created automatically.
the Invoice id that was created.

createInvoiceType(InvoiceType, long) method

Creates a new Invoice Type.

Syntax

```
long createInvoiceType ( InvoiceType ic , long callerId )
```

Parameters

- **ic** –
- **callerId** – The ID of the customer invoking the method

Returns

The ID (primary key) of the generated Invoice Type.

Usage

The ID (primary key) of the generated Invoice Type.

deactivateInvoiceConfiguration(long, long) method

Deactivates an Invoice Configuration.

Syntax

```
void deactivateInvoiceConfiguration ( long invoiceConfigurationId ,  
long callerId )
```

Parameters

- **invoiceConfigurationId** –
- **callerId** – The ID of the customer invoking the method

Usage

In case there are still active Invoices for an InvoiceConfiguration and Exception with code 5101 is thrown.

deactivateInvoiceType(long, long) method

Deactivates an existing Invoice Type so that it will no longer be usable.

Syntax

```
void deactivateInvoiceType ( long invoiceTypeId , long callerId )
```

Parameters

- **invoiceTypeId** –
- **callerId** – The ID of the customer invoking the method

getAllInvoiceTypes(boolean, long) method

Returns a list of all the (active) Invoice Types in the system.

Syntax

```
List< InvoiceType > getAllInvoiceTypes ( boolean onlyActive , long  
callerId )
```

Parameters

- **onlyActive** – Is set to `true` only the active Invoice Types are returned. If `false` all are returned.
- **callerId** – The ID of the customer invoking the method

getInvoice(long, long) method

Returns an existing Invoice by its ID.

Syntax

Invoice getInvoice (long *invoiceId* , long *callerId*)

Parameters

- **invoiceId** –
- **callerId** – The ID of the customer invoking the method

Returns

The requested Invoice or null if it was not found.

Usage

The requested Invoice or null if it was not found.

getInvoiceConfiguration(long, long) method

Returns an Invoice Configuration by its ID.

Syntax

InvoiceConfiguration getInvoiceConfiguration (long *invoiceConfigurationId* , long *callerId*)

Parameters

- **invoiceConfigurationId** –
- **callerId** – The ID of the customer invoking the method

getInvoiceConfigurationsByCustomer(long, Long, boolean, long) method

Returns all the Invoice Configurations for a customer.

Syntax

```
List< InvoiceConfiguration >  
getInvoiceConfigurationsByCustomer ( long customerId , Long  
invoiceTypeId , boolean onlyActive , long callerId )
```

Parameters

- **customerId** – The ID of the customer
- **invoiceTypeId** – Optionally filters for the provided Invoice Type.
- **onlyActive** – Is set to `true` only the active Invoice Types are returned. If `false` all are returned.
- **callerId** – The ID of the customer invoking the method

Usage

Various filters can be applied.

getInvoicesByCustomer(long, Long, Integer, long) method

Returns a list of Invoices for a customer.

Syntax

```
List< Invoice > getInvoicesByCustomer ( long customerId , Long
invoiceConfigurationId , Integer invoiceStatus , long callerId )
```

Parameters

- **customerId** – The ID of the customer
- **invoiceConfigurationId** – Optionally filters for the provided Invoice Configuration.
- **invoiceStatus** – Optionally filters the list for the provided Invoice Status.
- **callerId** – The ID of the customer invoking the method

Usage

Optional filters can be applied.

getInvoiceType(long, long) method

Returns an existing Invoice Type by its ID.

Syntax

```
InvoiceType getInvoiceType ( long invoiceTypeId , long callerId )
```

Parameters

- **invoiceTypeId** –
- **callerId** – The ID of the customer invoking the method

getInvoiceTypesByGroup(int, boolean, long) method

Returns Invoice Types by group.

Syntax

```
List< InvoiceType >getInvoiceTypesByGroup ( int groupId , boolean  
onlyActive , long callerId )
```

Parameters

- **groupId** – The mandatory group id to filter for.
- **onlyActive** –
- **callerId** – The ID of the customer invoking the method

Usage

There are various pre-defined groups but the list can be extended (in the Database).

getInvoiceTypesByIssuer(long, boolean, long) method

Returns all the Invoices for a given issuer (=customer, biller).

Syntax

```
List< InvoiceType >getInvoiceTypesByIssuer ( long issuerId ,  
boolean onlyActive , long callerId )
```

Parameters

- **issuerId** – The mandatory ID of the issuer.
- **onlyActive** – Is set to `true` only the active Invoice Types are returned. If `false` all are returned.
- **callerId** – The ID of the customer invoking the method

updateInvoiceConfiguration(InvoiceConfiguration, long) method

Updates an existing Invoice Configuration.

Syntax

```
void updateInvoiceConfiguration ( InvoiceConfiguration ic , long  
callerId )
```

Parameters

- **ic** –
- **callerId** – The ID of the customer invoking the method

updateInvoiceType(InvoiceType, long) method

Updates an existing Invoice Type with the provided data.

Syntax

```
void updateInvoiceType ( InvoiceType ic , long callerId )
```

Parameters

- **ic** –
- **callerId** – The ID of the customer invoking the method

InvoiceException class

InvoiceException.

Syntax

```
public class InvoiceException
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

InvoiceException(Status) constructor**Syntax**

```
InvoiceException ( Status status )
```

InvoiceException(String, Status) constructor**Syntax**

```
InvoiceException ( String message , Status status )
```

InvoiceException(Throwable, Status) constructor**Syntax**

```
InvoiceException ( Throwable cause , Status status )
```

InvoiceException(String, Throwable, Status) constructor**Syntax**

```
InvoiceException ( String message , Throwable cause , Status status )
```

getErrorCode() method

Syntax

```
int getErrorCode ()
```

getStatus() method

Syntax

```
Status getStatus ()
```

Status() enumeration

Enum Constant Summary

- **ACTIVE_INVOICES** –

normalizer package

Members

All public members of the normalizer package.

- **IConcreteIdentificationNormalizer interface** – The `IConcreteIdentificationNormalizer` provides wrapping around the `IndentificationNormalizer`.
- **IIdentificationNormalizerProvider interface** – The `IIdentificationNormalizerProvider` provides the functionality to look up a normalizer which matches the identification type of the identification which is to be normalized.
- **IIdentificationValueNormaliser interface** – Classes implementing `IIdentificationValueNormaliser` interface provide the normalizer implementation.

IConcreteIdentificationNormalizer interface

The `IConcreteIdentificationNormalizer` provides wrapping around the `IndentificationNormalizer`.

Syntax

```
public interface IConcreteIdentificationNormalizer
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

normalize(Identification) method

Normalizes the identification using the normalizer which matches the `identificationType` of the identification.

Syntax

```
abstract Identification normalize ( Identification identification )
```

Parameters

- **identification** – The identification to normalize

Returns

The normalized Identification

Usage

The normalized Identification

normalize(List< Identification >) method

A wrapper which applies the `normalize` method to all elements of the list.

Syntax

```
abstract List< Identification > normalize ( List<  
Identification > identificationL )
```

Parameters

- **identificationL** –

Returns

The normalized Identifications

Usage

The normalized Identifications

normalize(String, Integer) method

Normalizes the identification by looking for and invoking the normalizer which applies to the provided type.

Syntax

```
abstract String normalize ( String ident , Integer type )
```

Parameters

- **ident** – The identification which is to be normalized
- **type** – The type of the identification

Returns

The normalized identification String

Usage

The normalized identification String

IIIdentificationNormalizerProvider interface

The IIIdentificationNormalizerProvider provides the functionality to look up a normalizer which matches the identification type of the identification which is to be normalized.

Syntax

```
public interface IIIdentificationNormalizerProvider
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getNormalizerForType(Integer) method

Gets the normalizer which is registered for the type provided.

Syntax

```
IIIdentificationValueNormaliser getNormalizerForType ( Integer  
type )
```

Parameters

- **The** – identification type

Returns

The resolved normalizer

Usage

The resolved normalizer

getNormalizers() method

Gets all registered normalizers.

Syntax

```
Collection< IIIdentificationValueNormaliser > getNormalizers ()
```

Returns

All registered normalizers

Usage

All registered normalizers

IIIdentificationValueNormaliser interface

Classes implementing IIIdentificationValueNormaliser interface provide the normalizer implementation.

Syntax

```
public interface IIIdentificationValueNormaliser
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCoverage() method

Will be called by the IdentificationNormalizerProvider when searching for a Normalizer matching the identification type which is to be normalized.

Syntax

```
Set< Integer > getCoverage ()
```

Returns

A set of IdentificationType ids

Usage

A set of IdentificationType ids

normalizeUserIdentification(final String) method

Does the normalizing of the identification by using the underlying ConcreteIndentificationNormalizer.

Syntax

```
String normalizeUserIdentification ( final String identification )
```

Parameters

- **identification** – The identification which should be normalized

Returns

The normalized identification

Usage

The normalized identification

oauth2 package

Members

All public members of the oauth2 package.

- **IOAuth2Logic interface** – Logic for dealing with OAuth2 providers programmatically.

IOAuth2Logic interface

Logic for dealing with OAuth2 providers programmatically.

Syntax

```
public interface IOAuth2Logic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

linkOAuth2Account(final String, final String, final long, long) method

Links an oauth 2 account to an existing mobiliser account.

Syntax

```
long linkOAuth2Account ( final String provider , final String token ,  
final long customerId , long callerId )
```

Parameters

- **provider** – the name of the oauth2 provider. such a provider must have already been registered in the system
- **token** – the token from the oauth2 provider
- **customerId** – the customer to which the account should be linked
- **callerId** – caller id

Returns

the id of the newly created identification

Usage

the id of the newly created identification

payment package

Members

All public members of the payment package.

- **api package** –
- **handlers package** –

api package

Members

All public members of the api package.

- **exception package** –
- **types package** –
- **IExchangeRateBroker interface** –
- **IExchangeRateHandler interface** –
- **IPaymentBroker interface** –
- **IPaymentHandler interface** –
- **IPaymentInstrumentClassHandler interface** –

- **IPaymentInstrumentLookup** interface –
- **IPaymentInstrumentTypeHandler** interface –

exception package

Members

All public members of the exception package.

- **MobiliserPaymentHandlerException** class –
- **PaymentHandlerException** class –

MobiliserPaymentHandlerException class

Syntax

```
public class MobiliserPaymentHandlerException
```

Usage

This exception wraps a PaymentHandlerException into a MobiliserServiceException so that it can be handled by the Mobiliser framework and translated into an error code.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

MobiliserPaymentHandlerException(final Exception) constructor

Syntax

```
MobiliserPaymentHandlerException ( final Exception pe )
```

Parameters

- **pe** – the wrapped exception (should be a payment handler exception)

MobiliserPaymentHandlerException(final PaymentHandlerException) constructor

Syntax

```
MobiliserPaymentHandlerException ( final  
PaymentHandlerException pe )
```

Parameters

- **pe** – the wrapped payment handler exception

*MobiliserPaymentHandlerException(final String, final PaymentHandlerException)
constructor*

Syntax

`MobiliserPaymentHandlerException (final String message , final
PaymentHandlerException pe)`

Parameters

- **message** – additional exception message
- **pe** – the wrapped payment handler exception

*MobiliserPaymentHandlerException(final String, final Throwable, final
PaymentHandlerException) constructor*

Syntax

`MobiliserPaymentHandlerException (final String message , final
Throwable cause , final PaymentHandlerException pe)`

Parameters

- **message** – additional exception message
- **cause** – the original exception
- **pe** – the wrapped payment handler exception

*MobiliserPaymentHandlerException(final Throwable, final
PaymentHandlerException) constructor*

Syntax

`MobiliserPaymentHandlerException (final Throwable cause , final
PaymentHandlerException pe)`

Parameters

- **cause** – the original exception
- **pe** – the wrapped payment handler exception

getErrorCode() method

Returns the error code defined in the payment handler exception.

Syntax

`int getErrorCode ()`

Returns

the error code defined in the payment handler exception.

Usage

the error code defined in the payment handler exception.

PaymentHandlerException class

Syntax

```
public class PaymentHandlerException
```

Usage

This exception is used by IPaymentHandlers to communicate failures in payment processing to the IPaymentBroker. The broker will then react accordingly in the payment processing flow and wrap this exception in a MobiliserPaymentHandlerException.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

PaymentHandlerException(final int) constructor

Syntax

```
PaymentHandlerException ( final int errorCode )
```

Parameters

- **errorCode** – the exception's error code

Usage

Creates a new instance of

```
AuthenticationException
```

.

PaymentHandlerException(final int, final Map< String, String >) constructor

Syntax

```
PaymentHandlerException ( final int errorCode , final Map< String,  
String > parameters )
```

Parameters

- **errorCode** – the exception's error code
- **parameters** – the exception parameters

Usage

Creates a new instance of

`AuthenticationException`

.

`PaymentHandlerException`(final `String`, final `int`, final `Map< String, String >`) constructor

Syntax

`PaymentHandlerException ( final String message , final int errorCode , final Map< String, String > parameters )`

Parameters

- **message** – the exception's detail message
- **errorCode** – the exception's error code
- **parameters** – the exception parameters

Usage

Creates a new instance of

`AuthenticationException`

.

`PaymentHandlerException`(final `String`, final `Throwable`, final `int`, final `Map< String, String >`) constructor

Syntax

`PaymentHandlerException ( final String message , final Throwable cause , final int errorCode , final Map< String, String > parameters )`

Parameters

- **message** – the exception's detail message
- **cause** – the exception's cause

- **errorCode** – the exception's error code
- **parameters** – the exception parameters

Usage

Creates a new instance of

`AuthenticationException`

.

PaymentHandlerException(final Throwable, final int, final Map< String, String >)
constructor

Syntax

`PaymentHandlerException ( final Throwable cause , final int errorCode , final Map< String, String > parameters )`

Parameters

- **cause** – the exception's cause
- **errorCode** – the exception's error code
- **parameters** – the exception parameters

Usage

Creates a new instance of

`AuthenticationException`

.

getErrorCode() *method*

Syntax

`final int getErrorCode ()`

Returns

the `errorCode`

Usage

the `errorCode`

getParameters() method

Syntax

```
final Map< String, String >getParameters ()
```

Returns

an Object array containing the parameters

Usage

Returns the

parameters

available for this exception.

an Object array containing the parameters

types package

Members

All public members of the types package.

- **Balance class** –
- **ExchangeRate class** –
- **OrgUnitCurrencyCombination class** –
- **PaymentInstrumentClassCombination class** –
- **PaymentInstrumentTypeCombination class** –
- **TransactionData class** –

Balance class

Syntax

```
public class Balance
```

Usage

Balance object.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getBalance() method

Syntax

Long getBalance ()

Returns

the balance

Usage

the balance

getCreditLimit() method

Syntax

Long getCreditLimit ()

Returns

the creditLimit

Usage

the creditLimit

getCurrency() method

Syntax

String getCurrency ()

Returns

the currency

Usage

the currency

getDebitLimit() method

Syntax

Long getDebitLimit ()

Returns

the debitLimit

Usage

the debitLimit

setBalance(final Long) method

Syntax

```
void setBalance ( final Long balance )
```

Parameters

- **balance** – the balance to set

setCreditLimit(final Long) method

Syntax

```
void setCreditLimit ( final Long creditLimit )
```

Parameters

- **creditLimit** – the creditLimit to set

setCurrency(final String) method

Syntax

```
void setCurrency ( final String currency )
```

Parameters

- **currency** – the currency to set

setDebitLimit(final Long) method

Syntax

```
void setDebitLimit ( final Long debitLimit )
```

Parameters

- **debitLimit** – the debitLimit to set

ExchangeRate class

Syntax

```
public class ExchangeRate
```

Usage

This bean holds exchange rate information per the computation of an `com.sybase365.mobiliser.money.businesslogic.payment.api.IExchangeRateHandler`

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

ExchangeRate(*final String*, *final String*, *final long*, *final long*, *final BigDecimal*)
constructor

Create a new exchange rate bean.

Syntax

```
ExchangeRate ( final String fromCurrency , final String toCurrency ,  
final long fromAmount , final long toAmount , final BigDecimal  
rate )
```

Parameters

- **fromCurrency** – the source currency
- **toCurrency** – the target currency
- **fromAmount** – the source scaling amount
- **toAmount** – the target scaling amount
- **rate** – the exchange rate

getFromAmount() *method*

Syntax

```
long getFromAmount ()
```

Returns

the source scaling amount

Usage

the source scaling amount

getFromCurrency() *method*

Syntax

```
String getFromCurrency ()
```


Returns

the source currency

Usage

the source currency

getRate() method

Syntax

```
BigDecimal getRate ()
```

Returns

the exchange rate

Usage

the exchange rate

getToAmount() method

Syntax

```
long getToAmount ()
```

Returns

the target scaling amount

Usage

the target scaling amount

getToCurrency() method

Syntax

```
String getToCurrency ()
```

Returns

the target currency

Usage

the target currency

OrgUnitCurrencyCombination class

Syntax

```
public class OrgUnitCurrencyCombination
```

Usage

This bean defines a combination of handled currencies for com.sybase365.mobiliser.money.businesslogic.payment.api.IExchangeRateHandler s and org units.

- If both currencies are set, this specific combination is handled
- If only the from/to type is set, all exchange rates from/to the specified currency are handled by this exchange rate handler.
- If an orgunit is provided, the handler is only applicable for that org unit

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

OrgUnitCurrencyCombination() constructor

Syntax

```
OrgUnitCurrencyCombination ()
```

Usage

Creates a new

```
OrgUnitCurrencyCombination
```

.

OrgUnitCurrencyCombination(final String, final Currency, final Currency) constructor

Syntax

```
OrgUnitCurrencyCombination (final String orgUnit, final Currency  
fromCurrency, final Currency toCurrency)
```

Parameters

- **orgUnit** –
- **fromCurrency** –
- **toCurrency** –

Usage

Creates a new

OrgUnitCurrencyCombination

.

equals(final Object) method

Syntax

boolean equals (final Object *other*)

Usage

Checks orgunit, from currency, and to currency for equality (or both being

null

).

hashCode() method

Syntax

int hashCode ()

Returns

the sum of the property's hash codes.

Usage

The hash code for this bean.

the sum of the property's hash codes.

toString() method

Syntax

String toString ()

fromCurrency variable

The from currency.

Syntax

final Currency fromCurrency

orgUnit variable

The org unit.

Syntax

```
final String orgUnit
```

toCurrency variable

The to currency.

Syntax

```
final Currency toCurrency
```

PaymentInstrumentClassCombination class

Syntax

```
public class PaymentInstrumentClassCombination
```

Usage

This bean defines a combination of handled payment instrument classes for com.sybase365.mobiliser.money.businesslogic.payment.api.IPaymentHandler s.

- If both classes are set, this specific combination is handled
- If only the debit/credit class is set, debits/credits on payment instruments of this class are handled, the corresponding credit/debit is expected to be handled by another payment handler.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

PaymentInstrumentClassCombination(final PaymentInstrument.Class, final PaymentInstrument.Class) constructor

Syntax

```
PaymentInstrumentClassCombination ( final  
PaymentInstrument.Class debitPaymentInstrumentClass , final  
PaymentInstrument.Class creditPaymentInstrumentClass )
```

Parameters

- **debitPaymentInstrumentClass** –
- **creditPaymentInstrumentClass** –

Usage

Creates a new

PaymentInstrumentClassCombination

.

equals(final Object) method

Syntax

boolean equals (final Object *other*)

Usage

Checks debit and credit pi class for equality.

hashCode() method

Syntax

int hashCode ()

Returns

the sum of the property's hash codes.

Usage

The hash code for this bean.

the sum of the property's hash codes.

toString() method

Syntax

String toString ()

creditPaymentInstrumentClass variable

The credit payment instrument class.

Syntax

final PaymentInstrument.Class creditPaymentInstrumentClass

debitPaymentInstrumentClass variable

The debit payment instrument class.

Syntax

```
final PaymentInstrument.Class debitPaymentInstrumentClass
```

PaymentInstrumentTypeCombination class

Syntax

```
public class PaymentInstrumentTypeCombination
```

Usage

This bean defines a combination of handled payment instrument types for com.sybase365.mobiliser.money.businesslogic.payment.api.IPaymentHandler s.

- If both types are set, this specific combination is handled
- If only the debit/credit type is set, debits/credits on payment instruments of this type are handled, the corresponding credit/debit is expected to be handled by another payment handler.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

PaymentInstrumentTypeCombination(final Integer, final Integer) constructor

Syntax

```
PaymentInstrumentTypeCombination ( final Integer  
debitPaymentInstrumentType , final Integer creditPaymentInstrumentType )
```

Parameters

- **debitPaymentInstrumentType** –
- **creditPaymentInstrumentType** –

Usage

Creates a new

PaymentInstrumentTypeCombination

.

equals(final Object) method

Syntax

```
boolean equals ( final Object other )
```

Usage

Checks debit and credit pi type for equality.

hashCode() method

Syntax

```
int hashCode ()
```

Returns

the sum of the property's hash codes.

Usage

The hash code for this bean.

the sum of the property's hash codes.

toString() method

Syntax

```
String toString ()
```

creditPaymentInstrumentType variable

The credit payment instrument type.

Syntax

```
final Integer creditPaymentInstrumentType
```

debitPaymentInstrumentType variable

The debit payment instrument type.

Syntax

```
final Integer debitPaymentInstrumentType
```

TransactionData class

Syntax

```
public class TransactionData
```

Usage

This bean holds all the relevant data for payment handlers so that they can perform the execution of a payment.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

TransactionData(final Long, final long, final long, final SubTransaction, final SubTransaction, final List< IKeyValue >, final List< IKeyValue >) constructor
Create a new transaction data bean.

Syntax

```
TransactionData ( final Long actorId , final long payeeNetAmount ,  
final long payerNetAmount , final SubTransaction transaction , final  
SubTransaction refTxn , final List< IKeyValue >  
unstructuredDataRequest , final List< IKeyValue >  
unstructuredDataResponse )
```

Parameters

- **actorId** – the current authenticated user.
- **payeeNetAmount** – the net amount the payee must receive
- **payerNetAmount** – the net amount the maper must be charged
- **transaction** – the current sub transaction
- **refTxn** – the sub transaction this current payment action is referenced to (might be null if the current payment action is an authorization, i.e. the first step in payment processing).
- **unstructuredDataRequest** – additional data from the request.
- **unstructuredDataResponse** – can be used by the payment handler to add additional data into the Mobiliser response.

TransactionData(final Long, final long, final long, final SubTransaction, final TransactionData) constructor

Syntax

```
TransactionData ( final Long actorId , final long payeeNetAmount ,  
final long payerNetAmount , final SubTransaction transaction , final  
TransactionData data )
```


addUnstructuredDataResponse(final IKeyValue) method

Syntax

void addUnstructuredDataResponse (final IKeyValue *keyValue*)

getActorId() method

Syntax

Long getActorId ()

Returns

the actorId

Usage

the actorId

getPayeeNetAmount() method

Syntax

long getPayeeNetAmount ()

Returns

the payeeNetAmount

Usage

the payeeNetAmount

getPayerNetAmount() method

Syntax

long getPayerNetAmount ()

Returns

the payerNetAmount

Usage

the payerNetAmount

getRefTxn() method

Syntax

SubTransaction getRefTxn ()

Returns

the refTxn

Usage

the refTxn

getTransaction() method

Syntax

SubTransaction getTransaction ()

Returns

the transaction

Usage

the transaction

getUnstructuredDataRequest() method

Syntax

List< IKeyValue > getUnstructuredDataRequest ()

IExchangeRateBroker interface

Syntax

public interface IExchangeRateBroker

Usage

The exchange rate broker can be imported from the OSGi service registry and used to get access to the appropriate IExchangeRateHandler.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getExchangeRateHandler(OrgUnit, Currency, Currency) method

Syntax

```
IEExchangeRateHandler getExchangeRateHandler ( OrgUnit orgUnit ,  
Currency fromCurr , Currency toCurr )
```

Parameters

- **orgUnit** – the org unit for this exchange rate
- **fromCurr** – the source currency
- **toCurr** – the target currency

Returns

the appropriate ExchangeRateHandler or null if no handler is found

Usage

Returns the appropriate exchange rate handler for the specified org unit and (optional) currency combination.

the appropriate ExchangeRateHandler or null if no handler is found

IEExchangeRateHandler interface

Syntax

```
public interface IEExchangeRateHandler
```

Usage

Register an implementation of this interface in the OSGi service registry to provide an own implementation of an exchange rate handler.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCoverage() method

Syntax

```
OrgUnitCurrencyCombination[] getCoverage ()
```

Returns

an array of OrgUnitCurrencyCombinations

Usage

Returns the currency combinations this handler is responsible for. This is used by the
IExchangeRateBroker

when searching for an exchange rate handler.

an array of OrgUnitCurrencyCombinations

getExchangeRate(Transaction, Currency, Currency) method

Syntax

ExchangeRate getExchangeRate (Transaction *monTxn* , Currency
fromCurr , Currency *toCurr*)

Parameters

- **monTxn** – the money transaction
- **fromCurr** – the source currency
- **toCurr** – the target currency

Returns

the matching exchange rate or null if this handler cannot compute the exchange rate.

Usage

Performs an exchange rate retrieval for the specified currencies.

the matching exchange rate or null if this handler cannot compute the exchange rate.

getName() method

Syntax

String getName ()

Returns

the name.

Usage

Returns the name of the exchange rate handler. Can be the class name or a short description.
This is shown by the broker in the log file

the name.

IPaymentBroker interface**Syntax**

```
public interface IPaymentBroker
```

Usage

The payment broker is responsible for determining which IPaymentHandler to use for a given combination of payment instruments. You can retrieve the payment broker from the OSGi service registry.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCreditPaymentHandler(PaymentInstrument.Type) method

Find the matching IPaymentHandler.

Syntax

```
IPaymentHandler getCreditPaymentHandler  
( PaymentInstrument.Type creditType )
```

Parameters

- **creditType** – type of the payment instrument that is credited.

Returns

the matching payment handler or null if no match is found

Usage

the matching payment handler or null if no match is found

getDebitPaymentHandler(PaymentInstrument.Type) method

Find the matching IPaymentHandler.

Syntax

```
IPaymentHandler getDebitPaymentHandler ( PaymentInstrument.Type  
debitType )
```

Parameters

- **debitType** – type of the payment instrument that is debited.

Returns

the matching payment handler or null if no match is found

Usage

the matching payment handler or null if no match is found

getPaymentHandler(PaymentInstrument.Type, PaymentInstrument.Type) method
Find the matching IPaymentHandler.

Syntax

```
IPaymentHandler getPaymentHandler ( PaymentInstrument.Type  
debitType , PaymentInstrument.Type creditType )
```

Parameters

- **debitType** – type of the payment instrument that is debited.
- **creditType** – type of the payment instrument that is credited.

Returns

the matching payment handler or null if no type can handle both the credit and debit type in one step.

Usage

This method should always be tried first before searching for debit and credit payment handlers in isolation with

```
getDebitPaymentHandler(com.sybase365.mobiliser.money.persistence.model.pi.PaymentInstrument.Type)
```

and

```
getCreditPaymentHandler(com.sybase365.mobiliser.money.persistence.model.pi.PaymentInstrument.Type)
```

the matching payment handler or null if no type can handle both the credit and debit type in one step.

IPaymentHandler interface

Syntax

```
public interface IPaymentHandler
```

Derived classes

- *IPaymentInstrumentClassHandler* on page 206
- *IPaymentInstrumentTypeHandler* on page 208

Usage

The `PaymentHandler` is responsible for authorizing, capturing and canceling a transaction. If processing fails, the payment handler must throw a `PaymentHandlerException`.

To register a new payment handler with the `IPaymentBroker`, implement either the `IPaymentInstrumentClassHandler` or `IPaymentInstrumentTypeHandler` and register the implementation into the OSGi service registry under the specific interface you've implemented.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

authorise(final TransactionData, final PaymentMode) method

Syntax

```
void authorise ( final TransactionData txnData , final PaymentMode
paymentMode ) throws PaymentHandlerException
```

Parameters

- **txnData** – the transaction data
- **paymentMode** – the payment mode

Exceptions

- **PaymentHandlerException** – if anything went wrong and the transaction failed.

Usage

Performs an authorisation for the specified transaction.

balanceInquiry(final PaymentInstrument) method

Syntax

```
Balance balanceInquiry ( final PaymentInstrument paymentInstrument )
throws UnsupportedOperationException, PaymentHandlerException
```

Parameters

- **paymentInstrument** – the payment instrument on which to perform the balance inquiry

Returns

the balance

Exceptions

- **UnsupportedOperationException** – if the payment handler doesn't support balance inquiry
- **PaymentHandlerException** – if anything went wrong and the transaction failed.

Usage

Performs a balance inquiry for the specified payment instrument.

the balance

cancelAuthorisation(final TransactionData, final PaymentMode) method

Syntax

```
void cancelAuthorisation ( final TransactionData txnData , final  
PaymentMode paymentMode ) throws PaymentHandlerException
```

Parameters

- **txnData** – the transaction data
- **paymentMode** – the payment mode

Exceptions

- **PaymentHandlerException** – if anything went wrong and the transaction failed.

Usage

Performs an authorisation cancel for the specified transaction.

cancelCapture(final TransactionData, final PaymentMode) method

Syntax

```
void cancelCapture ( final TransactionData txnData , final  
PaymentMode paymentMode ) throws PaymentHandlerException
```


Parameters

- **txnData** – the transaction data
- **paymentMode** – the payment mode

Exceptions

- **PaymentHandlerException** – if anything went wrong and the transaction failed.

Usage

Performs a capture cancel for the specified transaction.

capture(final TransactionData, final PaymentMode) method

Syntax

```
void capture ( final TransactionData txnData ,   final PaymentMode
paymentMode ) throws PaymentHandlerException
```

Parameters

- **txnData** – the transaction data
- **paymentMode** – the payment mode

Exceptions

- **PaymentHandlerException** – if anything went wrong and the transaction failed.

Usage

Performs a capture for the specified transaction.

getName() method

Syntax

```
String getName ()
```

Returns

the name.

Usage

Returns the name of the payment handler. Can be the class name or a short description. This is shown by the broker in the log file

the name.

preAuthorise(final TransactionData, final PaymentMode) method

Syntax

```
void preAuthorise ( final TransactionData txnData , final  
PaymentMode paymentMode ) throws UnsupportedOperationException,  
PaymentHandlerException
```

Parameters

- **txnData** – the transaction data
- **paymentMode** – the payment mode

Exceptions

- **UnsupportedOperationException** – if pre-authorization is not supported. The payment broker will interpret this as a successful pre-authorization
- **PaymentHandlerException** – if anything went wrong and the transaction failed.

Usage

Performs a non-committal pre-authorization for the specified transaction.

supportsCaptureCancel() method

Syntax

```
boolean supportsCaptureCancel ()
```

Returns

true if the handler supports capture cancel

Usage

Defines if this PaymentHandler supports CaptureCancel

true if the handler supports capture cancel

IPaymentInstrumentClassHandler interface

Syntax

```
public interface IPaymentInstrumentClassHandler
```

Derived classes

- *AbstractCardPaymentHandler* on page 209

Usage

The `IPaymentInstrumentClassHandler` extends the core `IPaymentHandler` interface by defining the responsibility of the handler. Concrete implementations MUST specify for which payment-instrument class(es) they are responsible. Register your implementation in the OSGi registry to make it available to the `IPaymentBroker`.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCoverage() method

Syntax

```
PaymentInstrumentClassCombination[] getCoverage ()
```

Returns

an array of `PaymentInstrumentClassCombinations`

Usage

Returns the classes of payment instruments this handler is responsible for.

an array of `PaymentInstrumentClassCombinations`

IPaymentInstrumentLookup interface

Syntax

```
public interface IPaymentInstrumentLookup
```

Usage

`PaymentInstrument` is now a concrete class and the additional payment instrument must be loaded separately. Implementations of this interface make loading those types simpler. Simply reference this interface from the OSGi service registry to get a suitable implementation.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getPaymentInstrumentDelegateType(final PaymentInstrument) method

Get the payment instrument details for this instrument.

Syntax

```
Object getPaymentInstrumentDelegateType ( final  
PaymentInstrument paymentInstrument )
```

Parameters

- **paymentInstrument** – the payment instrument for which the details should be loaded

Returns

the particular JPA payment instrument bean, depending on the type of the passed paymentInstrument.

Usage

the particular JPA payment instrument bean, depending on the type of the passed paymentInstrument.

IPaymentInstrumentTypeHandler interface

Syntax

```
public interface IPaymentInstrumentTypeHandler
```

Usage

The IPaymentInstrumentTypeHandler extends the core IPaymentHandler interface by defining the responsibility of the handler. Concrete implementations MUST specify for which payment-instrument type(s) they are responsible. Register your implementation in the OSGi registry to make it available to the IPaymentBroker.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCoverage() method

Syntax

```
PaymentInstrumentTypeCombination[] getCoverage ()
```

Returns

an array of PaymentInstrumentTypeCombinations

Usage

Returns the types of payment instruments this handler is responsible for.
 an array of `PaymentInstrumentTypeCombinations`

handlers package*Members*

All public members of the handlers package.

- **model package** –

*model package**Members*

All public members of the model package.

- **AbstractCardPaymentHandler class** –

*AbstractCardPaymentHandler class**Syntax*

```
public class AbstractCardPaymentHandler
```

Usage

The `AbstractCardPaymentHandler` provides a base implementation of an `IPaymentInstrumentClassHandler` that handles credit cards with encrypted data. This implementation takes care of card data decryption and provides easy to implement methods to actually implement the integration with the card processor.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

*AbstractCardPaymentHandler() constructor***Syntax**

```
AbstractCardPaymentHandler ()
```

Usage

Creates a new instance of

```
SvaPaymentHandler
```

.

afterPropertiesSet() method

Syntax

```
void afterPropertiesSet ()
```

authorise(final TransactionData, final PaymentMode) method

Syntax

```
final void authorise ( final TransactionData txnData , final  
PaymentMode paymentMode ) throws PaymentHandlerException
```

Parameters

- **txnData** – the transaction data
- **paymentMode** – the payment mode

Exceptions

- **PaymentHandlerException** – if anything went wrong and the transaction failed.

Usage

Performs an authorisation for the specified transaction.

authorise(final Customer, final Card, final Transaction, final SubTransaction, final long) method

Implement an authorization call to the payment card processor interface.

Syntax

```
abstract void authorise ( final Customer cardHolder , final Card  
card , final Transaction mainTxn , final SubTransaction subTxn ,  
final long amount ) throws PaymentHandlerException
```

Parameters

- **cardHolder** – the card holder of this card
- **card** – the clear text card data
- **mainTxn** – the current transaction
- **subTxn** – the current subtransaction, remember setting a payer reference if applicable
- **amount** – the amount that should be charged

Exceptions

- **PaymentHandlerException** – if the payment processor call fails

balanceInquiry(final PaymentInstrument) method

Syntax

```
final Balance balanceInquiry ( final PaymentInstrument
paymentInstrument ) throws UnsupportedOperationException,
PaymentHandlerException
```

Parameters

- **paymentInstrument** – the payment instrument on which to perform the balance inquiry

Returns

the balance

Exceptions

- **UnsupportedOperationException** – if the payment handler doesn't support balance inquiry
- **PaymentHandlerException** – if anything went wrong and the transaction failed.

Usage

Performs a balance inquiry for the specified payment instrument.

the balance

balanceInquiry(final Customer, final Card) method

Implement this method to get the card balance from the external system.

Syntax

```
abstract Balance balanceInquiry ( final Customer cardHolder , final
Card card ) throws UnsupportedOperationException,
PaymentHandlerException
```

Parameters

- **cardHolder** – the card holder of this card
- **card** – the clear text card data

Returns

the current balance.

Exceptions

- **PaymentHandlerException** – if the payment processor call fails, communicate this by throwing a PaymentHandlerException.
- **UnsupportedOperationException** – if the payment card processor does not support balance inquiries, simply throw an UnsupportedOperationException in your implementation

Usage

the current balance.

cancelAuthorisation(final TransactionData, final PaymentMode) method

Syntax

```
final void cancelAuthorisation ( final TransactionData txnData ,  
final PaymentMode paymentMode ) throws PaymentHandlerException
```

Parameters

- **txnData** – the transaction data
- **paymentMode** – the payment mode

Exceptions

- **PaymentHandlerException** – if anything went wrong and the transaction failed.

Usage

Performs an authorisation cancel for the specified transaction.

cancelAuthorisation(Customer, Card, Transaction, SubTransaction, Authorisation, long) method

Implement an reversal of an existing authorization/reservation at the payment card processor interface.

Syntax

```
abstract void cancelAuthorisation ( Customer cardHolder , Card card ,  
Transaction mainTxn , SubTransaction subTxn , Authorisation  
authorisation , long amount ) throws PaymentHandlerException
```

Parameters

- **cardHolder** – the card holder of this card

- **card** – the clear text card data
- **mainTxn** – the current transaction
- **subTxn** – the current subtransaction, remember setting a payer reference if applicable
- **authorisation** – the original authorisation subtransaction, you can get reference ids from there if provided earlier in the authorisation implementation
- **amount** – the amount that should be charged

Exceptions

- **PaymentHandlerException** – if the payment processor call fails

cancelCapture(final TransactionData, final PaymentMode) method

Syntax

```
final void cancelCapture ( final TransactionData txnData , final
PaymentMode paymentMode ) throws PaymentHandlerException
```

Parameters

- **txnData** – the transaction data
- **paymentMode** – the payment mode

Exceptions

- **PaymentHandlerException** – if anything went wrong and the transaction failed.

Usage

Performs a capture cancel for the specified transaction.

cancelCapture(Customer, Card, Transaction, SubTransaction, Capture, long)
method

Implement an reversal on an existing capture at the payment card processor interface.

Syntax

```
abstract void cancelCapture ( Customer cardHolder , Card card ,
Transaction mainTxn , SubTransaction subTxn , Capture capture , long
amount ) throws PaymentHandlerException
```

Parameters

- **cardHolder** – the card holder of this card
- **card** – the clear text card data

- **mainTxn** – the current transaction
- **subTxn** – the current subtransaction, remember setting a payer reference if applicable
- **capture** – the original capture subtransaction, you can get reference ids from there if provided earlier in the capture implementation
- **amount** – the amount that should be charged

Exceptions

- **PaymentHandlerException** – if the payment processor call fails

capture(final TransactionData, final PaymentMode) method

Syntax

`final void capture ( final TransactionData txnData , final PaymentMode paymentMode ) throws PaymentHandlerException`

Parameters

- **txnData** – the transaction data
- **paymentMode** – the payment mode

Exceptions

- **PaymentHandlerException** – if anything went wrong and the transaction failed.

Usage

Performs a capture for the specified transaction.

capture(Customer, Card, Transaction, SubTransaction, Authorisation, long) method

Implement a capture of a previous authorization at the payment card processor interface.

Syntax

`abstract void capture ( Customer cardHolder , Card card , Transaction mainTxn , SubTransaction subTxn , Authorisation authorisation , long amount ) throws PaymentHandlerException`

Parameters

- **cardHolder** – the card holder of this card
- **card** – the clear text card data
- **mainTxn** – the current transaction
- **subTxn** – the current subtransaction, remember setting a payer reference if applicable

- **authorisation** – the earlier authorisation subtransaction, you can get reference ids from there if provided earlier in the authorisation implementation
- **amount** – the amount that should be charged

Exceptions

- **PaymentHandlerException** – if the payment processor call fails

decrypt(final Card) method

This is a hook to decrypt additional fields other than card number and security number, if required.

Syntax

```
Card decrypt ( final Card card )
```

Parameters

- **card** – the card number.

Returns

a modified card that has more fields decrypted.

Usage

Simply override this method. Card and security number will already be decrypted.

a modified card that has more fields decrypted.

getSecurityConfiguration() method

Implement this to get a proper instance of the security configuration, required for accessing the keystore.

Syntax

```
abstract SecurityConfiguration getSecurityConfiguration ()
```

Returns

the applicable security configuration.

Usage

Implementations should instantiate the

SecurityConfiguration

with a suitable preferences node and return it here.

the applicable security configuration.

preAuthorise(final TransactionData, final PaymentMode) method

Syntax

`final void preAuthorise ( final TransactionData txnData , final PaymentMode paymentMode )` throws `UnsupportedOperationException`, `PaymentHandlerException`

Parameters

- **txnData** – the transaction data
- **paymentMode** – the payment mode

Exceptions

- **UnsupportedOperationException** – if pre-authorization is not supported. The payment broker will interpret this as a successful pre-authorization
- **PaymentHandlerException** – if anything went wrong and the transaction failed.

Usage

Performs a non-committal pre-authorization for the specified transaction.

preAuthorise(final Customer, final Card, final Transaction, final SubTransaction, final long) method

Implement a preauthorization call (no reservation) to the payment card processor interface.

Syntax

`abstract void preAuthorise ( final Customer cardHolder , final Card card , final Transaction mainTxn , final SubTransaction subTxn , final long amount )` throws `UnsupportedOperationException`, `PaymentHandlerException`

Parameters

- **cardHolder** – the card holder of this card
- **card** – the clear text card data
- **mainTxn** – the current transaction
- **subTxn** – the current subtransaction, remember setting a payer reference if applicable
- **amount** – the amount that should be charged

Exceptions

- **PaymentHandlerException** – if the payment processor call fails
- **UnsupportedOperationException** – if the payment card processor does not support pre-authorizations, simply throw an UnsupportedOperationException

setAuthorisationDao(final AuthorisationDAO) method

Syntax

```
void setAuthorisationDao ( final AuthorisationDAO authorisationDao )
```

setCaptureDao(final CaptureDAO) method

Syntax

```
void setCaptureDao ( final CaptureDAO captureDao )
```

setCardDao(final CardDAO) method

Syntax

```
void setCardDao ( final CardDAO cardDAO )
```

setPiLookup(final IPaymentInstrumentLookup) method

Syntax

```
void setPiLookup ( final IPaymentInstrumentLookup piLookup )
```

Parameters

- **piLookup** – the piLookup to set

toString() method

Return a short description of your payment handler here.

Syntax

```
abstract String toString ()
```

Returns

a `String` representing this instance

Usage

This will be used in log messages.

a `String` representing this instance

system package

Members

All public members of the system package.

- **IBatchRunLock interface** – The IBatchRunLock can be used to synchronize threads (running possibly in multiple VMs) via a semaphore in the DB.
- **ICertificateLookup interface** – ICertificateLookup.
- **ICredentialPoliciesLogic interface** – The businesslogic service ICredentialPoliciesLogic provides all services around the credential policies including the configuration of customer type, credential type to credential policy.
- **IExchangeRateLogic interface** – Interface IExchangeRateLogic contains methods to handle exchange rates.
- **IGLAccountLogic interface** – The businesslogic service IGLAccountLogic provides services around the GLAccountTree management for loading the Tree and deleting a GLAccount.
- **IInterestLogic interface** – The IInterestLogic defines the businesslogic related to interest.
- **ILimitsLogic interface** – The interface ILimitsLogic contains CRUD type as well as some complex methods that all deal with Limits and the related entities.
- **IPrivateKeyLookup interface** – IPrivateKeyLookup.
- **IRsaPrivateKeyLogic interface** – IRsaPrivateKeyLogic.
- **IRsaPublicKeyLogic interface** – IRsaPublicKeyLogic.
- **ISystemLogic interface** – The interface ISystemLogic contains various methods to work on the system entity beans.
- **ITransactionRestrictionLogic interface** – The interface ITransactionRestrictionLogic contains mainly CRUD operations to work on transaction restrictions and the directly related entities.
- **ITxntemplLogic interface** – The interface ITxntemplLogic contains CRUD type methods to work on Transaction Templates.
- **LockingException class** – The LockingException is used when a lock on the Batchrun cannot be acquired.

IBatchRunLock interface

The IBatchRunLock can be used to synchronize threads (running possibly in multiple VMs) via a semaphore in the DB.

Syntax

```
public interface IBatchRunLock
```

Usage

All methods run in their own DB transaction and commit their work right after the call. Each different class should use its own batch run type which must be inserted into the table

MOB_BATCHRUN_TYPES - unless of course you want to synchronize multiple classes via the same semaphore.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getLock(final int, final int, final long) method

This function makes 2 things: it creates an entry in table BATCHRUNS and directly checks, if no other instance of the given batchType is running (finished is null).

Syntax

```
long getLock ( final int batchType , final int maxAgeInSeconds , final long callerId ) throws LockingException
```

Parameters

- **batchType** – Batch type as existing in MOB_BATCHRUN_TYPES
- **maxAgeInSeconds** – The maxAgeInSeconds specifies after which time an old, unfinished BatchRun is ignored
- **callerId** – The ID of the caller of the service.

Exceptions

- **LockingException class** – is thrown in case the lock can not be obtained (another instance is already running).

Usage

The maxAgeInSeconds specifies after which time an old, unfinished BatchRun is ignored. The function

```
releaseLock
```

has to be called to release the lock.

getLockInstance(final int, final int, final long) method

Utilises getLock to to acquire a lock, but instead of returning the batch runs ID, it returns the batchrun instance.

Syntax

```
BatchRun getLockInstance ( final int batchType , final int maxAgeInSeconds , final long callerId ) throws LockingException
```

Parameters

- **batchType** –
- **maxAgeInSeconds** –
- **callerId** –

Exceptions

- **LockingException** class –

releaseLock(final long, final long) method

Releases the lock that was obtained by getLock.

Syntax

```
void releaseLock ( final long id ,   final long callerId )
```

Parameters

- **id** –
- **callerId** –

ICertificateLookup interface

ICertificateLookup.

Syntax

```
public interface ICertificateLookup
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCertificate(final String) method

Find the certificate with the given name.

Syntax

```
Certificate getCertificate ( final String keyName )
```

Parameters

- **keyName** – the keyname

Returns

the certificate

Usage

the certificate

ICredentialPoliciesLogic interface

The businesslogic service `ICredentialPoliciesLogic` provides all services around the credential policies including the configuration of customer type, credential type to credential policy.

Syntax

```
public interface ICredentialPoliciesLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

createCrendetialPolicy(CredentialPolicy, List< CredentialPolicyRegex >, Long)
method

Creates a credential policy and all of the defined `CredentialPolicyRegEx`.

Syntax

```
Long createCrendetialPolicy ( CredentialPolicy policy , List<
CredentialPolicyRegex > rules , Long actor )
```

Parameters

- `policy` –

Returns

the id of the newly created entity

Usage

the id of the newly created entity

createCustomerTypeCredentialPolicy(CustomerTypeCredentialPolicy, Long)
method

Create a new configuration for customer type credential type.

Syntax

```
Long createCustomerTypeCredentialPolicy
( CustomerTypeCredentialPolicy fromContract , Long actor )
```

Parameters

- **fromContract** –

deleteCredentialPolicy(long) method

Delete an credential policy.

Syntax

```
void deleteCredentialPolicy ( long credentialPolicyId )
```

Parameters

- **credentialPolicyId** –

deleteCustomerTypeCredentialPolicy(long) method

Delete customer type credential type policy configuration.

Syntax

```
void deleteCustomerTypeCredentialPolicy ( long id )
```

Parameters

- **fromContract** –

getAllCredentialPolicies() method

Return all credential policies from the persistence layer.

Syntax

```
List< CredentialPolicy > getAllCredentialPolicies ()
```

getAllCustomerTypeCredentialPolicies() method

Return all credential type to policy configuration entries.

Syntax

```
List< CustomerTypeCredentialPolicy >  
getAllCustomerTypeCredentialPolicies ()
```

getCredentialPolicy(long) method

Get the policy by Id.

Syntax

```
CredentialPolicy getCredentialPolicy ( long credentialPolicyId )
```

Parameters

- **credentialPolicyId** –

updateCredentialPolicy(Long, CredentialPolicy, List< CredentialPolicyRegex >, Long) method

Update an existing Credential policy.

Syntax

```
void updateCredentialPolicy ( Long id , CredentialPolicy
newValues , List< CredentialPolicyRegex > newRules , Long actor )
```

Parameters

- **id** –
- **fromContract** –

IExchangeRateLogic interface

Interface IExchangeRateLogic contains methods to handle exchange rates.

Syntax

```
public interface IExchangeRateLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

deleteExchangeRate(Currency, Currency) method

Removes an exchange rate configuration from the database.

Syntax

```
void deleteExchangeRate ( Currency fromCurrency , Currency toCurrency )
throws EntityNotFoundException
```

Parameters

- **fom** –
- **to** –

getActiveExchangeRate(Currency, Currency) method

Retrieves the current exchange rate for the given to and from currencies or null if none is found.

Syntax

ExchangeRate getActiveExchangeRate (Currency *fromCurrency* ,
Currency *toCurrency*)

Parameters

- **fromCurrency** –
- **toCurrency** –

getExchangeRateHistory(Currency, Currency) method

Retrieves a list of former exchange rate configurations for the given from and to currencies.

Syntax

List< ExchangeRateHistory > getExchangeRateHistory (Currency
fromCurrency , Currency *toCurrency*)

Parameters

- **fromCurrency** –
- **toCurrency** –

getExchangeRates(Currency, Currency) method

Retrieves all exchange rate information for the given from and to currencies.

Syntax

List< ExchangeRate > getExchangeRates (Currency *fromCurrency* ,
Currency *toCurrency*)

Parameters

- **from** –
- **to** –

Usage

This is a legacy method and actually always only returns 0 or 1 element.

setExchangeRate(ExchangeRate, long) method

Persists the given exchange rate configuration and returns the persited instance.

Syntax

```
ExchangeRate setExchangeRate ( ExchangeRate exchangeRate , long
callerId ) throws IllegalArgumentException
```

IGLAccountLogic interface

The businesslogic service IGLAccountLogic provides services around the GLAccountTree management for loading the Tree and deleting a GLAccount.

Syntax

```
public interface IGLAccountLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

deleteGLAccount(long, Long, Identification, Customer) method

Tries to delete a GLAccount entry in the database (not set inactive)

Syntax

```
void deleteGLAccount ( long gLAAccountId , Long paymentInstrumentId ,
Identification identifi , Customer customer )
```

Parameters

- **gLAAccountId** – This contains the CustomerId of the GLAccount which will be deleted (mandatory)
- **paymentInstrumentId** – This contains the id of the Payment - Instrument, it's just mandatory if it exists otherwise the delete - Request will fail.
- **identifi** – This contains the Identification, it's just mandatory if it exists otherwise the delete - Request will fail.
- **customer** – The customer Object of the GLAccount (mandatory) otherwise an EntityNotFoundException will be thrown.

getGLAccountTree(String, List< Integer >, Boolean) method**Syntax**

```
List< GLAccountTreeElementBean > getGLAccountTree ( String
orgUnit , List< Integer > types , Boolean getActiveOnly )
```

InterestLogic interface

The IInterestLogic defines the businesslogic related to interest.

Syntax

```
public interface IInterestLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

createInterestCalculation(com.sybase365.mobiliser.money.persistence.model.system.InterestCalculation, long) method

Manually creates an Interestcalculation entry in the Database.

Syntax

```
long createInterestCalculation  
( com.sybase365.mobiliser.money.persistence.model.system.InterestCalculation bean , long callerId )
```

Parameters

- **bean** – The interestcalculation object to create
- **callerId** – The caller Id for auditing purposes

Usage

Since the entries are generated by the interestCalculation job this should never be necessary.

createInterestConfiguration(com.sybase365.mobiliser.money.persistence.model.system.InterestConfiguration, long) method

Manually creates an InterestConfiguration, interestconfigurations are automatically created for all SVAs at SVA creation time so this is not necessary for standard SVAs.

Syntax

```
long createInterestConfiguration  
( com.sybase365.mobiliser.money.persistence.model.system.InterestConfiguration bean , long callerId )
```

Parameters

- **bean** – The interestconfiguration object which shall be created
- **callerId** – The caller Id for auditing purposes

*createInterestPeriod(com.sybase365.mobiliser.money.persistence.model.system.I
nterestPeriod, long) method*

Creates an InterestPeriod.

Syntax

```
java.lang.String createInterestPeriod
( com.sybase365.mobiliser.money.persistence.model.system.Inte
restPeriod bean , long callerId )
```

Parameters

- **bean** – The interestperiod object which shall be created
- **callerId** – The caller Id for auditing purposes

*createInterestProduct(com.sybase365.mobiliser.money.persistence.model.system.
InterestProduct, long) method*

Creates an InterestProduct.

Syntax

```
long createInterestProduct
( com.sybase365.mobiliser.money.persistence.model.system.Inte
restProduct bean , long callerId )
```

Parameters

- **bean** – The interestProduct object which shall be created
- **callerId** – The caller Id for auditing purposes

*createInterestProductMapping(com.sybase365.mobiliser.money.persistence.model
.system.InterestProductMapping, long) method*

Creates an InterestProductMapping.

Syntax

```
long createInterestProductMapping
( com.sybase365.mobiliser.money.persistence.model.system.Inte
restProductMapping bean , long callerId )
```

Parameters

- **bean** – The interest product mapping object which shall be created
- **callerId** – The caller Id for auditing purposes

deleteInterestCalculation(long, long) method

Manually deletes an Interestcalculation entry in the Database.

Syntax

```
void deleteInterestCalculation ( long id ,   long callerId )
```

Parameters

- **id** – The id of the calculation which is to be deleted
- **callerId** – The caller Id for auditing purposes

Usage

Since the entries are generated by the interestCalculation job and may be required for keeping records this should never be necessary as a manual step.

deleteInterestConfiguration(long, long) method

Deletes a InterestConfiguration byId, should not be necessary and may mess up the interest calculation/clearing process.

Syntax

```
void deleteInterestConfiguration ( long id ,   long callerId )
```

Parameters

- **id** – The id of the interestconfiguration which shall be deleted
- **callerId** – The caller Id for auditing purposes

deleteInterestPeriod(java.lang.String, long) method

Deletes a InterestPeriod byId.

Syntax

```
void deleteInterestPeriod ( java.lang.String id ,   long callerId )
```

Parameters

- **id** – The id of the period which is to be deleted
- **callerId** – The caller Id for auditing purposes

deleteInterestProduct(long, long) method

Deletes a InterestProduct by Id.

Syntax

```
void deleteInterestProduct ( long id , long callerId )
```

Parameters

- **id** – The id of the product which is to be deleted
- **callerId** – The caller Id for auditing purposes

deleteInterestProductMapping(long, long) method

Deletes a InterestProductMapping byId.

Syntax

```
void deleteInterestProductMapping ( long id , long callerId )
```

Parameters

- **id** – The id of the productMapping which is to be deleted
- **callerId** – The caller Id for auditing purposes

getInterestCalculation(long, long) method

Retrieves a interestcalculation entry.

Syntax

```
com.sybase365.mobiliser.money.persistence.model.system.InterestCalculation getInterestCalculation ( long id , long callerId )
```

Parameters

- **id** – The id of the calculation to retrieve
- **callerId** – The caller Id for auditing purposes

Returns

The retrieved InterestCalculation

Usage

The retrieved InterestCalculation

getInterestConfiguration(long, long) method

Retrieves an interestConfiguration by Id.

Syntax

```
com.sybase365.mobiliser.money.persistence.model.system.InterestConfiguration getInterestConfiguration ( long id , long callerId )
```

Parameters

- **id** – The id to be retrieved
- **callerId** – The caller Id for auditing purposes

getInterestPeriod(java.lang.String, long) method

Getter method for InterestPeriod.

Syntax

```
com.sybase365.mobiliser.money.persistence.model.system.InterestPeriod getInterestPeriod ( java.lang.String id , long callerId )
```

Parameters

- **id** – The id to get
- **callerId** – The caller Id for auditing purposes

getInterestProduct(long, long) method

Getter method for InterestProduct.

Syntax

```
com.sybase365.mobiliser.money.persistence.model.system.InterestProduct getInterestProduct ( long id , long callerId )
```

Parameters

- **id** – The id to get
- **callerId** – The caller Id for auditing purposes

getInterestProductList(long) method

Gets a list of all available products.

Syntax

```
java.util.List<
com.sybase365.mobiliser.money.persistence.model.system.Inter
estProduct > getInterestProductList ( long callerId )
```

Parameters

- **callerId** – The caller Id for auditing purposes

Returns

A list of all products

Usage

A list of all products

getInterestProductMapping(long, long) method

Getter method for InterestProductMapping.

Syntax

```
com.sybase365.mobiliser.money.persistence.model.system.Inter
estProductMapping getInterestProductMapping ( long id , long
callerId )
```

Parameters

- **id** – The id to get
- **callerId** – The caller Id for auditing purposes

getInterestProductMappingList(long) method

Gets a list of all productMappings.

Syntax

```
java.util.List<
com.sybase365.mobiliser.money.persistence.model.system.Inter
estProductMapping > getInterestProductMappingList ( long
callerId )
```

Parameters

- **callerId** – The caller Id for auditing purposes

Returns

A list of all productMappings

Usage

A list of all productMappings

getScaleStepsByProduct(long, int, String) method

Retrieves a Map containing a List scale steps for the specified product, scaletype and currency.

Syntax

```
Map< String, List< Step > > getScaleStepsByProduct ( long  
productId, int scaleType, String currency )
```

Parameters

- **productId** – The id of the product to get the steps for
- **scaleType** – The scaletype. One of (CREDIT_SCALE | DEBIT_SCALE | TAX_SCALE)
- **currency** – The currency to get the steps for

Returns

A Map of scale steps with the currency as key.

Usage

A Map of scale steps with the currency as key.

setScaleStepsByProduct(long, int, Map< String, List< Step >>, long) method

Sets scalesteps for a specific product and scaletype.

Syntax

```
void setScaleStepsByProduct ( long productId, int scaleType, Map<  
String, List< Step >> currencyScaleSteps, long callerId ) throws  
EntityMandatoryException, EntityNotFoundException
```

Parameters

- **productId** – The product for which to set the scaleSteps
- **scaleType** – The scaletype for which to set the scaleSteps

- **currencyScaleSteps** – A list of scaleSteps
- **callerId** – The caller Id for auditing purposes

Exceptions

- **EntityMandatoryException** –
- **EntityNotFoundException** –

updateInterestCalculation(com.sybase365.mobiliser.money.persistence.model.system.InterestCalculation, long) method

Manually updates an Interestcalculation entry in the Database.

Syntax

```
void updateInterestCalculation  
( com.sybase365.mobiliser.money.persistence.model.system.InterestCalculation bean , long callerId )
```

Parameters

- **bean** – The interestcalculation object to update
- **callerId** – The caller Id for auditing purposes

Usage

Since the entries are generated by the interestCalculation job this should never be necessary.

updateInterestConfiguration(com.sybase365.mobiliser.money.persistence.model.system.InterestConfiguration, long) method

Updates an InterestConfiguration.

Syntax

```
void updateInterestConfiguration  
( com.sybase365.mobiliser.money.persistence.model.system.InterestConfiguration bean , long callerId )
```

Parameters

- **bean** – The interestconfiguration object which shall be updated
- **callerId** – The caller Id for auditing purposes

*updateInterestPeriod(com.sybase365.mobiliser.money.persistence.model.system.I
nterestPeriod, long) method*

Updates an InterestPeriod.

Syntax

```
void updateInterestPeriod  
( com.sybase365.mobiliser.money.persistence.model.system.Inte  
restPeriod bean , long callerId )
```

Parameters

- **bean** – The interest period object which shall be updated
- **callerId** – The caller Id for auditing purposes

*updateInterestProduct(com.sybase365.mobiliser.money.persistence.model.system
.InterestProduct, long) method*

Updates an InterestProduct.

Syntax

```
void updateInterestProduct  
( com.sybase365.mobiliser.money.persistence.model.system.Inte  
restProduct bean , long callerId )
```

Parameters

- **bean** – The interest product object which shall be updated
- **callerId** – The caller Id for auditing purposes

*updateInterestProductMapping(com.sybase365.mobiliser.money.persistence.mode
l.system.InterestProductMapping, long) method*

Updates an InterestProductMapping.

Syntax

```
void updateInterestProductMapping  
( com.sybase365.mobiliser.money.persistence.model.system.Inte  
restProductMapping bean , long callerId )
```

Parameters

- **bean** – The interest product mapping object which shall be updated
- **callerId** – The caller Id for auditing purposes

ILimitsLogic interface

The interface `ILimitsLogic` contains CRUD type as well as some complex methods that all deal with Limits and the related entities.

Syntax

```
public interface ILimitsLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

add(LimitSum, final long, final int, final boolean, final Date) method**Syntax**

```
void add ( LimitSum limitSum , final long amount , final int count ,  
final boolean credit , final Date fromDate ) throws  
IllegalDataException
```

Parameters

- **limitSum** –
- **amount** – the amount to add
- **count** – the count (usually 1 for regular transactions or -1 for cancels)
- **credit** – `true` to update the *credit* sums; `false` to update the *debit* sums
- **fromDate** –

Exceptions

- **IllegalDataException** –

Usage

Adds the specified amount to the absolute, monthly, weekly and daily sums for credit and debit, respectively.

checkLimitAmount(LimitSum, boolean) method**Syntax**

```
int checkLimitAmount ( LimitSum limitSum , boolean payee ) throws  
IllegalDataException
```

Returns

the error code as a result of a limit exceeded - if any, otherwise 0.

Usage

Checks if any limit is exceeded by the currently set sums.

the error code as a result of a limit exceeded - if any, otherwise 0.

clearLimitSetClass(long) method

Deletes all Limit Set to Limit Classes mappings for the given Limit Set ID.

Syntax

```
void clearLimitSetClass ( long limitSetId )
```

Parameters

- **limitSetId** –

createLimitClass(LimitClass, Long, ILimitsLogic.EntityType, Integer, long) method

Create a new limit class.

Syntax

```
long createLimitClass ( LimitClass limitClass , Long entityId ,  
ILimitsLogic.EntityType entityType , Integer useCase , long callerId )  
throws IllegalArgumentException, EntityNotFoundException
```

Parameters

- **limitClass** – The new Limit Class to create.
- **entityId** – This is either the ID (primary key) of a Customer, a Payment Instrument, or a Wallet entry
- **entityType** – The type of the entity (see entityType).
- **useCase** –
- **callerId** –

Exceptions

- **IllegalArgumentException** –
- **EntityNotFoundException** –

Usage

If the `entityId` is provided it will automatically be attached to the referenced entity, which can be a Customer, a Payment Instrument, a Wallet entry, or an existing Limit Set. In case it is any of the first three and there is no limit set yet, a new limit set is created.

createLimitSet(LimitSet, long) method

Creates a new Limit Set with the provided data and returns the ID of the newly created entity.

Syntax

```
long createLimitSet ( LimitSet limitSet ,   long callerId ) throws
EntityMandatoryException
```

Parameters

- **limitSet** –
- **callerId** –

Exceptions

- **EntityMandatoryException** –

createLimitSetClass(long, long, Integer, long) method

Creates a new link for an (optional) use case between limit set and limit class.

Syntax

```
long createLimitSetClass ( long limitSetId ,   long limitClassId ,   Integer
useCaseId ,   long callerId ) throws EntityNotFoundException
```

Parameters

- **limitSetId** –
- **limitClassId** –
- **useCaseId** –
- **callerId** –

Exceptions

- **EntityNotFoundException** –

deleteLimitClass(long, long) method

Deletes an existing Limit Class.

Syntax

```
void deleteLimitClass ( long limitClassId , long callerId ) throws  
EntityNotFoundException, EntityReferenceException
```

Parameters

- **limitClassId** –
- **callerId** –

Exceptions

- **EntityNotFoundException** –
- **EntityReferenceException** –

Usage

If the Limit Class is still referenced in the DB, the record is not deleted but an EntityReferenceException is thrown.

deleteLimitSet(long, long) method

Deletes an existing Limit Set.

Syntax

```
void deleteLimitSet ( long limitSetId , long callerId ) throws  
EntityNotFoundException
```

Parameters

- **limitSetId** –
- **callerId** –

Exceptions

- **EntityNotFoundException** –

Usage

Use with caution. The Limit Set is first detached from Customers, Customer Types, etc. then it is deleted.

deleteLimitSetClass(long) method

Deletes an existing limit set class by its ID.

Syntax

```
void deleteLimitSetClass ( long limitSetClassId ) throws
EntityNotFoundException
```

Parameters

- **limitSetClassId** –

Exceptions

- **EntityNotFoundException** –

getLimitClasses() method

Returns all available Limit Classes.

Syntax

```
List< LimitClass > getLimitClasses ()
```

getLimitClasses(boolean) method

Returns all available Limit Classes.

Syntax

```
List< LimitClass > getLimitClasses ( boolean filterIndividual )
```

Parameters

- **filterIndividual** – The individual Limit Classes are not returned in case this parameter is set to true

Usage

Optionally the individual ones can be filtered out.

getLimitSetClass(long) method

Retrieves the Limit Set Class by its ID.

Syntax

```
LimitSetClass getLimitSetClass ( long limitSetClassId )
```

Parameters

- **limitSetClassId** –

getLimitSetClasses(long, ILimitsLogic.EntityType) method

Returns all limit set classes for the given entity which is either customer, payment instrument, wallet entry, or limit set.

Syntax

```
List< LimitSetClass > getLimitSetClasses ( long entityId ,  
ILimitsLogic.EntityType entityType )
```

Parameters

- **entityId** –
- **entityType** –

getLimitSets(Boolean) method

This method returns all available Limit Sets.

Syntax

```
List< LimitSet > getLimitSets ( Boolean fetchIndividual )
```

Parameters

- **fetchIndividual** – If `true` only individual Limit Sets are returned. If `false` only global/generic Limit Sets are returned. If `null` all Limit Sets are returned.

Usage

The parameter `fetchIndividual` controls whether the individual (= unique for a customer) Limit Sets are to be fetched or only generic or all.

resetSums(LimitSum, final Date) method

Syntax

```
void resetSums ( LimitSum limitSum , final Date currentTimestamp ) throws  
IllegalDataException
```

Parameters

- **limitSum** – the limit sum to reset
- **currentTimestamp** – the Date representing the current point in time
- **timeZone** – time zone in which the computation should be done

Exceptions

- **IllegalDataException** –

Usage

Resets the daily and/or weekly and/or monthly sums if necessary. It is conceived necessary to reset the sums if the specified timestamp is a day older as the last update and a month older as the last update, respectively. If resetting the monthly sums, the daily sums are reset, too.

updateLimitClass(LimitClass, long) method

Updates an existing Limit Class with the provided data.

Syntax

```
void updateLimitClass ( LimitClass limitClass , long callerId ) throws  
IllegalDataException
```

Parameters

- **limitClass** –
- **callerId** –

Exceptions

- **IllegalDataException** –

updateLimitSet(LimitSet, long) method

Updates an existing Limit Set with the provided data.

Syntax

```
void updateLimitSet ( LimitSet limitSet , long callerId ) throws  
EntityMandatoryException
```

Parameters

- **limitSet** –
- **callerId** –

Exceptions

- **EntityMandatoryException** –

updateLimitSetClass(long, long, long, Integer, long) method

Updates an existing limit set class with the provided information.

Syntax

```
void updateLimitSetClass ( long limitSetClassId , long limitSetId , long  
limitClassId , Integer useCaseId , long callerId ) throws  
EntityNotFoundException
```

Parameters

- **limitSetClassId** –
- **limitSetId** –
- **limitClassId** –
- **useCaseId** –
- **callerId** –

Exceptions

- **EntityNotFoundException** –

EntityType() enumeration

Enum Constant Summary

- **CUSTOMER** –
- **PI** –
- **WALLET** –
- **LIMITSET** –

IPrivateKeyLookup interface

IPrivateKeyLookup.

Syntax

```
public interface IPrivateKeyLookup
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getPrivateKey(final String) method

Find the certificate with the given name.

Syntax

```
PrivateKey getPrivateKey ( final String keyName )
```

Parameters

- **keyName** – the keyname

Returns

the certificate

Usage

the certificate

getPrivateKeyByFingerprint(final String) method

Find the private key whose certificate matches this fingerprint.

Syntax

```
PrivateKey getPrivateKeyByFingerprint ( final String fingerprint )
```

Parameters

- **fingerprint** – the certificate's fingerprint

Returns

the key

Usage

the key

IRsaPrivateKeyLogic interface

IRsaPrivateKeyLogic.

Syntax

```
public interface IRsaPrivateKeyLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

decrypt(final String) method

Decrypt the given cipher text.

Syntax

```
String decrypt ( final String cipherText )
```

Parameters

- **cipherText** – text

Returns

the decrypted

Usage

the decrypted

decrypt(final String, final String) method

Decrypt the given cipher text.

Syntax

```
String decrypt ( final String cipherText , final String alias ) throws  
EncryptionException
```

Parameters

- **cipherText** – text
- **alias** – the key alias to use

Returns

the decrypted

Exceptions

- **EncryptionException** – if decryption fails

Usage

the decrypted

IRsaPublicKeyLogic interface

IRsaPublicKeyLogic.

Syntax

```
public interface IRsaPublicKeyLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getRsaPublicKeyDetails(final String) method

Returns the details for the given RSA public key.

Syntax

```
RsaPublicKey getRsaPublicKeyDetails ( final String keyName )
```

Parameters

- **keyName** – the name of the key

Returns

the details

Usage

the details

ISystemLogic interface

The interface ISystemLogic contains various methods to work on hte system entity beans.

Syntax

```
public interface ISystemLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

createJob(Job, long) method

Creates a new (cron) Job.

Syntax

```
long createJob ( Job job ,   long callerId )
```

Parameters

- **job** –
- **callerId** –

createUseCasePrivilege(UseCasePrivilege, long) method

Creates a new Use Case Privilege with the provided data.

Syntax

```
void createUseCasePrivilege ( UseCasePrivilege useCasePrivilege ,  
long callerId ) throws EntityMandatoryException
```

Parameters

- **useCasePrivilege** –
- **callerId** –

Exceptions

- **EntityMandatoryException** –

deleteJobAndHistory(long, long) method

Completely deletes a (cron) Job and all the history entries of the Job.

Syntax

```
void deleteJobAndHistory ( long jobId ,   long callerId )
```

Parameters

- **jobId** –
- **callerId** –

deleteUseCasePrivilege(int, long) method

Deletes and existing Use Case Privilege from the system.

Syntax

`void deleteUseCasePrivilege ( int useCaseId , long callerId ) throws EntityNotFoundException`

Parameters

- **useCaseId** –
- **callerId** –

Exceptions

- **EntityNotFoundException** –

getCustomerType(int) method

Return a CustomerType entity by its ID.

Syntax

`CustomerType getCustomerType ( int customerTypeId )`

Parameters

- **customerTypeId** –

getCustomerTypes() method

Returns all Customer Types that are in the system.

Syntax

`List< CustomerType > getCustomerTypes ()`

getJobs(long) method

Returns a list of all (cron) Jobs currently in the systems.

Syntax

`List< Job > getJobs ( long callerId )`

Parameters

- **callerId** –

getLookups(String, List< String >) method

Retrieves lookup values from the DB, checking if the required privilege is in the list of supplied ones.

Syntax

```
List< LookupEntity > getLookups ( String lookupQueryId , List< String > callerPrivileges ) throws EntityNotFoundException, AccessDeniedMobiliserServiceException
```

Parameters

- **lookupQueryId** – The name of the lookup query.
- **callerPrivileges** –

Exceptions

- **EntityNotFoundException** –
- **AccessDeniedMobiliserServiceException** –

getOrgUnits() method

Returns a list of all OrgUnits in the system.

Syntax

```
List< OrgUnit > getOrgUnits ()
```

getRiskCategories() method

Returns all Risk Categories that are in the system.

Syntax

```
List< RiskCategory > getRiskCategories ()
```

getUseCasePrivilege(int) method

Returns a single Use Case Privilege entity by its ID.

Syntax

```
UseCasePrivilege getUseCasePrivilege ( int useCaseId )
```

Parameters

- **useCaseId** –

getUseCasePrivileges() method

Returns all Use Case Privileges entities that are in the system.

Syntax

```
List< UseCasePrivilege > getUseCasePrivileges ()
```

updateCustomerType(CustomerType, long) method

Updates an existing Customer Type with the provided data.

Syntax

```
void updateCustomerType ( CustomerType customerType , long callerId )  
throws EntityMandatoryException
```

Parameters

- **customerType** –
- **callerId** –

Exceptions

- **EntityMandatoryException** –

updateJob(Job, long) method

Updates an existing (cron) Job with the provided data.

Syntax

```
void updateJob ( Job job , long callerId )
```

Parameters

- **job** –
- **callerId** –

updateOrgUnit(OrgUnit, long) method

Updates the data of an existing OrgUnit.

Syntax

```
void updateOrgUnit ( OrgUnit orgUnit , long callerId ) throws  
EntityMandatoryException
```

Parameters

- **orgUnit** –
- **callerId** –

Exceptions

- **EntityMandatoryException** –

updateRiskCategory(RiskCategory, long) method

Updates an existing Risk Category with the provided data.

Syntax

```
void updateRiskCategory ( RiskCategory riskCategory , long callerId )  
throws EntityMandatoryException
```

Parameters

- **riskCategory** –
- **callerId** –

Exceptions

- **EntityMandatoryException** –

updateUseCasePrivilege(UseCasePrivilege, long) method

Updates an existing Use Case Privilege with the provided data.

Syntax

```
void updateUseCasePrivilege ( UseCasePrivilege useCasePrivilege ,  
long callerId ) throws EntityMandatoryException,  
EntityNotFoundException
```

Parameters

- **useCasePrivilege** –
- **callerId** –

Exceptions

- **EntityMandatoryException** –
- **EntityNotFoundException** –

ITransactionRestrictionLogic interface

The interface ITransactionRestrictionLogic contains mainly CRUD operations to work on transaction restrictions and the directly related entities.

Syntax

```
public interface ITransactionRestrictionLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

createIndividualRestriction(IndividualRestriction, long) method

Creates an individual restriction (between a payer and a payee).

Syntax

```
long createIndividualRestriction ( IndividualRestriction  
restriction , long callerId )
```

Parameters

- **restriction** –
- **callerId** –

createRestriction(Restriction, List< String >, List< Integer >, List< String >, List< Integer >, List< Integer >, List< RestrictionRule >, long) method

Creates a new Restriction and registers it with the provided OrgUnits and Risk Categories.

Syntax

```
long createRestriction ( Restriction restriction , List< String >  
forOrgUnits , List< Integer > forRiskCategories , List< String >  
onOrgUnits , List< Integer > onRiskCategories , List< Integer >  
onUseCases , List< RestrictionRule > restrictionRules , long callerId )
```

Parameters

- **restriction** –
- **forOrgUnits** –
- **forRiskCategories** –
- **onOrgUnits** –
- **onRiskCategories** –

- **onUseCases** –
- **restrictionRules** –
- **callerId** –

createRestrictionsGroup(RestrictionsGroup, long) method

Creates a new Restriction Group.

Syntax

```
void createRestrictionsGroup ( RestrictionsGroup restrictionsGroup ,  
long callerId )
```

Parameters

- **restrictionsGroup** –
- **callerId** –

deleteIndividualRestriction(long, long) method

Deletes an individual Restriction.

Syntax

```
void deleteIndividualRestriction ( long restrictionId , long callerId )
```

Parameters

- **restrictionId** –
- **callerId** –

deleteRestriction(long, long) method

Deletes a restriction including all references to it.

Syntax

```
void deleteRestriction ( long restrictionId , long callerId )
```

Parameters

- **restrictionId** –
- **callerId** –

deleteRestrictionsGroup(int, long) method

Deletes a restriction group including all attached restrictions and references to it - use with care.

Syntax

```
void deleteRestrictionsGroup ( int restrictionGroupId , long callerId )
```


Parameters

- **restrictionGroupId** –
- **callerId** –

getIndividualRestrictions(Long, Long, Integer, long) method

Returns the existing Individual Restrictions for the given payer and payee and an optional use case.

Syntax

```
List< IndividualRestriction > getIndividualRestrictions ( Long
payer , Long payee , Integer useCase , long callerId )
```

Parameters

- **payer** –
- **payee** –
- **useCase** – this is optional.
- **callerId** –

getRestriction(long, long) method

Returns the restriction entity by its ID.

Syntax

```
RestrictionResponse getRestriction ( long restrictionId , long
callerId )
```

Parameters

- **restrictionId** –
- **callerId** –

getRestrictionsGroups(long) method

Returns all Restriction Groups.

Syntax

```
List< RestrictionsGroupResponse > getRestrictionsGroups ( long
callerId )
```

Parameters

- **callerId** –

swapRestriction(long, long, long) method

Swaps the priority (order) of two existing restrictions.

Syntax

```
void swapRestriction ( long id1 , long id2 , long callerId )
```

Parameters

- **id1** –
- **id2** –
- **callerId** –

updateIndividualRestriction(long, IndividualRestriction, long) method

Updates an existing individual Restriction with the provided data.

Syntax

```
void updateIndividualRestriction ( long restrictionId ,  
IndividualRestriction restriction , long callerId )
```

Parameters

- **restrictionId** –
- **restriction** –
- **callerId** –

updateRestriction(long, Restriction, List< String >, List< Integer >, List< String >, List< Integer >, List< Integer >, List< RestrictionRule >, long) method

Updates an existing Restriction with the provided data.

Syntax

```
void updateRestriction ( long restrictionId , Restriction restriction ,  
List< String > forOrgUnits , List< Integer > forRiskCategories , List<  
String > onOrgUnits , List< Integer > onRiskCategories , List< Integer  
> onUseCases , List< RestrictionRule > restrictionRules , long callerId )
```

Parameters

- **restrictionId** –
- **restriction** –
- **forOrgUnits** –
- **forRiskCategories** –

- **onOrgUnits** –
- **onRiskCategories** –
- **onUseCases** –
- **restrictionRules** –
- **callerId** –

ITxntemplLogic interface

The interface ITxntemplLogic contains CRUD type methods to work on Transaction Templates.

Syntax

```
public interface ITxntemplLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

createTransactionTemplateMapping(com.sybase365.mobiliser.money.persistence.model.system.TransactionTemplateMapping, long) method

Creates a new Transaction Template Mapping with the provided data.

Syntax

```
long createTransactionTemplateMapping  
( com.sybase365.mobiliser.money.persistence.model.system.TransactionTemplateMapping bean , long callerId )
```

Parameters

- **bean** –
- **callerId** –

deleteTransactionTemplateMapping(long, long) method

Deletes an existing Transaction Template Mapping from the DB.

Syntax

```
void deleteTransactionTemplateMapping ( long id , long callerId )
```

Parameters

- **id** –
- **callerId** –

getAllTransactionTemplateMappings(long) method

Returns all Transaction Template Mapping entities.

Syntax

```
java.util.List<  
com.sybase365.mobiliser.money.persistence.model.system.TransactionTemplateMapping > getAllTransactionTemplateMappings  
( long callerId )
```

Parameters

- **callerId** –

getTransactionTemplateMapping(long, long) method

Returns an existing Transaction Template Mapping by its ID.

Syntax

```
com.sybase365.mobiliser.money.persistence.model.system.TransactionTemplateMapping getTransactionTemplateMapping ( long id ,  
long callerId )
```

Parameters

- **id** –
- **callerId** –

getTransactionTemplateMappingsByOrgUnitAndUseCaseId(java.lang.String, int, long) method

Returns all Transaction Template Mapping entities for the given Use Case ID and OrgUnit.

Syntax

```
java.util.List<  
com.sybase365.mobiliser.money.persistence.model.system.TransactionTemplateMapping >  
getTransactionTemplateMappingsByOrgUnitAndUseCaseId  
( java.lang.String orgunit , int useCaseId , long callerId )
```

Parameters

- **orgunit** –
- **useCaseId** –
- **callerId** –

getTransactionTemplateMappingsByUseCaseId(int, long) method

Returns all Transaction Template Mapping entities for the given Use Case ID.

Syntax

```
java.util.List<
com.sybase365.mobiliser.money.persistence.model.system.TransactionTemplateMapping >
getTransactionTemplateMappingsByUseCaseId ( int useCaseId , long callerId )
```

Parameters

- **useCaseId** –
- **callerId** –

updateTransactionTemplateMapping(com.sybase365.mobiliser.money.persistence.model.system.TransactionTemplateMapping, long) method

Updates an existing Transaction Template Mapping.

Syntax

```
void updateTransactionTemplateMapping
( com.sybase365.mobiliser.money.persistence.model.system.TransactionTemplateMapping bean , long callerId )
```

Parameters

- **bean** –
- **callerId** –

LockingException class

The LockingException is used when a lock on the Batchrun cannot be acquired.

Syntax

```
public class LockingException
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

LockingException(String) constructor

Syntax

`LockingException ( String msg )`

getErrorCode() method

Syntax

`int getErrorCode ()`

transaction package

Members

All public members of the transaction package.

- **actions package** –
- **authcancel package** –
- **authcont package** –
- **authorisation package** –
- **capture package** –
- **capturecancel package** –
- **checkinvoice package** –
- **continueinvoice package** –
- **demandforpayment package** –
- **demarcation package** –
- **flow package** –
- **payinvoice package** –
- **preauth package** –
- **preauthcont package** –
- **process package** –
- **transcancel package** –
- **webcont package** –
- **webstart package** –
- **IAmountLogic interface** – Simple abstract to allow for calculating the amounts pertaining to a given SubTransaction.
- **IAuthorisationLogic interface** – Logic for authorizing transactions.
- **ICancelLogic interface** – Logic for canceling transactions.
- **ICaptureLogic interface** – Logic for capturing transactions.
- **ICommissionClearingLogic interface** – Logic for commission clearing.
- **ICommissionLogic interface** – Logic for commission calculation and handling.

- **IContinueLogic interface** – Logic for continuing a transaction after authentication.
- **IDelegationLogic interface** – Logic handling the delegation between actor and caller.
- **IEventLogic interface** – Logic to generate transaction events based on a given transaction/ system ID.
- **IExchangeRateLogic interface** – Logic for exchange rate handling and currency conversion.
- **IExpirationLogic interface** – Logic for transaction expiration handling.
- **IInitFollowUpTransaction< T extends SubTransaction > interface** – Logic for creating follow-up subtransactions.
- **IInitTransactionLogic interface** – Logic for initializing the different transaction types.
- **IInitWebTransaction interface** – Logic for initializing web transactions.
- **IInvoiceLogic interface** – Logic for Invoice Transactions.
- **ILimitAccess interface** – Allows clients to access the LimitSums configured on Mobiliser objects.
- **ILimitLogic interface** – Logic for limit handling.
- **IParticipantLogic interface** – Logic for handling transaction participant verification.
- **IRedirectUrlLookup interface** – Interface for retrieving the url redirects for web transactions.
- **IRestrictionLogic interface** – Logic for restriction checks.
- **ITransactionCRUDLogic interface** – Logic for transaction CRUD operations.
- **ITransactionLogic interface** – Miscellaneous transaction logic.
- **IUseCaseLogic interface** – Logic dealing with use cases.
- **IVoucherLogic interface** – Logic for voucher handling.
- **IWebTxnLogic interface** – Logic for web transaction handling.
- **TransactionConfiguration class** – Configuration bean for generic financial transaction properties.

actions package

Members

All public members of the actions package.

- **IAdjustBalancesAction interface** –
- **IAuthenticationAction interface** –
- **IAuthorisationAction interface** –
- **ICancelAction interface** – Common actions for canceling initial transactions.
- **ICaptureAction interface** –
- **ICommissionAction interface** –
- **IInitAction interface** –
- **IInvoiceAction interface** –
- **ILimitAction interface** – Common actions for processing limits.

- **IMakerCheckerAction< Req extends IMoneyRequest, Res extends ITransactionResponse > interface** –
- **IRestrictionAction interface** – Common actions for processing transaction restrictions.
- **ITransactionEventAction interface** –
- **IVoucherAction interface** –
- **IWebActions interface** – Common actions for processing web transactions.

IAdjustBalancesAction interface

Syntax

```
public interface IAdjustBalancesAction
```

Usage

Common actions for adjusting balances on limits during transactions. This action does not do any limit validation, but simply adjusts the balances based on follow-up sub transactions (if amounts differ from original authorization amount and in case of cancellations).

The reference implementation defines transaction demarcation on the action level and does not rely the client code starting a transaction themselves.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

adjustBalances(final long, final Long) method

Syntax

```
void adjustBalances ( final long subTransactionId , final Long actorId )  
throws MobiliserServiceException, DataAccessException,  
TransactionException
```

Parameters

- **subTransactionId** – the subtransaction's id
- **actorId** – the actorId (authenticated user)

Exceptions

- **MobiliserServiceException** – if adjusting the balances fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage

- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

Adjusts the balances for the given subtransaction. This does not perform any limit checking, but only adjusts the limit sums based on transaction amounts. This method should hence only be used for follow-up transactions, not for the authorization step.

Notes on transactional semantics:

This operation will run inside its own database transaction. Any failure will trigger a rollback. Exceptions will

always

cause the

SubTransaction

to be marked as failed in a fresh database transaction.

readjustBalancesAfterFailedAuthorisation(final long, final Long) method

Syntax

```
void readjustBalancesAfterFailedAuthorisation ( final long
authId ,   final Long actorId ) throws MobiliserServiceException,
DataAccessException, TransactionException
```

Parameters

- **authId** – the authorisation id
- **actorId** – the actorId (authenticated user)

Exceptions

- **MobiliserServiceException** – if adjusting the balances fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

Re-adjust, i.e. roll-back previous changes, of the balances after an authorization failed.

Notes on transactional semantics:

This operation will run inside its own database transaction. Any failure will trigger a rollback. Since the authorisation will have already failed, the error code will

NOT

be set on the authorisation transaction to avoid clobbering the old value.

IAuthenticationAction interface

Syntax

```
public interface IAuthenticationAction
```

Usage

Common actions for dealing with authentication of transactions. This includes determining the authentication to apply as well as performing authentications, i.e. triggering authentications through external channels.

The reference implementation defines transaction demarcation on the action level and does not rely the client code starting a transaction themselves.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

authenticate(final Long, final AuthenticationMethodComputation, final ITransactionCredential, final Long) method

Syntax

```
ConsumerAuthenticationResult authenticate ( final Long authId ,  
final AuthenticationMethodComputation computeRules , final  
ITransactionCredential credential , final Long actorId ) throws  
MobiliserServiceException, DataAccessException,  
TransactionException
```

Parameters

- **authId** – the id of the authorisation transaction to authenticate
- **computeRules** – the authentication rules
- **credential** – the credential to use for authentication
- **actorId** – the actorId (authenticated user)

Returns

the result of the authentication

Exceptions

- **MobiliserServiceException** – if authentication fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

Authenticate the

Authorisation

based on the previously computed authentication method.

Notes on transactional semantics:

If a business logic exception occurs, a

MobiliserServiceException

will be thrown and the

Authorisation

and the main

Transaction

will be marked with the error code from the exception and the database transaction committed.

Any other exception will trigger a rollback of the database transaction followed by a fresh transaction inside of which the

Authorisation

and the main

Transaction

will be marked with an error code.

the result of the authentication

computeAuthentication(final Long, final Long) method

Syntax

AuthenticationMethodComputation computeAuthentication (final Long *authId* , final Long *actorId*) throws MobiliserServiceException, DataAccessException, TransactionException

Parameters

- **authId** – the authorisation transaction id
- **actorId** – the actorId (authenticated user)

Returns

the result of the computation

Exceptions

- **MobiliserServiceException** – if computing the authentication methods fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

Compute the authentication methods for the given authorisation transaction. This will return the methods for the payer and the payee. When this method returns, the given authentication methods will have been set and saved on the main

Transaction

.

Notes on transactional semantics:

If a business logic exception occurs, a

MobiliserServiceException

will be thrown and the

Authorisation

will be marked with the error code from the exception and the database transaction committed.

Any other exception will trigger a rollback of the database transaction followed by a fresh transaction inside of which the

Authorisation

transaction will be marked with an error code.

the result of the computation

continueAuthentication(final ITransaction, final ITransactionCredential, final CallerInformation, final IAuthenticationContinueResponse) method

Syntax

ContinueParams continueAuthentication (final ITransaction *referencedTxn* , final ITransactionCredential *credential* , final CallerInformation *actor* , final IAuthenticationContinueResponse *response*) throws MobiliserServiceException, DataAccessException, TransactionException

Parameters

- **referencedTxn** – the referenced transaction
- **credential** – the credential to use for authentication
- **actor** – the actor for this operation
- **actorId** – the caller id
- **response** – the response object to fill

Returns

a ContinueParams object containing the authentication id and the details of the main Transaction

Exceptions

- **MobiliserServiceException** – if continuing the authentication methods fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

Continue the authentication for the given transaction. This is required if a transaction has been paused and is waiting for authentication credentials to be continued.

Notes on transactional semantics:

If a business logic exception occurs, a

MobiliserServiceException

will be thrown and the

Authorisation

will be marked with the error code from the exception and the database transaction committed.
If the

MobiliserServiceException#getErrorCode()

is anything

other

than

StatusCodes#ERROR_AUTHENTICATION_INVALID_CREDENTIALS

, the main

Transaction

will additionally be marked as failed. Any other exception will trigger a rollback of the database transaction followed by a fresh transaction inside of which the

Authorisation

transaction will be marked with an error code.

a ContinueParams object containing the authentication id and the details of the main Transaction

initContinueAuthentication(final ITransaction, final ITransactionCredential, final CallerInformation, final IAuthenticationContinueResponse) method
Initializes (returns the referenced transaction) the authenticationContinue.

Syntax

ContinueParams initContinueAuthentication (final ITransaction *referencedTxn* , final ITransactionCredential *credential* , final CallerInformation *caller* , final IAuthenticationContinueResponse *response*) throws MobiliserServiceException, DataAccessException, TransactionException

Parameters

- **referencedTxn** – the referenced transaction
- **credential** – the credential to use for authentication
- **caller** – the actor for this operation
- **response** – the response object to fill

Returns

a ContinueParams object containing the authentication id and the details of the main Transaction

Exceptions

- **MobiliserServiceException** – if continuing the authentication methods fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

a `ContinueParams` object containing the authentication id and the details of the main `Transaction`

IAuthorisationAction interface

Syntax

```
public interface IAuthorisationAction
```

Usage

This interface defines common actions for authorizing transactions.

The reference implementation defines transaction demarcation on the action level and does not rely the client code starting a transaction themselves.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

doAuthorisation(final Long, final List< IKeyValue >, final List< IKeyValue >, final Long)
method

Syntax

```
boolean doAuthorisation ( final Long authId , final List<
IKeyValue > unstructuredDataRequest , final List< IKeyValue >
unstructuredDataResponse , final Long actorId ) throws
MobiliserServiceException, DataAccessException,
TransactionException
```

Parameters

- **authId** – the id of the transaction to authorize
- **unstructuredDataRequest** – the unstructured data coming from the original request

- **unstructuredDataResponse** – this list can be filled with information to be returned with the response
- **actorId** – the actorId (authenticated user)

Returns

true if this authorization should be auto captured

Exceptions

- **MobiliserServiceException** – if authorizing the transaction fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

Perform the actual authorization for the given authorization sub-transaction.

Notes on transactional semantics:

If a business logic exception occurs, a

MobiliserServiceException

will be thrown and the database transaction committed. If the the cause stems from the underlying payment handler, both the main

Transaction

and the

Authorisation

transaction will be failed. Any other exception will trigger a rollback of the database transaction followed by a fresh transaction inside of which the

Authorisation

transaction will be marked with an error code.

true if this authorization should be auto captured

doAuthorisationCancel(final long, final List< IKeyValue >, final List< IKeyValue >, final Long, final Integer) method

Syntax

```
void doAuthorisationCancel ( final long authCancelId ,  final List< IKeyValue > unstructuredDataRequest ,  final List< IKeyValue >
```


unstructuredDataResponse , final Long *actorId* , final Integer *mainTxnErrorCode*) throws MobiliserServiceException, DataAccessException, TransactionException

Parameters

- **authCancelId** – the id of the transaction whose authorization should be cancelled
- **unstructuredDataRequest** – the unstructured data coming from the original request
- **unstructuredDataResponse** – this list can be filled with information to be returned with the response
- **actorId** – the actorId (authenticated user)
- **mainTxnErrorCode** – the error code with which the main transaction must be failed. If null, the main transaction must not be failed.

Exceptions

- **MobiliserServiceException** – if canceling the authorization for the transaction fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

Perform the actual authorization cancel for the given authorization transaction.

Notes on transactional semantics:

If a business logic exception occurs, a

MobiliserServiceException

will be thrown and the database transaction committed. If the the cause stems from the underlying payment handler, both the main

Transaction

and the

AuthorisationCancel

transaction will be failed. Any other exception will trigger a rollback of the database transaction followed by a fresh transaction inside of which the

AuthorisationCancel

transaction will be marked with an error code.

doPreAuthorisation(final Long, final List< IKeyValue >, final List< IKeyValue >, final Long) method

Syntax

```
void doPreAuthorisation ( final Long authId , final List<
IKeyValue > unstructuredDataRequest , final List< IKeyValue >
unstructuredDataResponse , final Long actorId ) throws
MobiliserServiceException, DataAccessException,
TransactionException
```

Parameters

- **authId** – the id of the transaction to pre-authorize
- **unstructuredDataRequest** – the unstructured data coming from the original request
- **unstructuredDataResponse** – this list can be filled with information to be returned with the response
- **actorId** – the actorId (authenticated user)

Exceptions

- **MobiliserServiceException** – if pre-authorizing the transaction fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

Perform the actual pre-authorization for the given authorization transaction in initial state

Notes on transactional semantics:

If a business logic exception occurs, a

MobiliserServiceException

will be thrown and the database transaction committed. If the the cause stems from the underlying payment handler, both the main

Transaction

and the

Authorisation

transaction will be failed. Any other exception will trigger a rollback of the database transaction followed by a fresh transaction inside of which the

Authorisation

transaction will be marked with an error code.

fillAuthenticationContinueResponse(final Long, final IAuthenticationContinueResponse, final Long) method

Syntax

```
void fillAuthenticationContinueResponse ( final Long txnId, final
IAuthenticationContinueResponse response, final Long actorId )
```

Parameters

- **txnId** – the transaction id of the authorization subtransaction to use
- **response** – the response that will be filled
- **actorId** – actor id of current user

Usage

Fill the authentication continue response with data from the database.

fillAuthorisationResponse(final Long, final IAuthorisationResponse, final Long) method

Syntax

```
void fillAuthorisationResponse ( final Long txnId, final
IAuthorisationResponse response, final Long actorId )
```

Parameters

- **txnId** – the transaction id of the authorization subtransaction to use
- **response** – the response that will be filled
- **actorId** – actor id of current user

Usage

Fill the authorisation response with data from the database.

*fillPreAuthorisationContinueResponse(final Long, final
IPreAuthorisationContinueResponse, final Long) method*

Syntax

```
void fillPreAuthorisationContinueResponse ( final Long txnId ,  
final IPreAuthorisationContinueResponse response , final Long  
actorId )
```

Parameters

- **txnId** – the transaction id of the authorization subtransaction to use
- **response** – the response that will be filled
- **actorId** – actor id of current user

Usage

Fill the pre-authorization continue response with data from the database.

*fillPreAuthorisationResponse(final Long, final IPreAuthorisationResponse, final
Long) method*

Syntax

```
void fillPreAuthorisationResponse ( final Long authId , final  
IPreAuthorisationResponse response , final Long actorId )
```

Parameters

- **authId** – the transaction id of the authorization subtransaction to use
- **response** – the response that will be filled
- **actorId** – actor id of current user

Usage

Fill the pre-authorization response with data from the database.

*fillWebContinueResponse(final Long, final IWebContinueResponse, final Long)
method*

Syntax

```
void fillWebContinueResponse ( final Long txnId , final  
IWebContinueResponse response , final Long actorId )
```

Parameters

- **txnId** – the transaction id of the authorization subtransaction to use
- **response** – the response that will be filled
- **actorId** – actor id of current user

Usage

Fill the web-continue response with data from the database.

KEY_AUTH_CODE variable

Syntax

```
final String KEY_AUTH_CODE
```

ICancelAction interface

Common actions for canceling initial transactions.

Syntax

```
public interface ICancelAction
```

Usage

The reference implementation defines transaction demarcation on the action level and does not rely the client code starting a transaction themselves.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

cancelTransaction(final ITransaction, final Integer, final Long) method

Cancel a transaction by setting an error code to it.

Syntax

```
void cancelTransaction ( final ITransaction referencedTransaction ,  
final Integer errorCode , final Long actorId ) throws  
MobiliserServiceException, DataAccessException,  
TransactionException
```

Parameters

- **referencedTransaction** – the transaction that must be cancelled.
- **errorCode** – the error code to set with the transaction.
- **actorId** – the current actor

Exceptions

- **MobiliserServiceException** – if canceling the transaction fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

No further processing is applied, if you need to cancel a transaction that is already authorized or captured, use the respective cancellation methods.

ICaptureAction interface

Syntax

```
public interface ICaptureAction
```

Usage

Common actions for capturing transactions.

The reference implementation defines transaction demarcation on the action level and does not rely the client code starting a transaction themselves.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

doCapture(final long, final Long, final List< IKeyValue >, final List< IKeyValue >, final Long) method

Syntax

```
void doCapture ( final long captureId , final Long authorisationId ,  
final List< IKeyValue > unstructuredDataRequest , final List<  
IKeyValue > unstructuredDataResponse , final Long actorId ) throws  
MobiliserServiceException, DataAccessException,  
TransactionException
```

Parameters

- **captureId** – the id of the transaction to capture

- **authorisationId** – the authorization that will be captured, this should be passed if available for performance reasons
- **unstructuredDataRequest** – the unstructured data coming from the original request
- **unstructuredDataResponse** – this list can be filled with information to be returned with the response
- **actorId** – the actorId (authenticated user)

Exceptions

- **MobiliserServiceException** – if the capturing the transaction fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

Perform the actual capture for the given capture transaction.

Notes on transactional semantics:

If a business logic exception occurs, a

`MobiliserServiceException`

will be thrown and the database transaction committed. If the the cause stems from the underlying payment handler, both the main

`Transaction`

and the

`Capture`

transaction will be failed. Any other exception will trigger a rollback of the database transaction followed by a fresh transaction inside of which the

`Capture`

transaction will be marked with an error code.

doCaptureCancel(final long, final List< IKeyValue >, final List< IKeyValue >, final Long) method

Syntax

```
void doCaptureCancel ( final long captureCancelId , final List<
IKeyValue > unstructuredDataRequest , final List< IKeyValue >
unstructuredDataResponse , final Long actorId )
```

Parameters

- **captureCancelId** – the id of the capture cancel transaction to cancel
- **unstructuredDataRequest** – the unstructured data coming from the original request
- **unstructuredDataResponse** – this list can be filled with information to be returned with the response
- **actorId** – the actorId (authenticated user)

Exceptions

- **MobiliserServiceException** – if the capturing canceling the transaction fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

Perform the actual capture cancel for the given capture transaction.

Notes on transactional semantics:

If a business logic exception occurs, a

`MobiliserServiceException`

will be thrown and the database transaction committed. If the the cause stems from the underlying payment handler, both the main

`Transaction`

and the

`CaptureCancel`

transaction will be failed. Any other exception will trigger a rollback of the database transaction followed by a fresh transaction inside of which the

`CaptureCancel`

transaction will be marked with an error code.

isAuthenticated(long) method

Helper function to retrieve the Transaction status is Authenticated.

Syntax

`boolean isAuthenticated ( long systemId )`

Parameters

- **systemId** – the transaction id to check

Returns

true if the transaction status is authenticated

Usage

true if the transaction status is authenticated

isAutoCapture(final long) method

Helper function to retrieve if the given txn is an auto capture transaction.

Syntax

```
boolean isAutoCapture ( final long systemId )
```

Parameters

- **systemId** – the transaction id to check

Returns

true if the transaction is a auto-capture transaction

Usage

true if the transaction is a auto-capture transaction

ICommissionAction interface*Syntax*

```
public interface ICommissionAction
```

Usage

Common actions for commission handling

The reference implementation defines transaction demarcation on the action level and does not rely the client code starting a transaction themselves.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

commissionCheck(final Long, final Long) method

Syntax

```
void commissionCheck ( final Long authorisationId, final Long actorId )  
throws MobiliserServiceException, DataAccessException,  
TransactionException
```

Parameters

- **authorisationId** – the subtransaction's id
- **actorId** – the actorId (authenticated user)

Exceptions

- **MobiliserServiceException** – if checking of the commissions fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

Perform a Commission check for the given

SubTransaction

.

Notes on transactional semantics:

This operation will run inside its own database transaction. Any failure will trigger a rollback and also if the user requests

`persist=false`

the changes will also be rolled back. Exception will

always

cause the

SubTransaction

and the main

Transaction

to be marked as failed in a fresh database transaction.

commissionRollback(final Long, final Long, final Long) method

Syntax

```
void commissionRollback ( final Long mainTransactionId , final Long
subTransactionId , final Long actorId) throws MobiliserServiceException,
DataAccessException, TransactionException
```

Parameters

- **mainTransactionId** – the main txn id
- **subTransactionId** – the canceling subtransaction's id
- **actorId** – the actorId (authenticated user)

Exceptions

- **MobiliserServiceException** – if the rollback fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

Performs a "rollback" for the given commission. The rollback is done by booking negative commissions for the existing positive commission, this should only be used as last resort, for example in case of a capturecancel

Notes on transactional semantics:

This operation will run inside its own database transaction. Any failure will trigger a rollback and also if the user requests

`persist=false`

the changes will also be rolled back. Exception will

always

cause the

`SubTransaction`

and the main

`Transaction`

to be marked as failed in a fresh database transaction.

IInitAction interface

Syntax

```
public interface IInitAction
```

Usage

Common actions for initializing transactions.

The reference implementation defines transaction demarcation on the action level and does not rely the client code starting a transaction themselves.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CreateMoneyTransactionResult class

Syntax

```
public class CreateMoneyTransactionResult
```

Usage

This bean holds the result of creating the initial money transaction.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CreateMoneyTransactionResult(final String, final Long, final String, final Long) constructor

Creates a new result bean.

Syntax

```
CreateMoneyTransactionResult ( final String authCode , final Long  
authId , final String orderId , final Long txnId )
```

Parameters

- **authCode** –
- **authId** –
- **orderId** –
- **txnId** –

CreateMoneyTransactionResult(final String, final Long, final String, final Long, final boolean) constructor

Creates a new result bean.

Syntax

```
CreateMoneyTransactionResult ( final String authCode , final Long authId , final String orderId , final Long txnId , final boolean autoCapture )
```

Parameters

- **authCode** –
- **authId** –
- **orderId** –
- **txnId** –
- **autoCapture** –

authCode variable

the authorization code of the main transaction.

Syntax

```
final String authCode
```

authId variable

the id of the authorization transaction.

Syntax

```
final Long authId
```

autocapture variable

indicated if the transaction must be auto-captured.

Syntax

```
boolean autocapture
```

orderId variable

the order id of the transaction.

Syntax

```
final String orderId
```

txnId variable

the main transaction id.

Syntax

```
final Long txnId
```

createMoneyTransactions(final IGenericAuthorisationRequest, final CallerInformation) method

Syntax

```
CreateMoneyTransactionResult createMoneyTransactions ( final  
IGenericAuthorisationRequest request , final CallerInformation  
caller ) throws MobiliserServiceException, DataAccessException,  
TransactionException
```

Parameters

- **request** – the authorization request
- **caller** – the actor (authenticated user)

Returns

a `CreateMoneyTransactionResult` object containing the details of the created transactions

Exceptions

- **MobiliserServiceException** – if the creation of the transactions fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

Creates the

Transaction

and

Authorisation

for the given

`IGenericAuthorisationRequest`

Notes on transactional semantics:

This operation will run inside its own database transaction. Any failure will trigger a rollback.

a `CreateMoneyTransactionResult` object containing the details of the created transactions

getMoneyTransactions(IWebContinueRequest, Long) method

Syntax

`CreateMoneyTransactionResult getMoneyTransactions`
 (`IWebContinueRequest request` , `Long actorId`) throws
`MobiliserServiceException`, `DataAccessException`,
`TransactionException`

Parameters

- **request** – the web-continue request
- **actorId** – the actor (authenticated user)

Returns

a `CreateMoneyTransactionResult` holding the transaction that must be continued.

Exceptions

- **MobiliserServiceException** – if initializing the transaction fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

Loads the transaction to be continued for the web-continue request.

Notes on transactional semantics:

If a business logic exception occurs, a

`MobiliserServiceException`

will be thrown which will trigger a rollback. Any other exception will also trigger a rollback of the database transaction.

a `CreateMoneyTransactionResult` holding the transaction that must be continued.

initAuthorisation(final Long, final IGenericAuthorisationRequest, final CallerInformation) method

Syntax

```
void initAuthorisation ( final Long authorisationId , final  
IGenericAuthorisationRequest request , final CallerInformation  
caller ) throws MobiliserServiceException, DataAccessException,  
TransactionException
```

Parameters

- **authorisationId** – the id of the authorization transaction to initialize
- **request** – the authorisation request
- **caller** – the actor (authenticated user)

Exceptions

- **MobiliserServiceException** – if initializing the transaction fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

Actually initializes the

Authorisation

transaction in the database with the values from the request, performing various checks in the process.

Notes on transactional semantics:

If a business logic exception occurs, a

MobiliserServiceException

will be thrown and both transactions will be marked with the error code from the exception and the database transaction committed. Any other exception will trigger a rollback of the database transaction followed by a fresh transaction inside of which the

Authorisation

transaction will be marked with an error code.

initAuthorisation(final Long, final IWebContinueRequest, final CallerInformation)
method

Syntax

```
void initAuthorisation ( final Long authId , final
IWebContinueRequest request , final CallerInformation caller )
```

Parameters

- **authId** – the id of the authorization transaction to initialize
- **request** – the web-continue request
- **caller** – the actor (authenticated user)

Exceptions

- **MobiliserServiceException** – if initializing the transaction fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

Actually initializes the

Authorisation

transaction in the database with the values from the request, performing various checks in the process.

Notes on transactional semantics:

If a business logic exception occurs, a

MobiliserServiceException

will be thrown and both transactions will be marked with the error code from the exception and the database transaction committed. Any other exception will trigger a rollback of the database transaction followed by a fresh transaction inside of which the

Authorisation

transaction will be marked with an error code.

initAuthorisationCancel(final *IAuthorisationCancelRequest*, final *CallerInformation*)
method

Syntax

InitTransactionResult initAuthorisationCancel (final
IAuthorisationCancelRequest *request* , final CallerInformation
caller) throws MobiliserServiceException, DataAccessException,
TransactionException

Parameters

- **request** – the authorisationcancel request
- **caller** – the actor (authenticated user)

Returns

a InitTransactionResult holding the id of the newly created
AuthorisationCancel transaction along with the parameters from the main
Transaction

Exceptions

- **MobiliserServiceException** – if initializing the transaction fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

Creates and initializes the

AuthorisationCancel

transaction in the database with the values from the request, performing various checks in the process.

Notes on transactional semantics:

If a business logic exception occurs, a

MobiliserServiceException

will be thrown which will trigger a rollback. Any other exception will also trigger a rollback of the database transaction.

a `InitTransactionResult` holding the id of the newly created
`AuthorisationCancel` transaction along with the parameters from the main
`Transaction`

initCapture(final ICaptureRequest, final CallerInformation) method

Syntax

`InitTransactionResult initCapture (final ICaptureRequest request ,
final CallerInformation caller)` throws `MobiliserServiceException`,
`DataAccessException`, `TransactionException`

Parameters

- **request** – the capture request
- **caller** – the actor (authenticated user)

Returns

a `InitTransactionResult` holding the id of the newly created `Capture` transaction
along with the parameters from the main `Transaction`

Exceptions

- **MobiliserServiceException** – if initializing the transaction fails for some reason
(business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading
persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to
persistent storage

Usage

Creates and initializes the

`Capture`

transaction in the database with the values from the request, performing various checks in the
process.

Notes on transactional semantics:

If a business logic exception occurs, a

`MobiliserServiceException`

will be thrown which will trigger a rollback. Any other exception will also trigger a rollback of
the database transaction.

a `InitTransactionResult` holding the id of the newly created `Capture` transaction along with the parameters from the main `Transaction`

`initCaptureCancel(final ICaptureCancelRequest, final CallerInformation)` method

Syntax

```
InitTransactionResult initCaptureCancel ( final  
ICaptureCancelRequest request , final CallerInformation caller )  
throws MobiliserServiceException, DataAccessException,  
TransactionException
```

Parameters

- **request** – the capture-cancel request
- **caller** – the actor (authenticated user)

Returns

a `InitTransactionResult` holding the id of the newly created `CaptureCancel` transaction along with the parameters from the main `Transaction`

Exceptions

- **MobiliserServiceException** – if initializing the transaction fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

Creates and initializes the

`CaptureCancel`

transaction in the database with the values from the request, performing various checks in the process.

Notes on transactional semantics:

If a business logic exception occurs, a

`MobiliserServiceException`

will be thrown which will trigger a rollback. Any other exception will also trigger a rollback of the database transaction.

a `InitTransactionResult` holding the id of the newly created `CaptureCancel` transaction along with the parameters from the main `Transaction`

initFollowUpAuthorisationCancel(final long, final CallerInformation) method

Syntax

```
long initFollowUpAuthorisationCancel ( final long authorisationId ,
final CallerInformation caller ) throws MobiliserServiceException,
DataAccessException, TransactionException
```

Parameters

- **authorisationId** – the id of the authorization transaction that should be cancelled
- **caller** – the actor (authenticated user)

Returns

the id of the newly created authorization cancel transaction

Exceptions

- **MobiliserServiceException** – if initializing the transaction fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicated / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

Creates and initializes the

`AuthorisationCancel`

transaction in the database with the values from the

`Authorisation`

, performing various checks in the process. This action can be used if an authorization cancel must be performed based on some internal processes, and not based on an external service call.

Notes on transactional semantics:

If a business logic exception occurs, a

`MobiliserServiceException`

will be thrown which will trigger a rollback. Any other exception will also trigger a rollback of the database transaction.

the id of the newly created authorization cancel transaction

initFollowUpCapture(final Long, final CallerInformation) method

Syntax

long initFollowUpCapture (final Long *authorisationId* , final CallerInformation *caller*) throws MobiliserServiceException, DataAccessException, TransactionException

Parameters

- **authorisationId** – the id of the authorization transaction to used to initialize the capture transaction
- **caller** – the actor (authenticated user)

Returns

the id of the newly created capture transaction

Exceptions

- **MobiliserServiceException** – if initializing the transaction fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

Creates and initializes the

Capture

transaction in the database with the values from the

Authorisation

, performing various checks in the process. This action can be used if a capture must be performed based on some internal processes, and not based on an external service call, e.g. for auto-capture.

Notes on transactional semantics:

If a business logic exception occurs, a

MobiliserServiceException

will be thrown which will trigger a rollback. Any other exception will also trigger a rollback of the database transaction.

the id of the newly created capture transaction

initPreAuthorisationContinue(final *IPreAuthorisationContinueRequest*, final *CallerInformation*) method

Syntax

```
InitTransactionResult initPreAuthorisationContinue ( final
IPreAuthorisationContinueRequest request , final
CallerInformation caller) throws MobiliserServiceException,
DataAccessException, TransactionException
```

Parameters

- **request** – the pre-authorization request
- **caller** – the actor (authenticated user)

Returns

a `InitTransactionResult` holding the id of the Authorisation transaction along with the parameters from the main Transaction

Exceptions

- **MobiliserServiceException** – if initializing the transaction fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

Continue the pre-authorization transaction based on the values from the request, performing various checks in the process.

Notes on transactional semantics:

If a business logic exception occurs, a

`MobiliserServiceException`

will be thrown which will trigger a rollback. Any other exception will also trigger a rollback of the database transaction.

a `InitTransactionResult` holding the id of the Authorisation transaction along with the parameters from the main `Transaction`

initTransactionCancel(final *ITransactionCancelRequest*, final *CallerInformation*)
method

Syntax

```
InitTransactionResult initTransactionCancel ( final
ITransactionCancelRequest request, final CallerInformation caller)
throws MobiliserServiceException, DataAccessException,
TransactionException
```

Parameters

- **request** – the transactioncancel request
- **caller** – the actor (authenticated user)

Returns

a `InitTransactionResult` with the parameters from the main `Transaction`

Exceptions

- **MobiliserServiceException** – if initialising the transaction fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

Creates and initializes the

`TransactionCancel`

transaction in the database with the values from the request, performing various checks in the process.

Notes on transactional semantics:

If a business logic exception occurs, a

`MobiliserServiceException`

will be thrown which will trigger a rollback. Any other exception will also trigger a rollback of the database transaction.

a `InitTransactionResult` with the parameters from the main `Transaction`

unlockTransaction(final Long, final Long) method

unlocks a transaction after it has been processed, i.e.

Syntax

```
void unlockTransaction ( final Long systemId , final Long actorId )
```

Parameters

- **systemId** – the transaction id of the transaction to unlock
- **actorId** – the actor (authenticated user)

Usage

resets the lock flag in the database

IIInvoiceAction interface*Syntax*

```
public interface IIInvoiceAction
```

Usage

Common actions for invoice payments.

The reference implementation defines transaction demarcation on the action level and does not rely the client code starting a transaction themselves.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

callCheckInvoice(final long, final Long, final Long) method

Performs a check-invoice on the passed invoice.

Syntax

```
Boolean callCheckInvoice ( final long authorisationId , final Long actorId , final Long invoiceId ) throws MobiliserServiceException, DataAccessException, TransactionException
```

Parameters

- **authorisationId** –
- **actorId** – the authenticated user

- **invoiceId** – the invoice to check; this id is mandatory.

Returns

always true

Exceptions

- **MobiliserServiceException** – if initializing the transaction fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

If the checking is successful, the passed authorization will be linked with the invoice through an

InvoiceTransaction

.

always true

callPayInvoice(final long, final Long) method

Calls pay-invoice on the invoice that belongs to the passed invoice transaction.

Syntax

Boolean callPayInvoice (final long *trxId* , final Long *actorId*) throws MobiliserServiceException, DataAccessException, TransactionException

Parameters

- **trxId** – the invoice transaction to pay
- **actorId** – the authenticated user

Returns

always true

Exceptions

- **MobiliserServiceException** – if initializing the transaction fails for some reason (business rules)

- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

always true

makePreAuthfromInvoiceId(final Long, final ICheckPayInvoiceRequest) method
Creates a check-pay-invoice (aka pre-authorization) request based on the data stored with the invoice itself.

Syntax

```
ICheckPayInvoiceRequest makePreAuthfromInvoiceId ( final Long  
actorId , final ICheckPayInvoiceRequest request ) throws  
MobiliserServiceException, DataAccessException,  
TransactionException
```

Parameters

- **actorId** – the authenticated user
- **request** – the request holding the invoice id (and other operational data like traceNo)

Returns

a ready-to-use check-pay-invoice request with all required information provided.

Exceptions

- **MobiliserServiceException** – if creating the pre-auth request fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

a ready-to-use check-pay-invoice request with all required information provided.

setInvoiceStatus(final long, final Long, final int) method

Updates the invoice status of the invoice, which belongs to passed invoice transaction.

Syntax

```
void setInvoiceStatus ( final long trxId, final Long actorId, final  
int InvoiceStatus ) throws MobiliserServiceException,  
DataAccessException, TransactionException
```

Parameters

- **trxId** – the invoice transaction that belongs to the invoice that should be updated.
- **actorId** – the authenticated user
- **InvoiceStatus** – the new invoice status.

Exceptions

- **MobiliserServiceException** – if initializing the transaction fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

The status will be updated without any further checking.

ILimitAction interface

Common actions for processing limits.

Syntax

```
public interface ILimitAction
```

Usage

The reference implementation defines transaction demarcation on the action level and does not rely the client code starting a transaction themselves.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

limitCheck(final Long, final Long, final boolean) method

Syntax

```
void limitCheck ( final Long subTransactionId , final Long actorId ,
final boolean persist ) throws MobiliserServiceException,
DataAccessException, TransactionException
```

Parameters

- **subTransactionId** – the subtransaction's id
- **actorId** – the actorId (authenticated user)
- **persist** – whether the changes on the limit sums should be persisted or discarded; if set to `false` this can be used for testing purposed w/o actually changing database content

Exceptions

- **MobiliserServiceException** – if checking of the transaction limits fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

Perform a limit sum adjustment and limit check for the given

```
SubTransaction
```

```
.
```

Notes on transactional semantics:

This operation will run inside its own database transaction. Any failure will trigger a rollback and also if the user requests

```
persist=false
```

the changes will also be rolled back. Exception will

always

cause the

```
SubTransaction
```

and the main

```
Transaction
```

to be marked as failed in a fresh database transaction.

IMakerCheckerAction< Req extends IMoneyRequest, Res extends ITransactionResponse > interface

Syntax

```
public interface IMakerCheckerAction< Req extends  
IMoneyRequest, Res extends ITransactionResponse >
```

Usage

Actions for creating a workflow process for a transaction request, e.g. to support maker/checker.

In the reference implementation of the authorization flow this method is called after a successful pre-authorization and authentication. That means that the authorization request may be rejected before the workflow process is started due to balance or limit checks failing (to name just a few).

The reference implementation defines transaction demarcation on the action level and does not rely the client code starting a transaction themselves.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

checkWorkflowConfiguration(final Req, Res, CallerInformation, Long, Long) method
Checks if there exists a maker/checker configuration for the given request (authorization) and starts a new Activiti workflow process if an appropriate configuration was found.

Syntax

```
void checkWorkflowConfiguration ( final Req request , Res response ,  
CallerInformation caller , Long authId , Long txnId ) throws  
AuthorisationPendingException
```

Parameters

- **request** – the current request
- **response** – the current response
- **caller** – the authenticated user
- **authId** – the authorization transaction id
- **txnId** – the main transaction id

Exceptions

- **AuthorisationPendingException** – if the request has been paused in a workflow

IRestrictionAction interface

Common actions for processing transaction restrictions.

Syntax

```
public interface IRestrictionAction
```

Usage

The reference implementation defines transaction demarcation on the action level and does not rely the client code starting a transaction themselves.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

restrictionCheck(final Long, final Long, final boolean) method

Syntax

```
void restrictionCheck ( final Long subTransactionId , final Long
actorId , final boolean persist ) throws MobiliserServiceException,
DataAccessException, TransactionException
```

Parameters

- **subTransactionId** – the sub transaction's id; this parameter is mandatory
- **actorId** – the actorId (authenticated user)
- **persist** – whether the changes should be persisted or discarded; you can discard changes if you just want to check for the outcome w/o actually changing data

Exceptions

- **MobiliserServiceException** – if checking of the transaction limits fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicated / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

Performs the restriction check for the given

SubTransaction

.

Notes on transactional semantics:

This operation will run inside its own database transaction. Any failure will trigger a rollback and also if the user requests

`persist=false`

the changes will also be rolled back. Exception will

always

cause the

SubTransaction

and the main

Transaction

to be marked as failed in a fresh database transaction.

ITransactionEventAction interface

Syntax

```
public interface ITransactionEventAction
```

Usage

Common actions for creating transaction-related events.

The reference implementation defines transaction demarcation on the action level and does not rely the client code starting a transaction themselves.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

generateSubTxnEvent(long, long) method

Generates a subTransaction based event.

Syntax

```
void generateSubTxnEvent ( long txnId , long actor )
```

Parameters

- **txnId** – the subtransaction id

- **actor** – the authenticated user

Usage

It retrieves the transaction status based on the referenced main transaction status and uses the subTransaction error code.

generateTxnEvent(long, long) method

Generates a transaction based event similar to generateSubTxnEvent(long, long), but only considers data and error code of the main transaction.

Syntax

```
void generateTxnEvent ( long systemId ,   long actor )
```

Parameters

- **systemId** – the main transaction id
- **actor** – the authenticated user

IVoucherAction interface

Syntax

```
public interface IVoucherAction
```

Usage

Common actions for voucher handling.

The reference implementation defines transaction demarcation on the action level and does not rely the client code starting a transaction themselves.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

verifyVoucherReceipient(final Long, final Long, final IGenericAuthorisationRequest)
method

Syntax

```
ITransactionParticipant verifyVoucherReceipient ( final Long  
authorisationId ,   final Long actorId ,   final  
IGenericAuthorisationRequest request ) throws  
MobiliserServiceException, DataAccessException,  
TransactionException
```

Parameters

- **authorisationId** – the id of the authorization transaction that is being initialized
- **actorId** – the actorId (authenticated user)
- **request** – the authorization request

Exceptions

- **MobiliserServiceException** – if initializing the transaction fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

Takes the payee identification from the request and check if the identification already exists in the system, then creates the new payment instrument as

InternalVoucher

. If not, create the new customer with internal voucher as payment instrument.

IWebActions interface

Common actions for processing web transactions.

Syntax

```
public interface IWebActions
```

Usage

The reference implementation defines transaction demarcation on the action level and does not rely the client code starting a transaction themselves.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CreateMoneyWebTransactionResult class

Syntax

```
public class CreateMoneyWebTransactionResult
```

Usage

Result of creating the initial money web transactions.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CreateMoneyWebTransactionResult(final String, final Long, final String, final String, final Long) constructor

Creates a new web transaction result bean.

Syntax

```
CreateMoneyWebTransactionResult ( final String authCode , final
Long authId , final String orderId , final String returnUrl , final Long
txnId )
```

Parameters

- **authCode** –
- **authId** –
- **orderId** –
- **returnUrl** –
- **txnId** –

authCode variable

the authorisation code of the main transaction.

Syntax

```
final String authCode
```

authId variable

the id of the authorisation transaction.

Syntax

```
final Long authId
```

orderId variable

the order id of the transaction.

Syntax

```
final String orderId
```

returnUrl variable

return url.

Syntax

```
final String returnUrl
```

txnId variable

the main transaction id.

Syntax

```
final Long txnId
```

createWebTransactions(final IWebStartRequest, final CallerInformation) method

Syntax

CreateMoneyWebTransactionResult createWebTransactions (final IWebStartRequest *request* , final CallerInformation *caller*) throws MobiliserServiceException, DataAccessException, TransactionException

Parameters

- **request** – the IWebStartRequest for which you want to create the transactions
- **caller** – the actor (authenticated user)

Returns

a bean holding the created and initialized transactions.

Exceptions

- **MobiliserServiceException** – if the creation of the transaction fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

This method creates the web transactions for the given request. That includes the main

Transaction

along with the

Authorisation

sub-transaction

Notes on transactional semantics:

This method will create both transactions inside one database transaction, rolling back completely if creation fails for any reason and any exception type.

a bean holding the created and initialized transactions.

initWebTransaction(final Long, final IWebStartRequest, final IWebStartResponse, final CallerInformation) method

Syntax

```
void initWebTransaction ( final Long authorisationId , final
IWebStartRequest request , final IWebStartResponse response , final
CallerInformation caller ) throws MobiliserServiceException,
DataAccessException, TransactionException
```

Parameters

- **authorisationId** – the id of the Authorisation transaction to initialize
- **request** – the web start request to process
- **response** – the web start response, which might get filled with additional data during processing.
- **caller** – the actor (authenticated user)

Exceptions

- **MobiliserServiceException** – if the initialization of the transaction fails for some reason (business rules)
- **DataAccessException** – if some error occurs communicating / writing / reading persistent storage
- **TransactionException** – if some error occurs while handling the transactions related to persistent storage

Usage

This method complements

`IWebActions#createWebTransactions(IWebStartRequest, Long)`

. After successfully creating the transactions, this action can be used to actually initialize all relevant fields in the

Authorisation

Notes on transactional semantics:

If a business logic exception occurs, a

`MobiliserServiceException`

will be thrown and both transactions will be marked with the error code from the exception and the database transaction committed. Any other exception will trigger a rollback of the database transaction followed by a fresh transaction inside of which the

`Authorisation`

transaction will be marked with an error code.

authcancel package

Members

All public members of the authcancel package.

- **`IAuthorisationCancel< Req extends IAuthorisationCancelRequest, Res extends IAuthorisationCancelResponse > interface`** – Implementations of this interface can cancel an authorisation.

`IAuthorisationCancel< Req extends IAuthorisationCancelRequest, Res extends IAuthorisationCancelResponse > interface`

Implementations of this interface can cancel an authorisation.

Syntax

```
public interface IAuthorisationCancel< Req extends
IAuthorisationCancelRequest, Res extends
IAuthorisationCancelResponse >
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

`authorisationCancel(final CallerInformation, final Req, final Res) method`

Cancel the referenced authorization.

Syntax

```
void authorisationCancel ( final CallerInformation caller , final
Req request , final Res response )
```

Parameters

- **caller** – the actorId
- **request** – the request parameters
- **response** – the response parameters

Usage

The caller is the customer authenticated before calling this method (e.g., http header)

authcont package*Members*

All public members of the authcont package.

- **IAuthenticationContinue< Req extends IAuthenticationContinueRequest, Res extends IAuthenticationContinueResponse > interface** – Implementations of this interface can continue authentication for a transaction.

IAuthenticationContinue< Req extends IAuthenticationContinueRequest, Res extends IAuthenticationContinueResponse > interface

Implementations of this interface can continue authentication for a transaction.

Syntax

```
public interface IAuthenticationContinue< Req extends
IAuthenticationContinueRequest, Res extends
IAuthenticationContinueResponse >
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

authenticationContinue(final CallerInformation, final Req, final Res) method
Continue authentication for the referenced transaction.

Syntax

```
void authenticationContinue ( final CallerInformation caller ,
final Req request , final Res response )
```

Parameters

- **caller** – the actor

- **request** – the request parameters
- **response** – the response parameters

authorisation package

Members

All public members of the authorisation package.

- **IAuthorisationService< Req extends IAuthorisationRequest, Res extends IAuthorisationResponse > interface** – This interface represents the authorization of a transaction.

IAuthorisationService< Req extends IAuthorisationRequest, Res extends IAuthorisationResponse > interface

This interface represents the authorization of a transaction.

Syntax

```
public interface IAuthorisationService< Req extends  
IAuthorisationRequest, Res extends IAuthorisationResponse >
```

Usage

The details of the authorization are passed as an IAuthorisationRequest object. The result of the authorization is written into the passed response object.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

authorise(final CallerInformation, final Req, final Res) method

Authorize the given transaction.

Syntax

```
void authorise ( final CallerInformation caller , final Req request ,  
final Res response )
```

Parameters

- **caller** – the current caller
- **request** – the request parameters
- **response** – the response parameters

continueAuthorise(final Long, final Req, final Res, final Long, final Long) method
Continues a pending authorization.

Syntax

```
void continueAuthorise ( final Long actorId , final Req request ,
final Res response , final Long authId , final Long txnId )
```

Parameters

- **actorId** – the current caller
- **request** – the request parameters
- **response** – the response parameters
- **authId** – the authorization subtransaction id
- **txnId** – the main transaction id

rejectAuthorise(final Long, final Req, final Res, final Long, final Long) method
Rejects a pending authorization.

Syntax

```
void rejectAuthorise ( final Long actorId , final Req request , final
Res response , final Long authId , final Long txnId )
```

Parameters

- **actorId** – the current caller
- **request** – the request parameters
- **response** – the response parameters
- **authId** – the authorization subtransaction id
- **txnId** – the main transaction id

capture package

Members

All public members of the capture package.

- **ICapture< Req extends ICaptureRequest, Res extends ICaptureResponse > interface** – The capture service is responsible for completely capturing a transaction.

ICapture< Req extends ICaptureRequest, Res extends ICaptureResponse > interface

The capture service is responsible for completely capturing a transaction.

Syntax

```
public interface ICapture< Req extends ICaptureRequest, Res
extends ICaptureResponse >
```

Usage

It takes as its parameters everything needed to actually capture the transaction and handling the transaction demarcation and setting of error codes.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

capture(final CallerInformation, final Req, final Res) method

Capture the transaction referenced in this request returning the result.

Syntax

```
void capture ( final CallerInformation caller , final Req request ,
final Res response )
```

Parameters

- **caller** – the actor
- **request** – the request
- **response** – the response object which the implementation should fill with the capture results

capturecancel package

Members

All public members of the capturecancel package.

- **ICaptureCancel< Req extends ICaptureCancelRequest, Res extends ICaptureCancelResponse > interface** – The capture cancel service implements the full flow of a capture cancel.

ICaptureCancel< Req extends ICaptureCancelRequest, Res extends ICaptureCancelResponse > interface

The capture cancel service implements the full flow of a capture cancel.

Syntax

```
public interface ICaptureCancel< Req extends
ICaptureCancelRequest, Res extends ICaptureCancelResponse >
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

captureCancel(final CallerInformation, final Req, final Res) method

Performs a capture-cancel.

Syntax

```
void captureCancel ( final CallerInformation caller , final Req
request , final Res response )
```

Parameters

- **caller** – the current caller
- **request** – the request data
- **response** – the response data

checkinvoice package

Members

All public members of the checkinvoice package.

- **ICheckPayInvoice< Req extends ICheckPayInvoiceRequest, Res extends ICheckPayInvoiceResponse > interface** – Defines the service to check (pre-authorize) an invoice payment.

ICheckPayInvoice< Req extends ICheckPayInvoiceRequest, Res extends ICheckPayInvoiceResponse > interface

Defines the service to check (pre-authorize) an invoice payment.

Syntax

```
public interface ICheckPayInvoice< Req extends
ICheckPayInvoiceRequest, Res extends ICheckPayInvoiceResponse
>
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

checkPayInvoice(final CallerInformation, final Req, final Res) method

Performs a check-pay-invoice.

Syntax

```
void checkPayInvoice ( final CallerInformation caller , final Req  
request , final Res response )
```

Parameters

- **caller** – the current caller
- **request** – the request data
- **response** – the response data

continueinvoice package

Members

All public members of the continueinvoice package.

- **IContinuePayInvoice< Req extends IContinuePayInvoiceRequest, Res extends IContinuePayInvoiceResponse > interface** – Defines the service to continue a previous check-pay-invoice transaction.

*IContinuePayInvoice< Req extends IContinuePayInvoiceRequest, Res extends
IContinuePayInvoiceResponse > interface*

Defines the service to continue a previous check-pay-invoice transaction.

Syntax

```
public interface IContinuePayInvoice< Req extends  
IContinuePayInvoiceRequest, Res extends  
IContinuePayInvoiceResponse >
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

continuePayInvoice(final CallerInformation, final Req, final Res) method
Continues an existing pay-invoice transaction (from a check-pay-invoice call).

Syntax

```
void continuePayInvoice ( final CallerInformation caller , final
Req request , final Res response )
```

Parameters

- **caller** – the current caller
- **request** – the request data
- **response** – the response data

demandforpayment package

Members

All public members of the demandforpayment package.

- **IDemandForPaymentService interface** – Provides the demand-for-payment service that allows customers requesting money from other customers by creating an invoice for the other customer.

IDemandForPaymentService interface

Provides the demand-for-payment service that allows customers requesting money from other customers by creating an invoice for the other customer.

Syntax

```
public interface IDemandForPaymentService
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

demandForPayment(final Long, final List< String >, final IDemandForPaymentRequest, final IDemandForPaymentResponse) method
Creates an invoice for the requesting customer, so that the other customer can pay the invoice to send the money to the requestor.

Syntax

```
void demandForPayment ( final Long actorId , final List< String >
actorPrivs , final IDemandForPaymentRequest request , final
IDemandForPaymentResponse response )
```

Parameters

- **actorId** – the actor
- **actorPrivs** – the actor's privileges
- **request** – the request data
- **response** – the response data

demarcation package

Members

All public members of the demarcation package.

- **DynamicMobiliserTxnAwareFatalSubTransactionCallback< T > interface** – A further extension of the MobiliserTxnAwareTransactionCallback which allows dynamic decision making of whether the main and sub transaction should be marked as failed.
- **DynamicMobiliserTxnAwareFatalTransactionCallback< T > interface** – A further extension of the MobiliserTxnAwareTransactionCallback which allows dynamic decision making of whether the main transaction should also be marked as failed.
- **ITransactionFailer interface** – Implementations of this interface can fail transactions.
- **MobiliserTransactionOperations interface** – When writing Mobiliser platform services, the whole transaction demarcation and exception handling aspect is tricky to get just right.
- **MobiliserTransactionTemplateFactory class** – This FactoryBean makes it simple to create a fully-configured MobiliserTransactionOperations.
- **MobiliserTxnAwareFatalTransactionCallback< T > interface** – A further extension of the MobiliserTxnAwareTransactionCallback interface which will cause the main transaction to also always be marked with the error code.
- **MobiliserTxnAwareTransactionCallback< T > interface** – This extension of the standard TransactionCallback allows com.sybase365.mobiliser.money.businesslogic.transaction.demarcation.MobiliserTransactionOperations implementations to know what subtransaction will be affected by the callback and the actor performing it.

DynamicMobiliserTxnAwareFatalSubTransactionCallback< T > interface

A further extension of the MobiliserTxnAwareTransactionCallback which allows dynamic decision making of whether the main and sub transaction should be marked as failed.

Syntax

```
public interface
DynamicMobiliserTxnAwareFatalSubTransactionCallback< T >
```

DynamicMobiliserTxnAwareFatalTransactionCallback< T > interface

A further extension of the MobiliserTxnAwareTransactionCallback which allows dynamic decision making of whether the main transaction should also be marked as failed.

Syntax

```
public interface
DynamicMobiliserTxnAwareFatalTransactionCallback< T >
```

Derived classes

- *DynamicMobiliserTxnAwareFatalSubTransactionCallback< T >* on page 315

failMainTransaction(final Throwable) method

Determine whether the given exception should cause the main transaction to be failed.

Syntax

```
boolean failMainTransaction ( final Throwable e )
```

Parameters

- **e** – the exception which was thrown

Returns

true if the main transaction should be failed

Usage

true if the main transaction should be failed

ITransactionFailer interface

Implementations of this interface can fail transactions.

Syntax

```
public interface ITransactionFailer
```

Usage

This is typically used for building complex "Actions" and then failing the transaction inside in the appropriate manner (rolling back, not rolling back). Failing a transaction will mean setting its status and error code properly.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

failTransaction(final MobiliserTxnAwareTransactionCallback<?>, final Throwable)
method

Fail the transaction referenced from the given MobiliserTxnAwareTransactionCallback.

Syntax

```
void failTransaction ( final  
MobiliserTxnAwareTransactionCallback<?> awareCallback ,   final  
Throwable e )
```

Parameters

- **awareCallback** – the callback
- **e** – the exception thrown

failTransactionInTransaction(final MobiliserTxnAwareTransactionCallback<?>, final Throwable)
method

Fail the transaction referenced from the given MobiliserTxnAwareTransactionCallback inside a database transaction.

Syntax

```
void failTransactionInTransaction ( final  
MobiliserTxnAwareTransactionCallback<?> awareCallback ,   final  
Throwable e )
```

Parameters

- **awareCallback** – the callback
- **e** – the exception thrown

MobiliserTransactionOperations interface

When writing Mobiliser platform services, the whole transaction demarcation and exception handling aspect is tricky to get just right.

Syntax

```
public interface MobiliserTransactionOperations
```

Usage

This extension of the standard TransactionOperations interface allows client to code to mark under which circumstances mobiliser transactions should be marked with an error code and when exceptions should cause a database rollback. Implementations are aware of com.sybase365.mobiliser.money.businesslogic.transaction.demarcation.MobiliserTxnAwareTransactionCallback and can set error codes on transactions accordingly. Use this marker interface in your code to enforce injecting a proper implementation (instead of a raw TransactionOperations)

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

MobiliserTransactionTemplateFactory class

This FactoryBean makes it simple to create a fully-configured MobiliserTransactionOperations.

Syntax

```
public class MobiliserTransactionTemplateFactory
```

Usage

It requires a TransactionDAO and a SubTransactionDAO as well at the PlatformTransactionManager to use. It allows configuring the isolation levels and Propagation for the transactions. These default to Isolation#DEFAULT and Propagation#REQUIRES_NEW.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

MobiliserTransactionTemplateFactory(final int, final int, final SubTransactionDAO, final TransactionDAO, final IMobiliserExceptionTranslator, final PlatformTransactionManager) constructor

Syntax

```
MobiliserTransactionTemplateFactory ( final int isolationLevel ,  
final int propagationBehavior , final SubTransactionDAO  
subTransactionDao , final TransactionDAO transactionDao , final  
IMobiliserExceptionTranslator translator , final  
PlatformTransactionManager transactionManager )
```

Parameters

- **isolationLevel** –
- **propagationBehavior** –
- **subTransactionDao** –
- **transactionDao** –
- **translator** –
- **transactionManager** –

MobiliserTransactionTemplateFactory(final SubTransactionDAO, final TransactionDAO, final IMobiliserExceptionTranslator, final PlatformTransactionManager) constructor

Syntax

```
MobiliserTransactionTemplateFactory ( final SubTransactionDAO  
subTransactionDao , final TransactionDAO transactionDao , final  
IMobiliserExceptionTranslator translator , final  
PlatformTransactionManager transactionManager )
```

Parameters

- **subTransactionDao** –
- **transactionDao** –
- **translator** –
- **transactionManager** –

getObject() method

Syntax

```
MobiliserTransactionOperations getObject () throws Exception
```

*getObjectType() method***Syntax**

```
Class<?> getObjectType ()
```

*isSingleton() method***Syntax**

```
boolean isSingleton ()
```

MobiliserTxnAwareFatalTransactionCallback< T > interface

A further extension of the MobiliserTxnAwareTransactionCallback interface which will cause the main transaction to also always be marked with the error code.

Syntax

```
public interface MobiliserTxnAwareFatalTransactionCallback< T
>
```

MobiliserTxnAwareTransactionCallback< T > interface

This extension of the standard TransactionCallback allows com.sybase365.mobiliser.money.businesslogic.transaction.demarcation.MobiliserTransactionOperations implementations to know what subtransaction will be affected by the callback and the actor performing it.

Syntax

```
public interface MobiliserTxnAwareTransactionCallback< T >
```

Derived classes

- *DynamicMobiliserTxnAwareFatalTransactionCallback< T >* on page 315
- *MobiliserTxnAwareFatalTransactionCallback< T >* on page 319

Usage

It allows configuring a hint of whether the current database transaction should be rolled back and then the transaction failed, or if the transaction should be committed with the error code set.

getActorId() method

The id of the actor for this operation.

Syntax

```
Long getActorId ()
```

Returns

the actor's id

Usage

the actor's id

getSubTransactionId() method

Returns the id of the subtransaction this operation will affect.

Syntax

```
long getSubTransactionId ()
```

Returns

the subtransaction's id

Usage

the subtransaction's id

isRollbackThenFail() method

If true, a rollack is performed and then afterward the transaction is fetched in a clean database transaction and then failed.

Syntax

```
boolean isRollbackThenFail ()
```

Returns

true or false

Usage

true or false

flow package

Members

All public members of the flow package.

- **AbstractMoneyCreateTransactionFlow< Req extends IGenericAuthorisationRequest, Res extends ITransactionResponse > class** – The AbstractMoneyCreateTransactionFlow extends the AbstractMoneyFlow and simply adds the new steps with pre and post processing hooks.

- **AbstractMoneyFlow< Req extends IMoneyRequest, Res extends ITransactionResponse > class** –
- **AbstractMoneyRefTransactionFlow< Req extends IMoneyRefTransactionRequest, Res extends ITransactionResponse > class** – Abstract wrapper for AbstractMoneyFlow.
- **ITransactionFlowOverlay< Req extends IMoneyRequest, Res extends ITransactionResponse > interface** –

AbstractMoneyCreateTransactionFlow< Req extends IGenericAuthorisationRequest, Res extends ITransactionResponse > class

The AbstractMoneyCreateTransactionFlow extends the AbstractMoneyFlow and simply adds the new steps with pre and post processing hooks.

Syntax

```
public class AbstractMoneyCreateTransactionFlow< Req extends
IGenericAuthorisationRequest, Res extends
ITransactionResponse >
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

afterCreateMoneyTransaction(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, AtomicReference< CreateMoneyTransactionResult >) method

Postprocessing step.

Syntax

```
void afterCreateMoneyTransaction ( final AtomicReference< Req >
requestRef , final AtomicReference< Res > responseRef , final
AtomicReference< CallerInformation > callerRef , AtomicReference<
CreateMoneyTransactionResult > resultRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **resultRef** –

Usage

This method does nothing and can be overridden

afterInitAuthorisation(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >) method
Post processing step.

Syntax

```
void afterInitAuthorisation ( final AtomicReference< Req >  
    requestRef, final AtomicReference< Res > responseRef, final  
    AtomicReference< CallerInformation > callerRef, final  
    AtomicReference< Long > authIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –

Usage

This method does nothing and can be overridden

beforeCreateMoneyTransaction(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >) method
Pre processing step.

Syntax

```
void beforeCreateMoneyTransaction ( final AtomicReference< Req  
> requestRef, final AtomicReference< Res > responseRef, final  
    AtomicReference< CallerInformation > callerRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –

Usage

This method does nothing and can be overridden

beforeInitAuthorisation(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >)

method

Pre processing step.

Syntax

```
void beforeInitAuthorisation ( final AtomicReference< Req >
    requestRef , final AtomicReference< Res > responseRef , final
    AtomicReference< CallerInformation > callerRef , final
    AtomicReference< Long > authIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –

Usage

This method does nothing and can be overridden

createMoneyTransaction(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >) *method*

Create a new money transaction by calling the pre processing step,

ITransactionFlowOverlay#beforeCreateTransaction(com.sybase365.mobiliser.money.businesslogic.util.beans.IMoneyRequest, ITransactionResponse, AtomicReference) , initAction and finally post processing step.

Syntax

```
final CreateMoneyTransactionResult createMoneyTransaction
    ( final AtomicReference< Req > requestRef , final AtomicReference<
    Res > responseRef , final AtomicReference< CallerInformation >
    callerRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –

initAuthorisation(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >) method
Initialize the Authorization with pre- and post step.

Syntax

```
final void initAuthorisation ( final AtomicReference< Req >  
requestRef, final AtomicReference< Res > responseRef, final  
AtomicReference< CallerInformation > callerRef, final  
AtomicReference< Long > authIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –

initAuthorisationChecked(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >, final AtomicReference< Boolean >) method
Initializes the transaction and call the limit and restriction check.

Syntax

```
final void initAuthorisationChecked ( final AtomicReference<  
Req > requestRef, final AtomicReference< Res > responseRef, final  
AtomicReference< CallerInformation > callerRef, final  
AtomicReference< Long > authIdRef, final AtomicReference<  
Boolean > persistRef )
```

Parameters

- **actorId** –
- **actorPrivs** –
- **request** –
- **result** –

Usage

Afterwards the

```
ITransactionFlowOverlay#afterInitTransaction(com.sybase365.mobiliser.money.businesslo  
gic.util.beans.IMoneyRequest, ITransactionResponse, AtomicReference, AtomicReference,  
boolean)
```


is called

AbstractMoneyFlow< Req extends IMoneyRequest, Res extends ITransactionResponse > class

Syntax

```
public class AbstractMoneyFlow< Req extends IMoneyRequest,
Res extends ITransactionResponse >
```

Usage

The `AbstractMoneyFlow` implements the Spring DM injection methods for common OSGi service used by the `MoneyFlow` service. It also provides the base implementation of the transaction steps Steps that can be influenced by `ITransactionFlowOverlay` implementation have pre- and post- processing steps.

Please use the `ITransactionFlowOverlay` to customize the transaction flow processing if needed.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

adjustBalances(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >) method
Adjust balance step with pre- and post processing.

Syntax

```
void adjustBalances ( final AtomicReference< Req > requestRef ,
final AtomicReference< Res > responseRef , final AtomicReference<
CallerInformation > callerRef , final AtomicReference< Long >
txnIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **txnIdRef** –

afterAdjustBalances(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >) method
Post processing step.

Syntax

```
void afterAdjustBalances ( final AtomicReference< Req >  
requestRef , final AtomicReference< Res > responseRef , final  
AtomicReference< CallerInformation > callerRef , final  
AtomicReference< Long > txnIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **txnIdRef** –

Usage

This method does nothing and can be overridden

afterAuthenticate(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >, final AtomicReference< AuthenticationMethodComputation >, final AtomicReference< ITransactionCredential >, final AtomicReference< ConsumerAuthenticationResult >) method
Post Authenticate processing step.

Syntax

```
void afterAuthenticate ( final AtomicReference< Req > requestRef ,  
final AtomicReference< Res > responseRef , final AtomicReference<  
CallerInformation > callerRef , final AtomicReference< Long >  
authIdRef , final AtomicReference<  
AuthenticationMethodComputation > authMethods , final  
AtomicReference< ITransactionCredential > credentials , final  
AtomicReference< ConsumerAuthenticationResult > authenticate )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –

- **authIdRef** –
- **authMethods** –
- **credentials** –
- **authenticate** –

Usage

This method does nothing and can be overridden

afterAuthorisationCancel(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Integer >, final AtomicReference< Long >) method

Post processing step.

Syntax

```
void afterAuthorisationCancel ( final AtomicReference< Req >
requestRef, final AtomicReference< Res > responseRef, final
AtomicReference< CallerInformation > callerRef, final
AtomicReference< Integer > errorCodeRef, final AtomicReference<
Long > authCancelIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **errorCodeRef** –
- **authCancelIdRef** –

Usage

This method does nothing and can be overridden

afterCommissionCheck(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >) method

Post processing step.

Syntax

```
void afterCommissionCheck ( final AtomicReference< Req >
requestRef, final AtomicReference< Res > responseRef, final
AtomicReference< CallerInformation > callerRef, final
AtomicReference< Long > authIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –

Usage

This method does nothing and can be overridden

afterComputeAuthentication(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >, final AtomicReference< AuthenticationMethodComputation >) method
After ComputeAuthentication processing step.

Syntax

```
void afterComputeAuthentication ( final AtomicReference< Req >
requestRef, final AtomicReference< Res > responseRef, final
AtomicReference< CallerInformation > callerRef, final
AtomicReference< Long > authIdRef, final AtomicReference<
AuthenticationMethodComputation > authMethods )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –
- **authMethods** –

Usage

This method does nothing and can be overridden

afterDoAuthorisation(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >, final AtomicReference< Boolean >) method
Post processing step.

Syntax

```
void afterDoAuthorisation ( final AtomicReference< Req >
requestRef, final AtomicReference< Res > responseRef, final
AtomicReference< CallerInformation > callerRef, final
```

```
AtomicReference< Long > authIdRef , final AtomicReference<
Boolean > autoCaptureRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –
- **autoCaptureRef** –

Usage

This method does nothing and can be overridden

afterDoCapture(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >, final AtomicReference< Long >) method
Post processing step.

Syntax

```
void afterDoCapture ( final AtomicReference< Req > requestRef ,
final AtomicReference< Res > responseRef , final AtomicReference<
CallerInformation > callerRef , final AtomicReference< Long >
captureIdRef , final AtomicReference< Long > authIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **captureIdRef** –
- **authIdRef** –

Usage

This method does nothing and can be overridden

afterDoPreAuthorisation(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >) method

Post processing step.

Syntax

```
void afterDoPreAuthorisation ( final AtomicReference< Req >
    requestRef, final AtomicReference< Res > responseRef, final
    AtomicReference< CallerInformation > callerRef, final
    AtomicReference< Long > authIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –

Usage

This method does nothing and can be overridden

afterInitFollowUpAuthorisationCancel(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >) method

Post processing step.

Syntax

```
void afterInitFollowUpAuthorisationCancel ( final
    AtomicReference< Req > requestRef, final AtomicReference< Res >
    responseRef, final AtomicReference< CallerInformation > callerRef,
    final AtomicReference< Long > authIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –

Usage

This method does nothing and can be overridden

afterInitFollowUpCapture(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >, final AtomicReference< Long >) method

Post processing step.

Syntax

```
void afterInitFollowUpCapture ( final AtomicReference< Req >
    requestRef, final AtomicReference< Res > responseRef, final
    AtomicReference< CallerInformation > callerRef, final
    AtomicReference< Long > authIdRef, final AtomicReference< Long >
    resultRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –
- **resultRef** –

Usage

This method does nothing and can be overridden

afterLimitCheck(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >, final AtomicReference< Boolean >) method

Post processing step.

Syntax

```
void afterLimitCheck ( final AtomicReference< Req > requestRef,
    final AtomicReference< Res > responseRef, final AtomicReference<
    CallerInformation > callerRef, final AtomicReference< Long >
    authIdRef, final AtomicReference< Boolean > persistRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –
- **persistRef** –

Usage

This method does nothing and can be overridden

afterPropertiesSet() method

This is called after Spring injection and verifies that all mandatory services are present/configured.

Syntax

```
void afterPropertiesSet ()
```

Exceptions

- **IllegalStateException** – when a mandatory OSGi service can not be injected/is not configured. Please see error message and make sure that this service is available and configured in the bundle context via property reference

afterReadjustBalances(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >)
method

Post processing step.

Syntax

```
void afterReadjustBalances ( final AtomicReference< Req >  
requestRef , final AtomicReference< Res > responseRef , final  
AtomicReference< CallerInformation > callerRef , final  
AtomicReference< Long > authIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –

Usage

This method does nothing and can be overridden

afterRestrictionCheck(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >, final AtomicReference< Boolean >) method

Post processing step.

Syntax

```
void afterRestrictionCheck ( final AtomicReference< Req >
    requestRef, final AtomicReference< Res > responseRef, final
    AtomicReference< CallerInformation > callerRef, final
    AtomicReference< Long > authIdRef, final AtomicReference<
    Boolean > persistRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –
- **persistRef** –

Usage

This method does nothing and can be overridden

afterSubTxnEventGeneration(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< Long >) method

Post processing step.

Syntax

```
void afterSubTxnEventGeneration ( final AtomicReference< Req >
    requestRef, final AtomicReference< Res > responseRef, final
    AtomicReference< Long > subTxnIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **subTxnIdRef** –

Usage

This method does nothing and can be overridden

afterTxnEventGeneration(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< Long >) method
Post processing step.

Syntax

```
void afterTxnEventGeneration ( final AtomicReference< Req >  
    requestRef, final AtomicReference< Res > responseRef, final  
    AtomicReference< Long > txnIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **txnIdRef** –

Usage

This method does nothing and can be overridden

afterUnlockTransaction(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >) method
Post processing step.

Syntax

```
void afterUnlockTransaction ( final AtomicReference< Req >  
    requestRef, final AtomicReference< Res > responseRef, final  
    AtomicReference< CallerInformation > callerRef, final  
    AtomicReference< Long > txnIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **txnIdRef** –

Usage

This method does nothing and can be overridden

authCancel(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >, final AtomicReference< Integer >) method
 Authorisation Cancel processing step.

Syntax

```
Long authCancel ( final AtomicReference< Req > requestRef, final
AtomicReference< Res > responseRef, final AtomicReference<
CallerInformation > callerRef, final AtomicReference< Long >
authIdRef, final AtomicReference< Integer > errorCodeRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –
- **errorCodeRef** –

Usage

This step retrieves the authorisation reference and calls the authorisation cancel

authCancel(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >, final AtomicReference< Integer >, final AtomicReference< Long >) method
 Performs the auth cancel by performing the authorizationCancel and adjusting the balances afterwards.

Syntax

```
Long authCancel ( final AtomicReference< Req > requestRef, final
AtomicReference< Res > responseRef, final AtomicReference<
CallerInformation > callerRef, final AtomicReference< Long >
authIdRef, final AtomicReference< Integer > errorCodeRef, final
AtomicReference< Long > authCancelIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –

- **errorCodeRef** –
- **authCancelIdRef** –

authenticate(final AtomicReference< Req > *requestRef* , final AtomicReference< Res > , final AtomicReference< CallerInformation > , final AtomicReference< Long > , final AtomicReference< ITransactionCredential >) method

Calculate the authentication methods and perform the authentication.

Syntax

```
ConsumerAuthenticationResult authenticate ( final  
AtomicReference< Req > requestRef , final AtomicReference< Res >  
responseRef , final AtomicReference< CallerInformation > callerRef ,  
final AtomicReference< Long > authIdRef , final AtomicReference<  
ITransactionCredential > credentials )
```

Parameters

- **actorId** –
- **result** –

authenticate(final AtomicReference< Req > , final AtomicReference< Res > , final AtomicReference< CallerInformation > , final AtomicReference< Long > , final AtomicReference< AuthenticationMethodComputation > , final AtomicReference< ITransactionCredential >) method

Authentication step with pre- and processing step.

Syntax

```
ConsumerAuthenticationResult authenticate ( final  
AtomicReference< Req > requestRef , final AtomicReference< Res >  
responseRef , final AtomicReference< CallerInformation > callerRef ,  
final AtomicReference< Long > authIdRef , final AtomicReference<  
AuthenticationMethodComputation > authMethods , final  
AtomicReference< ITransactionCredential > credentials )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –
- **authMethods** –
- **credentials** –

beforeAdjustBalances(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >)
method

Pre processing step.

Syntax

```
void beforeAdjustBalances ( final AtomicReference< Req >
    requestRef, final AtomicReference< Res > responseRef, final
    AtomicReference< CallerInformation > callerRef, final
    AtomicReference< Long > txnIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **txnIdRef** –

Usage

This method does nothing and can be overridden

beforeAuthenticate(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >, final AtomicReference< AuthenticationMethodComputation >, final AtomicReference< ITransactionCredential >) method

Pre Authenticate processing step.

Syntax

```
void beforeAuthenticate ( final AtomicReference< Req > requestRef,
    final AtomicReference< Res > responseRef, final AtomicReference<
    CallerInformation > callerRef, final AtomicReference< Long >
    authIdRef, final AtomicReference<
    AuthenticationMethodComputation > authMethods, final
    AtomicReference< ITransactionCredential > credentials )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –

- **authMethods** –
- **credentials** –

Usage

This method does nothing and can be overridden

beforeAuthorisationCancel(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Integer >, final AtomicReference< Long >) method

Pre processing step.

Syntax

```
void beforeAuthorisationCancel ( final AtomicReference< Req >
requestRef, final AtomicReference< Res > responseRef, final
AtomicReference< CallerInformation > callerRef, final
AtomicReference< Integer > errorCodeRef, final AtomicReference<
Long > authCancelIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **errorCodeRef** –
- **authCancelIdRef** –

Usage

This method does nothing and can be overridden

beforeCommissionCheck(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >) method

Pre processing step.

Syntax

```
void beforeCommissionCheck ( final AtomicReference< Req >
requestRef, final AtomicReference< Res > responseRef, final
AtomicReference< CallerInformation > callerRef, final
AtomicReference< Long > authIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –

Usage

This method does nothing and can be overridden

beforeComputeAuthentication(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >) method

Pre processing step on compute authentication.

Syntax

```
void beforeComputeAuthentication ( final AtomicReference< Req >
requestRef, final AtomicReference< Res > responseRef, final
AtomicReference< CallerInformation > callerRef, final
AtomicReference< Long > authIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –

Usage

This method does nothing and can be overridden

beforeDoAuthorisation(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >) method

Pre processing step.

Syntax

```
void beforeDoAuthorisation ( final AtomicReference< Req >
requestRef, final AtomicReference< Res > responseRef, final
AtomicReference< CallerInformation > callerRef, final
AtomicReference< Long > authIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –

Usage

This method does nothing and can be overridden

beforeDoCapture(final AtomicReference< Req > *requestRef* , final AtomicReference< Res > , final AtomicReference< CallerInformation > , final AtomicReference< Long > , final AtomicReference< Long >) method

Pre processing step.

Syntax

```
void beforeDoCapture ( final AtomicReference< Req > requestRef ,  
final AtomicReference< Res > responseRef , final AtomicReference<  
CallerInformation > callerRef , final AtomicReference< Long >   
captureIdRef , final AtomicReference< Long > authIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **captureIdRef** –
- **authIdRef** –

Usage

This method does nothing and can be overridden

beforeDoPreAuthorisation(final AtomicReference< Req > , final AtomicReference< Res > , final AtomicReference< CallerInformation > , final AtomicReference< Long >) method

Pre processing step.

Syntax

```
void beforeDoPreAuthorisation ( final AtomicReference< Req >   
requestRef , final AtomicReference< Res > responseRef , final  
AtomicReference< CallerInformation > callerRef , final  
AtomicReference< Long > authIdRef )
```


Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –

Usage

This method does nothing and can be overridden

beforeInitFollowUpAuthorisationCancel(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >) method

Pre processing step.

Syntax

```
void beforeInitFollowUpAuthorisationCancel ( final
AtomicReference< Req > requestRef, final AtomicReference< Res >
responseRef, final AtomicReference< CallerInformation > callerRef,
final AtomicReference< Long > authIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –

Usage

This method does nothing and can be overridden

beforeInitFollowUpCapture(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >) method

Pre processing step.

Syntax

```
void beforeInitFollowUpCapture ( final AtomicReference< Req >
requestRef, final AtomicReference< Res > responseRef, final
AtomicReference< CallerInformation > callerRef, final
AtomicReference< Long > authIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –

Usage

This method does nothing and can be overridden

beforeLimitCheck(final AtomicReference< Req > *requestRef* , final AtomicReference< Res > ,
final AtomicReference< CallerInformation > , final AtomicReference< Long > , final
AtomicReference< Boolean >) *method*

Pre processing step.

Syntax

```
void beforeLimitCheck ( final AtomicReference< Req > requestRef ,  
final AtomicReference< Res > responseRef , final AtomicReference<  
CallerInformation > callerRef , final AtomicReference< Long >  
authIdRef , final AtomicReference< Boolean > persistRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –
- **persistRef** –

Usage

This method does nothing and can be overridden

beforeReadjustBalances(final AtomicReference< Req > , final AtomicReference<
Res > , final AtomicReference< CallerInformation > , final AtomicReference< Long >)
method

Pre processing step.

Syntax

```
void beforeReadjustBalances ( final AtomicReference< Req >  
requestRef , final AtomicReference< Res > responseRef , final  
AtomicReference< CallerInformation > callerRef , final  
AtomicReference< Long > authIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –

Usage

This method does nothing and can be overridden

beforeRestrictionCheck(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >, final AtomicReference< Boolean >) method

Pre processing step.

Syntax

```
void beforeRestrictionCheck ( final AtomicReference< Req >
requestRef , final AtomicReference< Res > responseRef , final
AtomicReference< CallerInformation > callerRef , final
AtomicReference< Long > authIdRef , final AtomicReference<
Boolean > persistRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –
- **persistRef** –

Usage

This method does nothing and can be overridden

beforeSubTxnEventGeneration(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< Long >) method

Pre processing step.

Syntax

```
void beforeSubTxnEventGeneration ( final AtomicReference< Req >
requestRef , final AtomicReference< Res > responseRef , final
AtomicReference< Long > subTxnIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **subTxnIdRef** –

Usage

This method does nothing and can be overridden

beforeTxnEventGeneration(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< Long >) method

Pre processing step.

Syntax

```
void beforeTxnEventGeneration ( final AtomicReference< Req >  
requestRef , final AtomicReference< Res > responseRef , final  
AtomicReference< Long > txnIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **txnIdRef** –

Usage

This method does nothing and can be overridden

beforeUnlockTransaction(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >) method

Pre processing step.

Syntax

```
void beforeUnlockTransaction ( final AtomicReference< Req >  
requestRef , final AtomicReference< Res > responseRef , final  
AtomicReference< CallerInformation > callerRef , final  
AtomicReference< Long > txnIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –

- **callerRef** –
- **txnIdRef** –

Usage

This method does nothing and can be overridden

cleanup(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >, final AtomicReference< Long >) method

Performs a cleanup by unlocking the transaction and generate a transaction event.

Syntax

```
void cleanup ( final AtomicReference< Req > requestRef , final
AtomicReference< Res > responseRef , final AtomicReference<
CallerInformation > callerRef , final AtomicReference< Long >
txnIdRef , final AtomicReference< Long > subTxnIdRef )
```

Parameters

- **actorId** –
- **result** –
- **captureId** –
- **authCancelId** –

cleanupInit(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >) method

Performs a transaction cleanup by readjusting the balances(with pre- and post processing step) and call the transaction flow overlay if present.

Syntax

```
void cleanupInit ( final AtomicReference< Req > requestRef , final
AtomicReference< Res > responseRef , final AtomicReference<
CallerInformation > callerRef , final AtomicReference< Long >
authIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –

commissionCheck(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >) method
Perform the commission check action surrounded by pre and post call.

Syntax

```
final void commissionCheck ( final AtomicReference< Req >
requestRef , final AtomicReference< Res > responseRef , final
AtomicReference< CallerInformation > callerRef , final
AtomicReference< Long > authIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –

computeAuthentication(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >) method
Compute the authentication methods with pre- and post- processing step.

Syntax

```
AuthenticationMethodComputation computeAuthentication ( final
AtomicReference< Req > requestRef , final AtomicReference< Res >
responseRef , final AtomicReference< CallerInformation > callerRef ,
final AtomicReference< Long > authIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –

continueSuspendedAuthorisation(final Long, Req, Res, final Long, final Long) method
Continues a suspended transaction.

Syntax

```
void continueSuspendedAuthorisation ( final Long actorId , Req
request , Res response , final Long authId , final Long txnId )
```

Parameters

- **actorId** –
- **request** –
- **response** –
- **authId** –
- **txnId** –

doAuthorisation(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >, final AtomicReference< Long >) method

The authorization step is a multiple call step.

Syntax

```
boolean doAuthorisation ( final AtomicReference< Req > requestRef ,
    final AtomicReference< Res > responseRef , final AtomicReference<
    CallerInformation > callerRef , final AtomicReference< Long >
    authIdRef , final AtomicReference< Long > txnIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –
- **txnIdRef** –

Usage

First it calls the beforeDoAuthorisation. Afterwards the maker/checker workflow processing is invoked, followed by calling the authorisation logic. The following call is the post doAuthorisation. The final method is

```
ITransactionFlowOverlay#afterAuthorisationSuccess(IMoneyRequest,
ITransactionResponse, AtomicReference, AtomicReference)
```

doAuthorisationCancel(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Integer >, final AtomicReference< Long >, AtomicReference< Long >) method
 AuhtorizationCancel with pre and post step.

Syntax

```
void doAuthorisationCancel ( final AtomicReference< Req >
    requestRef , final AtomicReference< Res > responseRef , final
    AtomicReference< CallerInformation > callerRef , final
```

```
AtomicReference< Integer > errorCodeRef , final AtomicReference< Long > authCancelIdRef , AtomicReference< Long > authIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **errorCodeRef** –
- **authCancelIdRef** –
- **authIdRef** –

Usage

Finally the

```
ITransactionFlowOverlay#afterAuthorisationCancelSuccess(IMoneyRequest, ITransactionResponse, AtomicReference, AtomicReference, AtomicReference)
```

method is called

```
doCapture(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >, final AtomicReference< Long >) method
```

Performs the capture with pre and post processing step.

Syntax

```
void doCapture ( final AtomicReference< Req > requestRef , final AtomicReference< Res > responseRef , final AtomicReference< CallerInformation > callerRef , final AtomicReference< Long > captureIdRef , final AtomicReference< Long > authIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **captureIdRef** –
- **authIdRef** –

Usage

It also calls the commision check after successful capture. Finally the method

```
ITransactionFlowOverlay#afterCaptureSuccess(IMoneyRequest, ITransactionResponse, AtomicReference, AtomicReference, AtomicReference)
```


is called

doPreAuthorisation(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >) method
Calls the preAuthorisation logic with a pre and post processing step.

Syntax

```
void doPreAuthorisation ( final AtomicReference< Req > requestRef ,  
final AtomicReference< Res > responseRef , final AtomicReference<  
CallerInformation > callerRef , final AtomicReference< Long >  
authIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –

generateSubTxnEvent(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< Long >, final AtomicReference< CallerInformation >) method

Generate a transaction event (based on sub transaction reference) by calling the transaction event action service.

Syntax

```
void generateSubTxnEvent ( final AtomicReference< Req >  
requestRef , final AtomicReference< Res > responseRef , final  
AtomicReference< Long > subTxnIdRef , final AtomicReference<  
CallerInformation > callerRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **subTxnIdRef** –
- **callerRef** –

Usage

This also includes a pre and post processing step

generateTxnEvent(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< Long >, final AtomicReference< CallerInformation >) method
Generate a transaction event (based on main transaction reference) by calling the transaction event action service.

Syntax

```
void generateTxnEvent ( final AtomicReference< Req > requestRef ,  
    final AtomicReference< Res > responseRef , final AtomicReference<  
    Long > txnIdRef , final AtomicReference< CallerInformation >  
    callerRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **txnIdRef** –
- **callerRef** –

Usage

This also includes a pre and post processing step

initFollowUpAuthorisationCancel(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >) method

This step initializes the follow up authorization cancel including pre- and post processing step.

Syntax

```
Long initFollowUpAuthorisationCancel ( final AtomicReference<  
    Req > requestRef , final AtomicReference< Res > responseRef , final  
    AtomicReference< CallerInformation > callerRef , final  
    AtomicReference< Long > authIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –

initFollowUpCapture(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >) method
Initialize a capture after authorization.

Syntax

```
Long initFollowUpCapture ( final AtomicReference< Req >
    requestRef , final AtomicReference< Res > responseRef , final
    AtomicReference< CallerInformation > callerRef , final
    AtomicReference< Long > authIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –

Usage

This is called when the autocapture flag is present. The initialization is surrounded by pre and post call

isAuthenticated(final long) method

Check the authentication status of a transaction using the captureAction.

Syntax

```
boolean isAuthenticated ( final long systemId )
```

Parameters

- **systemId** –

isAutoCapture(final long) method

Check via captureAction if this transaction is a auto capture transaction.

Syntax

```
boolean isAutoCapture ( final long systemId )
```

Parameters

- **systemId** –

limitCheck(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >, final AtomicReference< Boolean >) method

Check the limits of payer and payee by calling the limit action.

Syntax

```
void limitCheck ( final AtomicReference< Req > requestRef, final  
AtomicReference< Res > responseRef, final AtomicReference<  
CallerInformation > callerRef, final AtomicReference< Long >  
authIdRef, final AtomicReference< Boolean > persistRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –
- **persistRef** –

Usage

A pre- and post processing step is included

restrictionCheck(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >, final AtomicReference< Boolean >) method

Perform the restriction check action with pre and post processing call.

Syntax

```
void restrictionCheck ( final AtomicReference< Req > requestRef,  
final AtomicReference< Res > responseRef, final AtomicReference<  
CallerInformation > callerRef, final AtomicReference< Long >  
authIdRef, final AtomicReference< Boolean > persistRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **authIdRef** –
- **persistRef** –

setAdjustBalancesAction(final IAdjustBalancesAction) method

Spring DM service injection.

Syntax

```
void setAdjustBalancesAction ( final IAdjustBalancesAction  
adjustBalancesAction )
```

Parameters

- **adjustBalancesAction** – the adjustBalancesAction to set

setAuthenticationAction(final IAuthenticationAction) method

Spring DM service injection.

Syntax

```
void setAuthenticationAction ( final IAuthenticationAction  
authenticationAction )
```

Parameters

- **authenticationAction** – the authenticationAction to set

setAuthorisationAction(final IAuthorisationAction) method

Spring DM service injection.

Syntax

```
void setAuthorisationAction ( final IAuthorisationAction  
authorisationAction )
```

Parameters

- **authorisationAction** – the authorisationAction to set

setCancelAction(ICancelAction) method

Spring DM service injection.

Syntax

```
void setCancelAction ( ICancelAction cancelAction )
```

Parameters

- **cancelAction** –

setCaptureAction(final ICaptureAction) method

Spring DM service injection.

Syntax

```
void setCaptureAction ( final ICaptureAction captureAction )
```

Parameters

- **captureAction** – the captureAction to set

setCommissionAction(final ICommissionAction) method

Spring DM service injection.

Syntax

```
void setCommissionAction ( final ICommissionAction  
commissionAction )
```

Parameters

- **commissionAction** – the commissionAction to set

setDelegate(ITransactionFlowOverlay< IMoneyRequest, ITransactionResponse >)
method

Spring DM service injection.

Syntax

```
void setDelegate ( ITransactionFlowOverlay< IMoneyRequest,  
ITransactionResponse > delegate )
```

Parameters

- **delegate** –

setInitAction(final IInitAction) method

Spring DM service injection.

Syntax

```
void setInitAction ( final IInitAction initAction )
```

Parameters

- **initAction** – the initAction to set

setLimitAction(final ILimitAction) method

Spring DM service injection.

Syntax

```
void setLimitAction ( final ILimitAction limitAction )
```

Parameters

- **limitAction** – the limitAction to set

setMakerCheckerAction(IMakerCheckerAction< Req, Res >) method

Spring DM service injection.

Syntax

```
void setMakerCheckerAction ( IMakerCheckerAction< Req, Res > makerCheckerAction )
```

Parameters

- **makerCheckerAction** –

setRestrictionAction(final IRestrictionAction) method

Spring DM service injection.

Syntax

```
void setRestrictionAction ( final IRestrictionAction restrictionAction )
```

Parameters

- **restrictionAction** – the restrictionAction to set

setTransactionEventAction(ITransactionEventAction) method

Spring DM service injection.

Syntax

```
void setTransactionEventAction ( ITransactionEventAction transactionEventAction )
```

Parameters

- **transactionEventAction** –

unlockTransaction(final AtomicReference< Req >, final AtomicReference< Res >, final AtomicReference< CallerInformation >, final AtomicReference< Long >) method
Unlock a locked transaction.

Syntax

```
void unlockTransaction ( final AtomicReference< Req > requestRef ,  
    final AtomicReference< Res > responseRef , final AtomicReference<  
    CallerInformation > callerRef , final AtomicReference< Long >  
    txnIdRef )
```

Parameters

- **requestRef** –
- **responseRef** –
- **callerRef** –
- **txnIdRef** –

Usage

This step also include pre- and post processing steps

adjustBalancesAction variable

Service reference to the AdjustBalancesAction service.

Syntax

```
IAdjustBalancesAction adjustBalancesAction
```

authenticationAction variable

Service reference to the AuthenticationAction service.

Syntax

```
IAuthenticationAction authenticationAction
```

authorisationAction variable

Service reference to the AuthorisationAction service.

Syntax

```
IAuthorisationAction authorisationAction
```


cancelAction variable

Service reference to the CancelAction service.

Syntax

```
ICancelAction cancelAction
```

captureAction variable

Service reference to the CaptureAction service.

Syntax

```
ICaptureAction captureAction
```

commissionAction variable

Service reference to the CommissionAction service.

Syntax

```
ICommissionAction commissionAction
```

delegate variable

Service reference to the TransactionFlowOverlay service.

Syntax

```
ITransactionFlowOverlay< IMoneyRequest, ITransactionResponse  
> delegate
```

Usage

This can be used to insert custom steps in the transaction processing. See
ITransactionFlowOverlay

initAction variable

Service reference to the InitAction service.

Syntax

```
IInitAction initAction
```

limitAction variable

Service reference to the LimitAction service.

Syntax

```
ILimitAction limitAction
```

makerCheckerAction variable

Service reference to the MakerCheckerAction service.

Syntax

```
IMakerCheckerAction< Req, Res > makerCheckerAction
```

restrictionAction variable

Service reference to the RestrictionAction service.

Syntax

```
IRestrictionAction restrictionAction
```

transactionEventAction variable

Service reference to the TransactionEventAction service.

Syntax

```
ITransactionEventAction transactionEventAction
```

AbstractMoneyRefTransactionFlow< Req extends IMoneyRefTransactionRequest, Res extends ITransactionResponse > class

Abstract wrapper for AbstractMoneyFlow.

Syntax

```
public class AbstractMoneyRefTransactionFlow< Req extends  
IMoneyRefTransactionRequest, Res extends ITransactionResponse  
>
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

ITransactionFlowOverlay< Req extends IMoneyRequest, Res extends ITransactionResponse > interface

Syntax

```
public interface ITransactionFlowOverlay< Req extends  
IMoneyRequest, Res extends ITransactionResponse >
```

Usage

The ITransactionFlowOverlay Interface can be used to customize the standard transaction processing by adding customized code that is called during the processing. By

implementing the TransactionFlowOverlay interface and registering the implementation as OSGi service you can directly inject your logic into the standard flow. An implementation of this interface can be used to

- modify the incoming request (for example change use case),
- add data to the outgoing response,
- add additional fees,
- perform additional limit checks.

Once the bundle is deployed in the container it will be invoked for all financial transactions. If you just want to execute your code for a subset of the transactions you have to filter for those. Please see the Money Mobiliser Customization Guide, Chapter 5.3, Transaction Overlay for more details

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

afterAuthorisationCancelSuccess(final Req, final Res, final AtomicReference< CallerInformation >, final AtomicReference< Long >, final AtomicReference< Long >)
method

Invoked after a successful authorisation cancel.

Syntax

```
void afterAuthorisationCancelSuccess ( final Req request , final
Res response , final AtomicReference< CallerInformation > callerRef ,
final AtomicReference< Long > authCancelIdRef , final
AtomicReference< Long > authIdRef )
```

Parameters

- **request** –
- **response** –
- **callerRef** –
- **authCancelIdRef** –
- **authIdRef** –

Usage

Authorisation Cancel can be either invoked as a separate service call or is invoked internally in case of a failed auto-capture.

Any exception thrown by this method will be swallowed and will not affect the transaction outcome - unless something bad happens on the DB and the commit charge can not be stored to the DB.

afterAuthorisationSuccess(final Req, final Res, final AtomicReference< CallerInformation >, final AtomicReference< Long >) method

This method is called in case the authorisation of the transaction was successful.

Syntax

```
void afterAuthorisationSuccess ( final Req request , final Res response , final AtomicReference< CallerInformation > callerRef , final AtomicReference< Long > authIdRef )
```

Parameters

- **request** –
- **response** –
- **callerRef** –
- **authIdRef** –

Usage

In case of auto-capture the capture will happen right away otherwise the response is returned to the caller.

Any exception thrown by this method will be swallowed and will not affect the transaction outcome - unless something bad happens on the DB and the commit charge can not be stored to the DB.

afterCaptureCancelSuccess(final Req, final Res, final AtomicReference< CallerInformation >, final AtomicReference< Long >, AtomicReference< Long >) method

This method is invoked when a Capture Cancel was successfully processed.

Syntax

```
void afterCaptureCancelSuccess ( final Req request , final Res response , final AtomicReference< CallerInformation > callerRef , final AtomicReference< Long > captCancelIdRef , AtomicReference< Long > captIdRef )
```

Parameters

- **request** –
- **response** –
- **callerRef** –
- **captCancelIdRef** –

- **captIdRef** –

Usage

This can happen either as a dedicated service call or as part of an internal Capture cleanup.

Any exception thrown by this method will be swallowed and will not affect the transaction outcome - unless something bad happens on the DB and the commit charge can not be stored to the DB.

afterCaptureSuccess(final Req, final Res, final AtomicReference< CallerInformation >, final AtomicReference< Long >, AtomicReference< Long >) method

Called after a successful Capture has occurred.

Syntax

```
void afterCaptureSuccess ( final Req request , final Res response ,
final AtomicReference< CallerInformation > callerRef , final
AtomicReference< Long > captIdRef , AtomicReference< Long >
authIdRef )
```

Parameters

- **request** –
- **response** –
- **callerRef** –
- **captIdRef** –
- **authIdRef** –

Usage

Capture is either invoked as a separate service call or as part of the standard authorisation flow.

Any exception thrown by this method will be swallowed and will not affect the transaction outcome - unless something bad happens on the DB and the commit charge can not be stored to the DB.

afterInitTransaction(final Req, final Res, final AtomicReference< CallerInformation >, final AtomicReference< Long >, boolean) method

At this stage the transaction is already fully initialized and all standard business logic checks have been executed.

Syntax

```
void afterInitTransaction ( final Req request , final Res response ,  
final AtomicReference< CallerInformation > callerRef , final  
AtomicReference< Long > authIdRef , boolean persist )
```

Parameters

- **request** –
- **response** –
- **callerRef** –
- **authIdRef** –

Usage

Fees are already attached to the transaction. If the transaction is failing after this call and before the

`afterAuthorisationSuccess`

is called the

`onFailure`

method is executed. Any BL checks that should fail the transactions should be executed here and a

`MobiliserServiceException`

should be thrown.

beforeCreateTransaction(final Req, final Res, final AtomicReference< CallerInformation >) method

Provides access to request and response for manipulation.

Syntax

```
void beforeCreateTransaction ( final Req request , final Res  
response , final AtomicReference< CallerInformation > callerRef )
```

Parameters

- **request** –
- **response** –
- **callerRef** –

Usage

No data should be persisted at this step. In case of failure, no method is called. Mainly provides access to request (and response) for manipulation.

cleanupInitTransaction(final Req, final Res, final AtomicReference< CallerInformation >, final AtomicReference< Long >) method

This method is invoked in case of a failure after the call to afterInit and before the Authorisation was successful.

Syntax

```
void cleanupInitTransaction (final Req request, final Res response,
final AtomicReference< CallerInformation > callerRef, final
AtomicReference< Long > authIdRef)
```

Parameters

- **request** –
- **response** –
- **callerRef** –
- **authIdRef** –

Usage

Any exception thrown by this method will be swallowed and will not affect the transaction outcome - unless something bad happens on the DB and the commit charge can not be stored to the DB.

getName() method

Syntax

```
String getName ()
```

Returns

the name.

Usage

Returns the name of the ITransactionFlowOverlay. Can be the class name or a short description. This is shown by the broker in the log file

the name.

getRanking() method

Returns the ranking of this implementation.

Syntax

```
int getRanking ()
```

Usage

Implementations with higher rankings are executed first. To provide a ranking usually only makes sense if you set a value in relation to another.

payinvoice package

Members

All public members of the payinvoice package.

- **IPayInvoice< Req extends IPayInvoiceRequest, Res extends IPayInvoiceResponse > interface** – Defines the pay-invoice service.

IPayInvoice< Req extends IPayInvoiceRequest, Res extends IPayInvoiceResponse > interface

Defines the pay-invoice service.

Syntax

```
public interface IPayInvoice< Req extends IPayInvoiceRequest,  
Res extends IPayInvoiceResponse >
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

payInvoice(final CallerInformation, final Req, final Res) method

Implements the pay-invoice flow to pay open invoices.

Syntax

```
void payInvoice ( final CallerInformation caller , final Req request ,  
final Res response )
```


Parameters

- **caller** – the current caller
- **request** – the request data
- **response** – the response data

preauth package*Members*

All public members of the preauth package.

- **IPreAuthorisation< Req extends IPreAuthorisationRequest, Res extends IPreAuthorisationResponse > interface** – Implements the pre-authorization service.
- **IPrePickupMoney< Req extends IPrePickupMoneyRequest, Res extends IPrePickupMoneyResponse > interface** – Implements the pre-pickup money service, i.e.
- **IStartVoucher< Req extends IStartVoucherRequest, Res extends IStartVoucherResponse > interface** – Implements the start-voucher service.

IPreAuthorisation< Req extends IPreAuthorisationRequest, Res extends IPreAuthorisationResponse > interface

Implements the pre-authorization service.

Syntax

```
public interface IPreAuthorisation< Req extends
IPreAuthorisationRequest, Res extends
IPreAuthorisationResponse >
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

preAuthorisation(final CallerInformation, final Req, final Res) method

Implements the pre-authorization flow.

Syntax

```
void preAuthorisation ( final CallerInformation caller , final Req
request , final Res response )
```

Parameters

- **caller** – the current caller

- **request** – the request data
- **response** – the response data

IPrePickupMoney< Req extends IPrePickupMoneyRequest, Res extends IPrePickupMoneyResponse > interface

Implements the pre-pickup money service, i.e.

Syntax

```
public interface IPrePickupMoney< Req extends
IPrePickupMoneyRequest, Res extends IPrePickupMoneyResponse
>
```

Usage

a validation/pre-authorization step.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

prePickupMoney(final CallerInformation, final Req, final Res, String) method

Provides the pre-pickup money flow to pre-auth a voucher pickup.

Syntax

```
void prePickupMoney ( final CallerInformation caller , final Req
request , final Res response , String pickupCode )
```

Parameters

- **caller** – the current caller
- **request** – the request data
- **response** – the response data
- **pickupCode** – the voucher pickup code

IStartVoucher< Req extends IStartVoucherRequest, Res extends IStartVoucherResponse > interface

Implements the start-voucher service.

Syntax

```
public interface IStartVoucher< Req extends
IStartVoucherRequest, Res extends IStartVoucherResponse >
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

startVoucher(final CallerInformation, final Req, final Res) method

Provides the start-voucher flow that initiates a voucher pickup transaction.

Syntax

```
void startVoucher ( final CallerInformation caller , final Req
request , final Res response )
```

Parameters

- **caller** – the current caller
- **request** – the request data
- **response** – the response data

preauthcont package

Members

All public members of the preauthcont package.

- **IConfirmVoucher< Req extends IConfirmVoucherRequest, Res extends IConfirmVoucherResponse > interface** – Implements the confirm voucher service, to complete a start-voucher transaction.
- **IPreAuthorisationContinue< Req extends IPreAuthorisationContinueRequest, Res extends IPreAuthorisationContinueResponse > interface** – Implements the pre-authorization continue service to continue on a previous pre-authorization.

IConfirmVoucher< Req extends IConfirmVoucherRequest, Res extends IConfirmVoucherResponse > interface

Implements the confirm voucher service, to complete a start-voucher transaction.

Syntax

```
public interface IConfirmVoucher< Req extends
IConfirmVoucherRequest, Res extends IConfirmVoucherResponse
>
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

confirmVoucher(final CallerInformation, final Req, final Res) method

Provides the confirm voucher flow.

Syntax

```
void confirmVoucher ( final CallerInformation caller , final Req  
request , final Res response )
```

Parameters

- **caller** – the current caller
- **request** – the request data
- **response** – the response data

IPreAuthorisationContinue< Req extends IPreAuthorisationContinueRequest, Res extends IPreAuthorisationContinueResponse > interface

Implements the pre-authorization continue service to continue on a previous pre-authorization.

Syntax

```
public interface IPreAuthorisationContinue< Req extends  
IPreAuthorisationContinueRequest, Res extends  
IPreAuthorisationContinueResponse >
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

preAuthorisationContinue(final CallerInformation, final Req, final Res) method

Provides the pre-auth continue flow.

Syntax

```
void preAuthorisationContinue ( final CallerInformation caller ,  
final Req request , final Res response )
```

Parameters

- **caller** – the current caller
- **request** – the request data
- **response** – the response data

process package

Members

All public members of the process package.

- **IMakerCheckerContinue interface** –
- **ITransactionFlowControl interface** –
- **TxnWorkflowFailedException class** – Extends `MobiliserServiceException` and is thrown if an error occurred during the transaction workflow processing.

IMakerCheckerContinue interface

Syntax

```
public interface IMakerCheckerContinue
```

Usage

This interface defines how a suspended transaction (authorisation) is continued.

The interface is used by the reference transaction workflow definition. It continues the authorisation where it was paused, which means no the pre-authorisation or limit/balance checks will not be repeated.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

continuePendingAuthorisation(final Long, final List< IKeyValue >, final List< IKeyValue >, final Long, final Long) method

This method is called in case a suspended transaction workflow task was approved by the `ITransactionFlowControl` and the activiti framework proceeds with the execution of the workflow.

Syntax

```
void continuePendingAuthorisation ( final Long actorId , final
List< IKeyValue > requestUnstructuredData , final List< IKeyValue >
responseUnstructuredData , final Long authId , final Long txnId ) throws
EntityNotFoundException
```

Parameters

- **actorId** –

- **requestUnstructuredData** – Any unstructured data of the original `IMoneyRequest` request
- **responseUnstructuredData** – Any unstructured data of the original `ITransactionResponse` response
- **authId** – The authorisation ID
- **txnId** – The reference txn ID

Exceptions

- **EntityNotFoundException** –

rejectPendingAuthorisation(final Long, final List< IKeyValue >, final List< IKeyValue >, final Long, final Long) method

This method is called in case a suspended transaction workflow task was rejected by the `ITransactionFlowControl` and the activiti framework proceeds with the execution of the workflow.

Syntax

```
void rejectPendingAuthorisation ( final Long actorId , final List< IKeyValue > requestUnstructuredData , final List< IKeyValue > responseUnstructuredData , final Long authId , final Long txnId )
```

Parameters

- **actorId** –
- **requestUnstructuredData** – Any unstructured data of the original `IMoneyRequest` request
- **responseUnstructuredData** – Any unstructured data of the original `ITransactionResponse` response
- **authId** – The authorisation ID
- **txnId** – The reference txn ID

ITransactionFlowControl interface

Syntax

```
public interface ITransactionFlowControl
```

Usage

This interface defines common actions required to implement a workflow handling a maker/checker process for the execution of transactions

The reference implementation uses the activiti workflow to start a new workflow and store a complete `IMoneyRequest` call by serializing it into the activiti context (database tables). The continue method then tells activiti to finish the workflow based on our workflow

definition and loads the previously stored request from the activiti context and either executes it in the approval case, or dismisses it and takes further defined actions in case of the reject case. The find method allows to search for "paused" transaction entries based on different criteria, which are matched against the stored transactions requests.

To avoid circular dependencies this interface is based in the transaction businesslogic rather than the transaction workflow bundle (which contains the reference implementation)

At the moment only `IAuthorisationRequest` are supported.

This interface (and its reference implementation) is used internally by the `IMakerCheckerContinue` implementation.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

continueProcess(final ContinuePendingTransactionRequest, final long, final List< String >) method

Continues a pending transaction request.

Syntax

```
ContinuePendingTransactionResponse continueProcess ( final
ContinuePendingTransactionRequest request , final long callerId ,
final List< String > privileges ) throws TxnWorkflowFailedException
```

Parameters

- **request** –
- **callerId** –
- **privileges** – Must contain all the caller privileges, which are checked against the privileges required to perform the continue operation.

Exceptions

- **TxnWorkflowFailedException class** –

getTransactionTaskId(final long, final long, final List< String >) method

Fetches the current task ID for a suspended transaction identified by its ID.

Syntax

```
GetTransactionTaskIdResponse getTransactionTaskId ( final long
txnId , final long callerId , final List< String > privileges ) throws
EntityNotFoundException
```

Parameters

- **txnId** –
- **callerId** –
- **privileges** – Must contain all the caller privileges, which are checked against the privileges required to perform the continue operation.

Exceptions

- **EntityNotFoundException** –

startProcess(final IMoneyRequest, final ITransactionResponse, final CallerInformation, final Long, final Long, final Integer) method

Starts a new maker/checker workflow process for the given transaction (authorisation) request.

Syntax

```
void startProcess ( final IMoneyRequest request , final  
ITransactionResponse response , final CallerInformation caller ,  
final Long authId , final Long txnId , final Integer useCaseId ) throws  
TxnWorkflowFailedException
```

Parameters

- **request** – The request to handle
- **response** –
- **caller** –
- **authId** – The authorisation txn ID
- **txnId** – The reference txn ID
- **useCaseId** –

Exceptions

- **TxnWorkflowFailedException class** –

TxnWorkflowFailedException class

Extends *MobiliserServiceException* and is thrown if an error occurred during the transaction workflow processing.

Syntax

```
public class TxnWorkflowFailedException
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

TxnWorkflowFailedException() constructor

Syntax

`TxnWorkflowFailedException ()`

TxnWorkflowFailedException(String) constructor

Syntax

`TxnWorkflowFailedException ( String message )`

TxnWorkflowFailedException(Throwable) constructor

Syntax

`TxnWorkflowFailedException ( Throwable cause )`

TxnWorkflowFailedException(String, Throwable) constructor

Syntax

`TxnWorkflowFailedException ( String message , Throwable cause )`

getErrorCode() method

Syntax

`int getErrorCode ()`

transcancel package

Members

All public members of the transcancel package.

- **ITransactionCancel< Req extends ITransactionCancelRequest, Res extends ITransactionCancelResponse > interface** – Implementations of this interface can cancel an initial transaction.

ITransactionCancel< Req extends ITransactionCancelRequest, Res extends ITransactionCancelResponse > interface

Implementations of this interface can cancel an initial transaction.

Syntax

```
public interface ITransactionCancel< Req extends  
ITransactionCancelRequest, Res extends  
ITransactionCancelResponse >
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

transactionCancel(final CallerInformation, final Req, final Res) method

Cancel the referenced transaction.

Syntax

```
void transactionCancel ( final CallerInformation caller, final Req  
request, final Res response )
```

Parameters

- **caller** – the current caller
- **request** – the request parameters
- **response** – the response parameters

webcont package

Members

All public members of the webcont package.

- **IWebContinue< Req extends IWebContinueRequest, Res extends IWebContinueResponse > interface** – Implements the web-continue service to complete a started web transaction.

IWebContinue< Req extends IWebContinueRequest, Res extends IWebContinueResponse > interface

Implements the web-continue service to complete a started web transaction.

Syntax

```
public interface IWebContinue< Req extends  
IWebContinueRequest, Res extends IWebContinueResponse >
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

webContinue(final CallerInformation, final Req, final Res) method

Provides the web continue flow.

Syntax

```
void webContinue ( final CallerInformation caller , final Req
request , final Res response )
```

Parameters

- **caller** – the current caller
- **request** – the request parameters
- **response** – the response parameters

webstart package

Members

All public members of the webstart package.

- **IWebStart< Req extends IWebStartRequest, Res extends IWebStartResponse > interface** – Implements the web start service to initiate a web-based transaction.

IWebStart< Req extends IWebStartRequest, Res extends IWebStartResponse > interface

Implements the web start service to initiate a web-based transaction.

Syntax

```
public interface IWebStart< Req extends IWebStartRequest, Res
extends IWebStartResponse >
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

webStart(final CallerInformation, final Req, final Res) method

Provides the web start flow.

Syntax

```
void webStart ( final CallerInformation caller , final Req request ,  
final Res response )
```

Parameters

- **caller** – the current caller
- **request** – the request parameters
- **response** – the response parameters

IAmountLogic interface

Simple abstract to allow for calculating the amounts pertaining to a given SubTransaction.

Syntax

```
public interface IAmountLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

calculateAmounts(final SubTransaction) method

Calculate the amounts for this subtransaction.

Syntax

```
CalculatedAmount calculateAmounts ( final SubTransaction txn )
```

Parameters

- **txn** – the sub transaction

Returns

the calculated amounts.

Usage

the calculated amounts.

IAuthorisationLogic interface

Logic for authorizing transactions.

Syntax

```
public interface IAuthorisationLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

authorisationCancel(final AuthorisationCancel, final List< IKeyValue >, final List< IKeyValue >, final Long) method

Syntax

```
void authorisationCancel ( final AuthorisationCancel txn , final
List< IKeyValue > unstructuredDataRequest , final List< IKeyValue >
unstructuredDataResponse , final Long actorId )
```

Parameters

- **txn** – the txn
- **unstructuredDataRequest** – additional data from the request
- **unstructuredDataResponse** – additional data for the response
- **actorId** – the current actor

Usage

Delegates the authorization cancel to the appropriate payment handler.

authorise(final Authorisation, final List< IKeyValue >, final List< IKeyValue >, final Long) method

Authorize the given transaction via a payment handler.

Syntax

```
String authorise ( final Authorisation txn , final List<
IKeyValue > unstructuredDataRequest , final List< IKeyValue >
unstructuredDataResponse , final Long actorId )
```

Parameters

- **txn** – the txn
- **unstructuredDataRequest** – additional data from the request
- **unstructuredDataResponse** – additional data for the response
- **actorId** – the current actor

Returns

the authorization code

Usage

the authorization code

fillAuthenticationContinueResponse(Long, IAuthenticationContinueResponse, Long) method

Syntax

```
void fillAuthenticationContinueResponse ( Long txnId ,  
IAAuthenticationContinueResponse response , Long actorId )
```

Parameters

- **txnId** – the transaction id of the authorization sub transaction
- **response** – the response to fill
- **actorId** – the current actor

Usage

Fill the authentication continue response.

fillAuthorisationResponse(final Long, final IAuthorisationResponse, final Long) method

Syntax

```
void fillAuthorisationResponse ( final Long authId , final  
IAuthorisationResponse response , final Long actorId )
```

Parameters

- **authId** – the transaction id of the authorization sub transaction
- **response** – the response to fill
- **actorId** – the current actor

Usage

Fill the authorisation response.

fillPreAuthenticationContinueResponse(Long, IPreAuthorisationContinueResponse, Long) method

Syntax

```
void fillPreAuthenticationContinueResponse ( Long txnId ,  
IPreAuthorisationContinueResponse response , Long actorId )
```

Parameters

- **txnId** – the transaction id of the authorization sub transaction
- **response** – the response to fill
- **actorId** – the current actor

Usage

Fill the pre-authentication continue response.

fillPreAuthorisationResponse(final Long, final IPreAuthorisationResponse, final Long) method

Syntax

```
void fillPreAuthorisationResponse ( final Long authId , final  
IPreAuthorisationResponse response , final Long actorId )
```

Parameters

- **authId** – the transaction id of the authorization sub transaction
- **response** – the response to fill
- **actorId** – the current actor

Usage

Fill the preauthorisation response.

fillWebContinueResponse(Long, IWebContinueResponse, Long) method

Syntax

```
void fillWebContinueResponse ( Long txnId , IWebContinueResponse  
response , Long actorId )
```

Parameters

- **txnId** – the transaction id of the authorization sub transaction
- **response** – the response to fill
- **actorId** – the current actor

Usage

Fill the web continue response.

preAuthorise(final Authorisation, final List< IKeyValue >, final List< IKeyValue >, final Long) method

Syntax

```
void preAuthorise ( final Authorisation txn , final List< IKeyValue > unstructuredDataRequest , final List< IKeyValue > unstructuredDataResponse , final Long actorId )
```

Parameters

- **txn** – the txn
- **unstructuredDataRequest** – additional data from the request
- **unstructuredDataResponse** – additional data for the response
- **actorId** – the current actor

Usage

Delegates the pre-authorisation to the appropriate payment handler. Note that the broker swallows

`UnsupportedOperationException`

s and proceeds as if the pre-authorisation was successful.

ICancelLogic interface

Logic for canceling transactions.

Syntax

```
public interface ICancelLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

cancel(final Transaction, final Integer, final Long) method

Cancels the transaction by setting an error code to it.

Syntax

```
void cancel ( final Transaction txn , final Integer errorCode , final Long actor )
```

Parameters

- **txn** – the transaction to cancel
- **errorCode** – the error code that should be set for the transaction
- **actor** – the current actor

ICaptureLogic interface

Logic for capturing transactions.

Syntax

```
public interface ICaptureLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

capture(final SubTransaction, Authorisation, final List< IKeyValue >, final List< IKeyValue >, final Long) method

Performs a capture on the passed transaction via the responsible payment handler.

Syntax

```
void capture ( final SubTransaction txn , Authorisation auth , final List< IKeyValue > unstructuredDataRequest , final List< IKeyValue > unstructuredDataResponse , final Long actorId )
```

Parameters

- **txn** – the capture txn
- **auth** – the authorization sub transaction
- **unstructuredDataRequest** – additional data from the request
- **unstructuredDataResponse** – additional data for the response
- **actorId** – the current actor

captureCancel(final SubTransaction, final List< IKeyValue >, final List< IKeyValue >, final Long) method

Performs a capture cancel on the passed transaction via the responsible payment handler.

Syntax

```
void captureCancel ( final SubTransaction txn , final List< IKeyValue > unstructuredDataRequest , final List< IKeyValue > unstructuredDataResponse , final Long actorId )
```

Parameters

- **txn** – the capture cancel txn
- **unstructuredDataRequest** – additional data from the request
- **unstructuredDataResponse** – additional data for the response
- **actorId** – the current actor

ICommissionClearingLogic interface

Logic for commission clearing.

Syntax

```
public interface ICommissionClearingLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CurrencyAmount class

This bean holds an amount with its currency.

Syntax

```
public class CurrencyAmount
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CurrencyAmount(final long, final String) constructor

Syntax

`CurrencyAmount ( final long amount , final String currency )`

CurrencyAmount(final long, final Currency) constructor

Syntax

`CurrencyAmount ( final long amount , final Currency currency )`

getAmount() method

Syntax

`long getAmount ()`

getCurrency() method

Syntax

`Currency getCurrency ()`

setAmount(final long) method

Syntax

`void setAmount ( final long amount )`

setCurrency(final Currency) method

Syntax

`void setCurrency ( final Currency currency )`

clearCommission(long, String, final String) method

Clears the commission for a customer, i.e.

Syntax

`ClearCommissionResponseBean clearCommission (long customerId ,
String cutoffTime , final String timeZone)`

Parameters

- **customerId** – the customer for who the commission should be cleared
- **cutoffTime** – the cut-off time - only commissions booked before that time will be settled

- **timeZone** – the time zone of the cut-off time

Returns

the outcome of the commission clearing.

Usage

the open commissions will be credited to that customer.

the outcome of the commission clearing.

getOpenCommissionAmountByConfiguration(final long) method

Returns the open commission amount.

Syntax

```
CurrencyAmount getOpenCommissionAmountByConfiguration ( final  
long commissionConfigurationId ) throws EntityNotFoundException
```

Parameters

- **commissionConfigurationId** – the configuration id to search for

Returns

the open commission amount for the passed configuration

Exceptions

- **EntityNotFoundException** – if the configuration is not found.

Usage

the open commission amount for the passed configuration

ICommissionLogic interface

Logic for commission calculation and handling.

Syntax

```
public interface ICommissionLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

calculateCommission(final SubTransaction, final Customer, final Customer, final Long) method

Calculates the commission based on the fee/commission configuration and creates commission (fee) entries for the passed transaction.

Syntax

```
void calculateCommission ( final SubTransaction txn , final
Customer payee , final Customer payer , final Long actor )
```

Parameters

- **txn** – the transaction to calculate the commission for
- **payee** – commission beneficiary
- **payer** – commission payer
- **actor** – the current actor

negateCommission(final SubTransaction, final SubTransaction) method

Creates reverse commission (fee) entries to negate an existing commission.

Syntax

```
void negateCommission ( final SubTransaction origSubTxn , final
SubTransaction cancelSubTxn )
```

Parameters

- **origSubTxn** – the original transaction having the commission assigned
- **cancelSubTxn** – the cancel transaction that must get the negative commission bookings.

Usage

This is required for transaction cancelations.

IContinueLogic interface

Logic for continuing a transaction after authentication.

Syntax

```
public interface IContinueLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

ContinueParams class

Result of continuing a transaction.

Syntax

```
public class ContinueParams
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

ContinueParams(final Long, final String, final Long) constructor

Syntax

```
ContinueParams ( final Long authorisationId , final String orderId ,  
final Long transactionId )
```

Parameters

- **authorisationId** –
- **orderId** –
- **transactionId** –

authorisationId variable

Syntax

```
final Long authorisationId
```

orderId variable

Syntax

```
final String orderId
```

transactionId variable

Syntax

```
final Long transactionId
```

authenticate(final AuthenticationMethodComputation, final ITransactionCredential, final Long, final Authorisation, final Transaction) method

Performs an authentication based on the passed authentication rules and (optional) credentials.

Syntax

```
ConsumerAuthenticationResult authenticate ( final
AuthenticationMethodComputation authenticationRules , final
ITransactionCredential credentials , final Long actorId , final
Authorisation auth , final Transaction txn )
```

Parameters

- **authenticationRules** – the authentication rules to apply
- **credentials** – the (optional) authentication credentials for payer and payee
- **actorId** – the current actor
- **auth** – the current authorization subtransaction.
- **txn** – the current main transaction.

Returns

the result of the authentication for payer and payee.

Usage

the result of the authentication for payer and payee.

continueAuthentication(final ITransaction, final ITransactionCredential, final CallerInformation) method

Continues the pending authentication with the required credentials.

Syntax

```
ContinueParams continueAuthentication ( final ITransaction
referencedTxn , final ITransactionCredential credentials , final
CallerInformation caller )
```

Parameters

- **referencedTxn** – the transaction that is pending authentication
- **credentials** – the credential(s) for authentication
- **caller** – the current actor

Returns

the authorization transaction id and additional information

Usage

the authorization transaction id and additional information

IDelegationLogic interface

Logic handling the delegation between actor and caller.

Syntax

```
public interface IDelegationLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

actorDelegation(final UseCase, final ITransactionParticipant, final CallerInformation)
method

Syntax

```
Identification actorDelegation ( final UseCase useCase , final  
ITransactionParticipant activeParticipant , final CallerInformation  
caller )
```

Parameters

- **useCase** – the use case
- **activeParticipant** – actor participant for the transaction
- **caller** – the caller

Returns

identification of the customer to be used as the actor

Usage

Handles the delegation for the actor. This method returns the customer that is to be used as actor. If no special actor element is given in the request, the caller will be returned. If the payer/payee element is given, this method retrieves the associated customer, checks if delegation is allowed and returns the acting payee/r's identification

identification of the customer to be used as the actor

IEventLogic interface

Logic to generate transaction events based on a given transaction/ system ID.

Syntax

```
public interface IEventLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

generateSubTxnEvent(long, long) method

Generates a subTransaction based event.

Syntax

```
void generateSubTxnEvent ( long txnId , long actor )
```

Parameters

- **txnId** – the subtransaction id
- **actor** – the current actor

Usage

It retrieves the transaction status based on the referenced main transaction status and uses the subTransaction error code

generateTxnEvent(long, long) method

Generates a transaction based event similar to generateSubTxnEvent(long, long).

Syntax

```
void generateTxnEvent ( long systemId , long actor )
```

Parameters

- **systemId** – the main transaction id
- **actor** – the current actor

IExchangeRateLogic interface

Logic for exchange rate handling and currency conversion.

Syntax

```
public interface IExchangeRateLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

convertByFractionCurrencyUnit(ExchangeRate, long) method

Syntax

BigDecimal convertByFractionCurrencyUnit (ExchangeRate *exchangeRate* , long *sourceAmount*)

Parameters

- **exchangeRate** – the exchange rate to use
- **sourceAmount** – the long amount to convert

Returns

the appropriate conversion amount as BigDecimal

Usage

Converts the specified amount of the source currency into the appropriate amount of the target currency.

the appropriate conversion amount as BigDecimal

convertByMainCurrencyUnit(ExchangeRate, BigDecimal) method

Syntax

BigDecimal convertByMainCurrencyUnit (ExchangeRate *exchangeRate* , BigDecimal *sourceAmount*)

Parameters

- **exchangeRate** – the exchange rate to use
- **sourceAmount** – the BigDecimal amount to convert

Returns

the appropriate conversion amount as BigDecimal

Usage

Converts the specified amount of the source currency into the appropriate amount of the target currency.

the appropriate conversion amount as `BigDecimal`

determineExchangeRates(final SubTransaction, Authorisation) method**Syntax**

```
ExchangeRateComputation determineExchangeRates ( final
SubTransaction txn , Authorisation authorisation )
```

Parameters

- **txn** – the `SubTransaction` to determine the exchange rates for
- **authorisation** – the authorisation subtransaction that can optionally be used to copy the exchange rates from. If the authorisation is not provided it must be loaded on demand by the implementation.

Returns

the computed exchange rates

Usage

Calculates the exchange rates that are to be considered for the specified transaction. The rates are stored into the database and available via

```
SubTransaction#getExchangeRatePayee()
```

and

```
SubTransaction#getExchangeRatePayer()
```

.

the computed exchange rates

getAdjustedRate(ExchangeRate) method**Syntax**

```
BigDecimal getAdjustedRate ( ExchangeRate exchangeRate ) throws
IllegalDataException
```

Parameters

- **exchangeRate** – the exchange rate to use

Returns

the adjusted exchange rate

Usage

Returns the exchange rate adjusted by the relation amounts (

getFromAmount()

and

getToAmount()

).

the adjusted exchange rate

reverseCalculationByFractionCurrencyUnit(ExchangeRate, long) method

Syntax

BigDecimal reverseCalculationByFractionCurrencyUnit
(ExchangeRate *exchangeRate* , long *targetAmount*)

Parameters

- **exchangeRate** – the exchange rate to use
- **targetAmount** – the long amount that is the intended result of a currency conversion

Returns

the required amount of the source currency to convert as BigDecimal

Usage

Performs a "reverse conversion", calculating the necessary amount of the source currency required to receive the specified amount of the target currency.

Note

that this method should NOT be used for a currency conversion in the other direction; it is intended to be used to calculate the "appropriate" source amount.

the required amount of the source currency to convert as BigDecimal

reverseCalculationByMainCurrencyUnit(ExchangeRate, BigDecimal) method

Syntax

BigDecimal reverseCalculationByMainCurrencyUnit (ExchangeRate
exchangeRate , BigDecimal *targetAmount*)

Parameters

- **exchangeRate** – the exchange rate to use
- **targetAmount** – the `BigDecimal` amount that is the intended result of a currency conversion

Returns

the required amount of the source currency to convert as `BigDecimal`

Usage

Performs a "reverse conversion", calculating the necessary amount of the source currency required to receive the specified amount of the target currency.

Note

that this method should NOT be used for a currency conversion in the other direction; it is intended to be used to calculate the "appropriate" source amount.

the required amount of the source currency to convert as `BigDecimal`

ExpirationLogic interface

Logic for transaction expiration handling.

Syntax

```
public interface IExpirationLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

computeAuthorisationExpirationDate(final Transaction, final Customer) method

Compute the expiration date for the given authorisation.

Syntax

```
Date computeAuthorisationExpirationDate ( final Transaction txn ,  
final Customer actingMerchant )
```

Parameters

- **txn** – the authorisation transaction
- **actingMerchant** – the acting merchant

Returns

the expiration date

Usage

the expiration date

IInitFollowUpTransaction< T extends SubTransaction > interface

Logic for creating follow-up subtransactions.

Syntax

```
public interface IInitFollowUpTransaction< T extends  
SubTransaction >
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

initialiseFollowUpTransaction(final SubTransaction, final CallerInformation) method

Returns a follow up transaction for the given SubTransaction.

Syntax

```
T initialiseFollowUpTransaction ( final SubTransaction origTxn ,  
final CallerInformation caller )
```

Parameters

- **origTxn** – the original transaction
- **caller** – the current caller

Returns

the subtraction having already been saved to persistent storage

Usage

the subtraction having already been saved to persistent storage

IInitTransactionLogic interface

Logic for initializing the different transaction types.

Syntax

```
public interface IInitTransactionLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

InitTransactionResult class

Result of initializing a transaction.

Syntax

```
public class InitTransactionResult
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

InitTransactionResult(final Long, final String, final Long) constructor

Creates a new result bean.

Syntax

```
InitTransactionResult ( final Long subTransactionId , final String  
orderId , final Long transactionId )
```

Parameters

- **subTransactionId** –
- **orderId** –
- **transactionId** –

orderId variable

Syntax

```
final String orderId
```

subTransactionId variable

Syntax

```
final Long subTransactionId
```

transactionId variable

Syntax

```
final Long transactionId
```

completeAuthorisation(ITransaction, SubTransaction, final ITransactionParticipant, final CallerInformation) method

Web-continue transactions use this method to complete the authorization subtransaction with the missing information about the payer of the transaction.

Syntax

```
void completeAuthorisation ( ITransaction referencedTxn ,  
SubTransaction auth , final ITransactionParticipant payer , final  
CallerInformation caller )
```

Parameters

- **referencedTxn** – the referenced transaction from the Mobiliser request
- **auth** – the open authorization subtransaction
- **payer** – the payer, which is missing to complete the authorization information
- **caller** – the current caller

createAuthorisation(final String, final String, final Transaction, final Long, final CallerInformation) method

Create the authorization transaction.

Syntax

```
Authorisation createAuthorisation ( final String origin , final  
String traceNo , final Transaction txn , final Long vatAmount , final  
CallerInformation caller )
```

Parameters

- **origin** – the origin for the transaction
- **traceNo** – the trace number for the transaction
- **txn** – the main transaction
- **vatAmount** – the VAT amount
- **caller** – the caller for this operation

Returns

the created authorization transaction.

Usage

the created authorization transaction.

initAuthorisation(final String, final String, final Transaction, final SubTransaction, final ITransactionParticipant, final ITransactionParticipant, final CallerInformation)

method

Initialize an authorization transaction.

Syntax

```
void initAuthorisation ( final String origin , final String traceNo ,  
final Transaction transaction , final SubTransaction auth , final  
ITransactionParticipant payer , final ITransactionParticipant  
payee , final CallerInformation caller )
```

Parameters

- **origin** – the origin for the transaction
- **traceNo** – the trace number for the transaction
- **transaction** – the main transaction
- **auth** – the authorization subtransaction
- **payer** – the payer of the transaction
- **payee** – the payee of the transaction
- **caller** – the caller for this operation

initAuthorisationCancel(final String, final String, final ITransaction, final CallerInformation) method

Initialize an authorization cancel transaction.

Syntax

```
InitTransactionResult initAuthorisationCancel ( final String  
origin , final String traceNo , final ITransaction referencedTxn ,  
final CallerInformation caller )
```

Parameters

- **origin** – the origin for the transaction
- **traceNo** – the trace number for the transaction
- **referencedTxn** – the referenced transaction
- **caller** – the caller for this operation

Returns

the id of the newly created transaction and additional information

Usage

the id of the newly created transaction and additional information

initCapture(final String, final String, final long, final Long, final ITransaction, final CallerInformation) method

Initialize a capture transaction.

Syntax

```
InitTransactionResult initCapture ( final String origin , final  
String traceNo , final long amount , final Long vatAmount , final  
ITransaction referencedTxn , final CallerInformation caller )
```

Parameters

- **origin** – the origin for the transaction
- **traceNo** – the trace number for the transaction
- **amount** – the capture amount
- **vatAmount** – the VAT amount
- **referencedTxn** – the referenced transaction
- **caller** – the caller for this operation

Returns

the id of the newly created transaction and additional information

Usage

the id of the newly created transaction and additional information

initCaptureCancel(final String, final String, final long, final Long, final ITransaction, final CallerInformation) method

Initialize a capture cancel transaction.

Syntax

```
InitTransactionResult initCaptureCancel ( final String origin ,  
final String traceNo , final long amount , final Long vatAmount ,  
final ITransaction referencedTxn , final CallerInformation caller )
```

Parameters

- **origin** – the origin for the transaction
- **traceNo** – the trace number for the transaction
- **amount** – the cancel amount

- **vatAmount** – the VAT amount
- **referencedTxn** – the referenced transaction
- **caller** – the caller for this operation

Returns

the id of the newly created transaction and additional information

Usage

the id of the newly created transaction and additional information

initTransactionCancel(final String, final String, final ITransaction, final CallerInformation) method

Initialize a transaction cancel.

Syntax

```
InitTransactionResult initTransactionCancel ( final String  
origin, final String traceNo, final ITransaction referencedTransaction ,  
final CallerInformation caller )
```

Parameters

- **origin** – the origin for the transaction
- **traceNo** – the trace number for the transaction
- **referencedTxn** – the referenced transaction
- **caller** – the caller for this operation

Returns

the id of the canceled transaction and additional information

Usage

the id of the canceled transaction and additional information

IInitWebTransaction interface

Logic for initializing web transactions.

Syntax

```
public interface IInitWebTransaction
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

initWebTransaction(final Transaction, final SubTransaction, final IWebStartRequest, final IWebStartResponse, final CallerInformation) method

Initialized an authorization for a web transaction.

Syntax

```
void initWebTransaction ( final Transaction txn , final  
SubTransaction auth , final IWebStartRequest request , final  
IWebStartResponse response , final CallerInformation caller )
```

Parameters

- **txn** – the main transaction
- **auth** – the authorization subtransaction
- **request** – the webstart request
- **response** – the webstart response
- **caller** – the current caller

IIInvoiceLogic interface

Logic for Invoice Transactions.

Syntax

```
public interface IIInvoiceLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

cancelInvoice(final long, final boolean, final Long) method

Syntax

```
void cancelInvoice ( final long invoiceId , final boolean  
cancelbyCustomer , final Long actorId )
```

Parameters

- **invoiceId** – invoice id
- **invoiceStatus** – invoice Status
- **actorId** – actor id

Usage

Call the InvoiceStatusChangeAdvice service from the responsible BillPaymentHandler implementation.

checkPayInvoice(final SubTransaction, final Long, final Long) method

Syntax

```
Boolean checkPayInvoice ( final SubTransaction txn , final Long actorId , final Long invoiceId )
```

Parameters

- **txn** – the newly created sub transaction
- **actorId** – actor id
- **invoiceId** – invoice id

Returns

true if the call was successful, otherwise false.

Usage

Call the checkPayInvoice service from the responsible BillPaymentHandler implementation.
true if the call was successful, otherwise false.

continuePayInvoice(final long, final Long, final Long) method

Syntax

```
Boolean continuePayInvoice ( final long invoiceId , final Long trxId , final Long actorId )
```

Parameters

- **invoiceId** – invoice id
- **trxId** – the id of transaction to be continued.
- **actorId** – actor id

Returns

true if the call was successful, otherwise false.

Usage

Call the payInvoice service from the responsible BillPaymentHandler implementation.
true if the call was successful, otherwise false.

makePreAuthRequest(final Long, final ICheckPayInvoiceRequest) method

Syntax

```
ICheckPayInvoiceRequest makePreAuthRequest ( final Long actorId ,  
final ICheckPayInvoiceRequest request )
```

Parameters

- **actorId** – actor id
- **request** – ICheckPayInvoiceRequest from request

Returns

the ICheckPayInvoiceRequest filled with valid data.

Usage

Fill the ICheckPayInvoiceRequest request for the PreAuthorisation call.

the ICheckPayInvoiceRequest filled with valid data.

ILimitAccess interface

Allows clients to access the LimitSums configured on Mobiliser objects.

Syntax

```
public interface ILimitAccess
```

Usage

It handles fetching and also creating them upon first use.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCustomerLimits(final Customer, final UseCase, final Long) method

Return the LimitSums for given customer and use case.

Syntax

```
List< LimitSum > getCustomerLimits ( final Customer customer ,  
final UseCase useCase , final Long actor )
```

Parameters

- **customer** – the customer
- **useCase** – the use case
- **actor** – the actor for the given action

Returns

the list of `LimitSums` for the customer, may be empty

Usage

This method will never return null. If a

`LimitSum`

for a configured

`com.sybase365.mobiliser.money.persistence.model.system.LimitClass`

has not yet been created for this customer, this method will create and save it on the fly.

the list of `LimitSums` for the customer, may be empty

getPaymentInstrumentLimits(final PaymentInstrument, final UseCase, final Long)
method

Return the `LimitSums` for given payment instrument and use case.

Syntax

```
List< LimitSum > getPaymentInstrumentLimits ( final
PaymentInstrument paymentInstrument ,   final UseCase useCase ,   final
Long actor )
```

Parameters

- **paymentInstrument** – the payment instrument
- **useCase** – the use case
- **actor** – the actor for the given action

Returns

the list of `LimitSums` for the payment instrument, may be empty

Usage

This method will never return null. If a

`LimitSum`

for a configured

com.sybase365.mobiliser.money.persistence.model.system.LimitClass

has not yet been created for this payment instrument, this method will create and save it on the fly.

the list of `LimitSums` for the payment instrument, may be empty

getWalletEntryLimits(final WalletEntry, final UseCase, final Long) method

Return the `LimitSums` for given wallet entry and use case.

Syntax

```
List< LimitSum > getWalletEntryLimits ( final WalletEntry  
walletEntry , final UseCase useCase , final Long actor )
```

Parameters

- **walletEntry** – the wallet entry
- **useCase** – the use case
- **actor** – the actor for the given action

Returns

the list of `LimitSums` for the wallet entry, may be empty

Usage

This method will never return null. If a

`LimitSum`

for a configured

com.sybase365.mobiliser.money.persistence.model.system.LimitClass

has not yet been created for this wallet entry, this method will create and save it on the fly.

the list of `LimitSums` for the wallet entry, may be empty

ILimitLogic interface

Logic for limit handling.

Syntax

```
public interface ILimitLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

adjustBalances(final SubTransaction, final Customer, final Customer, final Long)
method

Adjusts limit sums for payer and payee of a transaction.

Syntax

```
void adjustBalances ( final SubTransaction txn ,  final Customer
payee ,  final Customer payer ,  final Long actor )
```

Parameters

- **txn** – the current subtransaction
- **payee** – transaction payee
- **payer** – transaction payer
- **actor** – current actor

Usage

This does not check limit violations. Limits are checked after adjustment with
limitCheck(SubTransaction, Transaction, Long)

.

limitCheck(final SubTransaction, final Transaction, final Long)
method

Do limit checks.

Syntax

```
void limitCheck ( final SubTransaction txn ,  final Transaction
moneyTxn ,  final Long actor )
```

Parameters

- **txn** – the current subtransaction
- **moneyTxn** – the current main transaction
- **actor** – current actor

reverseAuthorisationAmounts(final Authorisation, final Customer, final Customer,
final Long)
method

This method rolls back the limit adjustments based on the authorization transaction amount.

Syntax

```
void reverseAuthorisationAmounts ( final Authorisation txn ,
final Customer payee ,  final Customer payer ,  final Long actor )
```

Parameters

- **txn** – authorization subtransaction
- **payee** – payer of transaction
- **payer** – payee of transaction
- **actor** – current actor

IParticipantLogic interface

Logic for handling transaction participant verification.

Syntax

```
public interface IParticipantLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

createMoneyCustomer(ITransactionParticipant, Transaction) method

Creates a new money customer in the system to facilitate anonymous requests.

Syntax

```
Identification createMoneyCustomer ( ITransactionParticipant  
participant , Transaction txn )
```

Parameters

- **participant** – the customer to create.
- **txn** – the current money transaction

Returns

the identification of the newly created money customer.

Usage

the identification of the newly created money customer.

getCustomer(final CallerInformation) method

Syntax

```
Customer getCustomer ( final CallerInformation caller )
```

Parameters

- **caller** – the current caller

Returns

the caller's customer object

Usage

Returns the customer object of the given caller

the caller's customer object

getIdentification(ITransactionParticipant, boolean) method**Syntax**

```
Identification getIdentification ( ITransactionParticipant
participant , boolean isPayee )
```

Parameters

- **participant** – the participant data
- **isPayee** – flag if the participant is payee

Returns

the participant's identification.

Usage

Returns the identification to use for the passed customer. If no explicit identification is used, the owner of the PI is tried if the customer is a participant. If no owner is set, this method returns

null

.

the participant's identification.

getPaymentInstrument(final UseCase, final ITransactionParticipant, final Customer, final boolean) method**Syntax**

```
PaymentInstrument getPaymentInstrument ( final UseCase useCase ,
final ITransactionParticipant participant , final Customer customer ,
final boolean payee )
```

Parameters

- **useCase** – the current transaction's use case
- **participant** – the transaction participant information
- **customer** – the corresponding customer entity

Returns

the payment instrument to use for this transaction / use case.

Usage

Returns the payment instrument to be used by this customer. It is either retrieved from the Participant info or from the default wallet settings for this customer.

the payment instrument to use for this transaction / use case.

verifyPayeePrivilege(Customer, UseCase) method

Verifies if the passed customer can participate in the use case as payee, based on the customer's privileges.

Syntax

```
void verifyPayeePrivilege ( Customer payee ,   UseCase useCase ) throws  
TransactionParticipantException
```

Parameters

- **payee** – the customer to check.
- **useCase** – the current use case.

Exceptions

- **TransactionParticipantException** – if the payee is lacking the required privilege.

verifyPayerPrivilege(Customer, UseCase) method

Verifies if the passed customer can participate in the use case as payer, based on the customer's privileges.

Syntax

```
void verifyPayerPrivilege ( Customer payer ,   UseCase useCase ) throws  
TransactionParticipantException
```

Parameters

- **payer** – the customer to check.

- **useCase** – the current use case.

Exceptions

- **TransactionParticipantException** – if the payer is lacking the required privilege.

IRedirectUrlLookup interface

Interface for retrieving the url redirects for web transactions.

Syntax

```
public interface IRedirectUrlLookup
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getRedirectUrl(final String) method

Lookup the configured redirect url for the order channel.

Syntax

```
String getRedirectUrl ( final String orderChannelName )
```

Parameters

- **orderChannelName** – the order channel name, may be null

Returns

the url

Usage

the url

getTokenSecret() method

Lookup for the preference configured secret that is used to obfuscate the transaction token hash.

Syntax

```
String getTokenSecret ()
```

Returns

the token

Usage

the token

getTokenTimeout() method

Returns the token validity timeout period.

Syntax

```
int getTokenTimeout ()
```

Returns

the token validity timeout period.

Usage

the token validity timeout period.

IRestrictionLogic interface

Logic for restriction checks.

Syntax

```
public interface IRestrictionLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

individualRestrictionCheck(final Customer, final Customer, UseCase) method

Check if there is a individual restriction and validates the restriction.

Syntax

```
boolean individualRestrictionCheck ( final Customer payer , final  
Customer payee , UseCase useCase ) throws  
TransactionRestrictionException
```

Parameters

- **payer** – the payer to check
- **payee** – the payee to check
- **useCase** – the current use case

Returns

true if an individual restriction existed, but hasn't been violated, false if no individual restriction exists.

Exceptions

- **TransactionRestrictionException** – if a restriction is violated.

Usage

true if an individual restriction existed, but hasn't been violated, false if no individual restriction exists.

restrictionCheck(Transaction) method

Check the restriction for the given transaction.

Syntax

```
void restrictionCheck ( Transaction moneyTxn ) throws  
TransactionRestrictionException
```

Parameters

- **moneyTxn** – the transaction to check for restriction violations.

Exceptions

- **TransactionRestrictionException** – if a restriction is violated.

ITransactionCRUDLogic interface

Logic for transaction CRUD operations.

Syntax

```
public interface ITransactionCRUDLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

checkTxnToken(long, String) method

Check if the supplied token matches the transaction given by its ID.

Syntax

```
void checkTxnToken ( long txnId , String token ) throws  
MobiliserServiceException
```

Parameters

- **txnId** – the transaction to check
- **token** – the token for validation

Exceptions

- **MobiliserServiceException** – if the token does not match.

findTransactions(Long, Long, Long, Date, Date, Boolean, Integer, String, Integer, int, Long, Boolean, Boolean) method

Find transaction logic searching for transactions matching the provided parameters.

Syntax

```
List< FindTransactionsResultBean > findTransactions ( Long  
customerId , Long joinedCustomerId , Long caller , Date fromDate , Date  
toDate , Boolean showFaulty , Integer statusFilter , String  
merchantOrderIdFilter , Integer maxHierarchyLevel , int maxRecords , Long  
paymentInstrumentId , Boolean customerIsPayer , Boolean showInitial ) throws  
IllegalArgumentException
```

Parameters

- **customerId** – limit results to transactions where this customer participated
- **joinedCustomerId** – limit results to transaction where this customer was the counterpart of the customerId - relevant if filtering by customerIsPayer.
- **caller** – filter by caller
- **fromDate** – search for transactions newer than this date (or equal)
- **toDate** – search for transactions older than this date (or equal)
- **showFaulty** – include transactions with error code not equal to zero
- **statusFilter** – only include transactions with this status
- **merchantOrderIdFilter** – filter by order id

- **maxHierarchyLevel** – if provided, also includes transactions of children of the customerId (up to the passed hierarchy level). When setting this you must also provide customerId.
- **maxRecords** – limit number of records

Returns

a list of found transactions

Exceptions

- **IllegalArgumentException** – if mandatory parameters are not provided.

Usage

a list of found transactions

getTransaction(long) method

Loads a transaction by id.

Syntax

```
Transaction getTransaction ( long txnId ) throws  
EntityNotFoundException
```

Parameters

- **txnId** – the transaction to load

Returns

the transaction entity.

Exceptions

- **EntityNotFoundException** – if the transaction does not exist.

Usage

the transaction entity.

getWebTxnDetails(long) method

Loads the details of a web transaction.

Syntax

```
Object[] getWebTxnDetails ( long txnId )
```

Parameters

- **txnId** – the web transaction id.

Returns

an array of the main transaction, all subtransactions, and the delivery address (if requested) - all being JPA beans.

Usage

an array of the main transaction, all subtransactions, and the delivery address (if requested) - all being JPA beans.

ITransactionLogic interface

Miscellaneous transaction logic.

Syntax

```
public interface ITransactionLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

checkForOpenTransactions(final Transaction) method

Syntax

```
void checkForOpenTransactions ( final Transaction txn ) throws  
TransactionParticipantException
```

Parameters

- **txn** – the transaction to check.

Exceptions

- **TransactionParticipantException** – if the payer has open transactions

Usage

Checks if the payer of the passed money transaction has open transactions pending and sets an appropriate error code if at least one open transaction (excluding the passed transaction) is found.

*checkIdentical(final Transaction) method***Syntax**

void checkIdentical (final Transaction *txn*) throws
TransactionParticipantException

Parameters

- **txn** – the MoneyTransaction to check

Exceptions

- **TransactionParticipantException** – if payer and payee are identical

Usage

Checks if payee and payer of the specified transaction are identical.

checkOpenWebTransactions(final String, final long, final long) method

Check for open web transactions.

Syntax

void checkOpenWebTransactions (final String *orderId* , final long
payeeId , final long *amount*) throws
TransactionParticipantException

Parameters

- **orderId** – the order id
- **payeeId** – the payee id
- **amount** – the amount of the transaction

Exceptions

- **TransactionParticipantException** – if the payee has open transactions with the passed amount

*checkTestStatus(Transaction) method***Syntax**

void checkTestStatus (Transaction *txn*) throws
TransactionFailedException

Parameters

- **txn** – the `MoneyTransaction` to check

Exceptions

- **TransactionFailedException** – if transaction and participants do not all have the same test/live status

Usage

Checks the test flags of the specified transaction and its participants.

initMoneyTransaction(IGenericAuthorisationRequest, final Long) method

Syntax

`Transaction initMoneyTransaction (IGenericAuthorisationRequest
authorisationRequest , final Long actor)`

Parameters

- **authorisationRequest** – the request that holds the data
- **actor** – the current actor

Returns

the initialized `MoneyTransaction`

Usage

Initialises the money transaction with the standard parameters from the

`IAuthorisationRequest`

. The standard parameters are:

- amount
- auth code
- currency
- order channel
- origin
- status
- test
- text
- order id
- timestamp

the initialized MoneyTransaction

initPreAuthorisationContinue(ITransaction, CallerInformation) method

Initializes a transaction for preauthorization continue.

Syntax

```
InitTransactionResult initPreAuthorisationContinue
( ITransaction refTransaction , CallerInformation actor )
```

Parameters

- **refTransaction** – the transaction data
- **actor** – the current actor

Returns

the transaction data

Usage

the transaction data

unlockTransaction(Long, final Long) method

Retrieves the referenced Money Transaction and unlocks it.

Syntax

```
Object unlockTransaction ( Long systemId , final Long actorId )
```

Parameters

- **systemId** – the transaction to unlock
- **actorId** – the current actor

Returns

always null

Usage

This must be used after the transaction has been processed

always null

verifyActorAndTransaction(ITransaction, SubTransaction, boolean, boolean, final CallerInformation, Integer...) method

Syntax

Transaction verifyActorAndTransaction (ITransaction *request* ,
SubTransaction *subTxn* , boolean *lockMoneyTxn* , boolean *checkExpiry* ,
final CallerInformation *actor* , Integer... *allowedStatus*) throws
TransactionFailedException

Parameters

- **request** – the MoneyRequest to process
- **subTxn** – the current sub transaction.
- **lockMoneyTxn** – true to lock the found MoneyTransaction; false otherwise
- **checkExpiry** – true to check the transaction's expiration; false to ignore it
- **allowedStatus** – the allowed Transaction.Status, the *Money Transaction* status must be in one of them - if at least one status is specified, the transaction's error code is checked to be 0; if no status is specified, neither the status nor the error code is checked

Returns

the referenced Transaction

Exceptions

- **TransactionFailedException** – if verification fails

Usage

Retrieves the referenced

Money Transaction

, locks it and performs basic checks:

- *Actor* is set
- Given *actor* matches *Money Transaction*
- Validity of the used transaction

the referenced Transaction

*verifyTransaction(ITransaction, boolean, Integer...) method***Syntax**

Transaction verifyTransaction (ITransaction *request* , boolean *checkExpiry* , Integer... *allowedStatus*) throws TransactionFailedException

Parameters

- **request** – the MoneyRequest to process
- **checkExpiry** – true to check the transaction's expiration; false to ignore it
- **allowedStatus** – the allowed Transaction.Status, the *Money Transaction* status must be in one of them - if at least one status is specified, the transaction's error code is checked to be 0; if no status is specified, neither the status nor the error code is checked

Returns

the referenced MoneyTransaction

Exceptions

- **TransactionFailedException** – if verification fails

Usage

Retrieves the referenced

Money Transaction

, locks it and performs basic checks on the transaction:

- Referenced Transaction exists and is (or refers to) a *Money Transaction*
- The *Money Transaction* has expired
- The *Money Transaction* is in the requested status with error code 0 (if any status is specified)
- The *Money Transaction* has expired (if requested)

If any violation is detected, the error will be set using

setError(int, Object[])

.

the referenced MoneyTransaction

IUseCaseLogic interface

Logic dealing with use cases.

Syntax

```
public interface IUseCaseLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

loadUseCase(final Integer, final boolean) method

Loads the given use case, checking for public access.

Syntax

```
UseCase loadUseCase ( final Integer useCaseId, final boolean  
isPublic )
```

Parameters

- **useCaseId** – the id of the use case
- **isPublic** – whether we are loading this use case for a public or private call

Returns

the use case

Usage

the use case

IVoucherLogic interface

Logic for voucher handling.

Syntax

```
public interface IVoucherLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

**verifyVoucherRemittanceReceipient(*final Long, final IGenericAuthorisationRequest*)
method**

Creates a new voucher and adds it to the recipients wallet.

Syntax

```
ITransactionParticipant verifyVoucherRemittanceReceipient
( final Long actorId, final IGenericAuthorisationRequest request )
```

Parameters

- **actorId** – current actor
- **request** – the authorization request, all data is pulled from here

Returns

the participant who received the voucher

Usage

If the recipient doesn't exist, the recipient is created on the fly, also.

the participant who received the voucher

IWebTxnLogic interface

Logic for web transaction handling.

Syntax

```
public interface IWebTxnLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

checkToken(*final Transaction, final String*) method

Checks if the given token was computed for the given transaction during the last n minutes.

Syntax

```
void checkToken ( final Transaction txn, final String token ) throws
MobiliserServiceException
```

Parameters

- **txn** – the current web transaction
- **token** – the token to check

Exceptions

- **MobiliserServiceException** – if the token does not match or is older than the configured timeframe

Usage

The timeframe is configured via preferences

getRedirectUrl(final Transaction, final String) method

Retrieved the redirect url for a transaction.

Syntax

```
String getRedirectUrl ( final Transaction txn , final String  
token )
```

Parameters

- **txn** – the transaction
- **token** – the token

Returns

the redirect url

Usage

the redirect url

getToken(final Transaction, final Date) method

Computes the security token to harden the redirect against fraud.

Syntax

```
String getToken ( final Transaction txn , final Date timestamp )
```

Parameters

- **txn** – the transaction to be continued.
- **timestamp** – the timestamp to use for hash computation.

Returns

the security token or `null` if the merchant has no credential set.

Usage

The security token is computed as a hash over transaction data, tokens and timestamp. The credential with largest ID (the newest one) is used.

the security token or `null` if the merchant has no credential set.

TransactionConfiguration class

Configuration bean for generic financial transaction properties.

Syntax

```
public class TransactionConfiguration
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getDefaultExpirationMinutes() method

Syntax

```
int getDefaultExpirationMinutes ()
```

Returns

default transaction expiration timeout in minutes, default is 14 days.

Usage

default transaction expiration timeout in minutes, default is 14 days.

isAlwaysCreateAuthorisationCode() method

Syntax

```
boolean isAlwaysCreateAuthorisationCode ()
```

Returns

if Mobiliser should always create an authorization code (even for failed transactions), default is `false`

Usage

if Mobiliser should always create an authorization code (even for failed transactions), default is `false`

isOpenTransactionCheckDisabled() method

Syntax

```
boolean isOpenTransactionCheckDisabled ()
```

Returns

if Mobiliser should check for open transactions, default is `false`

Usage

if Mobiliser should check for open transactions, default is `false`

setPreferences(final IPreferences) method

Sets the prefs node for configuration.

Syntax

```
void setPreferences ( final IPreferences preferences )
```

Parameters

- **preferences** –

preferences variable

Syntax

```
IPreferences preferences
```

umgr package

Members

All public members of the umgr package.

- **ICustomerPrivilegeProvider interface** – The methods of ICustomerPrivilegeProvider are invoked when the complete list of privileges of a user are to be retrieved (e.g.
- **ISecurityLogic interface** –

ICustomerPrivilegeProvider interface

The methods of ICustomerPrivilegeProvider are invoked when the complete list of privileges of a user are to be retrieved (e.g.

Syntax

```
public interface ICustomerPrivilegeProvider
```

Usage

after a successful login). You can add your own implementation to add or revoke privileges of a user.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getAdditionalCustomerPrivileges(long, long) method

Is invoked to return additional privileges for a customer, which are merged with the existing ones.

Syntax

```
Collection< String >getAdditionalCustomerPrivileges ( long  
customerId , long callerId )
```

Parameters

- **customerId** –
- **callerId** –

getRevokedCustomerPrivileges(long, long) method

Is invoked to revoke privileges from a customer.

Syntax

```
Collection< String >getRevokedCustomerPrivileges ( long  
customerId , long callerId )
```

Parameters

- **customerId** –
- **callerId** –

ISecurityLogic interface

Syntax

```
public interface ISecurityLogic
```

Usage

Set of services that deal with customer authentication (login), session handling.

An implementation of this class is exported via the OSGi service registry.

Use this to import the service into your code:

```
<osgi:reference id="securityLogic"
interface="com.sybase365.mobiliser.money.businesslogic.umgr.
ISecurityLogic" sticky="false" />
```

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

checkCredential(final long, final String, final int, final long) method

Similar to loginCustomer but does not create a session of the customer upon success.

Syntax

```
void checkCredential (final long customerId, final String credential,
final int credentialType, final long callerId) throws
EntityNotFoundException, CustomerStatusException,
CredentialValidationException, SessionException
```

Parameters

- **customerId** – the ID of the customer for whom the credential is checked
- **credential** – the credential to check
- **credentialType** – the type of the credential that is provided
- **callerId** – represents the ID of the caller of the operation. Used for tracking/audit purposes.

Exceptions

- **EntityNotFoundException** –
- **CustomerInactiveException** –
- **CustomerBlacklistedException** –
- **CredentialBlockedException** –

- **CredentialOkButExpiredException** –
- **CredentialWrongException** –
- **CredentialNotFoundException** –
- **CredentialOkButTemporaryException** –
- **CredentialWrongAndActionTakenException** –

Usage

Instead of an Identification the id of the customer must be provided. Other than that the same business logic is executed.

checkCredentialRetention(Customer, int, String) method

Checks if the credential matches any retained credential.

Syntax

```
void checkCredentialRetention ( Customer customer , int credentialType ,
String plainTextCredential )
```

Parameters

- **customer** –
- **credentialType** –
- **plainTextCredential** –

Usage

Mainly used internally.

checkCredentialStrength(CredentialPolicy, String, final long) method

Checks if the credential matches all of the requirements defined in the given policy.

Syntax

```
void checkCredentialStrength ( CredentialPolicy policy , String
plainTextCredential , final long callerId ) throws
CredentialStrengthException
```

Parameters

- **policy** – the policy that is used for checking the credntial against
- **plainTextCredential** – the credential in plain text (not hashed)
- **callerId** – represents the ID of the caller of the operation. Used for tracking/audit purposes.

Exceptions

- **CredentialTooLongException** –
- **CredentialTooShortException** –
- **CredentialHasWeakBlockException** –
- **CredentialHasInvalidPatternException** –

Usage

In case it does not an appropriate exception is thrown.

checkPasswordHashUpgrade(final String, final Credential, final long) method

Checks the given credential to see if the password hash should be upgraded.

Syntax

```
boolean checkPasswordHashUpgrade ( final String rawCredential ,  
final Credential credential , final long callerId )
```

Parameters

- **rawCredential** –
- **credential** –
- **callerId** –

Returns

true if the password hash was upgraded, false otherwise

Usage

If so , the raw credential is hashed and saved as the new credential.

true if the password hash was upgraded, false otherwise

generateCredential(CredentialPolicy, boolean, final long) method

Generates a new credential that is in compliance with the provided policy.

Syntax

```
String generateCredential ( CredentialPolicy policy , boolean  
numericOnly , final long callerId ) throws CredentialStrengthException
```

Parameters

- **policy** – the CredentialPolicy that the credential must adhere to.
- **numericOnly** – indicates if the credential should only contain numbers.

- **callerId** – represents the ID of the caller of the operation. Used for tracking/audit purposes.

Returns

the generated credential in plain text

Exceptions

- **CredentialGenerationException** – if it was not possible to generate a credential that is compliant to the policy. Probably means that the policy is wrongly configured.

Usage

the generated credential in plain text

getAllPrivileges(final long, final long) method

Returns all privileges that are attached to a customer, either via CustomerType, RiskCategory, Roles, or directly.

Syntax

```
List< String > getAllPrivileges ( final long customerId , final long callerId ) throws EntityNotFoundException, CustomerStatusException
```

Parameters

- **customerId** – the ID of the customer for whom to retrieve the privileges.
- **callerId** – represents the ID of the caller of the operation. Used for tracking/audit purposes.

Returns

the list of privileges (Strings)

Exceptions

- **EntityNotFoundException** –
- **CustomerInactiveException** –

Usage

Does not take the OrgUnit into account.

the list of privileges (Strings)

getDefaultPrivileges(int, Integer, long) method

Returns all privileges that are attached to the role of the CustomerType and RiskCategory.

Syntax

```
List< String > getDefaultPrivileges ( int customerTypeId , Integer  
riskCategoryId , long callerId )
```

Parameters

- **customerTypeId** –
- **riskCategoryId** –
- **callerId** – represents the ID of the caller of the operation. Used for tracking/audit purposes.

Exceptions

- **EntityNotFoundException** –

Usage

If the provided riskCategoryId is null, the default one from the CustomerType is used. This can be used to check the privileges before a new user gets created.

handleCorrectCredentialInput(Credential, long) method

Logs in an already authenticated customer.

Syntax

```
void handleCorrectCredentialInput ( Credential cred , long  
callerId )
```

Parameters

- **cred** –
- **callerId** –

handleWrongCredentialInput(Credential, long) method

Handles wrong credential input (update counter, black-list customer, ...) based on the settings in the security policy.

Syntax

```
boolean handleWrongCredentialInput ( Credential cred , long  
callerId )
```

Parameters

- **cred** –
- **callerId** –

Returns

`true` if the number of wrong entries was exceeded and the customer was either black-listed or the credential is temporarily blocked. Returns `false` otherwise.

Usage

Mainly used internally.

`true` if the number of wrong entries was exceeded and the customer was either black-listed or the credential is temporarily blocked. Returns `false` otherwise.

loginAuthenticatedCustomer(String, int, String, long, String, boolean, List< String >)
method

Syntax

`CustomerSession loginAuthenticatedCustomer (String identifier , int identifierType , String origin , long callerId , String orgUnit , boolean allPrivilegesRequired , List< String > requiredPrivileges )` throws
`EntityNotFoundException`, `CustomerStatusException`,
`SessionException`, `InternalCustomerException`,
`MissingLoginPrivilegeException`

Parameters

- **identifier** – the identification to load the customer by (MSISDN, username, ...).
- **identifierType** – the type of the identification provided in *identification*.
- **origin** – the origin is stored with the session that is created. Specifies where the login operation is initiating from.
- **callerId** – represents the ID of the caller of the operation. Used for tracking/audit purposes.
- **orgUnit** – the customers orgunit
- **allPrivilegesRequired** – this flag indicates if the customer must have all of the required privileges, or if one is sufficient
- **requiredPrivileges** – this is the list of privileges the customer must have to log in successfully; if the customer does not have these privileges, the login will fail and the customer status, wrong-credential-counter, etc. will not be changed (irrespective of whether or not the customer provided the right password), if this list is empty or null, no privilege checking will happen.

Returns

the created CustomerSession

Exceptions

- **EntityNotFoundException** –
- **CustomerStatusException** –
- **SessionException** –
- **InternalCustomerException** –
- **MissingLoginPrivilegeException** –

Usage

Logs in an already authenticated end-user into an application and creating a new session. The session id that is returned should be used for each subsequent request and is valid either until it times out or a logout is called.

the created CustomerSession

loginCustomer(String, String, int, int, String, long, String, boolean, List< String >)
method

Syntax

CustomerSession loginCustomer (String *identifier* , String *plainTextCredential* , int *identifierType* , int *credentialType* , String *origin* , long *callerId* , String *orgUnit* , boolean *allPrivilegesRequired* , List< String > *requiredPrivileges*) throws EntityNotFoundException, CustomerStatusException, CredentialValidationException, SessionException, InternalCustomerException, MissingLoginPrivilegeException

Parameters

- **identifier** – the identification to load the customer by (MSISDN, username, ...).
- **plainTextCredential** – the credential to check (PIN, password, ...).
- **identifierType** – the type of the identification provided in *identification*.
- **credentialType** – the type of the credential provided in *credential*
- **origin** – the origin is stored with the session that is created. Specifies where the login operation is initiating from.
- **callerId** – represents the ID of the caller of the operation. Used for tracking/audit purposes.
- **orgUnit** – the customers orgunit

- **allPrivilegesRequired** – this flag indicates if the customer must have all of the required privileges, or if one is sufficient
- **requiredPrivileges** – this is the list of privileges the customer must have to log in successfully; if the customer does not have these privileges, the login will fail and the customer status, wrong-credential-counter, etc. will not be changed (irrespective of whether or not the customer provided the right password), if this list is empty or null, no privilege checking will happen.

Returns

the created CustomerSession

Exceptions

- **EntityNotFoundException** –
- **CustomerStatusException** –
- **CredentialValidationException** –
- **SessionException** –
- **InternalCustomerException** –
- **MissingLoginPrivilegeException** –

Usage

Logs in an end-user into an application by checking username, credential, and creating a new session. This method runs in its

own transaction

to make sure that wrong credential checks are persisted. The session id that is returned should be used for each subsequent request and is valid either until it times out or a logout is called.

In case of an incorrect credential the Credential Policies are used to determine the next action. In case the credential was correct but it is temporary (system generated) or has expired the

CredentialOkButExpiredException

is thrown.

the created CustomerSession

loginCustomer(String, String, int, int, String, long, String) method

Syntax

```
CustomerSession loginCustomer ( String identifier , String
plainTextCredential , int identifierType , int credentialType , String origin ,
long callerId , String orgUnit ) throws EntityNotFoundException,
CustomerStatusException, CredentialValidationException,
SessionException, InternalCustomerException
```

logoutCustomer(final String, final long) method

Closes the session that was created by loginCustomer.

Syntax

```
void logoutCustomer ( final String sessionId ,   final long callerId )  
throws EntityNotFoundException
```

Parameters

- **sessionId** – the id of the session that is to be closed
- **callerId** – represents the ID of the caller of the operation. Used for tracking/audit purposes.

Exceptions

- **EntityNotFoundException** – if the provided sessionId was not found.

replaceCredential(final long, String, int, Integer, final long) method

Replaces an existing credential of the same type with the supplied one.

Syntax

```
void replaceCredential ( final long customerId ,   String credential ,   int  
credentialType ,   Integer credentialStatus ,   final long callerId ) throws  
EntityNotFoundException, CustomerStatusException
```

Parameters

- **customerId** – the ID of the customer for whom the credential will be stored
- **credential** – the credential to store
- **credentialType** – the type of the credential that is provided
- **credentialStatus** – specifies the status of the credential that is to be stored (usually 0 is used).
- **callerId** – represents the ID of the caller of the operation. Used for tracking/audit purposes.

Usage

The old one is marked as in-active and the new one is saved. The credential must be supplied as plain text and will be stored hashed. The credential is NOT checked against the policy. This must be done before the call to this method via

```
checkCredentialStrength
```

replaceCredential(final Credential, final long) method

Replaces an existing credential of the same type with the supplied one.

Syntax

```
void replaceCredential ( final Credential credential , final long
callerId ) throws EntityNotFoundException, CustomerStatusException
```

Parameters

- **credential** – a pre-hashed credential
- **callerId** – represents the ID of the caller of the operation. Used for tracking/audit purposes.

Usage

The old one is marked as in-active and the new one is saved. The credential must already be provided in hashed format. This method is mainly called internally.

umgrroles package***Members***

All public members of the umgrroles package.

- **IUmgrRolesPrivilegesLogic interface** – The interface IUmgrRolesPrivilegesLogic contains basic CRUD type operations to work on Umgr Roles and the directly related entities.

IUmgrRolesPrivilegesLogic interface

The interface IUmgrRolesPrivilegesLogic contains basic CRUD type operations to work on Umgr Roles and the directly related entities.

Syntax

```
public interface IUmgrRolesPrivilegesLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

addUmgrCustomerPrivilege(CustomerPrivilege, long) method

Syntax

long addUmgrCustomerPrivilege (CustomerPrivilege *customerPrivilege* ,
long *callerId*) throws EntityNotFoundException

addUmgrCustomerRole(CustomerRole, long) method

Syntax

long addUmgrCustomerRole (CustomerRole *customerRole* , long *callerId*)
throws EntityNotFoundException

createUmgrPrivilege(UmgrPrivilege, long) method

Syntax

void createUmgrPrivilege (UmgrPrivilege *privilege* , long *callerId*)
throws EntityNotUniqueException

createUmgrRole(UmgrRole, long) method

Syntax

void createUmgrRole (UmgrRole *role* , long *callerId*) throws
EntityNotUniqueException

createUmgrRolePrivilege(UmgrRolePrivilege, long) method

Syntax

void createUmgrRolePrivilege (UmgrRolePrivilege *rolePrivilege* , long
callerId) throws EntityNotUniqueException

deleteUmgrPrivilege(String, long) method

Syntax

void deleteUmgrPrivilege (String *privilegeId* , long *callerId*) throws
EntityNotFoundException

deleteUmgrRole(String, long) method

Syntax

void deleteUmgrRole (String *roleId* , long *callerId*) throws
EntityNotFoundException

deleteUmgrRolePrivilege(String, String, long) method

Syntax

void deleteUmgrRolePrivilege (String *roleId* , String *privilegeId* , long *callerId*) throws EntityNotFoundException

getUmgrCustomerPrivileges(long, Date) method

Syntax

List< CustomerPrivilege > getUmgrCustomerPrivileges (long *customerId* , Date *now*) throws EntityNotFoundException

getUmgrCustomerRoles(long, Date) method

Syntax

List< CustomerRole > getUmgrCustomerRoles (long *customerId* , Date *now*) throws EntityNotFoundException

getUmgrPrivileges(String, String) method

Syntax

List< UmgrPrivilege > getUmgrPrivileges (String *privilegeIdSearchParam* , String *descriptionSearchParam*)

getUmgrRolePrivileges(String, String) method

Syntax

List< UmgrRolePrivilege > getUmgrRolePrivileges (String *roleId* , String *privilegeId*)

getUmgrRoles(String, String) method

Syntax

List< UmgrRole > getUmgrRoles (String *roleIdSearchParam* , String *descriptionSearchParam*)

removeUmgrCustomerPrivilege(long, String, String) method

Syntax

void removeUmgrCustomerPrivilege (long *customerId* , String *umgrPrivilege* , String *orgUnit*) throws EntityNotFoundException

removeUmgrCustomerRole(long, String, String) method

Syntax

```
void removeUmgrCustomerRole ( long customerId , String umgrRole ,  
String orgUnit ) throws EntityNotFoundException
```

updateUmgrCustomerPrivilege(CustomerPrivilege, long) method

Syntax

```
void updateUmgrCustomerPrivilege ( CustomerPrivilege  
customerPrivilege , long callerId ) throws EntityNotFoundException
```

updateUmgrCustomerRole(CustomerRole, long) method

Syntax

```
void updateUmgrCustomerRole ( CustomerRole customerRole , long  
callerId ) throws EntityNotFoundException
```

updateUmgrPrivilege(UmgrPrivilege, long) method

Syntax

```
void updateUmgrPrivilege ( UmgrPrivilege privilege , long callerId )  
throws EntityNotFoundException
```

updateUmgrRole(UmgrRole, long) method

Syntax

```
void updateUmgrRole ( UmgrRole role , long callerId ) throws  
EntityNotFoundException
```

util package

Members

All public members of the util package.

- **AccessDeniedMobiliserServiceException class** – The AccessDeniedMobiliserServiceException is mostly used when the business logic layer encounters a missing privilege by the caller for an operation.
- **BlacklistReasonCode class** – Mostly internally used class that contains constants for black-list-reasons.

- **CredentialStatus class** – The CredentialStatus status contains some constants for credential status that have a meaning in the code.
- **EntityMandatoryException class** – The EntityMandatoryException is thrown when a required reference is missing from the input data.
- **EntityNotFoundException class** – The EntityNotFoundException is thrown when a referenced entity was not found.
- **EntityNotUniqueException class** – The EntityNotUniqueException is thrown when a to be created entity already exists with the same primary key.
- **EntityReferenceException class** – The EntityReferenceException is thrown when an entity that is to be deleted is still referenced by other entities.
- **EntityWrongStatusException class** – The EntityWrongStatusException is thrown when an entity (that is to be used or updated) is in a status that prevents the action from being executed.
- **IdentificationTypes class** – List of common identification types.
- **IllegalDataException class** – The IllegalDataException is thrown when the provided data is not valid or empty.
- **IllegalRequestException class** – `IllegalRequestException` indicates that the request was syntactically correct, but fails on logical check
- **LocationUtils class** – A collection of Location Information Utils.
- **MobiliserErrorCodeDescription interface** –
- **StatusCodes class** – The StatusCodes class contains constants for status codes / error codes that are used in the code.
- **SystemConfigurationErrorException class** – The `SystemConfigurationErrorException` is thrown when an internal configuration is missing/wrong.
- **TxnStatus class** – This class lists all available transaction status.
- **UserAccessingNotOwnedResourceException class** – The `UserAccessingNotOwnedResourceException` is thrown when a user (caller) tries to access a resource that belongs to another user.
- **VirusScanException class** –

AccessDeniedMobiliserServiceException class

The `AccessDeniedMobiliserServiceException` is mostly used when the business logic layer encounters a missing privilege by the caller for an operation.

Syntax

```
public class AccessDeniedMobiliserServiceException
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

AccessDeniedMobiliserServiceException() constructor

Syntax

```
AccessDeniedMobiliserServiceException ()
```

AccessDeniedMobiliserServiceException(String) constructor

Syntax

```
AccessDeniedMobiliserServiceException ( String message )
```

AccessDeniedMobiliserServiceException(Throwable) constructor

Syntax

```
AccessDeniedMobiliserServiceException ( Throwable cause )
```

AccessDeniedMobiliserServiceException(String, Throwable) constructor

Syntax

```
AccessDeniedMobiliserServiceException ( String message ,  
Throwable cause )
```

getErrorCode() method

Syntax

```
int getErrorCode ()
```

BlacklistReasonCode class

Mostly internally used class that contains constants for black-list-reasons.

Syntax

```
public class BlacklistReasonCode
```

Usage

There can be more than that but those have kind of a fixed meaning in the code.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

DEACTIVATED_AND_OBFUSCATED_IF_REQUIRED variable**Syntax**

```
final int DEACTIVATED_AND_OBFUSCATED_IF_REQUIRED
```

DEACTIVATION_STAGING variable**Syntax**

```
final int DEACTIVATION_STAGING
```

OBFUSCATION_PENDING variable**Syntax**

```
final int OBFUSCATION_PENDING
```

OK variable**Syntax**

```
final int OK
```

REGISTRATION_PENDING variable**Syntax**

```
final int REGISTRATION_PENDING
```

CredentialStatus class

The CredentialStatus status contains some constants for credential status that have a meaning in the code.

Syntax

```
public class CredentialStatus
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

BLOCKED variable**Syntax**

```
final int BLOCKED
```

OK variable

Syntax

```
final int OK
```

TEMPORARY variable

Syntax

```
final int TEMPORARY
```

EntityMandatoryException class

The EntityMandatoryException is thrown when a required reference is missing from the input data.

Syntax

```
public class EntityMandatoryException
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

EntityMandatoryException() constructor

Syntax

```
EntityMandatoryException ()
```

EntityMandatoryException(String) constructor

Syntax

```
EntityMandatoryException ( String message )
```

EntityMandatoryException(Throwable) constructor

Syntax

```
EntityMandatoryException ( Throwable cause )
```

EntityMandatoryException(String, Throwable) constructor

Syntax

```
EntityMandatoryException ( String message , Throwable cause )
```

*getErrorCode() method***Syntax**

```
int getErrorCode ()
```

EntityNotFoundException class

The EntityNotFoundException is thrown when a referenced entity was not found.

Syntax

```
public class EntityNotFoundException
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

*EntityNotFoundException() constructor***Syntax**

```
EntityNotFoundException ()
```

*EntityNotFoundException(String, String) constructor***Syntax**

```
EntityNotFoundException ( String message , String entity )
```

*EntityNotFoundException(Throwable, String) constructor***Syntax**

```
EntityNotFoundException ( Throwable cause , String entity )
```

*EntityNotFoundException(String, Throwable, String) constructor***Syntax**

```
EntityNotFoundException ( String message , Throwable cause , String entity )
```

*getErrorCode() method***Syntax**

```
int getErrorCode ()
```

EntityNotUniqueException class

The EntityNotUniqueException is thrown when a to be created entity already exists with the same primary key.

Syntax

```
public class EntityNotUniqueException
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

EntityNotUniqueException() constructor

Syntax

```
EntityNotUniqueException ()
```

EntityNotUniqueException(String) constructor

Syntax

```
EntityNotUniqueException ( String message )
```

EntityNotUniqueException(Throwable) constructor

Syntax

```
EntityNotUniqueException ( Throwable cause )
```

EntityNotUniqueException(String, Throwable) constructor

Syntax

```
EntityNotUniqueException ( String message , Throwable cause )
```

getErrorCode() method

Syntax

```
int getErrorCode ()
```


EntityReferenceException class

The EntityReferenceException is thrown when an entity that is to be deleted is still referenced by other entities.

Syntax

```
public class EntityReferenceException
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

EntityReferenceException() constructor**Syntax**

```
EntityReferenceException ()
```

EntityReferenceException(String, String) constructor**Syntax**

```
EntityReferenceException ( String message , String entity )
```

EntityReferenceException(Throwable, String) constructor**Syntax**

```
EntityReferenceException ( Throwable cause , String entity )
```

EntityReferenceException(String, Throwable, String) constructor**Syntax**

```
EntityReferenceException ( String message , Throwable cause ,  
String entity )
```

getErrorCode() method**Syntax**

```
int getErrorCode ()
```

EntityWrongStatusException class

The EntityWrongStatusException is thrown when an entity (that is to be used or updated) is in a status that prevents the action from being executed.

Syntax

```
public class EntityWrongStatusException
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

EntityWrongStatusException() constructor

Syntax

```
EntityWrongStatusException ()
```

EntityWrongStatusException(String, String, String) constructor

Syntax

```
EntityWrongStatusException (String message , String entity , String  
status )
```

getErrorCode() method

Syntax

```
int getErrorCode ()
```

IdentificationTypes class

List of common identification types.

Syntax

```
public class IdentificationTypes
```

Usage

Please note the listed identification types are the standard one and may be extended by a customisation

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

IDENTIFICATION_TYPE_CARD_NUMBER variable

Syntax

```
final int IDENTIFICATION_TYPE_CARD_NUMBER
```

IDENTIFICATION_TYPE_CUSTOMER_ID variable

Syntax

```
final int IDENTIFICATION_TYPE_CUSTOMER_ID
```

IDENTIFICATION_TYPE_EMAIL variable

Syntax

```
final int IDENTIFICATION_TYPE_EMAIL
```

IDENTIFICATION_TYPE_FOREIGN_CUSTOMER_ID variable

Syntax

```
final int IDENTIFICATION_TYPE_FOREIGN_CUSTOMER_ID
```

IDENTIFICATION_TYPE_IMEI_CO_USER variable

Syntax

```
final int IDENTIFICATION_TYPE_IMEI_CO_USER
```

IDENTIFICATION_TYPE_IMEI_MOBILE variable

Syntax

```
final int IDENTIFICATION_TYPE_IMEI_MOBILE
```

IDENTIFICATION_TYPE_MSISDN variable

Syntax

```
final int IDENTIFICATION_TYPE_MSISDN
```

IDENTIFICATION_TYPE_USER_NAME variable

Syntax

```
final int IDENTIFICATION_TYPE_USER_NAME
```

IllegalDataException class

The `IllegalDataException` is thrown when the provided data is not valid or empty.

Syntax

```
public class IllegalDataException
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

IllegalDataException() constructor

Syntax

```
IllegalDataException ()
```

IllegalDataException(String, String) constructor

Syntax

```
IllegalDataException ( String message , String illegalData )
```

IllegalDataException(Throwable, String) constructor

Syntax

```
IllegalDataException ( Throwable cause , String illegalData )
```

IllegalDataException(String, Throwable, String) constructor

Syntax

```
IllegalDataException ( String message , Throwable cause , String illegalData )
```

getErrorCode() method

Syntax

```
int getErrorCode ()
```

IllegalRequestException class

`IllegalRequestException` indicates that the request was syntactically correct, but fails on logical check

Syntax

```
public class IllegalRequestException
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

IllegalRequestException() constructor**Syntax**

```
IllegalRequestException ()
```

IllegalRequestException(String, String) constructor**Syntax**

```
IllegalRequestException ( String message , String illegalData )
```

IllegalRequestException(Throwable, String) constructor**Syntax**

```
IllegalRequestException ( Throwable cause , String illegalData )
```

IllegalRequestException(String, Throwable, String) constructor**Syntax**

```
IllegalRequestException ( String message , Throwable cause , String illegalData )
```

getErrorCode() method**Syntax**

```
int getErrorCode ()
```

LocationUtils class

A collection of Location Information Utils.

Syntax

```
public class LocationUtils
```

Usage

2012 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

distanceOf(double, double, double, double) method

Calculate the direct distance of X and Y via spherical law.

Syntax

```
double distanceOf ( double Xlatitude ,   double Xlongitude ,   double  
Ylatitude ,   double Ylongitude )
```

Parameters

- ***Xlatitude*** –
- ***Xlongitude*** –
- ***Ylatitude*** –
- ***Ylongitude*** –

Returns

distance in km

Usage

This is accurate for small and medium distances (<3.000km)

distance in km

toBigDecimal(double) method

Transform double value into database decimal format with a precision of 12.

Syntax

```
BigDecimal toBigDecimal ( double d )
```

Parameters

- **d** –

EARTH_RADIUS variable

The mean earth radius in km.

Syntax

```
final double EARTH_RADIUS
```

MobiliserErrorCodeDescription interface**Syntax**

```
public interface MobiliserErrorCodeDescription
```

Usage

This annotation can be used to define metadata for a constant. This metadata can later be used to generate documentation.

It is very important that this annotation is only used in conjunction with constant, that is static fields.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

description() method**Syntax**

```
String description ()
```

infoLevel() method**Syntax**

```
InfoMode infoLevel ()
```

name() method**Syntax**

```
String name ()
```

InfoMode() enumeration

Enum Constant Summary

- **INFO** –
- **DEBUG** –
- **WARN** –
- **ERROR** –
- **FATAL** –

StatusCodes class

The StatusCodes class contains constants for status codes / error codes that are used in the code.

Syntax

```
public class StatusCodes
```

Derived classes

- *StatusCodes* on page 536

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

ERROR_ACTIVATION_AMOUNT_WRONG variable

Syntax

```
final int ERROR_ACTIVATION_AMOUNT_WRONG
```

ERROR_ACTIVATION_FAILED_PERMANENTLY variable

Syntax

```
final int ERROR_ACTIVATION_FAILED_PERMANENTLY
```

ERROR_AMOUNT_DIFFERS variable

Syntax

```
final int ERROR_AMOUNT_DIFFERS
```


*ERROR_AUTH_EXPIRED variable**Syntax*

```
final int ERROR_AUTH_EXPIRED
```

*ERROR_AUTHENTICATION_BLACKLISTED variable**Syntax*

```
final int ERROR_AUTHENTICATION_BLACKLISTED
```

*ERROR_AUTHENTICATION_BLOCKED_USER variable**Syntax*

```
final int ERROR_AUTHENTICATION_BLOCKED_USER
```

*ERROR_AUTHENTICATION_CREDENTIALS_EXPIRED variable**Syntax*

```
final int ERROR_AUTHENTICATION_CREDENTIALS_EXPIRED
```

*ERROR_AUTHENTICATION_INACTIVE_USER variable**Syntax*

```
final int ERROR_AUTHENTICATION_INACTIVE_USER
```

*ERROR_AUTHENTICATION_INVALID variable**Syntax*

```
final int ERROR_AUTHENTICATION_INVALID
```

*ERROR_AUTHENTICATION_INVALID_AND_BLACKLISTED variable**Syntax*

```
final int ERROR_AUTHENTICATION_INVALID_AND_BLACKLISTED
```

*ERROR_AUTHENTICATION_INVALID_AND_BLOCKED variable**Syntax*

```
final int ERROR_AUTHENTICATION_INVALID_AND_BLOCKED
```

ERROR_AUTHENTICATION_INVALID_CREDENTIALS variable

Syntax

```
final int ERROR_AUTHENTICATION_INVALID_CREDENTIALS
```

ERROR_AUTHENTICATION_MISSING variable

Syntax

```
final int ERROR_AUTHENTICATION_MISSING
```

ERROR_AUTHENTICATION_UNKNOWN_USER variable

Syntax

```
final int ERROR_AUTHENTICATION_UNKNOWN_USER
```

ERROR_AUTHORISATION_FAILED variable

Syntax

```
final int ERROR_AUTHORISATION_FAILED
```

ERROR_AUTHORISATION_INACTIVE_AGENT variable

Syntax

```
final int ERROR_AUTHORISATION_INACTIVE_AGENT
```

ERROR_AUTHORISATION_INVALID_ACCESS variable

Syntax

```
final int ERROR_AUTHORISATION_INVALID_ACCESS
```

ERROR_AUTHORISATION_LOGIN_NOT_ALLOWED variable

Syntax

```
final int ERROR_AUTHORISATION_LOGIN_NOT_ALLOWED
```

ERROR_AUTHORISATION_NO_AGENT variable

Syntax

```
final int ERROR_AUTHORISATION_NO_AGENT
```

*ERROR_AUTHORIZATION_PAYER_DENIED variable**Syntax*

```
final int ERROR_AUTHORIZATION_PAYER_DENIED
```

*ERROR_AUTHORIZATION_PAYER_UNREACHABLE variable**Syntax*

```
final int ERROR_AUTHORIZATION_PAYER_UNREACHABLE
```

*ERROR_AUTO_CAPTURE_NOT_ALLOWED variable**Syntax*

```
final int ERROR_AUTO_CAPTURE_NOT_ALLOWED
```

*ERROR_AUTO_CAPTURE_REQUIRED variable**Syntax*

```
final int ERROR_AUTO_CAPTURE_REQUIRED
```

*ERROR_BACKSTOP_FAILED variable**Syntax*

```
final int ERROR_BACKSTOP_FAILED
```

*ERROR_BULK_PROCESSING_GENERAL variable**Syntax*

```
final int ERROR_BULK_PROCESSING_GENERAL
```

*ERROR_BULK_PROCESSING_INVALID_LINE variable**Syntax*

```
final int ERROR_BULK_PROCESSING_INVALID_LINE
```

*ERROR_BULK_PROCESSING_IO variable**Syntax*

```
final int ERROR_BULK_PROCESSING_IO
```

ERROR_BULK_PROCESSING_NO_FILE_HANDLER variable

Syntax

```
final int ERROR_BULK_PROCESSING_NO_FILE_HANDLER
```

ERROR_CAPTURE_AMOUNT variable

Syntax

```
final int ERROR_CAPTURE_AMOUNT
```

ERROR_CAPTURE_CANCEL_AMOUNT variable

Syntax

```
final int ERROR_CAPTURE_CANCEL_AMOUNT
```

ERROR_CODE_INVALID_RESOURCE_ACCESS variable

Syntax

```
final int ERROR_CODE_INVALID_RESOURCE_ACCESS
```

ERROR_CONFIGURATION_CUSTOMER_INVALID variable

Syntax

```
final int ERROR_CONFIGURATION_CUSTOMER_INVALID
```

ERROR_CONFIGURATION_DATABASE variable

Syntax

```
final int ERROR_CONFIGURATION_DATABASE
```

ERROR_CREDENTIAL_BLOCKED variable

Syntax

```
final int ERROR_CREDENTIAL_BLOCKED
```

ERROR_CREDENTIAL_EXPIRED variable

Syntax

```
final int ERROR_CREDENTIAL_EXPIRED
```

*ERROR_CREDENTIAL_INVALID variable**Syntax*

```
final int ERROR_CREDENTIAL_INVALID
```

*ERROR_CREDENTIAL_INVALID_PATTERN variable**Syntax*

```
final int ERROR_CREDENTIAL_INVALID_PATTERN
```

*ERROR_CREDENTIAL_NO_POLICY variable**Syntax*

```
final int ERROR_CREDENTIAL_NO_POLICY
```

*ERROR_CREDENTIAL_NOT_FOUND variable**Syntax*

```
final int ERROR_CREDENTIAL_NOT_FOUND
```

*ERROR_CREDENTIAL_OK_BUT_EXPIRED variable**Syntax*

```
final int ERROR_CREDENTIAL_OK_BUT_EXPIRED
```

*ERROR_CREDENTIAL_OK_BUT_TEMPORARY variable**Syntax*

```
final int ERROR_CREDENTIAL_OK_BUT_TEMPORARY
```

*ERROR_CREDENTIAL_RETENTION_MATCH variable**Syntax*

```
final int ERROR_CREDENTIAL_RETENTION_MATCH
```

*ERROR_CREDENTIAL_TOO_LONG variable**Syntax*

```
final int ERROR_CREDENTIAL_TOO_LONG
```

ERROR_CREDENTIAL_TOO_SHORT variable

Syntax

```
final int ERROR_CREDENTIAL_TOO_SHORT
```

ERROR_CREDENTIAL_WEAK_BLOCK variable

Syntax

```
final int ERROR_CREDENTIAL_WEAK_BLOCK
```

ERROR_CREDENTIAL_WRONG variable

Syntax

```
final int ERROR_CREDENTIAL_WRONG
```

ERROR_CREDENTIAL_WRONG_AND_ACTION_TAKEN variable

Syntax

```
final int ERROR_CREDENTIAL_WRONG_AND_ACTION_TAKEN
```

ERROR_CUSTOMER_BL variable

Syntax

```
final int ERROR_CUSTOMER_BL
```

ERROR_CUSTOMER_BLACKLISTED variable

Syntax

```
final int ERROR_CUSTOMER_BLACKLISTED
```

ERROR_CUSTOMER_DATA_MISSING variable

Syntax

```
final int ERROR_CUSTOMER_DATA_MISSING
```

ERROR_CUSTOMER_EMAIL_MISSING variable

Syntax

```
final int ERROR_CUSTOMER_EMAIL_MISSING
```

*ERROR_CUSTOMER_INACTIVE variable**Syntax*

```
final int ERROR_CUSTOMER_INACTIVE
```

*ERROR_CUSTOMER_MSISDN_MISSING variable**Syntax*

```
final int ERROR_CUSTOMER_MSISDN_MISSING
```

*ERROR_CUSTOMER_NOT_FOUND variable**Syntax*

```
final int ERROR_CUSTOMER_NOT_FOUND
```

*ERROR_DB variable**Syntax*

```
final int ERROR_DB
```

*ERROR_ENTITY_REFERENCED variable**Syntax*

```
final int ERROR_ENTITY_REFERENCED
```

*ERROR_ENTITY_WRONG_STATUS variable**Syntax*

```
final int ERROR_ENTITY_WRONG_STATUS
```

*ERROR_GENERIC_ACCESS_DENIED variable**Syntax*

```
final int ERROR_GENERIC_ACCESS_DENIED
```

*ERROR_GENERIC_BL variable**Syntax*

```
final int ERROR_GENERIC_BL
```

ERROR_GENERIC_ENTITY_MANDATORY variable

Syntax

```
final int ERROR_GENERIC_ENTITY_MANDATORY
```

ERROR_GENERIC_ENTITY_NOT_FOUND variable

Syntax

```
final int ERROR_GENERIC_ENTITY_NOT_FOUND
```

ERROR_GENERIC_ENTITY_NOT_UNIQUE variable

Syntax

```
final int ERROR_GENERIC_ENTITY_NOT_UNIQUE
```

ERROR_GENERIC_ILLEGAL_DATA variable

Syntax

```
final int ERROR_GENERIC_ILLEGAL_DATA
```

ERROR_GROUP_PAYEE_PI variable

Syntax

```
final int ERROR_GROUP_PAYEE_PI
```

ERROR_GROUP_PAYER_PI variable

Syntax

```
final int ERROR_GROUP_PAYER_PI
```

ERROR_IVR_BUSY variable

Syntax

```
final int ERROR_IVR_BUSY
```

ERROR_IVR_COMMUNICATION variable

Syntax

```
final int ERROR_IVR_COMMUNICATION
```


*ERROR_IVR_CONNECT variable**Syntax*

```
final int ERROR_IVR_CONNECT
```

*ERROR_IVR_HANGUP variable**Syntax*

```
final int ERROR_IVR_HANGUP
```

*ERROR_IVR_NO_PICKUP variable**Syntax*

```
final int ERROR_IVR_NO_PICKUP
```

*ERROR_IVR_NO_USER_INPUT variable**Syntax*

```
final int ERROR_IVR_NO_USER_INPUT
```

*ERROR_IVR_SERVICE_TIMEOUT variable**Syntax*

```
final int ERROR_IVR_SERVICE_TIMEOUT
```

*ERROR_IVR_UNKNOWN variable**Syntax*

```
final int ERROR_IVR_UNKNOWN
```

*ERROR_LICENSE variable**Syntax*

```
final int ERROR_LICENSE
```

*ERROR_MAX_SESSIONS_REACHED variable**Syntax*

```
final int ERROR_MAX_SESSIONS_REACHED
```

ERROR_MULTIPLE_OPEN_TRANSACTIONS variable

Syntax

```
final int ERROR_MULTIPLE_OPEN_TRANSACTIONS
```

ERROR_NO_ACTIVATION_TRANSACTION variable

Syntax

```
final int ERROR_NO_ACTIVATION_TRANSACTION
```

ERROR_NO_OPEN_TRANSACTION variable

Syntax

```
final int ERROR_NO_OPEN_TRANSACTION
```

ERROR_NOTIFICATION_TYPE_NOT_SUPPORTED variable

Syntax

```
final int ERROR_NOTIFICATION_TYPE_NOT_SUPPORTED
```

ERROR_NOTIFICATIONS_CONNECT_FAILURE variable

Syntax

```
final int ERROR_NOTIFICATIONS_CONNECT_FAILURE
```

ERROR_NOTIFICATIONS_CONNECT_TIMEOUT variable

Syntax

```
final int ERROR_NOTIFICATIONS_CONNECT_TIMEOUT
```

ERROR_NOTIFICATIONS_PROTOCOL variable

Syntax

```
final int ERROR_NOTIFICATIONS_PROTOCOL
```

ERROR_NOTIFICATIONS_READ_TIMEOUT variable

Syntax

```
final int ERROR_NOTIFICATIONS_READ_TIMEOUT
```

*ERROR_NOTIFICATIONS_SEND variable**Syntax*

```
final int ERROR_NOTIFICATIONS_SEND
```

*ERROR_OBFUSCATION_PROCESS_FAILED variable**Syntax*

```
final int ERROR_OBFUSCATION_PROCESS_FAILED
```

*ERROR_PAYEE_ABSOLUTE_LIMIT variable**Syntax*

```
final int ERROR_PAYEE_ABSOLUTE_LIMIT
```

*ERROR_PAYEE_ABSOLUTE_LIMIT_COUNT variable**Syntax*

```
final int ERROR_PAYEE_ABSOLUTE_LIMIT_COUNT
```

*ERROR_PAYEE_DAILY_LIMIT variable**Syntax*

```
final int ERROR_PAYEE_DAILY_LIMIT
```

*ERROR_PAYEE_DAILY_LIMIT_COUNT variable**Syntax*

```
final int ERROR_PAYEE_DAILY_LIMIT_COUNT
```

*ERROR_PAYEE_INVALID variable**Syntax*

```
final int ERROR_PAYEE_INVALID
```

*ERROR_PAYEE_INVALID_DELEGATE variable**Syntax*

```
final int ERROR_PAYEE_INVALID_DELEGATE
```

ERROR_PAYEE_KYC_INSUFFICIENT variable

Syntax

```
final int ERROR_PAYEE_KYC_INSUFFICIENT
```

ERROR_PAYEE_MONTHLY_LIMIT variable

Syntax

```
final int ERROR_PAYEE_MONTHLY_LIMIT
```

ERROR_PAYEE_MONTHLY_LIMIT_COUNT variable

Syntax

```
final int ERROR_PAYEE_MONTHLY_LIMIT_COUNT
```

ERROR_PAYEE_OPEN_TRANSACTION variable

Syntax

```
final int ERROR_PAYEE_OPEN_TRANSACTION
```

ERROR_PAYEE_PAYER_IDENTICAL variable

Syntax

```
final int ERROR_PAYEE_PAYER_IDENTICAL
```

ERROR_PAYEE_PAYMENTINSTRUMENT_ABSOLUTE_LIMIT variable

Syntax

```
final int ERROR_PAYEE_PAYMENTINSTRUMENT_ABSOLUTE_LIMIT
```

ERROR_PAYEE_PAYMENTINSTRUMENT_ABSOLUTE_LIMIT_COUNT variable

Syntax

```
final int ERROR_PAYEE_PAYMENTINSTRUMENT_ABSOLUTE_LIMIT_COUNT
```

ERROR_PAYEE_PAYMENTINSTRUMENT_BLOCKED variable

Syntax

```
final int ERROR_PAYEE_PAYMENTINSTRUMENT_BLOCKED
```

*ERROR_PAYEE_PAYMENTINSTRUMENT_CANCEL_CAPTURE_AMOUNT
variable*

Syntax

```
final int ERROR_PAYEE_PAYMENTINSTRUMENT_CANCEL_CAPTURE_AMOUNT
```

ERROR_PAYEE_PAYMENTINSTRUMENT_CAPTURE_FROZEN variable

Syntax

```
final int ERROR_PAYEE_PAYMENTINSTRUMENT_CAPTURE_FROZEN
```

ERROR_PAYEE_PAYMENTINSTRUMENT_CVD_FAILED variable

Syntax

```
final int ERROR_PAYEE_PAYMENTINSTRUMENT_CVD_FAILED
```

ERROR_PAYEE_PAYMENTINSTRUMENT_DAILY_LIMIT variable

Syntax

```
final int ERROR_PAYEE_PAYMENTINSTRUMENT_DAILY_LIMIT
```

ERROR_PAYEE_PAYMENTINSTRUMENT_DAILY_LIMIT_COUNT variable

Syntax

```
final int ERROR_PAYEE_PAYMENTINSTRUMENT_DAILY_LIMIT_COUNT
```

ERROR_PAYEE_PAYMENTINSTRUMENT_ERROR variable

Syntax

```
final int ERROR_PAYEE_PAYMENTINSTRUMENT_ERROR
```

ERROR_PAYEE_PAYMENTINSTRUMENT_EXPIRED variable

Syntax

```
final int ERROR_PAYEE_PAYMENTINSTRUMENT_EXPIRED
```

ERROR_PAYEE_PAYMENTINSTRUMENT_INACTIVE variable

Syntax

```
final int ERROR_PAYEE_PAYMENTINSTRUMENT_INACTIVE
```

ERROR_PAYEE_PAYMENTINSTRUMENT_INCONSISTENCY variable

Syntax

```
final int ERROR_PAYEE_PAYMENTINSTRUMENT_INCONSISTENCY
```

ERROR_PAYEE_PAYMENTINSTRUMENT_IO variable

Syntax

```
final int ERROR_PAYEE_PAYMENTINSTRUMENT_IO
```

ERROR_PAYEE_PAYMENTINSTRUMENT_MONTHLY_LIMIT variable

Syntax

```
final int ERROR_PAYEE_PAYMENTINSTRUMENT_MONTHLY_LIMIT
```

ERROR_PAYEE_PAYMENTINSTRUMENT_MONTHLY_LIMIT_COUNT variable

Syntax

```
final int ERROR_PAYEE_PAYMENTINSTRUMENT_MONTHLY_LIMIT_COUNT
```

ERROR_PAYEE_PAYMENTINSTRUMENT_NO_BALANCE variable

Syntax

```
final int ERROR_PAYEE_PAYMENTINSTRUMENT_NO_BALANCE
```

ERROR_PAYEE_PAYMENTINSTRUMENT_NO_FUNDS variable

Syntax

```
final int ERROR_PAYEE_PAYMENTINSTRUMENT_NO_FUNDS
```

ERROR_PAYEE_PAYMENTINSTRUMENT_NOT_ALLOWED variable

Syntax

```
final int ERROR_PAYEE_PAYMENTINSTRUMENT_NOT_ALLOWED
```

ERROR_PAYEE_PAYMENTINSTRUMENT_SINGLE_LIMIT variable

Syntax

```
final int ERROR_PAYEE_PAYMENTINSTRUMENT_SINGLE_LIMIT
```

ERROR_PAYEE_PAYMENTINSTRUMENT_SINGLE_MINIMUM_UNDERRUN variable

Syntax

```
final int  
ERROR_PAYEE_PAYMENTINSTRUMENT_SINGLE_MINIMUM_UNDERRUN
```

ERROR_PAYEE_PAYMENTINSTRUMENT_WEEKLY_LIMIT variable

Syntax

```
final int ERROR_PAYEE_PAYMENTINSTRUMENT_WEEKLY_LIMIT
```

ERROR_PAYEE_PAYMENTINSTRUMENT_WEEKLY_LIMIT_COUNT variable

Syntax

```
final int ERROR_PAYEE_PAYMENTINSTRUMENT_WEEKLY_LIMIT_COUNT
```

ERROR_PAYEE_PAYMENTINSTRUMENT_WRONG_STATUS variable

Syntax

```
final int ERROR_PAYEE_PAYMENTINSTRUMENT_WRONG_STATUS
```

ERROR_PAYEE_PI_CURRENCY variable

Syntax

```
final int ERROR_PAYEE_PI_CURRENCY
```

ERROR_PAYEE_PI_NO_CREDIT variable

Syntax

```
final int ERROR_PAYEE_PI_NO_CREDIT
```

ERROR_PAYEE_PI_NONE variable

Syntax

```
final int ERROR_PAYEE_PI_NONE
```

ERROR_PAYEE_PI_UNKNOWN variable

Syntax

```
final int ERROR_PAYEE_PI_UNKNOWN
```

ERROR_PAYEE_SINGLE_LIMIT variable

Syntax

```
final int ERROR_PAYEE_SINGLE_LIMIT
```

ERROR_PAYEE_SINGLE_MINIMUM_UNDERRUN variable

Syntax

```
final int ERROR_PAYEE_SINGLE_MINIMUM_UNDERRUN
```

ERROR_PAYEE_UNKNOWN variable

Syntax

```
final int ERROR_PAYEE_UNKNOWN
```

ERROR_PAYEE_WALLET_ABSOLUTE_LIMIT variable

Syntax

```
final int ERROR_PAYEE_WALLET_ABSOLUTE_LIMIT
```

ERROR_PAYEE_WALLET_ABSOLUTE_LIMIT_COUNT variable

Syntax

```
final int ERROR_PAYEE_WALLET_ABSOLUTE_LIMIT_COUNT
```

ERROR_PAYEE_WALLET_DAILY_LIMIT variable

Syntax

```
final int ERROR_PAYEE_WALLET_DAILY_LIMIT
```

ERROR_PAYEE_WALLET_DAILY_LIMIT_COUNT variable

Syntax

```
final int ERROR_PAYEE_WALLET_DAILY_LIMIT_COUNT
```

ERROR_PAYEE_WALLET_MONTHLY_LIMIT variable

Syntax

```
final int ERROR_PAYEE_WALLET_MONTHLY_LIMIT
```


*ERROR_PAYEE_WALLET_MONTHLY_LIMIT_COUNT variable**Syntax*

```
final int ERROR_PAYEE_WALLET_MONTHLY_LIMIT_COUNT
```

*ERROR_PAYEE_WALLET_SINGLE_LIMIT variable**Syntax*

```
final int ERROR_PAYEE_WALLET_SINGLE_LIMIT
```

*ERROR_PAYEE_WALLET_SINGLE_MINIMUM_UNDERRUN variable**Syntax*

```
final int ERROR_PAYEE_WALLET_SINGLE_MINIMUM_UNDERRUN
```

*ERROR_PAYEE_WALLET_WEEKLY_LIMIT variable**Syntax*

```
final int ERROR_PAYEE_WALLET_WEEKLY_LIMIT
```

*ERROR_PAYEE_WALLET_WEEKLY_LIMIT_COUNT variable**Syntax*

```
final int ERROR_PAYEE_WALLET_WEEKLY_LIMIT_COUNT
```

*ERROR_PAYEE_WALLETENTRY_INVALID variable**Syntax*

```
final int ERROR_PAYEE_WALLETENTRY_INVALID
```

*ERROR_PAYEE_WALLETENTRY_INVALID_ALIAS variable**Syntax*

```
final int ERROR_PAYEE_WALLETENTRY_INVALID_ALIAS
```

*ERROR_PAYEE_WALLETENTRY_INVALID_PRIORITY variable**Syntax*

```
final int ERROR_PAYEE_WALLETENTRY_INVALID_PRIORITY
```

ERROR_PAYEE_WEEKLY_LIMIT variable

Syntax

```
final int ERROR_PAYEE_WEEKLY_LIMIT
```

ERROR_PAYEE_WEEKLY_LIMIT_COUNT variable

Syntax

```
final int ERROR_PAYEE_WEEKLY_LIMIT_COUNT
```

ERROR_PAYER_ABSOLUTE_LIMIT variable

Syntax

```
final int ERROR_PAYER_ABSOLUTE_LIMIT
```

ERROR_PAYER_ABSOLUTE_LIMIT_COUNT variable

Syntax

```
final int ERROR_PAYER_ABSOLUTE_LIMIT_COUNT
```

ERROR_PAYER_DAILY_LIMIT variable

Syntax

```
final int ERROR_PAYER_DAILY_LIMIT
```

ERROR_PAYER_DAILY_LIMIT_COUNT variable

Syntax

```
final int ERROR_PAYER_DAILY_LIMIT_COUNT
```

ERROR_PAYER_INVALID variable

Syntax

```
final int ERROR_PAYER_INVALID
```

ERROR_PAYER_INVALID_DELEGATE variable

Syntax

```
final int ERROR_PAYER_INVALID_DELEGATE
```

*ERROR_PAYER_KYC_INSUFFICIENT variable**Syntax*

```
final int ERROR_PAYER_KYC_INSUFFICIENT
```

*ERROR_PAYER_MANDATE_REQUIRED variable**Syntax*

```
final int ERROR_PAYER_MANDATE_REQUIRED
```

*ERROR_PAYER_MONTHLY_LIMIT variable**Syntax*

```
final int ERROR_PAYER_MONTHLY_LIMIT
```

*ERROR_PAYER_MONTHLY_LIMIT_COUNT variable**Syntax*

```
final int ERROR_PAYER_MONTHLY_LIMIT_COUNT
```

*ERROR_PAYER_OPEN_TRANSACTION variable**Syntax*

```
final int ERROR_PAYER_OPEN_TRANSACTION
```

*ERROR_PAYER_PAYMENTINSTRUMENT_ABSOLUTE_LIMIT variable**Syntax*

```
final int ERROR_PAYER_PAYMENTINSTRUMENT_ABSOLUTE_LIMIT
```

*ERROR_PAYER_PAYMENTINSTRUMENT_ABSOLUTE_LIMIT_COUNT variable**Syntax*

```
final int ERROR_PAYER_PAYMENTINSTRUMENT_ABSOLUTE_LIMIT_COUNT
```

*ERROR_PAYER_PAYMENTINSTRUMENT_BLOCKED variable**Syntax*

```
final int ERROR_PAYER_PAYMENTINSTRUMENT_BLOCKED
```

ERROR_PAYER_PAYMENTINSTRUMENT_CANCEL_CAPTURE_AMOUNT variable

Syntax

```
final int ERROR_PAYER_PAYMENTINSTRUMENT_CANCEL_CAPTURE_AMOUNT
```

ERROR_PAYER_PAYMENTINSTRUMENT_CAPTURE_FROZEN variable

Syntax

```
final int ERROR_PAYER_PAYMENTINSTRUMENT_CAPTURE_FROZEN
```

ERROR_PAYER_PAYMENTINSTRUMENT_CVD_FAILED variable

Syntax

```
final int ERROR_PAYER_PAYMENTINSTRUMENT_CVD_FAILED
```

ERROR_PAYER_PAYMENTINSTRUMENT_DAILY_LIMIT variable

Syntax

```
final int ERROR_PAYER_PAYMENTINSTRUMENT_DAILY_LIMIT
```

ERROR_PAYER_PAYMENTINSTRUMENT_DAILY_LIMIT_COUNT variable

Syntax

```
final int ERROR_PAYER_PAYMENTINSTRUMENT_DAILY_LIMIT_COUNT
```

ERROR_PAYER_PAYMENTINSTRUMENT_ERROR variable

Syntax

```
final int ERROR_PAYER_PAYMENTINSTRUMENT_ERROR
```

ERROR_PAYER_PAYMENTINSTRUMENT_EXPIRED variable

Syntax

```
final int ERROR_PAYER_PAYMENTINSTRUMENT_EXPIRED
```

ERROR_PAYER_PAYMENTINSTRUMENT_INACTIVE variable

Syntax

```
final int ERROR_PAYER_PAYMENTINSTRUMENT_INACTIVE
```

*ERROR_PAYER_PAYMENTINSTRUMENT_INCONSISTENCY variable**Syntax*

```
final int ERROR_PAYER_PAYMENTINSTRUMENT_INCONSISTENCY
```

*ERROR_PAYER_PAYMENTINSTRUMENT_INVALID variable**Syntax*

```
final int ERROR_PAYER_PAYMENTINSTRUMENT_INVALID
```

*ERROR_PAYER_PAYMENTINSTRUMENT_IO variable**Syntax*

```
final int ERROR_PAYER_PAYMENTINSTRUMENT_IO
```

*ERROR_PAYER_PAYMENTINSTRUMENT_MONTHLY_LIMIT variable**Syntax*

```
final int ERROR_PAYER_PAYMENTINSTRUMENT_MONTHLY_LIMIT
```

*ERROR_PAYER_PAYMENTINSTRUMENT_MONTHLY_LIMIT_COUNT variable**Syntax*

```
final int ERROR_PAYER_PAYMENTINSTRUMENT_MONTHLY_LIMIT_COUNT
```

*ERROR_PAYER_PAYMENTINSTRUMENT_NO_BALANCE variable**Syntax*

```
final int ERROR_PAYER_PAYMENTINSTRUMENT_NO_BALANCE
```

*ERROR_PAYER_PAYMENTINSTRUMENT_NO_FUNDS variable**Syntax*

```
final int ERROR_PAYER_PAYMENTINSTRUMENT_NO_FUNDS
```

*ERROR_PAYER_PAYMENTINSTRUMENT_NOT_ALLOWED variable**Syntax*

```
final int ERROR_PAYER_PAYMENTINSTRUMENT_NOT_ALLOWED
```

ERROR_PAYER_PAYMENTINSTRUMENT_SINGLE_LIMIT variable

Syntax

```
final int ERROR_PAYER_PAYMENTINSTRUMENT_SINGLE_LIMIT
```

ERROR_PAYER_PAYMENTINSTRUMENT_SINGLE_MINIMUM_UNDERRUN variable

Syntax

```
final int  
ERROR_PAYER_PAYMENTINSTRUMENT_SINGLE_MINIMUM_UNDERRUN
```

ERROR_PAYER_PAYMENTINSTRUMENT_WEEKLY_LIMIT variable

Syntax

```
final int ERROR_PAYER_PAYMENTINSTRUMENT_WEEKLY_LIMIT
```

ERROR_PAYER_PAYMENTINSTRUMENT_WEEKLY_LIMIT_COUNT variable

Syntax

```
final int ERROR_PAYER_PAYMENTINSTRUMENT_WEEKLY_LIMIT_COUNT
```

ERROR_PAYER_PAYMENTINSTRUMENT_WRONG_STATUS variable

Syntax

```
final int ERROR_PAYER_PAYMENTINSTRUMENT_WRONG_STATUS
```

ERROR_PAYER_PI_CURRENCY variable

Syntax

```
final int ERROR_PAYER_PI_CURRENCY
```

ERROR_PAYER_PI_NO_DEBIT variable

Syntax

```
final int ERROR_PAYER_PI_NO_DEBIT
```

ERROR_PAYER_PI_NONE variable

Syntax

```
final int ERROR_PAYER_PI_NONE
```

*ERROR_PAYER_PI_NOT_SUPPORTED variable**Syntax*

```
final int ERROR_PAYER_PI_NOT_SUPPORTED
```

*ERROR_PAYER_PI_UNKNOWN variable**Syntax*

```
final int ERROR_PAYER_PI_UNKNOWN
```

*ERROR_PAYER_SINGLE_LIMIT variable**Syntax*

```
final int ERROR_PAYER_SINGLE_LIMIT
```

*ERROR_PAYER_SINGLE_MINIMUM_UNDERRUN variable**Syntax*

```
final int ERROR_PAYER_SINGLE_MINIMUM_UNDERRUN
```

*ERROR_PAYER_UNKNOWN variable**Syntax*

```
final int ERROR_PAYER_UNKNOWN
```

*ERROR_PAYER_WALLET_ABSOLUTE_LIMIT variable**Syntax*

```
final int ERROR_PAYER_WALLET_ABSOLUTE_LIMIT
```

*ERROR_PAYER_WALLET_ABSOLUTE_LIMIT_COUNT variable**Syntax*

```
final int ERROR_PAYER_WALLET_ABSOLUTE_LIMIT_COUNT
```

*ERROR_PAYER_WALLET_DAILY_LIMIT variable**Syntax*

```
final int ERROR_PAYER_WALLET_DAILY_LIMIT
```

ERROR_PAYER_WALLET_DAILY_LIMIT_COUNT variable

Syntax

```
final int ERROR_PAYER_WALLET_DAILY_LIMIT_COUNT
```

ERROR_PAYER_WALLET_MONTHLY_LIMIT variable

Syntax

```
final int ERROR_PAYER_WALLET_MONTHLY_LIMIT
```

ERROR_PAYER_WALLET_MONTHLY_LIMIT_COUNT variable

Syntax

```
final int ERROR_PAYER_WALLET_MONTHLY_LIMIT_COUNT
```

ERROR_PAYER_WALLET_SINGLE_LIMIT variable

Syntax

```
final int ERROR_PAYER_WALLET_SINGLE_LIMIT
```

ERROR_PAYER_WALLET_SINGLE_MINIMUM_UNDERRUN variable

Syntax

```
final int ERROR_PAYER_WALLET_SINGLE_MINIMUM_UNDERRUN
```

ERROR_PAYER_WALLET_WEEKLY_LIMIT variable

Syntax

```
final int ERROR_PAYER_WALLET_WEEKLY_LIMIT
```

ERROR_PAYER_WALLET_WEEKLY_LIMIT_COUNT variable

Syntax

```
final int ERROR_PAYER_WALLET_WEEKLY_LIMIT_COUNT
```

ERROR_PAYER_WALLETENTRY_INVALID variable

Syntax

```
final int ERROR_PAYER_WALLETENTRY_INVALID
```


*ERROR_PAYER_WALLETENTRY_INVALID_ALIAS variable**Syntax*

```
final int ERROR_PAYER_WALLETENTRY_INVALID_ALIAS
```

*ERROR_PAYER_WALLETENTRY_INVALID_PRIORITY variable**Syntax*

```
final int ERROR_PAYER_WALLETENTRY_INVALID_PRIORITY
```

*ERROR_PAYER_WEEKLY_LIMIT variable**Syntax*

```
final int ERROR_PAYER_WEEKLY_LIMIT
```

*ERROR_PAYER_WEEKLY_LIMIT_COUNT variable**Syntax*

```
final int ERROR_PAYER_WEEKLY_LIMIT_COUNT
```

*ERROR_PENDING_CONTINUE_MISSING_PRIVILEGE variable**Syntax*

```
final int ERROR_PENDING_CONTINUE_MISSING_PRIVILEGE
```

*ERROR_PENDING_FOR_APPROVAL variable**Syntax*

```
final int ERROR_PENDING_FOR_APPROVAL
```

*ERROR_PENDING_FOR_CLEARING_SVA variable**Syntax*

```
final int ERROR_PENDING_FOR_CLEARING_SVA
```

*ERROR_PENDING_INVALID_RESPONSE variable**Syntax*

```
final int ERROR_PENDING_INVALID_RESPONSE
```

ERROR_PENDING_START_MISSING_PRIVILEGE variable

Syntax

```
final int ERROR_PENDING_START_MISSING_PRIVILEGE
```

ERROR_PENDING_TASK_ACCESS variable

Syntax

```
final int ERROR_PENDING_TASK_ACCESS
```

ERROR_PENDING_TASK_CLAIMED variable

Syntax

```
final int ERROR_PENDING_TASK_CLAIMED
```

ERROR_PENDING_TASK_INCOMPLETE variable

Syntax

```
final int ERROR_PENDING_TASK_INCOMPLETE
```

ERROR_PENDING_TASK_NOT_FOUND variable

Syntax

```
final int ERROR_PENDING_TASK_NOT_FOUND
```

ERROR_PENDING_UNEXPECTED variable

Syntax

```
final int ERROR_PENDING_UNEXPECTED
```

ERROR_PERSISTENT_LOGIN_EXPIRED variable

Syntax

```
final int ERROR_PERSISTENT_LOGIN_EXPIRED
```

ERROR_PERSISTENT_LOGIN_NOT_SUPPORTED variable

Syntax

```
final int ERROR_PERSISTENT_LOGIN_NOT_SUPPORTED
```

*ERROR_PERSISTENT_LOGIN_STOLEN variable**Syntax*

```
final int ERROR_PERSISTENT_LOGIN_STOLEN
```

*ERROR_PERSISTENT_LOGIN_TOO_MANY variable**Syntax*

```
final int ERROR_PERSISTENT_LOGIN_TOO_MANY
```

*ERROR_PICKUP_ATTRIBUTE_NOT_UNIQUE variable**Syntax*

```
final int ERROR_PICKUP_ATTRIBUTE_NOT_UNIQUE
```

*ERROR_PICKUP_INTERNAL_VOUCHER_INACTIVE variable**Syntax*

```
final int ERROR_PICKUP_INTERNAL_VOUCHER_INACTIVE
```

*ERROR_PICKUP_INTERNAL_VOUCHER_NOT_FOUND variable**Syntax*

```
final int ERROR_PICKUP_INTERNAL_VOUCHER_NOT_FOUND
```

*ERROR_PICKUP_INTERNAL_VOUCHER_TXN_NOT_FOUND variable**Syntax*

```
final int ERROR_PICKUP_INTERNAL_VOUCHER_TXN_NOT_FOUND
```

*ERROR_PICKUP_INVALID_CODE variable**Syntax*

```
final int ERROR_PICKUP_INVALID_CODE
```

*ERROR_PICKUP_MAX_TRIES variable**Syntax*

```
final int ERROR_PICKUP_MAX_TRIES
```

ERROR_PICKUP_NO_CODE variable

Syntax

```
final int ERROR_PICKUP_NO_CODE
```

ERROR_REMOTE_SESSION_ACTIVE variable

Syntax

```
final int ERROR_REMOTE_SESSION_ACTIVE
```

ERROR_REMOTE_UPDATE variable

Syntax

```
final int ERROR_REMOTE_UPDATE
```

ERROR_REQUEST_BLACKLISTED_CUSTOMER variable

Syntax

```
final int ERROR_REQUEST_BLACKLISTED_CUSTOMER
```

ERROR_REQUEST_INACTIVE_CUSTOMER variable

Syntax

```
final int ERROR_REQUEST_INACTIVE_CUSTOMER
```

ERROR_REQUEST_INVALID variable

Syntax

```
final int ERROR_REQUEST_INVALID
```

ERROR_REQUEST_INVALID_MSISDN variable

Syntax

```
final int ERROR_REQUEST_INVALID_MSISDN
```

ERROR_REQUEST_INVALID_TXN variable

Syntax

```
final int ERROR_REQUEST_INVALID_TXN
```

*ERROR_REQUEST_INVALID_TXN_STATUS variable**Syntax*

```
final int ERROR_REQUEST_INVALID_TXN_STATUS
```

*ERROR_REQUEST_TEST_LIVE_MIX variable**Syntax*

```
final int ERROR_REQUEST_TEST_LIVE_MIX
```

*ERROR_REQUEST_UNKNOWN_CUSTOMER variable**Syntax*

```
final int ERROR_REQUEST_UNKNOWN_CUSTOMER
```

*ERROR_REQUEST_UNKNOWN_TXN variable**Syntax*

```
final int ERROR_REQUEST_UNKNOWN_TXN
```

*ERROR_RESTRICTION_INDIVIDUAL variable**Syntax*

```
final int ERROR_RESTRICTION_INDIVIDUAL
```

*ERROR_RESTRICTION_PAYEE_AMOUNT variable**Syntax*

```
final int ERROR_RESTRICTION_PAYEE_AMOUNT
```

*ERROR_RESTRICTION_PAYEE_AMOUNT_MIN variable**Syntax*

```
final int ERROR_RESTRICTION_PAYEE_AMOUNT_MIN
```

*ERROR_RESTRICTION_PAYEE_COUNT variable**Syntax*

```
final int ERROR_RESTRICTION_PAYEE_COUNT
```

ERROR_RESTRICTION_PAYEE_SUM variable

Syntax

```
final int ERROR_RESTRICTION_PAYEE_SUM
```

ERROR_RESTRICTION_PAYER_AMOUNT variable

Syntax

```
final int ERROR_RESTRICTION_PAYER_AMOUNT
```

ERROR_RESTRICTION_PAYER_AMOUNT_MIN variable

Syntax

```
final int ERROR_RESTRICTION_PAYER_AMOUNT_MIN
```

ERROR_RESTRICTION_PAYER_COUNT variable

Syntax

```
final int ERROR_RESTRICTION_PAYER_COUNT
```

ERROR_RESTRICTION_PAYER_SUM variable

Syntax

```
final int ERROR_RESTRICTION_PAYER_SUM
```

ERROR_SESSION_CLOSED variable

Syntax

```
final int ERROR_SESSION_CLOSED
```

ERROR_SESSION_EXPIRED variable

Syntax

```
final int ERROR_SESSION_EXPIRED
```

ERROR_SESSION_NOT_FOUND variable

Syntax

```
final int ERROR_SESSION_NOT_FOUND
```

*ERROR_SESSION_NOT_SUPPORTED variable**Syntax*

```
final int ERROR_SESSION_NOT_SUPPORTED
```

*ERROR_SESSION_TOO_MANY variable**Syntax*

```
final int ERROR_SESSION_TOO_MANY
```

*ERROR_SUBSCRIPTION_UNKNOWN variable**Syntax*

```
final int ERROR_SUBSCRIPTION_UNKNOWN
```

*ERROR_TRACING_NO_RESPONSE_CACHED variable**Syntax*

```
final int ERROR_TRACING_NO_RESPONSE_CACHED
```

*ERROR_TRACING_REPEAT_FLAG_MISSING variable**Syntax*

```
final int ERROR_TRACING_REPEAT_FLAG_MISSING
```

*ERROR_TRACING_REPEAT_SET_BUT_NO_MESSAGE_FOUND variable**Syntax*

```
final int ERROR_TRACING_REPEAT_SET_BUT_NO_MESSAGE_FOUND
```

*ERROR_TRACING_STILL_IN_PROCESS variable**Syntax*

```
final int ERROR_TRACING_STILL_IN_PROCESS
```

*ERROR_TRACING_WRONG_USER_FOR_REPEAT variable**Syntax*

```
final int ERROR_TRACING_WRONG_USER_FOR_REPEAT
```

ERROR_TRANSACTION_EXPIRED variable

Syntax

```
final int ERROR_TRANSACTION_EXPIRED
```

ERROR_TRANSACTION_LOCKED variable

Syntax

```
final int ERROR_TRANSACTION_LOCKED
```

ERROR_TRANSACTION_PERMANENT variable

Syntax

```
final int ERROR_TRANSACTION_PERMANENT
```

ERROR_TRANSACTION_REPETITION variable

Syntax

```
final int ERROR_TRANSACTION_REPETITION
```

ERROR_TXN_AUTH_REQUIRED variable

Syntax

```
final int ERROR_TXN_AUTH_REQUIRED
```

ERROR_TXN_AUTH_RESET_FAILED variable

Syntax

```
final int ERROR_TXN_AUTH_RESET_FAILED
```

ERROR_TXN_AUTH_WRONG variable

Syntax

```
final int ERROR_TXN_AUTH_WRONG
```

ERROR_TXN_AUTH_WRONG_FINAL variable

Syntax

```
final int ERROR_TXN_AUTH_WRONG_FINAL
```


*ERROR_TXN_MANUAL_DECISION_REQUIRED variable**Syntax*

```
final int ERROR_TXN_MANUAL_DECISION_REQUIRED
```

*ERROR_UNKNOWN variable**Syntax*

```
final int ERROR_UNKNOWN
```

*ERROR_VIRUS_INFECTION variable**Syntax*

```
final int ERROR_VIRUS_INFECTION
```

*ERROR_VIRUS_SCAN variable**Syntax*

```
final int ERROR_VIRUS_SCAN
```

*ERROR_VIRUS_SYSTEM variable**Syntax*

```
final int ERROR_VIRUS_SYSTEM
```

*ERROR_WORKFLOW_FAILED variable**Syntax*

```
final int ERROR_WORKFLOW_FAILED
```

*ERROR_WORKFLOW_REJECTED variable**Syntax*

```
final int ERROR_WORKFLOW_REJECTED
```

*OK variable**Syntax*

```
final int OK
```

SystemConfigurationErrorException class

The SystemConfigurationErrorException is thrown when an internal configuration is missing/wrong.

Syntax

```
public class SystemConfigurationErrorException
```

Usage

This means that the input data is correct but the processing requires system configuration that is invalid.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

SystemConfigurationErrorException() constructor

Syntax

```
SystemConfigurationErrorException ()
```

SystemConfigurationErrorException(String) constructor

Syntax

```
SystemConfigurationErrorException ( String message )
```

SystemConfigurationErrorException(Throwable) constructor

Syntax

```
SystemConfigurationErrorException ( Throwable cause )
```

SystemConfigurationErrorException(String, Throwable) constructor

Syntax

```
SystemConfigurationErrorException ( String message , Throwable cause )
```

getErrorCode() method

Syntax

```
int getErrorCode ()
```

TxnStatus class

This class lists all available transaction status.

Syntax

```
public class TxnStatus
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

ACKNOWLEDGED variable

Acknowledged.

Syntax

```
final int ACKNOWLEDGED
```

AUTH_REQUIRED_BOTH variable

Authentication required to continue transaction processing from both (payer and payee)

Syntax

```
final int AUTH_REQUIRED_BOTH
```

AUTH_REQUIRED_PAYEE variable

Authentication required to continue transaction by payee.

Syntax

```
final int AUTH_REQUIRED_PAYEE
```

AUTH_REQUIRED_PAYER variable

Authentication required to continue transaction by payer.

Syntax

```
final int AUTH_REQUIRED_PAYER
```

AUTHENTICATED variable

Authenticated.

Syntax

```
final int AUTHENTICATED
```

AUTHORIZATION_CANCELED variable

Authorization Cancel.

Syntax

```
final int AUTHORIZATION_CANCELED
```

AUTHORIZED variable

Authorized.

Syntax

```
final int AUTHORIZED
```

CAPTURE_CANCELED variable

Capture Cancel.

Syntax

```
final int CAPTURE_CANCELED
```

CAPTURED variable

Captured.

Syntax

```
final int CAPTURED
```

DECISION_WAITING variable

Waiting for Manual decision.

Syntax

```
final int DECISION_WAITING
```

DELIVERED variable

Delivered.

Syntax

```
final int DELIVERED
```

FULLY_PROCESSED variable

Fully processed.

Syntax

```
final int FULLY_PROCESSED
```

INITIAL variable

Initial.

Syntax

```
final int INITIAL
```

PRE_AUTHORIZED variable

Pre-Authorized.

Syntax

```
final int PRE_AUTHORIZED
```

UserAccessingNotOwnedResourceException class

The UserAccessingNotOwnedResourceException is thrown when a user (caller) tries to access a resource that belongs to another user.

Syntax

```
public class UserAccessingNotOwnedResourceException
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

UserAccessingNotOwnedResourceException(final String) constructor**Syntax**

```
UserAccessingNotOwnedResourceException ( final String message )
```

Parameters

- **message** –

UserAccessingNotOwnedResourceException(final String, final Throwable) constructor**Syntax**

```
UserAccessingNotOwnedResourceException ( final String message ,  
final Throwable cause )
```

Parameters

- **message** –
- **cause** –

UserAccessingNotOwnedResourceException(final Throwable) constructor

Syntax

`UserAccessingNotOwnedResourceException ( final Throwable cause )`

Parameters

- **cause** –

getErrorCode() method

Syntax

`int getErrorCode ()`

VirusScanException class

Syntax

`public class VirusScanException`

Usage

Common utility class that wraps any exception thrown by the virus scan framework into an Mobiliser Service Exception Usually this class is initialized via exception thrown by the virus scan framework

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

VirusScanException(Throwable, String) constructor

Syntax

`VirusScanException ( Throwable exception , String message )`

getErrorCode() method

Syntax

`int getErrorCode ()`

REASON() enumeration*Enum Constant Summary*

- **INFECTION** –
- **SCAN** –
- **SYSTEM** –
- **OTHER** –

wallet package*Members*

All public members of the wallet package.

- **IBalanceAlertLogic interface** – The IBalanceAlertLogic interface contains methods to manage Balance Alerts.
- **IPILogic interface** – The IPILogic interface defines methods that are used to manage payment instruments.
- **ISepaMandateLogic interface** – The ISepaMandateLogic interface defines methods used to handle (SEPA) mandates which regulate access to payment instruments (mainly bank accounts).

IBalanceAlertLogic interface

The IBalanceAlertLogic interface contains methods to manage Balance Alerts.

Syntax

```
public interface IBalanceAlertLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

createBalanceAlert(BalanceAlert, long) method**Syntax**

```
long createBalanceAlert ( BalanceAlert balanceAlert , long callerId )
```

Parameters

- **balanceAlert** –
- **callerId** –

Returns

balance alert ID

Exceptions

- **MobiliserServiceException** –

Usage

Create a Balance Alert

balance alert ID

deactivateBalanceAlert(long, long) method

Syntax

```
void deactivateBalanceAlert ( long balanceAlertId ,   long callerId )
```

Parameters

- **balanceAlertId** –
- **callerId** –

Exceptions

- **MobiliserServiceException** –

Usage

Deactivate an existing balance alert

getBalanceAlert(final long) method

Syntax

```
BalanceAlert getBalanceAlert ( final long balanceAlertId )
```

Parameters

- **balanceAlertId** – the id of the balance alert

Returns

the balance alert

Usage

Retrieve the balance alert by id.

the balance alert

getBalanceAlertsByPaymentInstrument(long, boolean, Long) method

Syntax

```
List< BalanceAlert > getBalanceAlertsByPaymentInstrument ( long  
paymentInstrumentId , boolean includeActive , Long minThreshold )
```

Parameters

- **paymentInstrumentId** –
- **includeActive** –
- **minThreshold** –

Returns

list of balance alerts

Exceptions

- **MobiliserServiceException** –

Usage

Retrieve all balance alerts by payment instrument

list of balance alerts

updateBalanceAlert(BalanceAlert, long) method

Syntax

```
void updateBalanceAlert ( BalanceAlert balanceAlert , long callerId )
```

Parameters

- **balanceAlert** –
- **callerId** –

Exceptions

- **MobiliserServiceException** –

Usage

Update an existing Balance Alert

IPILogic interface

The IPILogic interface defines methods that are used to manage payment instruments.

Syntax

```
public interface IPILogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

continuePendingWalletEntry(long, boolean, long) method

Continues a pending add wallet entry process.

Syntax

```
void continuePendingWalletEntry ( long taskId ,   boolean approve ,  
long callerId )
```

Parameters

- **request** –

createPaymentInstrument(PaymentInstrument, BankAccount, long) method

Syntax

```
long createPaymentInstrument ( PaymentInstrument paymentInstrument ,  
BankAccount bankAccount ,   long callerId )
```

Parameters

- **paymentInstrument** –
- **bankAccount** –
- **callerId** –

Returns

PI ID

Exceptions

- **MobiliserBusinessLogicException** –

Usage

Create a Bank Account

PI ID

createPaymentInstrument(PaymentInstrument, Card, long) method

Syntax

```
long createPaymentInstrument ( PaymentInstrument paymentInstrument ,  
Card card , long callerId )
```

Parameters

- **paymentInstrument** –
- **card** –
- **callerId** –

Returns

PI ID

Exceptions

- **MobiliserBusinessLogicException** –

Usage

Create a (Credit) Card

PI ID

createPaymentInstrument(PaymentInstrument, StoredValueAccount, long) method

Syntax

```
long createPaymentInstrument ( PaymentInstrument paymentInstrument ,  
StoredValueAccount storedValueAccount , long callerId )
```

Parameters

- **paymentInstrument** –
- **storedValueAccount** –
- **callerId** –

Returns

PI ID

Exceptions

- **MobiliserBusinessLogicException** –

Usage

Create a Stored Value Account

PI ID

createPaymentInstrument(PaymentInstrument, ExternalAccount, long) method

Syntax

long createPaymentInstrument (PaymentInstrument *paymentInstrument* ,
ExternalAccount *externalAccount* , long *callerId*)

Parameters

- **paymentInstrument** –
- **externalAccount** –
- **callerId** –

Returns

PI ID

Exceptions

- **MobiliserBusinessLogicException** –

Usage

Create a External Account

PI ID

createWalletEntry(WalletEntry, Boolean, Boolean, long) method

Syntax

long createWalletEntry (WalletEntry *walletEntry* , Boolean
primaryDebit , Boolean *primaryCredit* , long *callerId*)

Parameters

- **walletEntry** –
- **primaryDebit** –

- **primaryCredit** –

Returns

the wallet entry ID

Exceptions

- **MobiliserBusinessLogicException** –

Usage

Create a customer wallet entry

the wallet entry ID

createWrkWalletEntry(WrkWalletEntry, long) method

Creates a new add wallet entry process configuration.

Syntax

```
void createWrkWalletEntry ( WrkWalletEntry wrkWalletentry , long callerId ) throws EntityNotUniqueException
```

Parameters

- **wrkWalletentry** –

deactivatePaymentInstrument(long, long) method

Syntax

```
void deactivatePaymentInstrument ( long paymentInstrumentId , long callerId )
```

Parameters

- **paymentInstrumentId** –

Exceptions

- **MobiliserBusinessLogicException** –

Usage

Deactivate an existing payment instrument

deletePaymentInstrument(long, long) method

Syntax

```
void deletePaymentInstrument ( long paymentInstrumentId , long callerId )
```

Parameters

- **paymentInstrumentId** –

Exceptions

- **MobiliserBusinessLogicException** –

Usage

Delete an existing payment instrument

deleteWalletEntry(long, long) method

Syntax

```
void deleteWalletEntry ( long walletEntryId , long callerId )
```

Parameters

- **walletEntryId** –

Exceptions

- **MobiliserBusinessLogicException** –

Usage

Delete an existing customer wallet entry

deleteWrkWalletEntry(int, long) method

Deletes an existing add wallet entry process configuration.

Syntax

```
void deleteWrkWalletEntry ( int piTypeId , long callerId ) throws  
EntityNotFoundException
```

Parameters

- **request** –

findPaymentInstrument(final UseCase, final Customer, final boolean, final PaymentInstrumentCriteria) method

Syntax

```
PaymentInstrument findPaymentInstrument ( final UseCase useCase ,  
final Customer customer , final boolean payee , final  
PaymentInstrumentCriteria criteria )
```

Parameters

- **useCase** – use case
- **customer** – the Customer for which to fetch the payment instrument
- **payee** – true if customer is the payee; false if he is the payer
- **criteria** – criteria for the payment instrument

Returns

the payment instrument

Usage

Identifies the payment instrument for the specified customer.

the payment instrument

findPendingWalletEntries(Integer, Long, Date, Date) method

finds wallet entries waiting for the approval of the add wallet entry process

Syntax

```
Map< Long, WalletEntry > findPendingWalletEntries ( Integer  
piTypeId , Long CustomerId , Date fromDate , Date toDate )
```

Parameters

- **request** –

Returns

a map mapping the taskId to the actual wallet entry

Usage

a map mapping the taskId to the actual wallet entry

getActivePaymentInstrument(long) method

Syntax

`PaymentInstrument getActivePaymentInstrument ( long paymentInstrumentId )` throws `EntityNotFoundException`

Parameters

- **`paymentInstrumentId`** –

Returns

the active payment instrument

Exceptions

- **`EntityNotFoundException`** –
- **`PaymentInstrumentInactiveException`** –

Usage

the active payment instrument

getPaymentInstrument(long, boolean) method

Syntax

`PaymentInstrument getPaymentInstrument ( long paymentInstrumentId , boolean includeInactive )`

Parameters

- **`paymentInstrumentId`** –
- **`includeInactive`** –

Returns

a payment instrument

Exceptions

- **`MobiliserBusinessLogicException`** –

Usage

Retrieves a payment instrument

a payment instrument

getPaymentInstrumentBalance(long, long) method

Syntax

Balance getPaymentInstrumentBalance (long *paymentInstrumentId* , long *callerId*)

Parameters

- **paymentInstrumentId** –
- **callerId** –

Returns

the payment instrument balance

Usage

the payment instrument balance

getPaymentInstrumentsByCustomer(long, Integer, Integer, boolean) method

Syntax

List< PaymentInstrument > getPaymentInstrumentsByCustomer
(long *customerId* , Integer *piClassFilter* , Integer *piTypeFilter* , boolean *includeInactive*)

Parameters

- **customerId** –
- **piClassFilter** –
- **piTypeFilter** –
- **includeInactive** –

Returns

list of payment instruments

Exceptions

- **MobiliserBusinessLogicException** –

Usage

Retrieve all payment instrument by customer

list of payment instruments

getWalletEntriesByCustomer(long, Integer, Integer) method

Syntax

```
List< WalletEntry >getWalletEntriesByCustomer ( long customerId ,  
Integer piClassFilter , Integer piTypeFilter )
```

Parameters

- **customerId** –

Returns

the wallet entries

Exceptions

- **MobiliserBusinessLogicException** –
- **WalletNotFoundException** –

Usage

Retrieve the customers wallet entries.

the wallet entries

getWalletEntry(long) method

Syntax

```
WalletEntry getWalletEntry ( long walletEntryId )
```

Parameters

- **walletEntryId** –

Returns

a wallet entry

Exceptions

- **MobiliserBusinessLogicException** –

Usage

Retrieves a wallet entry

a wallet entry

getWrkWalletEntries() method

Fetches all add wallet entry process configurations.

Syntax

```
List< WrkWalletEntry > getWrkWalletEntries ()
```

getWrkWalletEntry(final int) method

Fetches wallet entry process configurations for a specific pi type.

Syntax

```
WrkWalletEntry getWrkWalletEntry ( final int piTypeId )
```

Returns

the configuration

Usage

the configuration

obfuscateWalletPiAccounts(long, long) method

Obfuscates the wallet, PI and related accounts for a customer.

Syntax

```
void obfuscateWalletPiAccounts ( long customerId , long callerId )
```

Parameters

- **customerId** –
- **callerId** –

updatePaymentInstrument(PaymentInstrument, BankAccount, long) method

Syntax

```
void updatePaymentInstrument ( PaymentInstrument paymentInstrument ,  
BankAccount bankAccount , long callerId )
```

Parameters

- **paymentInstrument** –
- **bankAccount** –
- **callerId** –

Exceptions

- **MobiliserBusinessLogicException** –

Usage

Update an existing Bank Account

updatePaymentInstrument(PaymentInstrument, Card, long) method

Syntax

```
void updatePaymentInstrument ( PaymentInstrument paymentInstrument ,  
Card card , long callerId )
```

Parameters

- **paymentInstrument** –
- **card** –
- **callerId** –

Exceptions

- **MobiliserBusinessLogicException** –

Usage

Update an existing (Credit) Card

updatePaymentInstrument(PaymentInstrument, StoredValueAccount, long) method

Syntax

```
void updatePaymentInstrument ( PaymentInstrument paymentInstrument ,  
StoredValueAccount storedValueAccount , long callerId )
```

Parameters

- **paymentInstrument** –
- **storedValueAccount** –
- **callerId** –

Exceptions

- **MobiliserBusinessLogicException** –

Usage

Update an existing Stored Value Account

updatePaymentInstrument(PaymentInstrument, ExternalAccount, long) method

Syntax

```
void updatePaymentInstrument ( PaymentInstrument paymentInstrument ,  
ExternalAccount externalAccount , long callerId )
```

Parameters

- **paymentInstrument** –
- **externalAccount** –
- **callerId** –

Exceptions

- **MobiliserBusinessLogicException** –

Usage

Update an existing External Account

updateWalletEntry(WalletEntry, Boolean, Boolean, long) method

Syntax

```
void updateWalletEntry ( WalletEntry walletEntry , Boolean  
primaryDebit , Boolean primaryCredit , long callerId ) throws  
IllegalDataException
```

Parameters

- **walletEntry** –
- **primaryDebit** –
- **primaryCredit** –

Exceptions

- **MobiliserBusinessLogicException** –

Usage

Update an existing customer wallet entry

updateWrkWalletEntry(WrkWalletEntry, long) method

Updates an existing add wallet entry process configuration.

Syntax

```
void updateWrkWalletEntry ( WrkWalletEntry wrkWalletEntry , long callerId ) throws EntityNotFoundException
```

Parameters

- **request** –

ISepaMandateLogic interface

The ISepaMandateLogic interface defines methods used to handle (SEPA) mandates which regulate access to payment instruments (mainly bank accounts).

Syntax

```
public interface ISepaMandateLogic
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

createSepaMandate(Mandate, long) method

Creates a new SEPA mandate.

Syntax

```
long createSepaMandate ( Mandate mandate , long callerId ) throws EntityNotUniqueException
```

Parameters

- **mandate** –

deleteSepaMandate(long, long) method

Deletes an existing SEPA mandate.

Syntax

```
void deleteSepaMandate ( long mandateId , long callerId ) throws EntityNotFoundException
```

Parameters

- **mandateId** –
- **callerId** –

getSepaMandate(final long) method

Retrieve a mandate by its ID.

Syntax

Mandate getSepaMandate (final long *mandateId*)

getSepaMandates(final Long, final Long, final boolean) method

Retrieve all mandates for a given payment instrument or customer.

Syntax

List< Mandate > getSepaMandates (final Long *customerId* , final Long *pid* , final boolean *includeInactive*)

updateSepaMandate(Mandate, long) method

Updates an existing SEPA mandate.

Syntax

void updateSepaMandate (Mandate *mandate* , long *callerId*) throws EntityNotFoundException

Parameters

- **mandate** –
- **callerId** –

coupon package

Members

All public members of the coupon package.

- **businesslogic package** –
- **dao package** –

businesslogic package

Members

All public members of the businesslogic package.

- **CouponStatus class** –
- **ICategoryDescriptionLogic interface** –
- **ICouponBatchFileParser interface** –
- **ICouponCategoryLogic interface** –
- **ICouponManager interface** –
- **ICouponServices interface** –
- **ICouponTypeDescriptionLogic interface** –
- **ISpmCouponLogic interface** –
- **StatusCodes class** –

CouponStatus class

Syntax

```
public class CouponStatus
```

Usage

Coupon Instance and Coupon Status Codes

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

ACTIVATION_PENDING variable

The coupon / coupon batch has been created, but is not activated.

Syntax

```
final int ACTIVATION_PENDING
```

ACTIVE variable

The coupon / batch can be assigned.

Syntax

```
final int ACTIVE
```

ASSIGNED variable

The coupon has been assigned to a customer.

Syntax

```
final int ASSIGNED
```


DELETED variable

the coupon was manually deleted

Syntax

```
final int DELETED
```

EXPIRED variable

the coupon has expired and is not available anymore

Syntax

```
final int EXPIRED
```

IN_CREATION_PHASE variable

During batch creation before all coupons has been created.

Syntax

```
final int IN_CREATION_PHASE
```

INVALID variable

The coupon/ batch is invalid.

Syntax

```
final int INVALID
```

OWNED variable

The coupon is owned by the customer and can be used.

Syntax

```
final int OWNED
```

REDEEMED variable

The coupon has been redeemed.

Syntax

```
final int REDEEMED
```

ICategoryDescriptionLogic interface***Syntax***

```
public interface ICategoryDescriptionLogic
```

Usage

OSGi Service Interface for the CategoryDescriptionLogic. Service implementing this interface provides a method to lookup a localized category description.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCaption(long, String) method

Returns the caption in the most appropriate locale.

Syntax

```
String getCaption ( long categoryId , String locale )
```

Parameters

- **categoryId** – category to look up
- **locale** – the language, optional country and optional variant

ICouponBatchFileParser interface

Syntax

```
public interface ICouponBatchFileParser
```

Usage

OSGi interface for coupon service that parses an binary array input and creates coupon batches, segments and coupons

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

parse(long, byte[], long) method

This is the main method used to parse the input.

Syntax

```
long parse ( long couponType , byte[] batch , long actor )
```

Parameters

- **couponType** – the coupon type ID

- **batch** – The batch content defining the batch and coupons
- **actor** – The actor/creator

Returns

created batchID

Usage

It parses the binary array and creates the batch, batch segments based on the preference configuration and coupons. The implementation should use its own session management due to the number of coupons created can be very high.

created batchID

ICouponCategoryLogic interface***Syntax***

```
public interface ICouponCategoryLogic
```

Usage

OSGi Interfaces, implemented by the Coupon Category Logic.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCouponCategoryTree(Integer) method

Retrieve all categories for a category group and construct a tree out of it.

Syntax

```
List< ICouponCategoryTreeNode > getCouponCategoryTree ( Integer  
categoryGroup )
```

Parameters

- **categoryGroup** – (optional) filter on a category group

Returns

list of root notes that references the children

Usage

Returns active and in-active coupon categories.

list of root notes that references the children

getCouponCategoryTree(Integer, boolean) method

Retrieve all categories for a category group and construct a tree out of it.

Syntax

```
List< ICouponCategoryTreeNode > getCouponCategoryTree ( Integer  
categoryGroup , boolean activeOnly )
```

Parameters

- **categoryGroup** – (optional) filter on a category group
- **activeOnly** – specifies if only active categories should be returned

Returns

list of root notes that references the children

Usage

list of root notes that references the children

getCouponCategoryTypeTree(Integer) method

Like getCouponCategoryTree this also returns the set of coupon types that belongs to a category group.

Syntax

```
List< ICouponCategoryTreeNode > getCouponCategoryTypeTree  
( Integer categoryGroup )
```

Parameters

- **categoryGroup** – (optional) filter on a category group

Returns

list of root notes that references the children

Usage

Returns active and in-active coupon categories.

list of root notes that references the children

getCouponCategoryTypeTree(Integer, boolean) method

Like getCouponCategoryTree this also returns the set of coupon types that belongs to a category group.

Syntax

```
List< ICouponCategoryTreeNode > getCouponCategoryTypeTree
( Integer categoryGroup ,  boolean activeOnly )
```

Parameters

- **categoryGroup** – (optional) filter on a category group
- **activeOnly** – specifies if only active categories should be returned

Returns

list of root notes that references the children

Usage

list of root notes that references the children

ICouponManager interface**Syntax**

```
public interface ICouponManager
```

Usage

OSGi service interface of the Coupon Manager. The coupon manager is responsible to assign/create new coupon instances, while distributing the load to a number of configurable coupon batch sequences and coupons. The coupon manager locks a number of configurable segments and distributes the available coupon instances.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

createCouponInstance(long, Customer, Long) method

Create a coupon instance based on a coupon type and return it.

Syntax

```
CouponInstance createCouponInstance ( long couponTypeId ,  Customer
customer ,  Long actorId )
```

Parameters

- **couponTypeId** –
- **customer** –
- **actorId** –

createCouponInstanceId(long, long, Long) method

Create coupon instance.

Syntax

```
long createCouponInstanceId ( long couponTypeId ,   long customerId ,  
Long actorId )
```

Parameters

- **couponTypeId** – the type
- **customerId** – the customerId this instance is created for
- **actorId** – the one who is creating it

Returns

the id of the instance

Usage

the id of the instance

ICouponServices interface

Syntax

```
public interface ICouponServices
```

Usage

OSGi service interface for coupon persistence beans

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

activateBatch(long, long) method

Activates all coupons, batch and segments of a given batch.

Syntax

```
void activateBatch ( long batchId ,   long callerId )
```

Parameters

- **batchId** –
- **callerId** –

assignCoupons(int, List< String >, long, long) method

Assigns coupons to a given list of customers.

Syntax

```
IAssignCouponsResponse assignCoupons ( int identificationType , List< String > identifications , long couponTypeId , long callerId )
```

Parameters

- **identificationType** –
- **identifications** –
- **couponTypeId** –
- **callerId** –

createCoupon(Coupon, long) method

Create a Coupon.

Syntax

```
Long createCoupon ( Coupon coupon , long callerId )
```

Parameters

- **coupon** –
- **callerId** –

createCouponBatch(CouponBatch, long) method

Create a CouponBatch.

Syntax

```
Long createCouponBatch ( CouponBatch batch , long callerId )
```

Parameters

- **batch** –
- **callerId** –

createCouponBatchFromCoupon(long, String, byte[], String, Date, Date, Date, long, long) method

Create a new coupon batch and segments for a coupon type.

Syntax

```
long createCouponBatchFromCoupon ( long couponType , String code ,  
byte[] binaryCode , String serialNumber , Date issueFrom , Date issueTo ,  
Date validUntil , long quantity , long callerId )
```

Parameters

- **couponType** –
- **code** – either this field or binary code must be present/not null
- **binaryCode** – either this field or code must be present/not null
- **serialNumber** – serial number for all new coupons
- **issueFrom** – optional issue from
- **issueTo** – optional issue to
- **validUntil** – optional valid until
- **quantity** – number of coupons to be created
- **callerId** –

createCouponBatchSegment(CouponBatchSegment, long) method

Create a CouponBatchSegment.

Syntax

```
Long createCouponBatchSegment ( CouponBatchSegment segment , long  
callerId )
```

Parameters

- **segment** –
- **callerId** –

createCouponCategory(CouponCategory, long) method

Create a CouponCategory.

Syntax

```
Long createCouponCategory ( CouponCategory coupon , long callerId )
```


Parameters

- **coupon** –
- **callerId** –

createCouponCategoryDescription(CouponCategoryDescription, long) method

Create a CouponCategoryDescription.

Syntax

```
Long createCouponCategoryDescription
( CouponCategoryDescription description ,   long callerId )
```

Parameters

- **description** –
- **callerId** –

createCouponCodeType(CouponCodeType, long) method

Create a CouponCodeType.

Syntax

```
Integer createCouponCodeType ( CouponCodeType type ,   long callerId )
```

Parameters

- **type** –
- **callerId** –

createCouponInstance(CouponInstance, long) method

Create a coupon instance.

Syntax

```
Long createCouponInstance ( CouponInstance instance ,   long callerId )
```

Parameters

- **instance** –
- **callerId** –

Usage

This method should be used only internal. AssignCoupon should be used instead

createCouponType(CouponType, long) method

Create a CouponType.

Syntax

Long createCouponType (CouponType *couponType* , long *callerId*)

Parameters

- **couponType** –
- **callerId** –

createCouponTypeCategory(CouponCategoryType, long) method

Create a CouponCategoryType.

Syntax

void createCouponTypeCategory (CouponCategoryType *category* , long *callerId*)

Parameters

- **category** –
- **callerId** –

createCouponTypeDescription(CouponTypeDescription, long) method

Create a CouponTypeDescription.

Syntax

Long createCouponTypeDescription (CouponTypeDescription *description* , long *callerId*)

Parameters

- **description** –
- **callerId** –

createCouponTypeLocation(final CouponTypeLocation, long) method

Create a new CouponTypeLocation.

Syntax

Long createCouponTypeLocation (final CouponTypeLocation *fromContract* , long *callerId*)

Parameters

- **fromContract** –
- **callerId** –

createCouponTypeTag(CouponTypeTag, long) method

Create a CouponTypeTag.

Syntax

Long createCouponTypeTag (CouponTypeTag *tag* , long *callerId*)

Parameters

- **tag** –
- **callerId** –

deleteCoupon(long, long) method

Delete a Coupon from the database/persistence provide.

Syntax

void deleteCoupon (long *couponId* , long *callerId*)

Parameters

- **couponId** –
- **callerId** –

deleteCouponBatch(long, long) method

Delete a CouponBatch from the database / persistence layer.

Syntax

void deleteCouponBatch (long *batchId* , long *callerId*)

Parameters

- **batchId** –
- **callerId** –

deleteCouponBatchSegment(long, long) method

Delete a CouponBatchSegment from the database.

Syntax

void deleteCouponBatchSegment (long *segmentId* , long *callerId*)

Parameters

- **segmentId** –
- **callerId** –

deleteCouponCategory(long, long) method

Delete a CouponCategory from database.

Syntax

```
void deleteCouponCategory ( long couponId ,   long callerId )
```

Parameters

- **couponId** –
- **callerId** –

deleteCouponCategoryDescription(long, long) method

Delete a CouponCategoryDescription from database.

Syntax

```
void deleteCouponCategoryDescription ( long descriptionId ,   long callerId )
```

Parameters

- **descriptionId** –
- **callerId** –

deleteCouponCodeType(int, long) method

Delete a CouponCodeType from the database.

Syntax

```
void deleteCouponCodeType ( int typeId ,   long callerId )
```

Parameters

- **typeId** –
- **callerId** –

deleteCouponInstance(long, long) method

Delete a CouponInstance.

Syntax

```
void deleteCouponInstance ( long instance ,   long callerId )
```

Parameters

- **instance** –
- **callerId** –

deleteCouponType(long, long) method

Deletes a CouponType from the persistence layer/database.

Syntax

```
void deleteCouponType ( long couponTypeId , long callerId )
```

Parameters

- **couponTypeId** –
- **callerId** –

deleteCouponTypeCategory(long, long, long) method

Delete a CouponCategoryType from database.

Syntax

```
void deleteCouponTypeCategory ( long typeId , long categoryId , long callerId )
```

Parameters

- **typeId** –
- **categoryId** –
- **callerId** –

deleteCouponTypeDescription(long, long) method

Delete a CouponTypeDescription from the database.

Syntax

```
void deleteCouponTypeDescription ( long descriptionId , long callerId )
```

Parameters

- **descriptionId** –
- **callerId** –

deleteCouponTypeLocation(long, long) method

Deletes a CouponTypeLocation (removes it from the persistence provider)

Syntax

```
void deleteCouponTypeLocation ( long locationId ,   long callerId )
```

Parameters

- **locationId** –
- **callerId** –

deleteCouponTypeTag(long, long) method

Delete a CouponTypeTag from database.

Syntax

```
void deleteCouponTypeTag ( long tagId ,   long callerId )
```

Parameters

- **tagId** –
- **callerId** –

findAllCategoriesForCouponType(long) method

Find all coupon categories for a coupon type.

Syntax

```
List< CouponCategory > findAllCategoriesForCouponType ( long  
couponType )
```

Parameters

- **couponType** –

getAllCouponBatchesByCouponType(long) method

Get all batches for a coupon type.

Syntax

```
List< CouponBatch > getAllCouponBatchesByCouponType ( long  
couponType )
```

Parameters

- **couponType** –

getAllCouponCategories(Integer) method

Get all Coupon Categories of a group.

Syntax

```
List< CouponCategory > getAllCouponCategories ( Integer groupId )
```

Parameters

- **groupId** –

getAllCouponCategoryDescriptionsByCategory(long) method

Get all descriptions of a category.

Syntax

```
List< CouponCategoryDescription >  
getAllCouponCategoryDescriptionsByCategory ( long category )
```

Parameters

- **category** –

getAllCouponMimeType() method

Get all defined mime types of coupons.

Syntax

```
List< CouponMimeType > getAllCouponMimeType ()
```

getAllCouponTagsByCouponType(long) method

Get all tags for a given coupon type.

Syntax

```
List< CouponTypeTag > getAllCouponTagsByCouponType ( long  
couponType )
```

Parameters

- **couponType** –

getAllCouponTypeDescriptionsByCouponType(long) method

Get all coupon type descriptions for a given coupon type.

Syntax

```
List< CouponTypeDescription >  
getAllCouponTypeDescriptionsByCouponType ( long couponType )
```

Parameters

- **couponType** –

getAllCouponTypeLocations(long) method

Get all locations of a coupon type.

Syntax

```
List< CouponTypeLocation > getAllCouponTypeLocations ( long  
type )
```

Parameters

- **type** –

getAllCouponTypes(boolean, Long) method

Return a filtered list of coupon types.

Syntax

```
List< CouponType > getAllCouponTypes ( boolean activeOnly , Long  
issuerId )
```

Parameters

- **activeOnly** –
- **issuerId** –

getAllIssuer() method

Find all coupon issuer from the issuer field of the coupon type.

Syntax

```
List< Customer > getAllIssuer ()
```


getCildCategories(long) method

Get all child categories (with parent of this category)

Syntax

```
List< CouponCategory > getCildCategories ( long category )
```

Parameters

- **category** –

getCoupon(long) method

Get a Coupon by its ID.

Syntax

```
Coupon getCoupon ( long couponId )
```

Parameters

- **couponId** –

getCouponBatch(long) method

Get a CouponBatch by ID.

Syntax

```
CouponBatch getCouponBatch ( long batchId )
```

Parameters

- **batchId** –

getCouponBatchSegment(long) method

Get a CouponBatchSegment by ID.

Syntax

```
CouponBatchSegment getCouponBatchSegment ( long segmentId )
```

Parameters

- **segmentId** –

getCouponCategory(long) method

Get a CouponCategory by ID.

Syntax

CouponCategory getCouponCategory (long *couponId*)

Parameters

- **couponId** –

getCouponCategoryDescription(long) method

Get a CouponCategoryDescription by ID.

Syntax

CouponCategoryDescription getCouponCategoryDescription (long *descriptionId*)

Parameters

- **descriptionId** –

getCouponCodeType(int) method

Get a CouponCodeType by ID.

Syntax

CouponCodeType getCouponCodeType (int *typeId*)

Parameters

- **typeId** –

getCouponInstance(long) method

Get a CouponInstance by ID.

Syntax

CouponInstance getCouponInstance (long *instanceId*)

Parameters

- **instanceId** –

getCouponType(long) method

Get a CouponType by its ID.

Syntax

CouponType getCouponType (long *couponTypeId*)

Parameters

- **couponTypeId** –

getCouponTypeCategory(long, long) method

Get a CouponCategoryType by ID.

Syntax

CouponCategoryType getCouponTypeCategory (long *typeId* , long *categoryId*)

Parameters

- **typeId** –
- **categoryId** –

getCouponTypeDescription(long) method

Get a CouponTypeDescription by ID.

Syntax

CouponTypeDescription getCouponTypeDescription (long *descriptionId*)

Parameters

- **descriptionId** –

getCouponTypeLocation(long) method

Get CouponTypeLocation with its ID.

Syntax

CouponTypeLocation getCouponTypeLocation (long *locationId*)

Parameters

- **locationId** –

getCouponTypes(String, Long, Boolean, Boolean) method

Find a list of coupons based on a filter.

Syntax

```
List< CouponType > getCouponTypes ( String name , Long issuerId ,  
Boolean isActive , Boolean isPublic )
```

Parameters

- **name** –
- **issuerId** –
- **isActive** –
- **isPublic** –

getCouponTypeTag(long) method

Get a CouponTypeTag by ID.

Syntax

```
CouponTypeTag getCouponTypeTag ( long tagId )
```

Parameters

- **tagId** –

getMimeType(String) method

Get a coupon mime type by its ID.

Syntax

```
CouponMimeType getMimeType ( String type )
```

Parameters

- **type** –

getRootCategories(int) method

Get the root categories (no parent) for a category group.

Syntax

```
List< CouponCategory > getRootCategories ( int groupId )
```

Parameters

- **groupId** –

replaceAllCouponTags(long, List< String >, long) method

replace all coupon type tags with the new tags

Syntax

```
void replaceAllCouponTags ( long couponType , List< String > tag ,  
long callerId )
```

Parameters

- **couponType** –
- **tag** –
- **callerId** –

searchCoupons(IIdentification, String) method

Search coupon instances based on filter by customer or serial number.

Syntax

```
List< CouponInstance > searchCoupons ( IIdentification customer ,  
String serialNumber )
```

Parameters

- **customer** – an identifier and type that identifies a customer
- **serialNumber** –

updateCoupon(Long, Coupon, long) method

Update a Coupon.

Syntax

```
void updateCoupon ( Long id , Coupon coupon , long callerId )
```

Parameters

- **id** –
- **coupon** –
- **callerId** –

updateCouponBatch(Long, CouponBatch, long) method

Update a CouponBatch.

Syntax

```
void updateCouponBatch ( Long id ,  CouponBatch batch ,  long callerId )
```

Parameters

- **id** –
- **batch** –
- **callerId** –

updateCouponBatchSegment(Long, CouponBatchSegment, long) method

Update a CouponBatchSegment.

Syntax

```
void updateCouponBatchSegment ( Long id ,  CouponBatchSegment  
segment ,  long callerId )
```

Parameters

- **id** –
- **segment** –
- **callerId** –

updateCouponCategory(Long, CouponCategory, long) method

Update a CouponCategory.

Syntax

```
void updateCouponCategory ( Long id ,  CouponCategory coupon ,  long  
callerId )
```

Parameters

- **id** –
- **coupon** –
- **callerId** –

updateCouponCategoryDescription(Long, CouponCategoryDescription, long) method

Update a CouponCategoryDescription.

Syntax

```
void updateCouponCategoryDescription ( Long id ,  
CouponCategoryDescription description , long callerId )
```

Parameters

- **id** –
- **description** –
- **callerId** –

updateCouponCodeType(Integer, CouponCodeType, long) method

Update a CouponCodeType.

Syntax

```
void updateCouponCodeType ( Integer id , CouponCodeType type , long  
callerId )
```

Parameters

- **id** –
- **type** –
- **callerId** –

updateCouponInstance(Long, CouponInstance, long) method

Update a CouponInstance.

Syntax

```
void updateCouponInstance ( Long id , CouponInstance instance , long  
callerId )
```

Parameters

- **id** –
- **instance** –
- **callerId** –

updateCouponType(Long, CouponType, long) method

Updates a CouponType.

Syntax

```
void updateCouponType ( Long id , CouponType couponType , long callerId )
```

Parameters

- **id** –
- **couponType** –
- **callerId** –

updateCouponTypeDescription(Long, CouponTypeDescription, long) method

Update a CouponTypeDescription.

Syntax

```
void updateCouponTypeDescription ( Long id ,  
CouponTypeDescription description , long callerId )
```

Parameters

- **id** –
- **description** –
- **callerId** –

updateCouponTypeLocation(Long, CouponTypeLocation, long) method

Update a CouponTypeLocation.

Syntax

```
void updateCouponTypeLocation ( Long id , CouponTypeLocation  
fromContract , long callerId )
```

Parameters

- **id** –
- **fromContract** –
- **callerId** –

updateCouponTypeTag(Long, CouponTypeTag, long) method

Update a CouponTypeTag.

Syntax

```
void updateCouponTypeTag ( Long id ,  CouponTypeTag tag ,  long
callerId )
```

Parameters

- **id** –
- **tag** –
- **callerId** –

*uploadCouponBatch(long, byte[], long) method***Syntax**

```
long uploadCouponBatch ( long couponType ,  byte[] binaryContent ,  long
callerId )
```

ICouponTypeDescriptionLogic interface**Syntax**

```
public interface ICouponTypeDescriptionLogic
```

Usage

OSGi service interface that is implemented to find a localized coupon type description

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCouponTypeDescription(long, String, String) method

Get the (best) localized coupon type description for a given locale and mime type.

Syntax

```
CouponTypeDescription getCouponTypeDescription ( long
couponTypeId ,  String locale ,  String contentType )
```

Parameters

- **couponTypeId** –

- **locale** –
- **contentType** –

ISpmCouponLogic interface

Syntax

```
public interface ISpmCouponLogic
```

Usage

OSGi service interface that is implemented for the Smartphone Coupon Logic

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

deleteCouponInstance(long, long, long) method

Remove a coupon instance for a given customer.

Syntax

```
void deleteCouponInstance ( long customerId , long couponInstanceId ,  
long callerId )
```

Parameters

- **customerId** –
- **couponInstanceId** –
- **callerId** –

Usage

Checks also if the instance is owned by the customer with the given ID

findNextCouponTypes(BigDecimal, BigDecimal, Integer, Long, List< String >, Integer) method

Find the next coupon location (sorted by the distance) based on a given filter.

Syntax

```
List< CouponTypeLocation > findNextCouponTypes ( BigDecimal  
latitude , BigDecimal longitude , Integer radius , Long categoryId , List<  
String > tag , Integer maxRecords )
```

Parameters

- **latitude** – mandatory latitude in GPS format
- **longitude** – mandatory longitude in GPS format
- **radius** – optional filter on radius in meter around the GPS coordinate
- **categoryId** – optional filter on a category
- **tag** – optional filter on a tag
- **maxRecords** – optional maximum number of records. Default and maximum is 100

Returns

the location type. The returned radius is the distance to the coordinate in meter

Usage

the location type. The returned radius is the distance to the coordinate in meter

getCouponByCustomer(long, long) method

Get all coupons of a customer.

Syntax

```
List< CouponInstance > getCouponByCustomer ( long customerId , long callerId )
```

Parameters

- **customerId** –
- **callerId** –

getCouponTypeByCategory(long, long) method

Get all coupon types for a category.

Syntax

```
List< CouponType > getCouponTypeByCategory ( long categoryId , long callerId )
```

Parameters

- **categoryId** –
- **callerId** –

getCouponTypeByTags(List< String >, long) method

Get all coupon types for a given tag list.

Syntax

```
List< CouponType > getCouponTypeByTags ( List< String > tags ,  
long callerId )
```

Parameters

- **tags** –
- **callerId** –

purchaseCouponInstance(long, long, Long, long) method

Purchase a coupon instance by debiting the payment instrument.

Syntax

```
AuthorisationResponse purchaseCouponInstance ( long customerId ,  
long couponInstanceId , Long paymentInstrumentId , long callerId )
```

Parameters

- **customerId** –
- **couponInstanceId** –
- **paymentInstrumentId** –
- **callerId** –

StatusCodes class

Syntax

```
public class StatusCodes
```

Usage

Status codes provided by the coupon services/logic

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

ERROR_COUPON_BATCH_FORMAT variable

Syntax

```
final int ERROR_COUPON_BATCH_FORMAT
```

*ERROR_COUPON_BATCH_IO variable**Syntax*

```
final int ERROR_COUPON_BATCH_IO
```

*ERROR_COUPON_NO_REMAINING variable**Syntax*

```
final int ERROR_COUPON_NO_REMAINING
```

*ERROR_COUPON_STATUS_WRONG variable**Syntax*

```
final int ERROR_COUPON_STATUS_WRONG
```

dao package*Members*

All public members of the dao package.

- **api package** –
- **model package** –

api package*Members*

All public members of the api package.

- **CouponBatchDao interface** – The Coupon Batch contains one or multiple Coupons.
- **CouponBatchSegmentDao interface** – To allow parallel access to Coupons by multiple servers / threads each Coupon Batch is broken up into segments.
- **CouponCategoryDao interface** – Contains the list of coupon categories.
- **CouponCategoryDescriptionDao interface** –
- **CouponCategoryGroupDao interface** – DAO access for CouponCategoryGroup.
- **CouponCategoryTypeDao interface** – DAO access for CouponCategoryType.
- **CouponCodeTypeDao interface** – DAO access for CouponCodeType.
- **CouponDao interface** – DAO access for Coupon.
- **CouponInstanceDao interface** – Whenever a coupon is assigned to a user a record in this table gets created.
- **CouponLocaleDao interface** – Contains the list of supported locales along with a user friendly description.

- **CouponMimeTypeDao interface** – Contains the list of supported mime type along with a user friendly description and details, such as if it is a binary mime type or alternatives.
- **CouponTypeDao interface** – DAO access for CouponType.
- **CouponTypeDescriptionDao interface** – Holds detailed descriptions for Coupon Types in multiple languages and formats.
- **CouponTypeLocationDao interface** – Stores location information for distribution, search and redemption of the coupon (maximum) radius.
- **CouponTypeTagDao interface** – Holds a list of searchable tags that are assigned to Coupon Types.

CouponBatchDao interface

The Coupon Batch contains one or multiple Coupons.

Syntax

```
public interface CouponBatchDao
```

Usage

The complete Coupon Batch belongs to a Coupon Type. Internally the Coupon Batch is broken up into Coupon Batch Segments to allow faster Coupon handling by multiple threads. The Coupon Batch can be provided as a CSV file and is uploaded in the UI or by a WebService call. Multiple Batches for one Coupon Type can be active at the same time. The oldest, active batch is always selected first to hand out Coupons.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getAllByType(long) method

Returns all of a given coupon type.

Syntax

```
List< CouponBatch > getAllByType ( long couponType )
```

Parameters

- **couponType** –

getQuantity(Long) method

Returns the quantity (sum) of all coupons in linked batch segments.

Syntax

```
Long getQuantity ( Long id )
```

Parameters

- **id** –

getUsed(Long) method

Returns the used sum of all linked batch segments.

Syntax

Long getUsed (Long *id*)

Parameters

- **id** –

CouponBatchSegmentDao interface

To allow parallel access to Coupons by multiple servers / threads each Coupon Batch is broken up into segments.

Syntax

```
public interface CouponBatchSegmentDao
```

Usage

The number of segments is a system wide configuration. Segments are purely for internal management and do not fulfill any business purpose.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

acquireBatchSegment(long, int, Long) method

Acquire the next unlocked batch segment for a given coupon type and locks it.

Syntax

```
CouponBatchSegment acquireBatchSegment ( long couponTypeId , int  
lockSeconds , Long updater )
```

Parameters

- **couponTypeId** – the requested coupon type
- **lockSeconds** – initial lock duration.
- **updater** –

Usage

The manager is responsible to renew the lock before the lock expires or release an unused segment

releaseBatchSegment(CouponBatchSegment) method

Releases a lock on a batch segment and makes it available for other workers.

Syntax

```
void releaseBatchSegment ( CouponBatchSegment couponBatchSegment )
```

Parameters

- **couponBatchSegment** –

CouponCategoryDao interface

Contains the list of coupon categories.

Syntax

```
public interface CouponCategoryDao
```

Usage

The exact name that is shown to end-users is maintained (in multiple languages) in CouponCategoryDescription

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getAllByGroup(Integer) method

Filter all categories by a given group ID.

Syntax

```
List< CouponCategory > getAllByGroup ( Integer groupId )
```

Parameters

- **groupId** – filter

Returns

all categories that belongs to the group

Usage

all categories that belongs to the group

getAllChildren(long) method

Retrieve all child categories of a given category.

Syntax

```
List< CouponCategory > getAllChildren ( long category )
```

Parameters

- **category** – ID of the category

Returns

list of child categories

Usage

list of child categories

getAllRoot(int) method

Get all categories that are not linked to a parent category.

Syntax

```
List< CouponCategory > getAllRoot ( int groupId )
```

Parameters

- **groupId** – the filter

Returns

list of root categories

Usage

Filter by group

list of root categories

CouponCategoryDescriptionDao interface

Syntax

```
public interface CouponCategoryDescriptionDao
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

getAllByCategory(long) method

Get all descriptions for a category.

Syntax

```
List< CouponCategoryDescription > getAllByCategory ( long  
category )
```

Parameters

- **category** –

getByCategoryandLocale(long, String) method

Get the 'best' description for a locale.

Syntax

```
CouponCategoryDescription getByCategoryandLocale ( long category ,  
String locale )
```

Parameters

- **category** –
- **locale** –

CouponCategoryGroupDao interface

DAO access for CouponCategoryGroup.

Syntax

```
public interface CouponCategoryGroupDao
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CouponCategoryTypeDao interface

DAO access for CouponCategoryType.

Syntax

```
public interface CouponCategoryTypeDao
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

findCategoriesForCouponType(long) method

Returns all CouponCategory for a given type.

Syntax

```
List< CouponCategory > findCategoriesForCouponType ( long typeId )
```

Parameters

- **typeId** –

getAllByGroup(Integer) method

Returns active and inactive CouponCategoryTypes for the provided group.

Syntax

```
List< CouponCategoryType > getAllByGroup ( Integer group )
```

Parameters

- **group** – is optional but it is highly recommended to provide it, otherwise all are returned.

getAllByGroup(Integer, boolean) method

Returns CouponCategoryTypes for the provided group.

Syntax

```
List< CouponCategoryType > getAllByGroup ( Integer group , boolean activeOnly )
```

Parameters

- **group** – is optional but it is highly recommended to provide it, otherwise all are returned.
- **activeOnly** – can be set to filter for only active entries (also looking at valid from/to date range)

CouponCodeTypeDao interface

DAO access for CouponCodeType.

Syntax

```
public interface CouponCodeTypeDao
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CouponDao interface

DAO access for Coupon.

Syntax

```
public interface CouponDao
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

acquireNext(Long, Long) method

Acquire next free coupon for a segment.

Syntax

Coupon acquireNext (Long *id* , Long *callerId*)

Parameters

- **id** – segment
- **callerId** –

CouponInstanceDao interface

Whenever a coupon is assigned to a user a record in this table gets created.

Syntax

```
public interface CouponInstanceDao
```

Usage

Some Coupons can be assigned to multiple customers, some can only be assigned to a single customer. Number of views and uses is tracked in this table.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getActiveByCustomer(Long) method

get all active assigned coupons by customer

Syntax

```
List< CouponInstance > getActiveByCustomer ( Long id )
```

Parameters

- **id** –

getByCustomer(Long) method

Get all assigned coupons of a customer.

Syntax

```
List< CouponInstance > getByCustomer ( Long id )
```

Parameters

- **id** –

getBySerialNumber(String) method

Get all assigned coupons by the serial number.

Syntax

```
List< CouponInstance > getBySerialNumber ( String serialNumber )
```

Parameters

- **serialNumber** –

CouponLocaleDao interface

Contains the list of supported locales along with a user friendly description.

Syntax

```
public interface CouponLocaleDao
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

CouponMimeTypeDao interface

Contains the list of supported mime type along with a user friendly description and details, such as if it is a binary mime type or alternatives.

Syntax

```
public interface CouponMimeTypeDao
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CouponTypeDao interface

DAO access for CouponType.

Syntax

```
public interface CouponTypeDao
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

findNextCouponTypes(BigDecimal, BigDecimal, Long, List< String >, long, Integer)
method

Find the next coupons based on filter.

Syntax

```
List< CouponTypeLocation > findNextCouponTypes ( BigDecimal  
latitude , BigDecimal longitude , Long categoryId , List< String > tags ,  
long maxDistance , Integer maxRecords )
```

Parameters

- **latitude** –
 - current position
- **longitude** –

- current position
- **categoryId** –
 - optional filter on categoryId
- **tags** –
 - optional filter on tags
- **maxDistance** –
 - mandatory max distance in meter

getActiveByTags(List< String >) method

Get all active coupon types by tag value.

Syntax

```
List< CouponType > getActiveByTags ( List< String > tags )
```

Parameters

- **tags** –

getAll(boolean, Long) method

Find all coupon type by issuer.

Syntax

```
List< CouponType > getAll ( boolean activeOnly , Long issuerId )
```

Parameters

- **activeOnly** – if true, only active coupon types are returned
- **issuerId** – optional, the issuer of the coupon type

getAll(String, Boolean, Boolean, Long) method

Return list of coupons based on the filter criteria.

Syntax

```
List< CouponType > getAll ( String name , Boolean activeOnly ,  
Boolean isPublic , Long issuerId )
```

Parameters

- **name** – optional, wildcards allowed : Filter based on name. If null this filter is not applied
- **isPublic** – optional : Filter on isActive. If not given, this filter is not applied
- **issuerId** – optional : Filter on issuer. If not given, this filter is not applied

getAllIssuer() method

Get all customers that are listed as coupon issuer.

Syntax

```
List< Customer > getAllIssuer ()
```

Usage

This is not the list of customers that can be issuers. Usually this is indirectly specified as privilege, role or customer type

getByCategory(long, boolean) method

Get all active coupon types by category.

Syntax

```
List< CouponType > getByCategory ( long categoryId , boolean  
publicOnly )
```

Parameters

- **categoryId** –
- **publicOnly** –

CouponTypeDescriptionDao interface

Holds detailed descriptions for Coupon Types in multiple languages and formats.

Syntax

```
public interface CouponTypeDescriptionDao
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getAllByType(long) method

Get all descriptions for a coupon type.

Syntax

```
List< CouponTypeDescription > getAllByType ( long couponType )
```


Parameters

- **couponType** –

getByCouponTypeLocaleAndMimeType(long, String, String) method
get the description based on coupon type, locale and mime type

Syntax

```
CouponTypeDescription getByCouponTypeLocaleAndMimeType ( long
couponType , String locale , String mimeType )
```

Parameters

- **couponType** –
- **locale** –
- **mimeType** –

CouponTypeLocationDao interface

Stores location information for distribution, search and redemption of the coupon (maximum) radius.

Syntax

```
public interface CouponTypeLocationDao
```

Usage

For finding the next coupon type, see CouponTypeDao

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getAllByType(long) method
get all locations for a type

Syntax

```
List< CouponTypeLocation > getAllByType ( long type )
```

Parameters

- **type** –

CouponTypeTagDao interface

Holds a list of searchable tags that are assigned to Coupon Types.

Syntax

```
public interface CouponTypeTagDao
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getAllbyCouponType(long) method

List all tags of a coupon type.

Syntax

```
List< CouponTypeTag > getAllbyCouponType ( long couponType )
```

Parameters

- **couponType** –

model package

Members

All public members of the model package.

- **Coupon class** – The Coupon itself.
- **CouponBatch class** – The Coupon Batch contains one or multiple Coupons.
- **CouponBatchSegment class** – To allow parallel access to Coupons by multiple servers / threads each Coupon Batch is broken up into segments.
- **CouponCategory class** – Contains the list of coupon categories.
- **CouponCategoryDescription class** – A human readable localized description of a category.
- **CouponCategoryGroup class** – Allows to classify and filter CouponCategories into a group.
- **CouponCategoryType class** – The CouponCategoryType maps coupon categories to coupon types.
- **CouponCodeType class** – A possible Coupon Code Types.
- **CouponInstance class** – Whenever a coupon is assigned to a user a record in this table gets created.

- **CouponLocale class** – Contains the list of supported locales along with a user friendly description.
- **CouponMimeType class** – Contains the list of supported mime type along with a user friendly description and details, such as if it is a binary mime type or alternatives.
- **CouponType class** – Each Coupon is assigned to a Coupon Type.
- **CouponTypeDescription class** – A detailed descriptions for Coupon Types in multiple languages and formats.
- **CouponTypeLocation class** – A location information for distribution,search and redemption of the coupon type (maximum) radius.
- **CouponTypeTag class** – A search able tag that is assigned to Coupon Types.

Coupon class

The Coupon itself.

Syntax

```
public class Coupon
```

Usage

A Coupon always belongs to a Coupon Type. A Coupon is usually not assigned to a customer when being created. It can be assigned to one or many customers (depending on the Coupon quantity). The assigned coupon is represented by a CouponInstance

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getBatchSegment() method

Specifies to which batch segment this coupon belongs to.

Syntax

```
CouponBatchSegment getBatchSegment ()
```

getCode() method

The code as a byte array (either STR_CODE or BIN_CODE is used, depending on the ID_CODE_TYPE)

Syntax

```
byte[] getCode ()
```

getCodeStr() method

The code as a string (either STR_CODE or BIN_CODE is used, depending on the ID_CODE_TYPE)

Syntax

String getCodeStr ()

getCouponType() method

short cut to going via the Batch Segment

Syntax

CouponType getCouponType ()

getGone() method

The number of times this Coupon was already issued.

Syntax

Long getGone ()

getHashStr() method

This allows easier search for a coupon by its code.

Syntax

String getHashStr ()

getQuantity() method

The number of times this Coupon can be issued to customers.

Syntax

Long getQuantity ()

Usage

0 means that there is no limitation

getRedeemedAt() method

The customer ID who redeemed the Coupon.

Syntax

Customer getRedeemedAt ()

Usage

Only applicable to Coupons with INT_QUANTITY = 1

getRedeemedOn() method

The date the coupon was redeemed.

Syntax

Date getRedeemedOn ()

Usage

Only applicable to Coupons with INT_QUANTITY = 1

getSerialNumber() method

The serial number.

Syntax

String getSerialNumber ()

Usage

All Coupons from one Issuers should have a unique serial number, otherwise Coupons can not be marked as redeemed

getStatus() method

The status of the Coupon.

Syntax

Integer getStatus ()

getValidTo() method

The date until when this coupon is valid.

Syntax

Date getValidTo ()

Usage

;

*isSetBatchSegment() method***Syntax**

boolean isSetBatchSegment ()

isSetCode() method

Syntax

boolean isSetCode ()

isSetCodeStr() method

Syntax

boolean isSetCodeStr ()

isSetCouponType() method

Syntax

boolean isSetCouponType ()

isSetGone() method

Syntax

boolean isSetGone ()

isSetHashStr() method

Syntax

boolean isSetHashStr ()

isSetQuantity() method

Syntax

boolean isSetQuantity ()

isSetRedeemedAt() method

Syntax

boolean isSetRedeemedAt ()

isSetRedeemedOn() method

Syntax

boolean isSetRedeemedOn ()

isSetSerialNumber() method

Syntax

`boolean isSetSerialNumber ()`

isSetStatus() method

Syntax

`boolean isSetStatus ()`

isSetValidTo() method

Syntax

`boolean isSetValidTo ()`

setBatchSegment(CouponBatchSegment) method

Syntax

`void setBatchSegment ( CouponBatchSegment batchSegment )`

setCode(byte[]) method

Syntax

`void setCode ( byte[] code )`

setCodeStr(String) method

Syntax

`void setCodeStr ( String codeStr )`

setCouponType(CouponType) method

Syntax

`void setCouponType ( CouponType couponType )`

setGone(Long) method

Syntax

`void setGone ( Long gone )`

setHashStr(String) method

Syntax

void setHashStr (String *hashStr*)

setQuantity(Long) method

Syntax

void setQuantity (Long *quantity*)

setRedeemedAt(Customer) method

Syntax

void setRedeemedAt (Customer *redeemedAt*)

setRedeemedOn(Date) method

Syntax

void setRedeemedOn (Date *redeemedOn*)

setSerialNumber(String) method

Syntax

void setSerialNumber (String *serialNumber*)

setStatus(Integer) method

Syntax

void setStatus (Integer *status*)

setValidTo(Date) method

Syntax

void setValidTo (Date *validTo*)

CouponBatch class

The Coupon Batch contains one or multiple Coupons.

Syntax

public class CouponBatch

Usage

The complete Coupon Batch belongs to a Coupon Type. Internally the Coupon Batch is broken up into Coupon Batch Segments to allow faster Coupon handling by multiple threads. The Coupon Batch can be provided as a CSV file and is uploaded in the UI or by a Webservice call. Multiple Batches for one Coupon Type can be active at the same time. The oldest, active batch is always selected first to hand out Coupons.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getActivated() method

The date and time when this batch was activated.

Syntax

```
Date getActivated ()
```

getActivator() method

The the customer who has activated the batch'.

Syntax

```
Customer getActivator ()
```

getCouponType() method

The coupon type to which this batch belongs to.

Syntax

```
CouponType getCouponType ()
```

getIssueFrom() method

Indicates from when on coupons of this batch are allowed to be issued.

Syntax

```
Date getIssueFrom ()
```

getIssueTo() method

Indicates until when coupons of this batch are allowed to be issued.

Syntax

```
Date getIssueTo ()
```

getStatus() method

The coupon status CouponStatus.

Syntax

Integer getStatus ()

isActive() method

Indicates if the batch is active or not (switched manually)

Syntax

boolean isActive ()

isSetActivated() method

Syntax

boolean isSetActivated ()

isSetActivator() method

Syntax

boolean isSetActivator ()

isSetActive() method

Syntax

boolean isSetActive ()

isSetCouponType() method

Syntax

boolean isSetCouponType ()

isSetIssueFrom() method

Syntax

boolean isSetIssueFrom ()

isSetIssueTo() method

Syntax

boolean isSetIssueTo ()

isSettatus() method

Syntax

boolean isSettatus ()

setActivated(Date) method

Syntax

void setActivated (Date *activated*)

setActivator(Customer) method

Syntax

void setActivator (Customer *activator*)

setActive(boolean) method

Syntax

void setActive (boolean *active*)

setCouponType(CouponType) method

Syntax

void setCouponType (CouponType *couponType*)

setIssueFrom(Date) method

Syntax

void setIssueFrom (Date *issueFrom*)

setIssueTo(Date) method

Syntax

void setIssueTo (Date *issueTo*)

setStatus(Integer) method

Syntax

void setStatus (Integer *status*)

CouponBatchSegment class

To allow parallel access to Coupons by multiple servers / threads each Coupon Batch is broken up into segments.

Syntax

```
public class CouponBatchSegment
```

Usage

The number of segments is a system wide configuration. Segments are purely for internal management and do not fulfill any business purpose. The number of segments created for a batch are configurable via preference CouponConfiguration

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getBatch() method

The Coupon Batch to which this segment belongs.

Syntax

```
CouponBatch getBatch ()
```

getLockedUntil() method

Indicates for how long the segment is still locked'.

Syntax

```
Date getLockedUntil ()
```

getQuantity() method

The quantity of Coupons in this segment.

Syntax

```
Long getQuantity ()
```

getUsed() method

The number of already used Coupons.

Syntax

```
Long getUsed ()
```

isLocked() method

Indicates if the segment is currently locked by a worker.

Syntax

```
boolean isLocked ()
```

*isSetBatch() method***Syntax**

```
boolean isSetBatch ()
```

*isSetLocked() method***Syntax**

```
boolean isSetLocked ()
```

*isSetLockedUtil() method***Syntax**

```
boolean isSetLockedUtil ()
```

*isSetQuantity() method***Syntax**

```
boolean isSetQuantity ()
```

*isSetUsed() method***Syntax**

```
boolean isSetUsed ()
```

*setBatch(CouponBatch) method***Syntax**

```
void setBatch ( CouponBatch batch )
```

*setLocked(boolean) method***Syntax**

```
void setLocked ( boolean active )
```

setLockedUntil(Date) method

Syntax

void setLockedUntil (Date *lockedUntil*)

setQuantity(Long) method

Syntax

void setQuantity (Long *quantity*)

setUsed(Long) method

Syntax

void setUsed (Long *used*)

CouponCategory class

Contains the list of coupon categories.

Syntax

public class CouponCategory

Usage

The exact name that is shown to end-users is maintained (in multiple languages) in CouponCategoryDescription

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCategoryGroup() method

Categories can be grouped via category groups.

Syntax

CouponCategoryGroup getCategoryGroup ()

getName() method

Contains the list of coupon categories.

Syntax

String getName ()

Usage

The exact name that is shown to end-users is maintained (in multiple languages) in CPN_COUPON_CAT_DESCR

getParent() method

To build up a hierarchy within categories, a parent category can be set.

Syntax

```
CouponCategory getParent ()
```

Usage

Categories without a parent or root categories

getPriority() method

Categories with higher priority will be shown first.

Syntax

```
Integer getPriority ()
```

isActive() method

Indicates if this category (including its children) is active.

Syntax

```
boolean isActive ()
```

*isSetActive() method***Syntax**

```
boolean isSetActive ()
```

*isSetCategoryGroup() method***Syntax**

```
boolean isSetCategoryGroup ()
```

*isSetName() method***Syntax**

```
boolean isSetName ()
```

isSetParent() method

Syntax

boolean isSetParent ()

isSetPriority() method

Syntax

boolean isSetPriority ()

setActive(boolean) method

Syntax

void setActive (boolean *active*)

setCategoryGroup(CouponCategoryGroup) method

Syntax

void setCategoryGroup (CouponCategoryGroup *category*)

setName(String) method

Syntax

void setName (String *name*)

setParent(CouponCategory) method

Syntax

void setParent (CouponCategory *parent*)

setPriority(Integer) method

Syntax

void setPriority (Integer *priority*)

CouponCategoryDescription class

A human readable localized description of a category.

Syntax

public class CouponCategoryDescription

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCaption() method

The description of the category in the given language (locale)

Syntax

```
String getCaption ()
```

getCategory() method

The Category that is described by this entity.

Syntax

```
CouponCategory getCategory ()
```

getLocale() method

The (Java) locale for which this description is.

Syntax

```
String getLocale ()
```

Usage

Null means default.

isSetCaption() method

Syntax

```
boolean isSetCaption ()
```

isSetCategory() method

Syntax

```
boolean isSetCategory ()
```

isSetLocale() method

Syntax

```
boolean isSetLocale ()
```

setCaption(String) method

Syntax

```
void setCaption ( String caption )
```

setCategory(CouponCategory) method

Syntax

```
void setCategory ( CouponCategory category )
```

setLocale(String) method

Syntax

```
void setLocale ( String locale )
```

CouponCategoryGroup class

Allows to classify and filter CouponCategories into a group.

Syntax

```
public class CouponCategoryGroup
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CouponCategoryGroup() constructor

Syntax

```
CouponCategoryGroup ( )
```

CouponCategoryGroup(int) constructor

Syntax

```
CouponCategoryGroup ( int id )
```

CouponCategoryType class

The CouponCategoryType maps coupon categories to coupon types.

Syntax

```
public class CouponCategoryType
```

Usage

This is a n:n relationship

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CouponCategoryTypePk class**Syntax**

```
public class CouponCategoryTypePk
```

CouponCategoryTypePk() constructor**Syntax**

```
CouponCategoryTypePk ()
```

CouponCategoryTypePk(Long, Long) constructor**Syntax**

```
CouponCategoryTypePk ( Long couponType , Long category )
```

equals(Object) method**Syntax**

```
boolean equals ( Object o )
```

getCategory() method

Reference to the category.

Syntax

```
Long getCategory ()
```

getCouponType() method

Reference to coupon type.

Syntax

```
Long getCouponType ()
```

hashCode() method

Syntax

`int hashCode ()`

isSetCategory() method

Syntax

`boolean isSetCategory ()`

isSetCouponType() method

Syntax

`boolean isSetCouponType ()`

setCategory(Long) method

Syntax

`void setCategory ( Long category )`

setCouponType(Long) method

Syntax

`void setCouponType ( Long couponType )`

toString() method

Syntax

`String toString ()`

CouponCategoryType() constructor

Syntax

`CouponCategoryType ()`

CouponCategoryType(CouponCategoryTypePk) constructor

Syntax

`CouponCategoryType ( CouponCategoryTypePk key )`

CouponCodeType class

A possible Coupon Code Types.

Syntax

```
public class CouponCodeType
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CouponCodeType() constructor**Syntax**

```
CouponCodeType ()
```

CouponCodeType(int) constructor**Syntax**

```
CouponCodeType ( int i )
```

getCryptHandler() method

The optional cryptographic handler.

Syntax

```
String getCryptHandler ()
```

getKey() method

The optional cryptographic key alias.

Syntax

```
String getKey ()
```

getMimeType() method

The mime type of the coupon code.

Syntax

```
String getMimeType ()
```

getPattern() method

The optional code pattern as regular expression.

Syntax

String getPattern ()

isBlob() method

indicates if Coupon Code is binary

Syntax

boolean isBlob ()

isSetBlob() method

Syntax

boolean isSetBlob ()

isSetCryptHandler() method

Syntax

boolean isSetCryptHandler ()

isSetKey() method

Syntax

boolean isSetKey ()

isSetMimeType() method

Syntax

boolean isSetMimeType ()

isSetPattern() method

Syntax

boolean isSetPattern ()

setBlob(boolean) method

Syntax

void setBlob (boolean *isBlob*)

setCryptHandler(String) method

Syntax

```
void setCryptHandler ( String cryptHandler )
```

setKey(String) method

Syntax

```
void setKey ( String key )
```

setMimeType(String) method

Syntax

```
void setMimeType ( String mimeType )
```

setPattern(String) method

Syntax

```
void setPattern ( String pattern )
```

CouponInstance class

Whenever a coupon is assigned to a user a record in this table gets created.

Syntax

```
public class CouponInstance
```

Usage

Some Coupons can be assigned to multiple customers, some can only be assigned to a single customer. Number of views and uses is tracked in this table.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCoupon() method

The Coupon reference.

Syntax

```
Coupon getCoupon ( )
```

getCustomer() method

The owner of this instance.

Syntax

Customer getCustomer ()

getStatus() method

The status of this record.

Syntax

Integer getStatus ()

getTransaction() method

The successful purchase txn for this coupon instance.

Syntax

Transaction getTransaction ()

getUses() method

The number of times the coupon was used.

Syntax

Long getUses ()

getViews() method

The number of times the coupon was viewed.

Syntax

Long getViews ()

isSetCoupon() method

Syntax

boolean isSetCoupon ()

isSetCustomer() method

Syntax

boolean isSetCustomer ()

isSettransaction() method

Syntax

boolean isSettransaction ()

isSetStatus() method

Syntax

boolean isSetStatus ()

isSetUses() method

Syntax

boolean isSetUses ()

isSetViews() method

Syntax

boolean isSetViews ()

setCoupon(Coupon) method

Syntax

void setCoupon (Coupon *coupon*)

setCustomer(Customer) method

Syntax

void setCustomer (Customer *customer*)

setStatus(Integer) method

Syntax

void setStatus (Integer *status*)

setTransaction(Transaction) method

Syntax

void setTransaction (Transaction *transaction*)

setUses(Long) method

Syntax

```
void setUses ( Long uses )
```

setViews(Long) method

Syntax

```
void setViews ( Long views )
```

CouponLocale class

Contains the list of supported locales along with a user friendly description.

Syntax

```
public class CouponLocale
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getDescription() method

The user friendly description of the locale.

Syntax

```
String getDescription ()
```

getLocale() method

The primary key.

Syntax

```
String getLocale ()
```

isActive() method

Indicates if the locale is active or not.

Syntax

```
boolean isActive ()
```

isSetActive() method

Syntax

```
boolean isSetActive ()
```

isSetDescription() method

Syntax

```
boolean isSetDescription ()
```

isSetLocale() method

Syntax

```
boolean isSetLocale ()
```

setActive(boolean) method

Syntax

```
void setActive ( boolean active )
```

setDescription(String) method

Syntax

```
void setDescription ( String description )
```

setLocale(String) method

Syntax

```
void setLocale ( String locale )
```

CouponMimeType class

Contains the list of supported mime type along with a user friendly description and details, such as if it is a binary mime type or alternatives.

Syntax

```
public class CouponMimeType
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getAlternative() method

an alternative mime type

Syntax

CouponMimeType getAlternative ()

getDescription() method

The user friendly description of the mime type.

Syntax

String getDescription ()

getMimeType() method

The primary key.

Syntax

String getMimeType ()

isActive() method

Indicates if the mime type is active or not.

Syntax

boolean isActive ()

isBinary() method

Indicates if the mime type is binary or not.

Syntax

boolean isBinary ()

isSetActive() method

Syntax

boolean isSetActive ()

isSetAlternative() method

Syntax

boolean isSetAlternative ()

isSetBinary() method

Syntax

boolean isSetBinary ()

isSetDescription() method

Syntax

boolean isSetDescription ()

isSetMimeType() method

Syntax

boolean isSetMimeType ()

setActive(boolean) method

Syntax

void setActive (boolean *active*)

setAlternative(CouponMimeType) method

Syntax

void setAlternative (CouponMimeType *alternative*)

setBinary(boolean) method

Syntax

void setBinary (boolean *binary*)

setDescription(String) method

Syntax

void setDescription (String *description*)

setMimeType(String) method

Syntax

void setMimeType (String *locale*)

CouponType class

Each Coupon is assigned to a Coupon Type.

Syntax

```
public class CouponType
```

Usage

The Coupon Type defines the overall validity period, the name, the issuer and some other attributes

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getAvailableThreshold() method

Minimum number of vouchers.

Syntax

```
Long getAvailableThreshold ()
```

Usage

On under run an event is created.

getCouponCodeType() method

Indicates in which format the code is stored and rendered.

Syntax

```
CouponCodeType getCouponCodeType ()
```

getIssueFrom() method

The start date from when Coupons of this type can be issued.

Syntax

```
Date getIssueFrom ()
```

getIssuer() method

The Issuer.

Syntax

```
Customer getIssuer ()
```

getIssueTo() method

The end date until when Coupons of this type can be issued.

Syntax

```
Date getIssueTo ()
```

getLastAvailable() method

Number of available vouchers during last check.

Syntax

```
Long getLastAvailable ()
```

getMaxUses() method

The maximum number the code of the voucher can be used (0 = not limited)

Syntax

```
Long getMaxUses ()
```

Usage

- requires feedback from the redemption system

*getMaxViews() method***Syntax**

```
Long getMaxViews ()
```

getName() method

The name of the Coupon Type.

Syntax

```
String getName ()
```

getPurchaseAmount() method

A not null amount indicates that this coupon must be purchased before redemption.

Syntax

```
Long getPurchaseAmount ()
```

getPurchaseCurrency() method

The currency of the purchase amount.

Syntax

String getPurchaseCurrency ()

isActive() method

Specifies if the Coupon Type is active.

Syntax

boolean isActive ()

isPublic() method

Indicates if this voucher type is browse able and can be searched for by customers.

Syntax

boolean isPublic ()

isRestoreable() method

Indicates if the user can restore the voucher after max views has been exceeded.

Syntax

boolean isRestoreable ()

isActive() method

Syntax

boolean isActive ()

isSetAvailableThreshold() method

Syntax

boolean isSetAvailableThreshold ()

isSetCouponType() method

Syntax

boolean isSetCouponType ()

isSetIssueFrom() method

Syntax

`boolean isSetIssueFrom ()`

isSetIssuer() method

Syntax

`boolean isSetIssuer ()`

isSetIssueTo() method

Syntax

`boolean isSetIssueTo ()`

isSetLastAvailable() method

Syntax

`boolean isSetLastAvailable ()`

isSetMaxUses() method

Syntax

`boolean isSetMaxUses ()`

isSetMaxViews() method

Syntax

`boolean isSetMaxViews ()`

isSetName() method

Syntax

`boolean isSetName ()`

isSetPublic() method

Syntax

`boolean isSetPublic ()`

isSetPurchaseAmount() method

Syntax

boolean isSetPurchaseAmount ()

isSetPurchaseCurrency() method

Syntax

boolean isSetPurchaseCurrency ()

isSetRestoreable() method

Syntax

boolean isSetRestoreable ()

setActive(boolean) method

Syntax

void setActive (boolean *active*)

setAvailableThreshold(Long) method

Syntax

void setAvailableThreshold (Long *availableThreshold*)

setCouponCodeType(CouponCodeType) method

Syntax

void setCouponCodeType (CouponCodeType *codeType*)

setIssueFrom(Date) method

Syntax

void setIssueFrom (Date *issueFrom*)

setIssuer(Customer) method

Syntax

void setIssuer (Customer *issuer*)

setIssueTo(Date) method

Syntax

void setIssueTo (Date *issueTo*)

setLastAvailable(Long) method

Syntax

void setLastAvailable (Long *lastAvailable*)

setMaxUses(Long) method

Syntax

void setMaxUses (Long *maxUses*)

setMaxViews(Long) method

Syntax

void setMaxViews (Long *maxViews*)

setName(String) method

Syntax

void setName (String *name*)

setPublic(boolean) method

Syntax

void setPublic (boolean *isPublic*)

setPurchaseAmount(Long) method

Syntax

void setPurchaseAmount (Long *purchaseAmount*)

setPurchaseCurrency(String) method

Syntax

void setPurchaseCurrency (String *dbPurchaseCurrency*)

setRestorable(boolean) method

Syntax

```
void setRestorable ( boolean restorable )
```

CouponTypeDescription class

A detailed descriptions for Coupon Types in multiple languages and formats.

Syntax

```
public class CouponTypeDescription
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCaption() method

The caption for the voucher type.

Syntax

```
String getCaption ()
```

getContent() method

The full description as a BLOB (text, image, PDF, ...)

Syntax

```
byte[] getContent ()
```

getCouponType() method

The reference to the Coupon Type for which this description is.

Syntax

```
CouponType getCouponType ()
```

getImageUrl() method

The URL to the full sized image.

Syntax

```
String getImageUrl ()
```

getLocale() method

The (Java) locale for which this description is.

Syntax

```
String getLocale ()
```

Usage

Null means default.

getMimeType() method

The mime type.

Syntax

```
String getMimeType ()
```

getThumbnailUrl() method

The URL to a thumbnail image.

Syntax

```
String getThumbnailUrl ()
```

*isSetCaption() method***Syntax**

```
boolean isSetCaption ()
```

*isSetContent() method***Syntax**

```
boolean isSetContent ()
```

*isSetCouponType() method***Syntax**

```
boolean isSetCouponType ()
```

*isSetImageUrl() method***Syntax**

```
boolean isSetImageUrl ()
```

isSetLocale() method

Syntax

boolean isSetLocale ()

isSetThumbnailUrl() method

Syntax

boolean isSetThumbnailUrl ()

isSetMimeType() method

Syntax

boolean isSetMimeType ()

setCaption(String) method

Syntax

void setCaption (String *caption*)

setContent(byte[]) method

Syntax

void setContent (byte[] *content*)

setCouponType(CouponType) method

Syntax

void setCouponType (CouponType *couponType*)

setImageUrl(String) method

Syntax

void setImageUrl (String *imageUrl*)

setLocale(String) method

Syntax

void setLocale (String *locale*)

setMimeType(String) method

Syntax

```
void setMimeType ( String mimeType )
```

setThumbnailUrl(String) method

Syntax

```
void setThumbnailUrl ( String thumbnailUrl )
```

CouponTypeLocation class

A location information for distribution, search and redemption of the coupon type (maximum) radius.

Syntax

```
public class CouponTypeLocation
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCouponType() method

The coupon type reference.

Syntax

```
CouponType getCouponType ()
```

getLatitude() method

Latitude of location in GPS degree format (-90.0 to +90.0 degree) with decimal.

Syntax

```
BigDecimal getLatitude ()
```

Usage

-90 is South pole, +90 is the North pole fraction.

getLongitude() method

Longitude of location in GPS degree format (-180.0 to +180.0) with decimal fraction.

Syntax

BigDecimal getLongitude ()

Usage

Negative values are western longitude.

getRadius() method

The radius/distance in meter.

Syntax

Long getRadius ()

isSetCouponType() method

Syntax

boolean isSetCouponType ()

isSetLatitude() method

Syntax

boolean isSetLatitude ()

isSetLongitude() method

Syntax

boolean isSetLongitude ()

isSetRadius() method

Syntax

boolean isSetRadius ()

setCouponType(CouponType) method

Syntax

void setCouponType (CouponType *couponType*)

setLatitude(BigDecimal) method

Syntax

void setLatitude (BigDecimal *latitude*)

setLongitude(BigDecimal) method

Syntax

void setLongitude (BigDecimal *longitude*)

setRadius(Long) method

Syntax

void setRadius (Long *radius*)

CouponTypeTag class

A search able tag that is assigned to Coupon Types.

Syntax

public class CouponTypeTag

Usage

Tags are used for easy filtering of multiple coupon types.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCouponType() method

the referenced coupon type

Syntax

CouponType getCouponType ()

getTag() method

the tag

Syntax

String getTag ()

isSetCouponType() method

Syntax

boolean isSetCouponType ()

isSetTag() method

Syntax

boolean isSetTag ()

setCouponType(CouponType) method

Syntax

void setCouponType (CouponType *couponType*)

setTag(String) method

Syntax

void setTag (String *tag*)

flows package

Members

All public members of the flows package.

- **bulkfileprocessing package** –
- **customer package** –
- **wallet package** –

bulkfileprocessing package

Members

All public members of the bulkfileprocessing package.

- **api package** –

api package

Members

All public members of the api package.

- **BulkWorkflowFailedException class** – Extends `MobiliserServiceException` and offers an enumeration of different reason codes which can be set depending on the error this exception is caused by.
- **IBulkFileProcessingFlowControl interface** –

BulkWorkflowFailedException class

Extends `MobiliserServiceException` and offers an enumeration of different reason codes which can be set depending on the error this exception is caused by.

Syntax

```
public class BulkWorkflowFailedException
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

BulkWorkflowFailedException(Reason) constructor

Syntax

```
BulkWorkflowFailedException ( Reason reason )
```

BulkWorkflowFailedException(String, Reason) constructor

Syntax

```
BulkWorkflowFailedException ( String message , Reason reason )
```

BulkWorkflowFailedException(Throwable, Reason) constructor

Syntax

```
BulkWorkflowFailedException ( Throwable cause , Reason reason )
```

BulkWorkflowFailedException(String, Throwable, Reason) constructor

Syntax

```
BulkWorkflowFailedException ( String message , Throwable cause ,  
Reason reason )
```

getErrorCode() method

Syntax

```
int getErrorCode ()
```

getReason() method

Syntax

Reason getReason ()

Reason() enumeration

Enum Constant Summary

- **TASK_CLAIMED** –
- **TASK_NOT_FOUND** –
- **CONTINUE_MISSING_PRIVILEGE** –
- **START_MISSING_PRIVILEGE** –
- **TASK_INCOMPLETE** –
- **INVALID_RESPONSE** –
- **TASK_ACCESS** –

IBulkFileProcessingFlowControl interface

Syntax

```
public interface IBulkFileProcessingFlowControl
```

Usage

This interface defines common actions required to implement a workflow handling a maker/checker process for the upload of bulk processing files.

The reference implementation uses the activiti workflow to start a new workflow and store a complete `UploadBulkProcessingFileRequest` call by serializing it into the activiti context (database tables). The continue method then tells activiti to finish the workflow based on our workflow definition and loads the previously stored request from the activiti context and either executes it in the approval case, or dismisses it and takes further defined actions in case of the reject case. The find method allows to search for "paused" upload bulk file entries based on different criteria, which are matched against the stored create customer requests.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

continueProcess(ContinuePendingBulkProcessingFileRequest, final long, final Set<String>) method

Tells activiti to continue a suspended workflow by continuing the task defined by the ContinuePendingBulkProcessingFileRequest request.

Syntax

```
ContinuePendingBulkProcessingFileResponse continueProcess
( ContinuePendingBulkProcessingFileRequest request , final long
callerId , final Set< String > privileges ) throws
BulkWorkflowFailedException
```

Parameters

- **request** –
- **callerId** –
- **privileges** – Must contain all the caller privileges, which are checked against the privileges required to perform the continue operation.

Exceptions

- **BulkWorkflowFailedException class** –

findPendingProcesses(FindPendingBulkProcessingFilesRequest, final long, final Set<String>) method

Returns all suspended UploadBulkProcessingFileRequest requests that match the given criteria.

Syntax

```
FindPendingBulkProcessingFilesResponse findPendingProcesses
( FindPendingBulkProcessingFilesRequest request , final long
callerId , final Set< String > privileges ) throws
BulkWorkflowFailedException
```

Parameters

- **request** –
- **callerId** –
- **privileges** – Must contain all the caller privileges, which are checked against the privileges required to perform the continue operation.

Returns

A list of BulkFile entries matching the criteria including the task ID in which the workflow process is paused in.

Exceptions

- **BulkWorkflowFailedException** class –

Usage

A list of `BulkFile` entries matching the criteria including the task ID in which the workflow process is paused in.

getPendingProcessDetails(final String, final long, final Set< String >) method

Fetches detailed information about the suspended upload bulk file request including the actual file payload.

Syntax

`GetPendingBulkProcessingFileDetailsResponse`

`getPendingProcessDetails ( final String taskId, final long callerId, final Set< String > privileges )` throws `BulkWorkflowFailedException`

Parameters

- **taskId** – The task ID for which the detailed customer information should be fetched.
- **callerId** –
- **privileges** – Must contain all the caller privileges, which are checked against the privileges required to perform the continue operation.

Exceptions

- **BulkWorkflowFailedException** class –

startProcess(UploadBulkProcessingFileRequest, final long) method

Starts an new upload bulk file workflow process and serializes the `UploadBulkProcessingFileRequest` request in the activiti context.

Syntax

`UploadBulkProcessingFileResponse startProcess`

`( UploadBulkProcessingFileRequest request, final long callerId )` throws `BulkWorkflowFailedException`

Parameters

- **request** –
- **callerId** –

Exceptions

- **BulkWorkflowFailedException** class –

customer package

Members

All public members of the customer package.

- **api package** –

api package

Members

All public members of the api package.

- **ICustomerDeactivationConfig** interface –
- **ICustomerDeactivationFlowControl** interface –
- **ICustomerFlowControl** interface –
- **ICustomerRegistrationConfig** interface –
- **WorkflowFailedException** class – Extends `MobiliserServiceException` and offers an enumeration of different reason codes which can be set depending on the error this exception is caused by.

ICustomerDeactivationConfig interface

Syntax

```
public interface ICustomerDeactivationConfig
```

Usage

Configuration interface for the reference implementation of the `ICustomerDeactivationFlowControl` interface

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getMaxObfuscationDelayDays() method

Syntax

```
int getMaxObfuscationDelayDays ()
```

ICustomerDeactivationFlowControl interface

Syntax

```
public interface ICustomerDeactivationFlowControl
```

Usage

This interface defines common actions required to implement a workflow handling a maker/checker process for the deletion of a customer

The reference implementation uses the activiti workflow to start a new workflow and store a complete `DeleteCustomerRequest` call by serializing it into the activiti context (database tables). The `continue` method then tells activiti to finish the workflow based on our workflow definition and loads the previously stored request from the activiti context and either executes it in the approval case, or dismisses it and takes further defined actions in case of the reject case. The `find` method allows to search for "paused" delete customer request entries based on different criteria, which are matched against the stored delete customer requests.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

continueProcess(ContinueDeactivationPendingCustomerForApprovalRequest, long, final Set< String >) method

Tells activiti to continue a suspended workflow by continuing the task defined by the `ContinueDeactivationPendingCustomerForApprovalRequest` requests.

Syntax

```
ContinueDeactivationPendingCustomerForApprovalResponse  
continueProcess  
( ContinueDeactivationPendingCustomerForApprovalRequest request ,  
long callerId , final Set< String > privileges ) throws  
WorkflowFailedException
```

Parameters

- **request** –
- **callerId** –
- **privileges** – Must contain all the caller privileges, which are checked against the privileges required to perform the continue operation.

Exceptions

- **WorkflowFailedException** class –

continueProcess(ContinueDeactivationPendingCustomerForSVAResponse, long, final Set< String >) method

Tells activiti to continue a suspended workflow by continuing the task defined by the ContinueDeactivationPendingCustomerForSVAResponse requests.

Syntax

```
ContinueDeactivationPendingCustomerForSVAResponse
continueProcess
( ContinueDeactivationPendingCustomerForSVAResponse request , long
callerId , final Set< String > privileges ) throws
WorkflowFailedException
```

Parameters

- **request** –
- **callerId** –
- **privileges** – Must contain all the caller privileges, whichh are checked against the privileges required to perform the continue operation.

Exceptions

- **WorkflowFailedException** class –

Usage

This method handles customers who have a SVA with balance left at time of deletion by branching the workflow to handle the transfer of the balance.

findDeactivationPendingProcesses(FindDeactivationPendingCustomersRequest, long, final Set< String >) method

Returns all suspended DeleteCustomerRequest requests that match the given criteria.

Syntax

```
FindDeactivationPendingCustomersResponse
findDeactivationPendingProcesses
( FindDeactivationPendingCustomersRequest request , long callerId ,
final Set< String > privileges ) throws WorkflowFailedException
```

Parameters

- **request** –
- **callerId** –

- **privileges** – Must contain all the caller privileges, which are checked against the privileges required to perform the continue operation.

Returns

A list of `DeactivationPendingCustomer` matching the criteria including the task ID in which the workflow process is paused in.

Exceptions

- **WorkflowFailedException** class –

Usage

A list of `DeactivationPendingCustomer` matching the criteria including the task ID in which the workflow process is paused in.

startProcess(DeleteCustomerRequest, int, String, String, String, String, String, String, final Set< String >, long) method

Starts an new create customer workflow process and serializes the `DeleteCustomerRequest` request along will all additional parameters in the activiti context.

Syntax

```
DeleteCustomerResponse startProcess ( DeleteCustomerRequest  
request , int customerId , String firstName , String lastName , String  
msisdn , String username , String displayName , String email , final  
Set< String > privileges , long callerId ) throws  
WorkflowFailedException
```

Parameters

- **request** –
- **customerId** –
- **firstName** –
- **lastName** –
- **msisdn** –
- **username** –
- **displayName** –
- **email** –
- **privileges** – Must contain all the caller privileges, which are checked against the privileges required to perform the continue operation.
- **callerId** –

Exceptions

- **WorkflowFailedException** class –

ICustomerFlowControl interface

Syntax

```
public interface ICustomerFlowControl
```

Usage

This interface defines common actions required to implement a workflow handling a maker/checker process for the creation of a new customer

The reference implementation uses the activiti workflow to start a new workflow and store a complete `Create[Full[Money]]CustomerRequest` call by serializing it into the activiti context (database tables). The `continue` method then tells activiti to finish the workflow based on our workflow definition and loads the previously stored request from the activiti context and either executes it in the approval case, or dismisses it and takes further defined actions in case of the reject case. The `find` method allows to search for "paused" create customer entries based on different criteria, which are matched against the stored create customer requests.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

continueProcess(ContinuePendingCustomerRequest, long, final Set< String >)
method

Tells activiti to continue a suspended workflow by continuing the task defined by the `ContinuePendingCustomerRequest` request.

Syntax

```
ContinuePendingCustomerResponse continueProcess  
( ContinuePendingCustomerRequest request, long callerId, final Set<  
String > privileges ) throws WorkflowFailedException
```

Parameters

- **request** –
- **callerId** –
- **privileges** – Must contain all the caller privileges, which are checked against the privileges required to perform the continue operation.

Exceptions

- **WorkflowFailedException** class –

findPendingProcesses(FindPendingCustomersRequest, long, final Set< String >)
method

Returns all suspended Create[Full[Money]]Customer requests that match the given criteria.

Syntax

FindPendingCustomersResponse findPendingProcesses
(FindPendingCustomersRequest *request* , long *callerId* , final Set<
String > *privileges*) throws WorkflowFailedException

Parameters

- **request** –
- **callerId** –
- **privileges** – Must contain all the caller privileges, which are checked against the privileges required to perform the continue operation.

Returns

A list of PendingCustomer matching the criteria including the task ID in which the workflow process is paused in.

Exceptions

- **WorkflowFailedException** class –

Usage

A list of PendingCustomer matching the criteria including the task ID in which the workflow process is paused in.

getPendingProcessDetails(final String, final long, final Set< String >) *method*
Fetches detailed information about the suspended create customer request (e.g.

Syntax

GetPendingcustomerDetailsResponse getPendingProcessDetails
(final String *taskId* , final long *callerId* , final Set< String >
privileges)

Parameters

- **taskId** – The task ID for which the detailed customer information should be fetched.

- **callerId** –
- **privileges** – Must contain all the caller privileges, which are checked against the privileges required to perform the continue operation.

Usage

identifications, addresses, credentials, attributes, etc.).

startProcess(CreateCustomerRequest, long) method

Starts an new create customer workflow process and serializes the CreateCustomerRequest request in the activiti context.

Syntax

```
CreateCustomerResponse startProcess ( CreateCustomerRequest
request , long callerId ) throws WorkflowFailedException
```

Parameters

- **request** –
- **callerId** –

Returns

The CreateCustomerResponse does not yet contains the new customer ID, since the customer has not been stored, yet

Exceptions

- **WorkflowFailedException** class –

Usage

The CreateCustomerResponse does not yet contains the new customer ID, since the customer has not been stored, yet

startProcess(CreateFullCustomerRequest, long) method

Starts an new create customer workflow process and serializes the CreateFullCustomerRequest request in the activiti context.

Syntax

```
CreateFullCustomerResponse startProcess
( CreateFullCustomerRequest request , long callerId ) throws
WorkflowFailedException
```

Parameters

- **request** –
- **callerId** –

Returns

The `CreateFullCustomerResponse` does not yet contains the new customer ID, since the customer has not been stored, yet

Exceptions

- **WorkflowFailedException** class –

Usage

The `CreateFullCustomerResponse` does not yet contains the new customer ID, since the customer has not been stored, yet

startProcess(CreateFullMoneyCustomerRequest, long) method

Starts an new create customer workflow process and serializes the `CreateFullMoneyCustomerRequest` request in the activiti context.

Syntax

```
CreateFullMoneyCustomerResponse startProcess  
( CreateFullMoneyCustomerRequest request , long callerId ) throws  
WorkflowFailedException
```

Parameters

- **request** –
- **callerId** –

Returns

The `CreateFullCustomerResponse` does not yet contains the new customer ID, since the customer has not been stored, yet

Exceptions

- **WorkflowFailedException** class –

Usage

The `CreateFullCustomerResponse` does not yet contains the new customer ID, since the customer has not been stored, yet

*ICustomerRegistrationConfig interface***Syntax**

```
public interface ICustomerRegistrationConfig
```

Usage

Configuration interface for the reference implementation of the ICustomerFlowControl interface

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

*getRejectedCustomerRegistrationBlacklistReason() method***Syntax**

```
int getRejectedCustomerRegistrationBlacklistReason ()
```

*getRejectedCustomerRegistrationCancellationReason() method***Syntax**

```
int getRejectedCustomerRegistrationCancellationReason ()
```

*isObfuscateRejectedCustomerRegistrationData() method***Syntax**

```
boolean isObfuscateRejectedCustomerRegistrationData ()
```

WorkflowFailedException class

Extends MobiliserServiceException and offers an enumeration of different reason codes which can be set depending on the error this exception is caused by.

Syntax

```
public class WorkflowFailedException
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

WorkflowFailedException(Reason) constructor

Syntax

`WorkflowFailedException ( Reason reason )`

WorkflowFailedException(String, Reason) constructor

Syntax

`WorkflowFailedException ( String message , Reason reason )`

WorkflowFailedException(Throwable, Reason) constructor

Syntax

`WorkflowFailedException ( Throwable cause , Reason reason )`

WorkflowFailedException(String, Throwable, Reason) constructor

Syntax

`WorkflowFailedException ( String message , Throwable cause , Reason reason )`

getErrorCode() method

Syntax

`int getErrorCode ()`

getReason() method

Syntax

`Reason getReason ()`

Reason() enumeration

Enum Constant Summary

- **TASK_CLAIMED** –
- **TASK_NOT_FOUND** –
- **CONTINUE_MISSING_PRIVILEGE** –
- **START_MISSING_PRIVILEGE** –
- **TASK_INCOMPLETE** –
- **INVALID_RESPONSE** –

- **TASK_ACCESS** –

wallet package

Members

All public members of the wallet package.

- **api package** –

api package

Members

All public members of the api package.

- **IWalletFlowControl interface** –
- **WalletWorkflowFailedException class** – Extends `MobiliserServiceException` and offers an enumeration of different reason codes which can be set depending on the error this exception is caused by.

IWalletFlowControl interface

Syntax

```
public interface IWalletFlowControl
```

Usage

This interface defines common actions required to implement a workflow handling a maker/checker process for the creation of new customer wallet entries

The reference implementation uses the activiti workflow to start a new workflow and store a complete `CreateWalletEntry` call by serializing it into the activiti ontext (database tables). The continue method then tells activiti to finish the workflow based on our workflow definition and loads the previously stored request from the activiti context and either executes it in the approval case, or dismisses it and takes further defined actions in case of the reject case. The find method allows to search for "paused" create wallet entries based on different criteria, which are matched against the stored create wallet requests.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

addBalanceAlertRequest(CreateBalanceAlertRequest, final int, final String, final long, final Set< String >) method

Adds an additional balance alert to the pending create wallet entry process instance.

Syntax

```
void addBalanceAlertRequest ( CreateBalanceAlertRequest  
baRequest, final int piTypeId, final String processInstanceId, final  
long callerId, final Set< String > privileges )
```

Parameters

- **baRequest** – The balance alert request
- **piTypeId** –
- **processInstanceId** – The workflow process instance the balance alert has to be attached to
- **callerId** –
- **privileges** – Must contain all the caller privileges, whichh are checked against the privileges required to perform the continue operation.

continueProcess(ContinuePendingWalletEntryRequest, final long, final Set< String >) method

Tells activiti to continue a suspended workflow by continuing the task defined by the ContinuePendingWalletEntryRequest.

Syntax

```
ContinuePendingWalletEntryResponse continueProcess  
( ContinuePendingWalletEntryRequest request, final long callerId,  
final Set< String > privileges ) throws  
WalletWorkflowFailedException
```

Parameters

- **request** –
- **callerId** –
- **privileges** – Must contain all the caller privileges, whichh are checked against the privileges required to perform the continue operation.

Exceptions

- **WalletWorkflowFailedException** class –

findPendingProcesses(FindPendingWalletEntriesRequest, final long, final Set<String >) method

Returns all suspended CreateWalletEntry requests that match the given criteria.

Syntax

```
FindPendingWalletEntriesResponse findPendingProcesses
( FindPendingWalletEntriesRequest request ,   final long callerId ,
  final Set< String > privileges ) throws
WalletWorkflowFailedException
```

Parameters

- **request** –
- **callerId** –
- **privileges** – Must contain all the caller privileges, whichh are checked against the privileges required to perform the continue operation.

Returns

A list of PendingWalletEntry matching the criteria including the task ID in which the workflow process is paused in.

Exceptions

- **WalletWorkflowFailedException** class –

Usage

A list of PendingWalletEntry matching the criteria including the task ID in which the workflow process is paused in.

startProcess(CreateWalletEntryRequest, final long) method

Starts an new create wallet entry workflow process and serializes the request in the activiti context.

Syntax

```
CreateWalletEntryResponse startProcess
( CreateWalletEntryRequest request ,   final long callerId ) throws
WalletWorkflowFailedException
```

Parameters

- **request** –

- **callerId** –

Returns

The `CreateWalletEntryResponse` does not yet contains the new wallet entry ID, since the wallet entry has not been stored, yet

Exceptions

- **WalletWorkflowFailedException** class –

Usage

The `CreateWalletEntryResponse` does not yet contains the new wallet entry ID, since the wallet entry has not been stored, yet

WalletWorkflowFailedException class

Extends `MobiliserServiceException` and offers an enumeration of different reason codes which can be set depending on the error this exception is caused by.

Syntax

```
public class WalletWorkflowFailedException
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

WalletWorkflowFailedException(Reason) constructor

Syntax

```
WalletWorkflowFailedException ( Reason reason )
```

WalletWorkflowFailedException(String, Reason) constructor

Syntax

```
WalletWorkflowFailedException ( String message , Reason reason )
```

WalletWorkflowFailedException(Throwable, Reason) constructor

Syntax

```
WalletWorkflowFailedException ( Throwable cause , Reason reason )
```

WalletWorkflowFailedException(String, Throwable, Reason) constructor

Syntax

`WalletWorkflowFailedException ( String message , Throwable cause , Reason reason )`

getErrorCode() method

Syntax

`int getErrorCode ()`

getReason() method

Syntax

`Reason getReason ()`

Reason() enumeration

Enum Constant Summary

- **TASK_CLAIMED** –
- **TASK_NOT_FOUND** –
- **CONTINUE_MISSING_PRIVILEGE** –
- **START_MISSING_PRIVILEGE** –
- **TASK_INCOMPLETE** –
- **INVALID_RESPONSE** –
- **TASK_ACCESS** –

jobs package

Members

All public members of the jobs package.

- **alerts package** –
- **event package** –
- **tasks package** –

alerts package

Members

All public members of the alerts package.

- **model package** –

model package

Members

All public members of the model package.

- **AccountBalanceSummary** class –
- **AccountBalanceThreshold** class –
- **CustomerLockout** class –
- **InvalidAccountAccess** class –
- **Location** class – The `Location`
- **PasswordChanged** class –
- **TransactionSummary** class –

AccountBalanceSummary class

Syntax

```
public class AccountBalanceSummary
```

Usage

The `AccountBalanceSummary` notifies a customer with the end of day balance for a specified account.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

AccountBalanceSummary() constructor

Syntax

```
AccountBalanceSummary ()
```

AccountBalanceSummary(EventData) constructor

Syntax

```
AccountBalanceSummary ( EventData data )
```

AccountBalanceSummary(String, EventData) constructor

Syntax

```
AccountBalanceSummary ( String name , EventData data )
```

createNew() method

Syntax

IAAlert createNew ()

getAccountPiAlias() method

Syntax

String getAccountPiAlias ()

getAccountPiBalance() method

Syntax

long getAccountPiBalance ()

getAccountPiCurrency() method

Syntax

String getAccountPiCurrency ()

getAccountPild() method

Syntax

long getAccountPiId ()

getCustomerCountry() method

Syntax

long getCustomerCountry ()

getCustomerLanguage() method

Syntax

long getCustomerLanguage ()

getCustomerOrgUnit() method

Syntax

long getCustomerOrgUnit ()

getCustomerPild() method

Syntax

long getCustomerPiId ()

init() method

Syntax

void init ()

setAccountPiAlias(String) method

Syntax

void setAccountPiAlias (String *value*)

setAccountPiBalance(long) method

Syntax

void setAccountPiBalance (long *value*)

setAccountPiCurrency(String) method

Syntax

void setAccountPiCurrency (String *value*)

setAccountPild(long) method

Syntax

void setAccountPiId (long *value*)

setCustomerCountry(String) method

Syntax

void setCustomerCountry (String *value*)

setCustomerLanguage(String) method

Syntax

void setCustomerLanguage (String *value*)

setCustomerOrgUnit(String) method

Syntax

```
void setCustomerOrgUnit ( String value )
```

setCustomerPild(long) method

Syntax

```
void setCustomerPiId ( long value )
```

ACCOUNT_PI_ALIAS variable

Syntax

```
final String ACCOUNT_PI_ALIAS
```

ACCOUNT_PI_BALANCE variable

Syntax

```
final String ACCOUNT_PI_BALANCE
```

ACCOUNT_PI_BALANCE_DISPLAY variable

Syntax

```
final String ACCOUNT_PI_BALANCE_DISPLAY
```

ACCOUNT_PI_CURRENCY variable

Syntax

```
final String ACCOUNT_PI_CURRENCY
```

ACCOUNT_PI_ID variable

Syntax

```
final String ACCOUNT_PI_ID
```

CUSTOMER_COUNTRY variable

Syntax

```
final String CUSTOMER_COUNTRY
```

CUSTOMER_LANGUAGE variable

Syntax

```
final String CUSTOMER_LANGUAGE
```

CUSTOMER_ORGUNIT variable

Syntax

```
final String CUSTOMER_ORGUNIT
```

CUSTOMER_PI_ID variable

Syntax

```
final String CUSTOMER_PI_ID
```

EVENT_TYPE variable

note: must match to database table ALR_TYPE.STR_NAME

Syntax

```
final String EVENT_TYPE
```

AccountBalanceThreshold class

Syntax

```
public class AccountBalanceThreshold
```

Usage

The `AccountBalanceThreshold` notifies customer when a specific an account has gone below or above a specified minimum or maximum value.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

AccountBalanceThreshold() constructor

Syntax

```
AccountBalanceThreshold ()
```

AccountBalanceThreshold(EventData) constructor

Syntax

```
AccountBalanceThreshold ( EventData data )
```

AccountBalanceThreshold(String, EventData) constructor

Syntax

`AccountBalanceThreshold ( String name ,   EventData data )`

createNew() method

Syntax

`IAAlert createNew ()`

getAccountPiAlias() method

Syntax

`String getAccountPiAlias ()`

getAccountPiBalance() method

Syntax

`long getAccountPiBalance ()`

getAccountPiCurrency() method

Syntax

`String getAccountPiCurrency ()`

getAccountPild() method

Syntax

`long getAccountPiId ()`

getCustomerCountry() method

Syntax

`long getCustomerCountry ()`

getCustomerLanguage() method

Syntax

`long getCustomerLanguage ()`

getCustomerMaxAmount() method

Syntax

long getCustomerMaxAmount ()

getCustomerMinAmount() method

Syntax

long getCustomerMinAmount ()

getCustomerOrgUnit() method

Syntax

long getCustomerOrgUnit ()

getCustomerPild() method

Syntax

long getCustomerPiId ()

init() method

Syntax

void init ()

setAccountPiAlias(String) method

Syntax

void setAccountPiAlias (String *value*)

setAccountPiBalance(long) method

Syntax

void setAccountPiBalance (long *value*)

setAccountPiCurrency(String) method

Syntax

void setAccountPiCurrency (String *value*)

setAccountPild(long) method

Syntax

`void setAccountPiId ( long value )`

setCustomerCountry(String) method

Syntax

`void setCustomerCountry ( String value )`

setCustomerLanguage(String) method

Syntax

`void setCustomerLanguage ( String value )`

setCustomerMaxAmount(long) method

Syntax

`void setCustomerMaxAmount ( long value )`

setCustomerMinAmount(long) method

Syntax

`void setCustomerMinAmount ( long value )`

setCustomerOrgUnit(String) method

Syntax

`void setCustomerOrgUnit ( String value )`

setCustomerPild(long) method

Syntax

`void setCustomerPiId ( long value )`

ACCOUNT_PI_ALIAS variable

Syntax

`final String ACCOUNT_PI_ALIAS`

ACCOUNT_PI_BALANCE variable

Syntax

```
final String ACCOUNT_PI_BALANCE
```

ACCOUNT_PI_BALANCE_DISPLAY variable

Syntax

```
final String ACCOUNT_PI_BALANCE_DISPLAY
```

ACCOUNT_PI_CURRENCY variable

Syntax

```
final String ACCOUNT_PI_CURRENCY
```

ACCOUNT_PI_ID variable

Syntax

```
final String ACCOUNT_PI_ID
```

CUSTOMER_COUNTRY variable

Syntax

```
final String CUSTOMER_COUNTRY
```

CUSTOMER_LANGUAGE variable

Syntax

```
final String CUSTOMER_LANGUAGE
```

CUSTOMER_MAX_AMOUNT variable

Syntax

```
final String CUSTOMER_MAX_AMOUNT
```

CUSTOMER_MAX_AMOUNT_DISPLAY variable

Syntax

```
final String CUSTOMER_MAX_AMOUNT_DISPLAY
```

CUSTOMER_MIN_AMOUNT variable

Syntax

```
final String CUSTOMER_MIN_AMOUNT
```

CUSTOMER_MIN_AMOUNT_DISPLAY variable

Syntax

```
final String CUSTOMER_MIN_AMOUNT_DISPLAY
```

CUSTOMER_ORGUNIT variable

Syntax

```
final String CUSTOMER_ORGUNIT
```

CUSTOMER_PI_ID variable

Syntax

```
final String CUSTOMER_PI_ID
```

EVENT_TYPE variable

note: must match to database table ALR_TYPE.STR_NAME

Syntax

```
final String EVENT_TYPE
```

CustomerLockout class

Syntax

```
public class CustomerLockout
```

Usage

The CustomerLockout notifies customer when customer is locked out due to invalid credentials.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CustomerLockout() constructor

Syntax

```
CustomerLockout ()
```

CustomerLockout(EventData) constructor

Syntax

`CustomerLockout ( EventData data )`

CustomerLockout(String, EventData) constructor

Syntax

`CustomerLockout ( String name , EventData data )`

createNew() method

Syntax

`IAlert createNew ()`

init() method

Syntax

`void init ()`

EVENT_TYPE variable

note: must match to database table ALR_TYPE.STR_NAME

Syntax

`final String EVENT_TYPE`

InvalidAccountAccess class

Syntax

`public class InvalidAccountAccess`

Usage

The InvalidAccountAccess notifies customer if there has been an invalid account access at initial login.

The InvalidAccountAccess

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

InvalidAccountAccess() constructor

Syntax

```
InvalidAccountAccess ()
```

InvalidAccountAccess(EventData) constructor

Syntax

```
InvalidAccountAccess ( EventData data )
```

InvalidAccountAccess(String, EventData) constructor

Syntax

```
InvalidAccountAccess ( String name ,   EventData data )
```

createNew() method

Syntax

```
IAlert createNew ()
```

init() method

Syntax

```
void init ()
```

EVENT_TYPE variable

Syntax

```
final String EVENT_TYPE
```

Location class

The Location

Syntax

```
public class Location
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Location() constructor

Syntax

`Location ()`

Location(EventData) constructor

Syntax

`Location ( EventData data )`

Location(String, EventData) constructor

Syntax

`Location ( String name ,   EventData data )`

createNew() method

Syntax

`IAlert createNew ()`

getAlertLatitude() method

Syntax

`double getAlertLatitude ()`

getAlertLogitude() method

Syntax

`double getAlertLogitude ()`

getAlertRadius() method

Syntax

`long getAlertRadius ()`

getCustomerLatitude() method

Syntax

`double getCustomerLatitude ()`

getCustomerLogitude() method

Syntax

double getCustomerLogitude ()

getMessageParams() method

Syntax

Map< String, String > getMessageParams ()

init() method

Syntax

void init ()

setAlertLatitude(double) method

Syntax

void setAlertLatitude (double *latitude*)

setAlertLongitude(double) method

Syntax

void setAlertLongitude (double *longitude*)

setAlertRadius(long) method

Syntax

void setAlertRadius (long *radius*)

setCustomerLatitude(double) method

Syntax

void setCustomerLatitude (double *latitude*)

setCustomerLongitude(double) method

Syntax

void setCustomerLongitude (double *longitude*)

setMessageParams(Map< String, String >) method

Syntax

```
void setMessageParams ( Map< String, String > map )
```

ALERT_LATITUDE variable

Syntax

```
final String ALERT_LATITUDE
```

ALERT_LONGITUDE variable

Syntax

```
final String ALERT_LONGITUDE
```

ALERT_RADIUS variable

Syntax

```
final String ALERT_RADIUS
```

CUSTOMER_LATITUDE variable

Syntax

```
final String CUSTOMER_LATITUDE
```

CUSTOMER_LONGITUDE variable

Syntax

```
final String CUSTOMER_LONGITUDE
```

EVENT_TYPE variable

Syntax

```
final String EVENT_TYPE
```

ISSUER variable

Syntax

```
final String ISSUER
```

*MESSAGE_PARAMS variable**Syntax*

```
final String MESSAGE_PARAMS
```

*PasswordChanged class**Syntax*

```
public class PasswordChanged
```

Usage

The PasswordChanged notifies customer if their password has been changed.

The PasswordChanged

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

*PasswordChanged() constructor**Syntax*

```
PasswordChanged ()
```

*PasswordChanged(EventData) constructor**Syntax*

```
PasswordChanged ( EventData data )
```

*PasswordChanged(String, EventData) constructor**Syntax*

```
PasswordChanged ( String name , EventData data )
```

*createNew() method**Syntax*

```
IAlert createNew ()
```

init() method

Syntax

```
void init ()
```

EVENT_TYPE variable

note: must match to database table ALR_TYPE.STR_NAME

Syntax

```
final String EVENT_TYPE
```

TransactionSummary class

Syntax

```
public class TransactionSummary
```

Usage

The `TransactionSummary` notifies customer when a specific type of transaction has been performed on the specified account with an optional specified amount value.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Match class

Syntax

```
public class Match
```

createNew() method

Syntax

```
IAlert createNew ()
```

getAccountPild() method

Syntax

```
long getAccountPiId ()
```

getCustomerAmount() method

Syntax

long getCustomerAmount ()

getCustomerPild() method

Syntax

long getCustomerPiId ()

getCustomerUseCase() method

Syntax

long getCustomerUseCase ()

getMatchOperation() method

Syntax

String getMatchOperation ()

getTxnAmount() method

Syntax

long getTxnAmount ()

getTxnAmountDisplay() method

Syntax

String getTxnAmountDisplay ()

getTxnCurrency() method

Syntax

String getTxnCurrency ()

getTxnDateTime() method

Syntax

String getTxnDateTime ()

getTxnId() method

Syntax

long getTxnId ()

getTxnParticipant() method

Syntax

String getTxnParticipant ()

getTxnPiAlias() method

Syntax

String getTxnPiAlias ()

getTxnUseCase() method

Syntax

int getTxnUseCase ()

init() method

Syntax

void init ()

setAccountPild(long) method

Syntax

void setAccountPiId (long *value*)

setCustomerAmount(long) method

Syntax

void setCustomerAmount (long *value*)

setCustomerPild(long) method

Syntax

void setCustomerPiId (long *value*)

setCustomerUseCase(long) method

Syntax

void setCustomerUseCase (long *value*)

setMatchOperation(String) method

Syntax

void setMatchOperation (String *value*)

setTxnAmount(long) method

Syntax

void setTxnAmount (long *value*)

setTxnAmountDisplay(String) method

Syntax

void setTxnAmountDisplay (String *value*)

setTxnCurrency(String) method

Syntax

void setTxnCurrency (String *value*)

setTxnDateTime(String) method

Syntax

void setTxnDateTime (String *value*)

setTxnId(long) method

Syntax

void setTxnId (long *value*)

setTxnParticipant(String) method

Syntax

void setTxnParticipant (String *value*)

setTxnPiAlias(String) method

Syntax

void setTxnPiAlias (String *value*)

setTxnUseCase(int) method

Syntax

void setTxnUseCase (int *value*)

ACCOUNT_PI_ID variable

Syntax

final String ACCOUNT_PI_ID

CUSTOMER_AMOUNT variable

Syntax

final String CUSTOMER_AMOUNT

CUSTOMER_PI_ID variable

Syntax

final String CUSTOMER_PI_ID

CUSTOMER_USE_CASE variable

Syntax

final String CUSTOMER_USE_CASE

MATCH_OPERATION variable

Syntax

final String MATCH_OPERATION

TXN_AMOUNT variable

Syntax

final String TXN_AMOUNT

TXN_AMOUNT_DISPLAY variable

Syntax

```
final String TXN_AMOUNT_DISPLAY
```

TXN_CURRENCY variable

Syntax

```
final String TXN_CURRENCY
```

TXN_DATE_TIME variable

Syntax

```
final String TXN_DATE_TIME
```

TXN_ID variable

Syntax

```
final String TXN_ID
```

TXN_PARTICIPANT variable

Syntax

```
final String TXN_PARTICIPANT
```

TXN_PI_ALIAS variable

Syntax

```
final String TXN_PI_ALIAS
```

TXN_USE_CASE variable

Syntax

```
final String TXN_USE_CASE
```

TransactionSummary() constructor

Syntax

```
TransactionSummary ()
```

TransactionSummary(EventData) constructor

Syntax

TransactionSummary (EventData *data*)

TransactionSummary(String, EventData) constructor

Syntax

TransactionSummary (String *name* , EventData *data*)

addTransaction(final TransactionSummary.Match) method

Syntax

void addTransaction (final TransactionSummary.Match *txnMatch*)

createNew() method

Syntax

IAAlert createNew ()

getCustomerCountry() method

Syntax

long getCustomerCountry ()

getCustomerLanguage() method

Syntax

long getCustomerLanguage ()

getCustomerOrgUnit() method

Syntax

long getCustomerOrgUnit ()

getTransaction() method

Syntax

List< TransactionSummary.Match > getTransaction ()

init() method

Syntax

```
void init ()
```

setCustomerCountry(String) method

Syntax

```
void setCustomerCountry ( String value )
```

setCustomerLanguage(String) method

Syntax

```
void setCustomerLanguage ( String value )
```

setCustomerOrgUnit(String) method

Syntax

```
void setCustomerOrgUnit ( String value )
```

CUSTOMER_COUNTRY variable

Syntax

```
final String CUSTOMER_COUNTRY
```

CUSTOMER_LANGUAGE variable

Syntax

```
final String CUSTOMER_LANGUAGE
```

CUSTOMER_ORGUNIT variable

Syntax

```
final String CUSTOMER_ORGUNIT
```

EVENT_TYPE variable

note: must match to database table ALR_TYPE.STR_NAME

Syntax

```
final String EVENT_TYPE
```

TXN_ID_KEY_PREFIX variable

Syntax

```
final String TXN_ID_KEY_PREFIX
```

event package

Members

All public members of the event package.

- **handler package** –
- **model package** –

handler package

Members

All public members of the handler package.

- **util package** –

util package

Members

All public members of the util package.

- **AuthenticatingMoneyEventHandler< C extends IMoneyEventHandlerConfiguration > class** –
- **IMoneyEventHandlerConfiguration interface** – The IMoneyEventHandlerConfiguration
- **MoneyEventHandler< C extends IMoneyEventHandlerConfiguration > class** –
- **MoneyEventHandlerConfiguration class** – The MoneyEventHandlerConfiguration

AuthenticatingMoneyEventHandler< C extends IMoneyEventHandlerConfiguration > class

Syntax

```
public class AuthenticatingMoneyEventHandler< C extends  
IMoneyEventHandlerConfiguration >
```

Usage

The `AuthenticatingMoneyEventHandler` is the base class for all Mobiliser event handlers which need to do internal service calls with an authenticated user. This super class

will handler setting up the spring security context so that does not need to be repeated for every handler.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

afterPropertiesSet() method

Syntax

```
final void afterPropertiesSet () throws Exception
```

doAfterPropertiesSet() method

Syntax

```
abstract void doAfterPropertiesSet ()
```

doProcess(final Event) method

Hook for subclasses to actually process the event.

Syntax

```
abstract boolean doProcess ( final Event e )
```

Parameters

- **e** – the event

Returns

result

Usage

result

process(final Event) method

Syntax

```
final boolean process ( final Event e )
```

setAuthenticatorFactory(final InternalServiceCallAuthenticatorFactory) method

Syntax

```
void setAuthenticatorFactory ( final  
InternalServiceCallAuthenticatorFactory authenticatorFactory )
```

Parameters

- **authenticatorFactory** – the authenticatorFactory to set

logger variable

Syntax

```
final Logger logger
```

IMoneyEventHandlerConfiguration interface

The IMoneyEventHandlerConfiguration

Syntax

```
public interface IMoneyEventHandlerConfiguration
```

Derived classes

- *MoneyEventHandlerConfiguration* on page 642

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getExpiryPeriod() method

Gets the event expiry timeframe.

Syntax

```
int getExpiryPeriod ()
```

Returns

the event expiry timeframe

Usage

the event expiry timeframe

getMaxActive() method

Max Active controls the maximum number of objects that can be allocated by the pool (checked out to clients, or idle awaiting checkout) at a given time.

Syntax

```
int getMaxActive ()
```

Returns

integer value

Usage

When non-positive, there is no limit to the number of objects that can be managed by the pool at one time. When maxActive is reached, the pool is said to be exhausted. The default setting for this parameter is 4

integer value

getMaxIdle() method

Max Idle controls the maximum number of objects that can sit idle in the pool at any time.

Syntax

```
int getMaxIdle ()
```

Returns

integer value

Usage

When negative, there is no limit to the number of objects that may be idle at one time. The default setting for this parameter is 2.

integer value

getMaxRetries() method

Gets the maximum number of retries.

Syntax

```
long getMaxRetries ()
```

Returns

the maximum number of retries

Usage

the maximum number of retries

getRetryDelay() method

Gets the delay in seconds to wait before retrying a failed event.

Syntax

```
long getRetryDelay ()
```

Returns

the delay in seconds to wait before retrying a failed event

Usage

the delay in seconds to wait before retrying a failed event

MoneyEventHandler< C extends IMoneyEventHandlerConfiguration > class

Syntax

```
public class MoneyEventHandler< C extends  
IMoneyEventHandlerConfiguration >
```

Usage

The `MoneyEventHandler` provides a generic retry mechanism for event handling.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

afterPropertiesSet() method

Checks whether the configuration is set.

Syntax

```
void afterPropertiesSet () throws Exception
```

Usage

If the configuration is not set, an

`IllegalStateException`

will be thrown.

IMPLEMENTATION NOTE: Every subclass that overrides this method should call this implementation of

afterPropertiesSet

.

createRetryEvent(final Event, boolean) method

Creates a new event with the same name and data as the `currentEvent` and the handler name of this instance.

Syntax

```
final boolean createRetryEvent ( final Event currentEvent , boolean
requiresNewTransaction )
```

Parameters

- **currentEvent** – the event that should be retried
- **requiresNewTransaction** – indicates whether a new transaction should be used if another transaction has already been opened

Returns

`true`, if the new (retry) event was created successfully. `false` is returned, if the maximum number of retries has been reached or the new event was not accepted and processed respectively.

Exceptions

- **NullPointerException** – thrown if `currentEvent` is null.

Usage

This event is passed to the event system if the maximum number of retries is not reached.

The retry event will be queued with a configurable delay.

`true`, if the new (retry) event was created successfully. `false` is returned, if the maximum number of retries has been reached or the new event was not accepted and processed respectively.

getConfiguration() method

Syntax

```
final C getConfiguration ()
```

getEventGenerator() method

Syntax

```
final EventGenerator getEventGenerator ()
```

getExpiryPeriod() method

Syntax

```
int getExpiryPeriod ()
```

getMaxActive() method

Syntax

```
int getMaxActive ()
```

getMaxIdle() method

Syntax

```
int getMaxIdle ()
```

getTransactionManager() method

Syntax

```
final PlatformTransactionManager getTransactionManager ()
```

setConfiguration(C) method

Syntax

```
final void setConfiguration ( C configuration )
```

setEventGenerator(EventGenerator) method

Syntax

```
final void setEventGenerator ( EventGenerator eventGenerator )
```

setTransactionManager(PlatformTransactionManager) method

Syntax

```
final void setTransactionManager ( PlatformTransactionManager  
transactionManager )
```

verifyEventGenerator() method

Checks whether the event generator is set.

Syntax

```
void verifyEventGenerator ()
```

Usage

If the event generator is not set, an

`IllegalStateException`

will be thrown.

IMPLEMENTATION NOTE: Call this method from subclasses'

`afterPropertiesSet`

method, as soon as the retry event mechanism or the event generator is used directly.

verifyTransactionManager() method

Checks whether the transaction manager is set.

Syntax

```
void verifyTransactionManager ()
```

Usage

If the transaction manager is not set, an

`IllegalStateException`

will be thrown.

IMPLEMENTATION NOTE: Call this method from subclasses'

`afterPropertiesSet`

method, as soon as the retry event mechanism or the transaction manager is used directly.

EVENT_DATA_KEY_RETRY_COUNT variable

The key for the event's retry counter.

Syntax

```
final String EVENT_DATA_KEY_RETRY_COUNT
```

MoneyEventHandlerConfiguration class

The MoneyEventHandlerConfiguration

Syntax

```
public class MoneyEventHandlerConfiguration
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCallerId() method

Syntax

```
long getCallerId ()
```

getExpiryPeriod() method

Syntax

```
int getExpiryPeriod ()
```

getMaxActive() method

Syntax

```
int getMaxActive ()
```

getMaxIdle() method

Syntax

```
int getMaxIdle ()
```

getMaxRetries() method

Syntax

```
long getMaxRetries ()
```

getRetryDelay() method

Syntax

```
long getRetryDelay ()
```

DEFAULT_CALLER_ID variable

The default value for the caller ID.

Syntax

```
final long DEFAULT_CALLER_ID
```

PREFS_CALLER_ID variable

The preferences key for the caller ID.

Syntax

```
final String PREFS_CALLER_ID
```

PREFS_EVENT_EXPIRY variable

The preferences key for the event expiry.

Syntax

```
final String PREFS_EVENT_EXPIRY
```

PREFS_MAX_ACTIVE variable

The preferences key for the maximum number of active handler pool threads.

Syntax

```
final String PREFS_MAX_ACTIVE
```

PREFS_MAX_IDLE variable

The preferences key for the maximum number of idle handler pool threads.

Syntax

```
final String PREFS_MAX_IDLE
```

PREFS_MAX_RETRIES variable

The preferences key for the maximum number of retries.

Syntax

```
final String PREFS_MAX_RETRIES
```

PREFS_RETRY_DELAY variable

The preferences key for the retry delay.

Syntax

```
final String PREFS_RETRY_DELAY
```

model package

Members

All public members of the model package.

- **customer package** –
- **transaction package** –

customer package

Members

All public members of the customer package.

- **CredentialChangeEvent class** – This `CredentialChangeEvent` is fired when a customer's credential has been changed.
- **CustomerBlacklistChangeEvent class** – This `CustomerTemporarilyBlockEvent` is fired when a customer's blacklist reason has changed.
- **CustomerDeactivatedEvent class** – This `CustomerDeactivatedEvent` is fired when a customer is deactivated.
- **CustomerEvent class** – This abstract `CustomerEvent` can be used as the basis for customer related events.
- **CustomerObfuscatedEvent class** – This abstract `CustomerObfuscatedEvent` is fired when a customer has been obfuscated.
- **CustomerTemporarilyBlockedEvent class** – This `CustomerTemporarilyBlockedEvent` is fired when a customer's credential's have been wrongly presented more than the maximum number of times for their security policy AND the policy is set to block customer access for a specified time interval.
- **RegistrationCompleteEvent class** – This `RegistrationCompleteEvent` is fired when a new customer was registered completely and its BLR changed from preregistered - 5 to none - 0.
- **WrongCredentialEvent class** – This `WrongCredentialEvent` is fired when a customer's account has been access with an incorrect credential.

CredentialChangeEvent class

This `CredentialChangeEvent` is fired when a customer's credential has been changed.

Syntax

```
public class CredentialChangeEvent
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

CredentialChangeEvent() constructor

Syntax

```
CredentialChangeEvent ()
```

CredentialChangeEvent(EventData) constructor

Syntax

```
CredentialChangeEvent ( EventData data )
```

getCredentialStatus() method

Syntax

```
int getCredentialStatus ()
```

getCredentialType() method

Syntax

```
int getCredentialType ()
```

getTimestamp() method

Syntax

```
long getTimestamp ()
```

isTransient() method

Syntax

```
boolean isTransient ()
```

setCredentialStatus(int) method

Syntax

```
void setCredentialStatus ( int credStatus )
```

setCredentialType(int) method

Syntax

```
void setCredentialType ( int credType )
```

setTimestamp(long) method

Syntax

```
void setTimestamp ( long timestamp )
```

CustomerBlacklistChangeEvent class

This `CustomerTemporarilyBlockEvent` is fired when a customer's blacklist reason has changed.

Syntax

```
public class CustomerBlacklistChangeEvent
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CustomerBlacklistChangeEvent() constructor

Syntax

```
CustomerBlacklistChangeEvent ()
```

CustomerBlacklistChangeEvent(EventData) constructor

Syntax

```
CustomerBlacklistChangeEvent ( EventData data )
```

getNewBlacklistReason() method

Syntax

```
int getNewBlacklistReason ()
```

getOldBlacklistReason() method

Syntax

```
int getOldBlacklistReason ()
```

getTimestamp() method

Syntax

```
long getTimestamp ()
```

isTransient() method

Syntax

```
boolean isTransient ()
```

setNewBlacklistReason(int) method

Syntax

```
void setNewBlacklistReason ( int blr )
```

setOldBlacklistReason(int) method

Syntax

```
void setOldBlacklistReason ( int blr )
```

setTimestamp(long) method

Syntax

```
void setTimestamp ( long timestamp )
```

CustomerDeactivatedEvent class

This CustomerDeactivatedEvent is fired when a customer is deactivated.

Syntax

```
public class CustomerDeactivatedEvent
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CustomerDeactivatedEvent() constructor

Syntax

```
CustomerDeactivatedEvent ()
```

CustomerDeactivatedEvent(EventData) constructor

Syntax

```
CustomerDeactivatedEvent ( EventData data )
```

getCancellationReason() method

Syntax

```
int getCancellationReason ()
```

getTimestamp() method

Syntax

```
long getTimestamp ()
```

isObfuscationNeeded() method

Syntax

```
boolean isObfuscationNeeded ()
```

isTransient() method

Syntax

```
boolean isTransient ()
```

setCancellationReason(int) method

Syntax

```
void setCancellationReason ( int cancellationReason )
```

setObfuscationNeeded(boolean) method

Syntax

```
void setObfuscationNeeded ( boolean obfuscationNeeded )
```

setTimestamp(long) method

Syntax

```
void setTimestamp ( long timestamp )
```

CustomerEvent class

This abstract `CustomerEvent` can be used as the basis for customer related events.

Syntax

```
public class CustomerEvent
```

Derived classes

- *CredentialChangeEvent* on page 644
- *CustomerBlacklistChangeEvent* on page 646
- *CustomerDeactivatedEvent* on page 647
- *CustomerObfuscatedEvent* on page 650
- *CustomerTemporarilyBlockedEvent* on page 651
- *RegistrationCompleteEvent* on page 652
- *WrongCredentialEvent* on page 653

Usage

It already provides access to the customer id and type.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

*CustomerEvent() constructor***Syntax**

```
CustomerEvent ()
```

*CustomerEvent(EventData) constructor***Syntax**

```
CustomerEvent ( EventData data )
```

*CustomerEvent(String, EventData) constructor***Syntax**

```
CustomerEvent ( String name , EventData data )
```

*getCustomerId() method***Syntax**

```
long getCustomerId ()
```

*getCustomerType() method***Syntax**

```
int getCustomerType ()
```

init() method

Syntax

```
void init ()
```

setCustomerId(long) method

Syntax

```
void setCustomerId ( long customerId )
```

setCustomerType(int) method

Syntax

```
void setCustomerType ( int customerType )
```

CustomerObfuscatedEvent class

This abstract `CustomerObfuscatedEvent` is fired when a customer has been obfuscated.

Syntax

```
public class CustomerObfuscatedEvent
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CustomerObfuscatedEvent() constructor

Syntax

```
CustomerObfuscatedEvent ()
```

CustomerObfuscatedEvent(EventData) constructor

Syntax

```
CustomerObfuscatedEvent ( EventData data )
```

getObfuscationMessage() method

Syntax

```
String getObfuscationMessage ()
```

getTimestamp() method

Syntax

long getTimestamp ()

isTransient() method

Syntax

boolean isTransient ()

setObfuscationMessage(String) method

Syntax

void setObfuscationMessage (String *obfuscationMessage*)

setTimestamp(long) method

Syntax

void setTimestamp (long *timestamp*)

CustomerTemporarilyBlockedEvent class

This CustomerTemporarilyBlockedEvent is fired when a customer's credential's have been wrongly presented more than the maximum number of times for their security policy AND the policy is set to block customer access for a specified time interval.

Syntax

public class CustomerTemporarilyBlockedEvent

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CustomerTemporarilyBlockedEvent() constructor

Syntax

CustomerTemporarilyBlockedEvent ()

CustomerTemporarilyBlockedEvent(EventData) constructor

Syntax

`CustomerTemporarilyBlockedEvent ( EventData data )`

getBlockedUntil() method

Syntax

`long getBlockedUntil ()`

isTransient() method

Syntax

`boolean isTransient ()`

setBlockedUntil(long) method

Syntax

`void setBlockedUntil ( long blockedUntil )`

RegistrationCompleteEvent class

This `RegistrationCompleteEvent` is fired when a new customer was registered completely and its BLR changed from preregistered - 5 to none - 0.

Syntax

`public class RegistrationCompleteEvent`

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

RegistrationCompleteEvent() constructor

Syntax

`RegistrationCompleteEvent ()`

RegistrationCompleteEvent(EventData) constructor

Syntax

`RegistrationCompleteEvent ( EventData data )`

WrongCredentialEvent class

This `WrongCredentialEvent` is fired when a customer's account has been access with an incorrect credential.

Syntax

```
public class WrongCredentialEvent
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

WrongCredentialEvent() constructor**Syntax**

```
WrongCredentialEvent ()
```

WrongCredentialEvent(EventData) constructor**Syntax**

```
WrongCredentialEvent ( EventData data )
```

getAttemptMax() method**Syntax**

```
long getAttemptMax ()
```

getAttemptNo() method**Syntax**

```
long getAttemptNo ()
```

getCredentialStatus() method**Syntax**

```
int getCredentialStatus ()
```

getCredentialType() method**Syntax**

```
int getCredentialType ()
```

getTimestamp() method

Syntax

long getTimestamp ()

isTransient() method

Syntax

boolean isTransient ()

setAttemptMax(int) method

Syntax

void setAttemptMax (int *attemptMax*)

setAttemptNo(int) method

Syntax

void setAttemptNo (int *attemptNo*)

setCredentialStatus(int) method

Syntax

void setCredentialStatus (int *credStatus*)

setCredentialType(int) method

Syntax

void setCredentialType (int *credType*)

setTimestamp(long) method

Syntax

void setTimestamp (long *timestamp*)

transaction package

Members

All public members of the transaction package.

- **ClearingEvent class** – ClearingEvents are generated by the AMS clearing process, once the clearing is finished.
- **InvoiceCreatedEvent class** – InvoiceCreatedEvent are created when a new Invoice was created.
- **TransactionEvent class** – TransactionEvents are created at the end of the transaction processing indicating that the status and error code of a transaction has changed or the transaction has been created.

ClearingEvent class

ClearingEvents are generated by the AMS clearing process, once the clearing is finished.

Syntax

```
public class ClearingEvent
```

Usage

All newly created payments will have the batch run id as indicated in this event but it is probably safer to only look at the payment status (but in this case multiple parallel events might be triggered).

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

ClearingEvent() constructor

Syntax

```
ClearingEvent ()
```

ClearingEvent(EventData) constructor

Syntax

```
ClearingEvent ( EventData data )
```

getBatchRunIds() method

Syntax

```
List< Long > getBatchRunIds ()
```

init() method

Syntax

```
void init ()
```

setBatchRunIds(List< Long >) method

Syntax

```
void setBatchRunIds ( List< Long > ids )
```

EVENT_TYPE variable

Syntax

```
final String EVENT_TYPE
```

InvoiceCreatedEvent class

InvoiceCreatedEvent are created when a new Invoice was created.

Syntax

```
public class InvoiceCreatedEvent
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

InvoiceCreatedEvent() constructor

Syntax

```
InvoiceCreatedEvent ()
```

InvoiceCreatedEvent(EventData) constructor

Syntax

```
InvoiceCreatedEvent ( EventData data )
```

getCustomerCanCreateInvoice() method

Syntax

```
boolean getCustomerCanCreateInvoice ()
```

getInvoiceId() method

Syntax

```
long getInvoiceId ()
```

getInvoiceTypeGroupId() method

Syntax

```
int getInvoiceTypeGroupId ()
```

init() method

Syntax

```
void init ()
```

setCustomerCanCreateInvoice(boolean) method

Syntax

```
void setCustomerCanCreateInvoice ( boolean customerCanCreateInvoice )
```

setInvoiceId(long) method

Syntax

```
void setInvoiceId ( long invId )
```

setInvoiceTypeGroupId(int) method

Syntax

```
void setInvoiceTypeGroupId ( int groupId )
```

EVENT_TYPE variable

Syntax

```
final String EVENT_TYPE
```

TransactionEvent class

TransactionEvents are created at the end of the transaction processing indicating that the status and error code of a transaction has changed or the transaction has been created.

Syntax

```
public class TransactionEvent
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

TransactionEvent() constructor

Syntax

TransactionEvent ()

TransactionEvent(EventData) constructor

Syntax

TransactionEvent (EventData *data*)

getErrorCode() method

Syntax

int getErrorCode ()

getStatus() method

Syntax

int getStatus ()

getTransactionId() method

Syntax

long getTransactionId ()

init() method

Syntax

void init ()

setErrorCode(int) method

Syntax

void setErrorCode (int *errorCode*)

setStatus(int) method

Syntax

void setStatus (int *status*)

setTransactionId(long) method

Syntax

```
void setTransactionId ( long id )
```

EVENT_TYPE variable

Syntax

```
final String EVENT_TYPE
```

tasks package

Members

All public members of the tasks package.

- **util package** –

util package

Members

All public members of the util package.

- **AuthenticatingMobiliserTask< C extends IMobiliserTaskConfiguration > class** –
- **IMobiliserTaskConfiguration interface** – The IMobiliserTaskConfiguration
- **MobiliserTask< C extends IMobiliserTaskConfiguration > class** –
- **MobiliserTaskConfiguration class** – The MobiliserTaskConfiguration

AuthenticatingMobiliserTask< C extends IMobiliserTaskConfiguration > class

Syntax

```
public class AuthenticatingMobiliserTask< C extends  
IMobiliserTaskConfiguration >
```

Usage

The AuthenticatingMobiliserTask is the base class for all Mobiliser task which need to do internal service calls with an authenticated user. This super class will handler setting up the spring security context so that does not need to be repeated for every task.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

afterPropertiesSet() method

Syntax

`final void afterPropertiesSet () throws Exception`

doAfterPropertiesSet() method

Syntax

`abstract void doAfterPropertiesSet ()`

doProcess(final Event) method

Hook for subclasses to actually process the event.

Syntax

`abstract boolean doProcess ( final Event e )`

Parameters

- **e** – the event

Returns

result

Usage

result

process(final Event) method

Syntax

`final boolean process ( final Event e )`

setAuthenticatorFactory(final InternalServiceCallAuthenticatorFactory) method

Syntax

`void setAuthenticatorFactory (final
InternalServiceCallAuthenticatorFactory authenticatorFactory)`

Parameters

- **authenticatorFactory** – the authenticatorFactory to set

logger variable

Syntax

```
final Logger logger
```

IMobiliserTaskConfiguration interface

The IMobiliserTaskConfiguration

Syntax

```
public interface IMobiliserTaskConfiguration
```

Derived classes

- *MobiliserTaskConfiguration* on page 663

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCronPattern() method

Returns the cron pattern for this task.

Syntax

```
String getCronPattern ()
```

Returns

the cron pattern for this task

Usage

the cron pattern for this task

MobiliserTask< C extends IMobiliserTaskConfiguration > class

Syntax

```
public class MobiliserTask< C extends  
IMobiliserTaskConfiguration >
```

Usage

The `MobiliserTask` is the base class for all Mobiliser task implementations. To register a task with the events engine, please do not export it directly into the OSGi registry, but rather use this snippet

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

```
<!-- use this dynamic exporter to refresh based on prefs, since we
have
    no other way to change the cron interval -->
    <bean id="osgiExporter"

class="com.sybase365.mobiliser.util.tools.spring.DynamicOsgiService
FactoryBean">
    <property name="target" ref="expireInitialTransaction" />
    <property name="interfaces">
        <list>

<value>com.sybase365.mobiliser.framework.task.model.TaskHandler

        </list>
    </property>
</bean>

    <prefs:node id="node"

class="com.sybase365.mobiliser.money.jobs.tasks.expireinitial.Expir
eInitialTransaction"
    refreshable-beans="osgiExporter" />
```

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

afterPropertiesSet() method

Syntax

void afterPropertiesSet () throws Exception

getConfiguration() method

Syntax

final C getConfiguration ()

getCronExpr() method

Syntax

String getCronExpr ()

getHandlerName() method

Syntax

String getHandlerName ()

getTaskData() method

Syntax

HashMap< String, String > getTaskData ()

setConfiguration(final C) method

Syntax

final void setConfiguration (final C *configuration*)

MobiliserTaskConfiguration class

The MobiliserTaskConfiguration

Syntax

public class MobiliserTaskConfiguration

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCallerId() method

Syntax

long getCallerId ()

getCronPattern() method

Syntax

String getCronPattern ()

DEFAULT_CALLER_ID variable

The default value for the caller ID.

Syntax

final long DEFAULT_CALLER_ID

PREFS_CALLER_ID variable

The preferences key for the caller ID.

Syntax

```
final String PREFS_CALLER_ID
```

PREFS_CRONPATTERN variable

Syntax

```
final String PREFS_CRONPATTERN
```

odc package

Members

All public members of the odc package.

- **businesslogic package** – 2013 SAP AG or an SAP affiliate company.
- **dao package** –

businesslogic package

2013 SAP AG or an SAP affiliate company.

Members

All public members of the businesslogic package.

- **IOdcConstant interface** – This interface contains all the defined operation types for Odc Implementation.
- **IOdcLogicService interface** – This interface contains all the method declarations defining the business logic implementation for ODC.

Usage

All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

IOdcConstant interface

This interface contains all the defined operation types for Odc Implementation.

Syntax

```
public interface IOdcConstant
```

NotifyMode() enumeration

enum defined for Notification Mode all values are as per DB table

MOB_IDENTIFICATION_TYPES

Enum Constant Summary

- **SMS** –
- **GCM** –
- **IOS_PN** –

*FUND_OUT_USE_CASE variable**Syntax*

```
int FUND_OUT_USE_CASE
```

*ID_TYPE_CUSTOMER_ID variable**Syntax*

```
int ID_TYPE_CUSTOMER_ID
```

*OFFLINE_OPERATION_STATUS_FAILURE variable**Syntax*

```
Character OFFLINE_OPERATION_STATUS_FAILURE
```

*OFFLINE_OPERATION_STATUS_PENDING variable**Syntax*

```
Character OFFLINE_OPERATION_STATUS_PENDING
```

*OFFLINE_OPERATION_STATUS_SETTLEMENT_FAILED variable**Syntax*

```
Character OFFLINE_OPERATION_STATUS_SETTLEMENT_FAILED
```

*OFFLINE_OPERATION_STATUS_SUCCESS variable**Syntax*

```
Character OFFLINE_OPERATION_STATUS_SUCCESS
```

*OFFLINE_OPERATION_TYPE_CANCEL_SUBSCRIPTION variable**Syntax*

```
int OFFLINE_OPERATION_TYPE_CANCEL_SUBSCRIPTION
```

OFFLINE_OPERATION_TYPE_CREATE_MERCHANT_SUBSCRIPTION variable

Syntax

int OFFLINE_OPERATION_TYPE_CREATE_MERCHANT_SUBSCRIPTION

OFFLINE_OPERATION_TYPE_MER_CREATE variable

Syntax

int OFFLINE_OPERATION_TYPE_MER_CREATE

OFFLINE_OPERATION_TYPE_MER_PIN_RESET variable

Syntax

int OFFLINE_OPERATION_TYPE_MER_PIN_RESET

OFFLINE_OPERATION_TYPE_MER_REMOVE variable

Syntax

int OFFLINE_OPERATION_TYPE_MER_REMOVE

OFFLINE_OPERATION_TYPE_MER_UPDATE variable

Syntax

int OFFLINE_OPERATION_TYPE_MER_UPDATE

OFFLINE_OPERATION_TYPE_SVA_CREATE variable

Syntax

int OFFLINE_OPERATION_TYPE_SVA_CREATE

OFFLINE_OPERATION_TYPE_SVA_CREDIT variable

Syntax

int OFFLINE_OPERATION_TYPE_SVA_CREDIT

OFFLINE_OPERATION_TYPE_SVA_DEACTIVATE variable

Syntax

int OFFLINE_OPERATION_TYPE_SVA_DEACTIVATE

OFFLINE_OPERATION_TYPE_SVA_DEBIT variable*Syntax*

```
int OFFLINE_OPERATION_TYPE_SVA_DEBIT
```

OFFLINE_OPERATION_TYPE_SVA_INQUIRY variable*Syntax*

```
int OFFLINE_OPERATION_TYPE_SVA_INQUIRY
```

OFFLINE_OPERATION_TYPE_SVA_UPDATE variable*Syntax*

```
int OFFLINE_OPERATION_TYPE_SVA_UPDATE
```

OFFLINE_OPERATION_TYPE_SYNC variable*Syntax*

```
int OFFLINE_OPERATION_TYPE_SYNC
```

OFFLINE_OPERATION_TYPE_SYNC_BALACE variable*Syntax*

```
int OFFLINE_OPERATION_TYPE_SYNC_BALACE
```

OFFLINE_OPERATION_TYPE_SYNC_TOKEN variable*Syntax*

```
int OFFLINE_OPERATION_TYPE_SYNC_TOKEN
```

OFFLINE_SVA_PI_TYPE variable*Syntax*

```
int OFFLINE_SVA_PI_TYPE
```

OFFLINE_SVA_STATUS_ACTIVE variable*Syntax*

```
int OFFLINE_SVA_STATUS_ACTIVE
```

OFFLINE_SVA_STATUS_BLOCKED variable

Syntax

int OFFLINE_SVA_STATUS_BLOCKED

OFFLINE_SVA_STATUS_CLOSED variable

Syntax

int OFFLINE_SVA_STATUS_CLOSED

OFFLINE_SVA_STATUS_LOCKED variable

Syntax

int OFFLINE_SVA_STATUS_LOCKED

OFFLINE_SVA_STATUS_PENDING variable

Syntax

int OFFLINE_SVA_STATUS_PENDING

STR_IMEI variable

Syntax

String STR_IMEI

STR_LOWER_CASE_SMS variable

Syntax

String STR_LOWER_CASE_SMS

STR_MSISDN variable

Syntax

String STR_MSISDN

STR_SMS variable

Syntax

String STR_SMS

IOdcLogicService interface

This interface contains all the method declarations defining the business logic implementation for ODC.

Syntax

```
public interface IOdcLogicService
```

createOdcMerchantRequest(long) method**Syntax**

```
CreateOdcMerchantCatalogRequest createOdcMerchantRequest ( long  
merchantId )
```

Parameters

- **merchantId** –

***createOfflineSVA(PaymentInstrument,
com.sybase365.mobiliser.money.odc.dao.model.OfflineSVA, long) method*****Syntax**

```
long createOfflineSVA ( PaymentInstrument paymentInstrument ,  
com.sybase365.mobiliser.money.odc.dao.model.OfflineSVA bean ,  
long callerId )
```

Parameters

- **paymentInstrument** –
- **bean** –
- **callerId** –

deleteOfflineSVA(long, CancelOdcSvaResponse, long) method**Syntax**

```
void deleteOfflineSVA ( long id , CancelOdcSvaResponse response ,  
long callerId )
```

Parameters

- **id** –
- **response** –
- **callerId** –

getAllCustomerSubscriptions(long) method

Syntax

```
List< ISubscription > getAllCustomerSubscriptions ( long  
customerId )
```

Parameters

- **customerId** –

getAllOfflineSVA(long) method

Syntax

```
List< OfflineSVA > getAllOfflineSVA ( long customerId )
```

Parameters

- **customerId** –

getFindOdcCustomerTokenRequest(SynchronizeCustomerRequest) method

Syntax

```
FindOdcCustomerTokenRequest getFindOdcCustomerTokenRequest  
( SynchronizeCustomerRequest request )
```

Parameters

- **request** –

getMaintainOdcMerPINRequest(long, long) method

Syntax

```
MaintainOdcMerPINRequest getMaintainOdcMerPINRequest ( long  
customerId , long callerId )
```

Parameters

- **customerId** –
- **callerId** –

getMaintainOdcMerProfileRequestData(DeployMERRequest) method

Syntax

```
MaintainOdcMerProfileRequest  
getMaintainOdcMerProfileRequestData ( DeployMERRequest request )
```

Parameters

- **request** –

getMobiliserSubscriptionIdentificationValue(long) method

Syntax

```
String getMobiliserSubscriptionIdentificationValue ( long  
customerId )
```

Parameters

- **customerId** –

getOfflineSVA(long, long) method

Syntax

```
OfflineSVA getOfflineSVA ( long id , long callerId )
```

Parameters

- **id** –
- **callerId** –

getSyncHistory(long, Date, Date, Integer, long) method

Syntax

```
List<  
com.sybase365.mobiliser.money.odc.dao.model.OfflineOperation  
> getSyncHistory ( long customerId , Date fromDate , Date toDate ,  
Integer offlineOperation , long callerId )
```

Parameters

- **customerId** –
- **fromDate** –

- **toDate** –
- **offlineOperation** –
- **callerId** –

getSyncTokensByCustomer(long, long) method

Syntax

```
java.util.List<  
com.sybase365.mobiliser.money.odc.dao.model.SyncToken >  
getSyncTokensByCustomer ( long customerId , long callerId )
```

Parameters

- **customerId** –
- **callerId** –

resetMERPIN(String, long, MaintainOdcMerPINResponse, long) method

Syntax

```
void resetMERPIN ( String newpin , long customerId ,  
MaintainOdcMerPINResponse response , long callerId )
```

Parameters

- **newpin** –
- **customerId** –
- **response** –
- **callerId** –

setCancelOdcSvaRequestData(long, long, CancelOdcSvaRequest) method

Syntax

```
void setCancelOdcSvaRequestData ( long paymentInstrumentId , long  
callerId , CancelOdcSvaRequest odcRequest )
```

Parameters

- **paymentInstrumentId** –
- **callerId** –
- **odcRequest** –

setDeployOdcMerOnMobileRequestData(DeployOdcMerOnMobileRequest, DeployMERRequest) method

Syntax

```
void setDeployOdcMerOnMobileRequestData  
( DeployOdcMerOnMobileRequest odcrequest , DeployMERRequest  
request )
```

Parameters

- **odcrequest** –
- **request** –

updateForPerformedOperation(MobiliserResponseType, long, long) method

Syntax

```
void updateForPerformedOperation ( MobiliserResponseType  
response , long customerId , long callerId )
```

Parameters

- **response** –
- **customerId** –
- **callerId** –

updateOfflineSVA(PaymentInstrument, com.sybase365.mobiliser.money.odc.dao.model.OfflineSVA, long) method

Syntax

```
void updateOfflineSVA ( PaymentInstrument paymentInstrument ,  
com.sybase365.mobiliser.money.odc.dao.model.OfflineSVA bean ,  
long callerId )
```

Parameters

- **paymentInstrument** –
- **bean** –
- **callerId** –

updateOpsForUpdateOfflineSVAOperation(MobiliserResponseType, long, long)
method

Syntax

```
void updateOpsForUpdateOfflineSVAOperation  
( MobiliserResponseType response , long customerId , long callerId )
```

Parameters

- **response** –
- **customerId** –
- **callerId** –

dao package

Members

All public members of the dao package.

- **api package** –
- **model package** – 2013 SAP AG or an SAP affiliate company.

api package

Members

All public members of the api package.

- **OfflineOperationDao interface** – 2013 SAP AG or an SAP affiliate company.
- **OfflineOperationTypeDao interface** – 2013 SAP AG or an SAP affiliate company.
- **OfflineSVADao interface** – 2013 SAP AG or an SAP affiliate company.
- **OfflineSVAStatusDao interface** – 2013 SAP AG or an SAP affiliate company.
- **SyncTokenDao interface** – 2013 SAP AG or an SAP affiliate company.

OfflineOperationDao interface

2013 SAP AG or an SAP affiliate company.

Syntax

```
public interface OfflineOperationDao
```

Usage

All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getByOdcOperation(long, int) method

Returns the OfflineOperation by envelopeId and partition id.

Syntax

```
OfflineOperation getByOdcOperation ( long envelopeID , int partitionID )
```

Parameters

- **envelopeID** –
- **partitionID** –

getByOdcOperationList(long, int) method

Gets the list of the offline operation for the given envelopeId and pertitionId.

Syntax

```
List< OfflineOperation > getByOdcOperationList ( long envelopeID , int partitionID )
```

Parameters

- **envelopeID** –
- **partitionID** –

getByTransaction(Transaction) method

Returns the Offline Operations by TransactionId.

Syntax

```
OfflineOperation getByTransaction ( Transaction txn )
```

Parameters

- **txn** –

getOperationHistory(Customer, Date, Date, OfflineOperationType) method

Gets the Offline Operations between given date, for a given operation type for a given customer.

Syntax

```
List< OfflineOperation > getOperationHistory ( Customer customerId , Date fromDate , Date toDate , OfflineOperationType offlineOperation )
```

Parameters

- **customerId** –
- **fromDate** –
- **toDate** –
- **offlineOperation** –

OfflineOperationTypeDao interface

2013 SAP AG or an SAP affiliate company.

Syntax

```
public interface OfflineOperationTypeDao
```

Usage

All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

OfflineSVADao interface

2013 SAP AG or an SAP affiliate company.

Syntax

```
public interface OfflineSVADao
```

Usage

All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

findOfflineSVAs(long) method

This method finds all the OfflineSVAs for the given customer.

Syntax

```
List< OfflineSVA > findOfflineSVAs ( long customerId )
```

Parameters

- **customerId** –

```
getOfflineSVA(@MobiliserParam(name="customerId", required=true,  
description="the customer ID for the search") Customer,  
@MobiliserParam(name="type", required=true, description="the
```


PaymentInstrument Type") Type, @MobiliserParam(name="merchantId", required=true, description="the Merchant ID in the given Offline SVA") Long) method
 This method returns the OfflineSVA based on the customer.

Syntax

```
OfflineSVA getOfflineSVA ( @MobiliserParam(name="customerId",
required=true, description="the customer ID for the search")
Customer customerId, @MobiliserParam(name="type", required=true,
description="the PaymentInstrument Type") Type type ,
@MobiliserParam(name="merchantId", required=true,
description="the Merchant ID in the given Offline SVA") Long
merchantId )
```

Parameters

- **customerId** –
- **type** –
- **merchantId** –

Usage

merchant and type

```
save(@MobiliserParam(name="paymentInstrument", required=true,
description="The Payment Instrument to be Created") PaymentInstrument,
@MobiliserParam(name="offlineSva", required=true, description="The Offline SVA
account to be created") OfflineSVA, @MobiliserParam(name="creator",
required=true, description="Id Of the Creator") Long) method
```

This method saves the OfflineSVA.

Syntax

```
void save ( @MobiliserParam(name="paymentInstrument",
required=true, description="The Payment Instrument to be
Created") PaymentInstrument paymentInstrument ,
@MobiliserParam(name="offlineSva", required=true,
description="The Offline SVA account to be created")
OfflineSVA offlineSva , @MobiliserParam(name="creator",
required=true, description="Id Of the Creator") Long creator )
```

Parameters

- **paymentInstrument** –

- **offlineSva** –
- **creator** –

OfflineSVAStatusDao interface

2013 SAP AG or an SAP affiliate company.

Syntax

```
public interface OfflineSVAStatusDao
```

Usage

All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

SyncTokenDao interface

2013 SAP AG or an SAP affiliate company.

Syntax

```
public interface SyncTokenDao
```

Usage

All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getSyncTokenByCustomer(@MobiliserParam(name="customerId", required=true, description="the customer ID for the search") long) method

This method returns all the sync tokens by the customer.

Syntax

```
List< SyncToken > getSyncTokenByCustomer  
( @MobiliserParam(name="customerId", required=true,  
description="the customer ID for the search") long customerId )
```

Parameters

- **customerId** –

model package

2013 SAP AG or an SAP affiliate company.

Members

All public members of the model package.

- **OfflineOperation** class –
- **OfflineOperationType** class –
- **OfflineSVA** class –
- **OfflineSVAStatus** class –
- **SyncToken** class –

Usage

All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

OfflineOperation class

Syntax

```
public class OfflineOperation
```

OfflineOperation() constructor

Syntax

```
OfflineOperation ()
```

getCustomer() method

Syntax

```
Customer getCustomer ()
```

Returns

the customerId

Usage

the customerId

getDateCreation() method

Syntax

```
Date getDateCreation ()
```

Returns

dateCreation

Usage

dateCreation

getEnvelopeId() method

Syntax

Long getEnvelopeId ()

Returns

envelopeId

Usage

envelopeId

getMerchant() method

Syntax

Customer getMerchant ()

Returns

merchant

Usage

merchant

getOfflineOperationType() method

Syntax

OfflineOperationType getOfflineOperationType ()

Returns

the offlineOperationType

Usage

the offlineOperationType

getOpStatus() method

Syntax

Character getOpStatus ()

Returns

the opStatus

Usage

the opStatus

getPartitionId() method

Syntax

Integer getPartitionId ()

Returns

partitionId

Usage

partitionId

getTxn() method

Syntax

Transaction getTxn ()

Returns

the txn

Usage

the txn

isSetCustomer() method

Syntax

boolean isSetCustomer ()

Returns

false if the variable is not set

Usage

false if the variable is not set

isSetDateCreation() method

Syntax

boolean isSetDateCreation ()

Returns

dateCreation

Usage

dateCreation

isSetEnvelopeId(Long) method

Checks if the envelopeId is set.

Syntax

boolean isSetEnvelopeId (Long *envelopeId*)

Parameters

- **envelopeId** –

isSetMerchant() method

Syntax

boolean isSetMerchant ()

Returns

merchant

Usage

merchant

isSetOfflineOperationType() method

Syntax

boolean isSetOfflineOperationType ()

Returns

false if the variable is not set

Usage

false if the variable is not set

isSetOpStatus() method

Syntax

```
boolean isSetOpStatus ()
```

Returns

false if the variable is not set

Usage

false if the variable is not set

isSetPartitionId() method

Checks if the PartitionId is set.

Syntax

```
boolean isSetPartitionId ()
```

isTxn() method

Syntax

```
boolean isTxn ()
```

Returns

false if the variable is not set

Usage

false if the variable is not set

setCustomer(Customer) method

Syntax

```
void setCustomer ( Customer customerId )
```

Parameters

- **customerId** – the customerId to set

setDateCreation(Date) method

Syntax

```
void setDateCreation ( Date dateCreation )
```

Parameters

- **dateCreation** –

setEnvelopeId(Long) method

Syntax

void setEnvelopeId (Long *envelopeId*)

Parameters

- **envelopeId** –

setMerchant(Customer) method

Syntax

void setMerchant (Customer *merchant*)

Parameters

- **merchant** –

setOfflineOperationType(OfflineOperationType) method

Syntax

void setOfflineOperationType (OfflineOperationType
offlineOperationType)

Parameters

- **offlineOperationType** – the offlineOperationType to set

setOpStatus(Character) method

Syntax

void setOpStatus (Character *opStatus*)

Parameters

- **opStatus** – the opStatus to set

setPartitionId(Integer) method

Syntax

void setPartitionId (Integer *partitionId*)

Parameters

- **partitionId** –

setTxn(Transaction) method

Syntax

```
void setTxn ( Transaction txn )
```

Parameters

- **txn** – the txn to set

OfflineOperationType class**Syntax**

```
public class OfflineOperationType
```

OfflineOperationType() constructor

Constructor.

Syntax

```
OfflineOperationType ()
```

OfflineOperationType(int) constructor**Syntax**

```
OfflineOperationType ( int id )
```

OfflineSVA class**Syntax**

```
public class OfflineSVA
```

OfflineSVA() constructor**Syntax**

```
OfflineSVA ()
```

Usage

Creates a new instance of

```
OfflineSVA
```

.

OfflineSVA(long) constructor

Syntax

OfflineSVA (long *id*)

Parameters

- **id** – the PK value

Usage

Creates a new instance of

OfflineSVA

with a preset primary key.

getBalanceAmount() method

Syntax

long getBalanceAmount ()

Returns

the balanceAmount

Usage

the balanceAmount

getDescription() method

Syntax

String getDescription ()

Returns

the description

Usage

the description

getLastSyncDate() method

Syntax

Date getLastSyncDate ()

Returns

the lastSyncDate

Usage

the lastSyncDate

getMerchant() method

Syntax

Long getMerchant ()

Returns

the merchant

Usage

the merchant

getOfflineSVAStatus() method

Syntax

Long getOfflineSVAStatus ()

Returns

the offlineSVAStatus

Usage

the offlineSVAStatus

getSvaCode() method

Syntax

String getSvaCode ()

Returns

the svaCode

Usage

the svaCode

isDefault() method

Syntax

```
boolean isDefault ()
```

Returns

the dbDefault

Usage

the dbDefault

isSetBalanceAmount() method

Syntax

```
boolean isSetBalanceAmount ()
```

Returns

false if the variable is not set

Usage

false if the variable is not set

isSetDbDefault() method

Syntax

```
boolean isSetDbDefault ()
```

Returns

false if the variable is not set

Usage

false if the variable is not set

isSetDbShared() method

Syntax

```
boolean isSetDbShared ()
```

Returns

false if the variable is not set

Usage

false if the variable is not set

isSetDescription() method

Syntax

```
boolean isSetDescription ()
```

Returns

false if the variable is not set

Usage

false if the variable is not set

isSetLastSyncDate() method

Syntax

```
boolean isSetLastSyncDate ()
```

Returns

false if the variable is not set

Usage

false if the variable is not set

isSetMerchant() method

Syntax

```
boolean isSetMerchant ()
```

Returns

false if the variable is not set

Usage

false if the variable is not set

isSetOfflineSVAStatus() method

Syntax

boolean isSetOfflineSVAStatus ()

Returns

false if the variable is not set

Usage

false if the variable is not set

isSetSvaCode() method

Syntax

boolean isSetSvaCode ()

Returns

false if the variable is not set

Usage

false if the variable is not set

isShared() method

Syntax

boolean isShared ()

Returns

the dbShared

Usage

the dbShared

setBalanceAmount(long) method

Syntax

void setBalanceAmount (long *balanceAmount*)

Parameters

- **balanceAmount** – the balanceAmount to set

setDefault(boolean) method

Syntax

`void setDefault ( boolean defaults )`

Parameters

- **default** – the dbDefault to set

setDescription(String) method

Syntax

`void setDescription ( String description )`

Parameters

- **description** – the description to set

setLastSyncDate(Date) method

Syntax

`void setLastSyncDate ( Date lastSyncDate )`

Parameters

- **lastSyncDate** – the lastSyncDate to set

setMerchant(Long) method

Syntax

`void setMerchant ( Long merchant )`

Parameters

- **merchant** – the merchant to set

setOfflineSVAStatus(Long) method

Syntax

`void setOfflineSVAStatus ( Long offlineSVAStatus )`

Parameters

- **offlineSVASStatus** – the offlineSVASStatus to set

setShared(boolean) method

Syntax

```
void setShared ( boolean shared )
```

Parameters

- **sharedthe** – dbShared to set

setSvaCode(String) method

Syntax

```
void setSvaCode ( String svaCode )
```

Parameters

- **svaCode** – the svaCode to set

OfflineSVASStatus class

Syntax

```
public class OfflineSVASStatus
```

OfflineSVASStatus() constructor

Syntax

```
OfflineSVASStatus ()
```

OfflineSVASStatus(int) constructor

Syntax

```
OfflineSVASStatus ( int id )
```

SyncToken class

Syntax

```
public class SyncToken
```


SyncToken() constructor

Constructor.

Syntax

`SyncToken ()`

getCustomer() method

Syntax

`Customer getCustomer ()`

Returns

the customer

Usage

the customer

getEnvelopeId() method

Syntax

`long getEnvelopeId ()`

Returns

the envelopeId

Usage

the envelopeId

getOfflineSVA() method

Syntax

`long getOfflineSVA ()`

Returns

the offlineSVA

Usage

the offlineSVA

getPayeeId() method

Syntax

long getPayeeId ()

Returns

the payeeId

Usage

the payeeId

getPayerId() method

Syntax

long getPayerId ()

Returns

the payerId

Usage

the payerId

getSubscriptionId() method

Syntax

String getSubscriptionId ()

Returns

the subscriptionId

Usage

the subscriptionId

getTokenAmt() method

Syntax

Long getTokenAmt ()

Returns

the tokenAmt

Usage

the tokenAmt

getTokenCreationDate() method

Syntax

Date getTokenCreationDate ()

Returns

the tokenCreationDate

Usage

the tokenCreationDate

getTokenSignature() method

Syntax

String getTokenSignature ()

Returns

the tokenSignature

Usage

the tokenSignature

getUseCase() method

Syntax

short getUseCase ()

Returns

the useCase

Usage

the useCase

isSetCustomer() method

Syntax

boolean isSetCustomer ()

Returns

false if the variable is not set

Usage

false if the variable is not set

isSetEnvelopeId() method

Syntax

`boolean isSetEnvelopeId ()`

Returns

false if the variable is not set

Usage

false if the variable is not set

isSetOfflineSVA() method

Syntax

`boolean isSetOfflineSVA ()`

Returns

false if the variable is not set

Usage

false if the variable is not set

isSetPayeeId() method

Syntax

`boolean isSetPayeeId ()`

Returns

false if the variable is not set

Usage

false if the variable is not set

isSetPayerId() method

Syntax

`boolean isSetPayerId ()`

Returns

false if the variable is not set

Usage

false if the variable is not set

isSetSubscriptionId() method

Syntax

`boolean isSetSubscriptionId ()`

Returns

false if the variable is not set

Usage

false if the variable is not set

isSetTokenAmt() method

Syntax

`boolean isSetTokenAmt ()`

Returns

false if the variable is not set

Usage

false if the variable is not set

isSetTokenCreationDate() method

Syntax

`boolean isSetTokenCreationDate ()`

Returns

false if the variable is not set

Usage

false if the variable is not set

isSetTokenSignature() method

Syntax

```
boolean isSetTokenSignature ()
```

Returns

false if the variable is not set

Usage

false if the variable is not set

isSetUseCase() method

Syntax

```
boolean isSetUseCase ()
```

Returns

false if the variable is not set

Usage

false if the variable is not set

setCustomer(Customer) method

Syntax

```
void setCustomer ( Customer customer )
```

Parameters

- **customer** – the customer to set

setEnvelopeId(long) method

Syntax

```
void setEnvelopeId ( long envelopeId )
```

Parameters

- **envelopeId** – the envelopeId to set

setOfflineSVA(long) method

Syntax

void setOfflineSVA (long *offlineSVA*)

Parameters

- **offlineSVA** – the offlineSVA to set

setPayeeId(long) method

Syntax

void setPayeeId (long *payeeId*)

Parameters

- **payeeId** – the payeeId to set

setPayerId(long) method

Syntax

void setPayerId (long *payerId*)

Parameters

- **payerId** – the payerId to set

setSubscriptionId(String) method

Syntax

void setSubscriptionId (String *subscriptionId*)

Parameters

- **subscriptionId** – the subscriptionId to set

setTokenAmt(Long) method

Syntax

void setTokenAmt (Long *tokenAmt*)

Parameters

- **tokenAmt** – the tokenAmt to set

setTokenCreationDate(Date) method

Syntax

```
void setTokenCreationDate ( Date tokenCreationDate )
```

Parameters

- **tokenCreationDate** – the tokenCreationDate to set

setTokenSignature(String) method

Syntax

```
void setTokenSignature ( String tokenSignature )
```

Parameters

- **tokenSignature** – the tokenSignature to set

setUseCase(short) method

Syntax

```
void setUseCase ( short useCase )
```

Parameters

- **useCase** – the useCase to set

persistence package

Members

All public members of the persistence package.

- **dao package** –
- **model package** –

dao package

Members

All public members of the dao package.

- **api package –**

api package

Members

All public members of the api package.

- **customer package –**
- **job package –**
- **pi package –**
- **system package –**
- **transaction package –**
- **util package –**

customer package

Members

All public members of the customer package.

- **AddressDAO interface –**
- **AttachmentDAO interface –**
- **AuditLogDAO interface –**
- **CommissionConfigurationDAO interface –**
- **CredentialDAO interface –**
- **CustomerAttributeDAO interface –**
- **CustomerDAO interface –**
- **CustomerNetworkDAO interface –**
- **CustomerOtpDAO interface –**
- **CustomerPersistentLoginDAO interface –** Dao contract for CustomerPersistentLogins.
- **CustomerPrivilegeDAO interface –**
- **CustomerRoleDAO interface –**
- **CustomerSessionDAO interface –**
- **HistoryEntryDAO interface –**
- **IdentificationDAO interface –**
- **IdentityDAO interface –**
- **InvoiceAttributeDAO interface –**
- **InvoiceConfigurationAttributeDAO interface –**
- **InvoiceConfigurationDAO interface –**
- **InvoiceDAO interface –**
- **LimitSumDAO interface –**
- **NoteDAO interface –**

- **OtpDAO interface** –
- **WrkCustomerDAO interface** –

AddressDAO interface

Syntax

```
public interface AddressDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

deleteAddresses(final long, final int, final int) method

Delete all addresses for a customer with the given type and location type.

Syntax

```
int deleteAddresses ( final long customer ,   final int addressType ,  
final int locationType )
```

Parameters

- **customer** – customer id
- **addressType** – address type
- **locationType** – location type

Returns

the number of addresses deleted

Usage

the number of addresses deleted

findNextAddresses(BigDecimal, BigDecimal, Long, Integer, long, Integer) method

Find locations that are in an area defined by latitude +/- radius and longitude +/- radius.

Syntax

```
List< DAddress > findNextAddresses ( BigDecimal latitude ,  
BigDecimal longitude , Long customerId , Integer locationType , long radius ,  
Integer maxRecords )
```

Parameters

- **latitude** –
- **longitude** –
- **customerId** –
- **locationType** –
- **radius** –
- **maxRecords** –

getActiveCustomerAddressByType(long, int) method

Get a single active customer address by customer and address type.

Syntax

Address getActiveCustomerAddressByType (long *customerId* , int *addresstype*)

Parameters

- **customerId** –
- **addresstype** –

getCustomerAddresses(long, boolean) method

Get all customer addresses.

Syntax

List< Address > getCustomerAddresses (long *customerId* , boolean *returnAll*)

Parameters

- **customerId** –
- **returnAll** –

AttachmentDAO interface

Syntax

```
public interface AttachmentDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getAttachments(Long, Long, Integer, Integer) method

Get attachments based on filter.

Syntax

```
List< Attachment > getAttachments ( Long customerId , Long  
transactionId , Integer attachmentType , Integer status )
```

Parameters

- **customerId** – filter on customer
- **transactionId** – filter on transaction ID
- **attachmentType** – filter on type
- **status** – filter on attachment status

getAttachmentsByCustomer(long, Integer) method

Get attachments for a customer.

Syntax

```
List< Attachment > getAttachmentsByCustomer ( long customerId ,  
Integer attachmentType )
```

Parameters

- **customerId** – the customer filter
- **attachmentType** – optional filter on type

getAttachmentsByTransaction(long, Integer) method

Get attachments for a transaction.

Syntax

```
List< Attachment > getAttachmentsByTransaction ( long transactionId ,  
Integer attachmentType )
```

Parameters

- **transactionId** – the transaction filter
- **attachmentType** – optional filter on type

AuditLogDAO interface

Syntax

```
public interface AuditLogDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getAuditLogs(Long, Long) method

get all audit logs for a callerID and transaction ID.

Syntax

```
List< AuditLog > getAuditLogs ( Long customerId , Long transactionId )
```

Parameters

- **customerId** –

Usage

Do not use! please use AuditLogDAO#getAuditLogs(Date, Date, Long, Long, Long, Boolean, List, Integer)

getAuditLogs(Date, Date, Long, Long, Long, Boolean, List< String >, Integer) method

Retrieve a list of audit records based on a filter.

Syntax

```
List< AuditLog > getAuditLogs ( Date start , Date end , Long callerId ,  
Long customerId , Long transactionId , Boolean faultyOnly , List< String >  
category , Integer rows )
```

Parameters

- **start** –
- **end** –
- **callerId** – optional filter on caller
- **customerId** – optional filter on customer
- **transactionId** – optional filter on transaction
- **faultyOnly** – optional filter on calls with a result code not 0
- **category** – optional filter on a audit categories
- **rows** – optional maximum number of records. Default is 1000

getAuditLogsByCustomer(long) method
get all audit logs for a customer ID / callerID.

Syntax

```
List< AuditLog > getAuditLogsByCustomer ( long customerId )
```

Parameters

- **customerId** –

Usage

Do not use! please use AuditLogDAO#getAuditLogs(Date, Date, Long, Long, Long, Boolean, List, Integer)

getAuditLogsByTransaction(long) method
get all audit logs for a transaction ID.

Syntax

```
List< AuditLog > getAuditLogsByTransaction ( long transactionId )
```

Parameters

- **transactionId** –

Usage

Do not use! please use AuditLogDAO#getAuditLogs(Date, Date, Long, Long, Long, Boolean, List, Integer)

CommissionConfigurationDAO interface

Syntax

```
public interface CommissionConfigurationDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCommissionConfig(long) method

Get the commission configuration for a given customer.

Syntax

```
CommissionConfiguration getCommissionConfig ( long customerID )
```

Parameters

- **customerID** –

CredentialDAO interface

Syntax

```
public interface CredentialDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

findById(final long) method

Find all Credential object for the given customer.

Syntax

```
List< Credential > findById ( final long customerId )
```

Parameters

- **customerId** – the customer id

Returns

the list of credentials

Usage

the list of credentials

findByCustomerId(final long, Integer, boolean, int, Boolean) method

Find all Credential object for the given customer.

Syntax

```
List< Credential > findByCustomerId ( final long customerId ,  
Integer credentialType , boolean onlyActive , int count , Boolean  
oldestFirst )
```

Parameters

- **customerId** – the customer id
- **credentialType** – the credential type filter
- **count** – the number of credentials to be retrieved
- **oldestFirst** – the order in which the credentials are retrieved

Returns

the list of credentials

Usage

the list of credentials

CustomerAttributeDAO interface

Syntax

```
public interface CustomerAttributeDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getAttributesByCustomerId(long) method

Get all customer attributes for a customer.

Syntax

```
List< CustomerAttribute > getAttributesByCustomerId ( long  
customerId )
```


Parameters

- **customerId** –

CustomerDAO interface

Syntax

```
public interface CustomerDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

computeParentListJava(long, Integer, Integer) method

This method will retrieve all customers in the hierarchy of callingAgentId restricted by the min- and max-HierarchyLevel values.

Syntax

```
Map< Long, Integer > computeParentListJava ( long callingAgentId ,  
Integer minHierarchyLevel , Integer maxHierarchyLevel )
```

Parameters

- **callingAgentId** – the agentId the search is based on
- **minHierarchyLevel** –
- **maxHierarchyLevel** –

Returns

A map with customerIds (Key) and corresponding Hierarchylevel(Value)

Usage

A map with customerIds (Key) and corresponding Hierarchylevel(Value)

findCustomer(Address, Identification, Long, String, Integer, int) method

see findCustomer(Address, Identification, boolean, Long, String, Integer, int, String)

Syntax

```
List< Customer > findCustomer ( Address address , Identification  
identification , Long blacklistReason , String displayName , Integer  
customerType , int maxRecords )
```

findCustomer(Address, Identification, boolean, Long, String, Integer, int, String)
method

Finds Customers without caring about hierarchy Note that only a few fields of address and identification are considered in the search.

Syntax

```
List< Customer > findCustomer ( Address address , Identification  
identification , boolean anyIdOrgUnit , Long blacklistReason , String  
displayName , Integer customerType , int maxRecords , String orgUnit )
```

Parameters

- **address** – The address of the Customer to be found, pass null if address is to be ignored
- **identification** – The identification of the Customer to be found, pass null if identification is to be ignored
- **blacklistReason** – The blacklistReason of the customers which are to be found. Pass null if this is to be ignored
- **displayName** – The displayName of the customers which are to be found
- **customerType** – The customer type the customer to be found has to have
- **maxRecords** – The maximum number of records which is to be returned by the service

findCustomerIncludeIdentifications(Address, Identification, boolean, Long, String, Integer, int, String) *method*

This findCustomer method not only returns customer, but also a list of their identifications.

Syntax

```
Map< Customer, List< Identification > >  
findCustomerIncludeIdentifications ( Address address ,  
Identification identification , boolean anyIdOrgUnit , Long blacklistReason ,  
String displayName , Integer customerType , int maxRecords , String  
orgUnit )
```

Usage

The parameters are identical to the currently most elaborate findCustomer method

see

findCustomer(Address, Identification, boolean, Long, String, Integer, int, String)

findDeactivatedCustomer(int, Integer) method

Finds deactivated customers.

Syntax

```
List< Customer > findDeactivatedCustomer ( int customerType ,
Integer blacklistReason )
```

Parameters

- **customerType** – The customer type the customer to be found has to have
- **blacklistReason** – The blacklistReason of the customers which are to be found. Pass null if this is to be ignored

findGLAccountTree(String, List< Integer >, Boolean) method

Loads all the information needed to create the GLAccountTree.

Syntax

```
List< GLAccountTreeElementBean > findGLAccountTree ( String
orgUnit , List< Integer > Types , Boolean getActiveOnly )
```

Parameters

- **orgUnit** – Restriction, just Elements assigned to this Orgunit will be loaded (mandatory)
- **Types** – Restriction, just Elements assigned to this CustomerTypes will be loaded (mandatory)
- **getActiveOnly** – Restriction, if it's 'true', just active GLAccountTreeElements will be loaded. Otherwise both (optional)
- **orgUnit** –
- **Types** –
- **getActiveOnly** –

Returns

get root GLAccounts

Usage

get root GLAccounts

findHierarchicalCustomer(long, Address, Identification, Long, String, Integer, Integer, Integer, int) method

See { findHierarchicalCustomer(long, Address, Identification, boolean, Long, String, Integer, Integer, Integer, int, String).

Syntax

```
Map< Customer, Integer > findHierarchicalCustomer ( long  
callingAgentId , Address address , Identification identification , Long  
blacklistReason , String displayName , Integer customerType , Integer  
minHierarchyLevel , Integer maxHierarchyLevel , int maxRecords )
```

findHierarchicalCustomer(long, Address, Identification, boolean, Long, String, Integer, Integer, Integer, int, String) method

Find customers considering the hierarchy the customers are in.

Syntax

```
Map< Customer, Integer > findHierarchicalCustomer ( long  
callingAgentId , Address address , Identification identification , boolean  
anyIdOrgUnit , Long blacklistReason , String displayName , Integer  
customerType , Integer minHierarchyLevel , Integer maxHierarchyLevel , int  
maxRecords , String orgUnit )
```

Parameters

- **callingAgentId** – The Id of the Customer which is calling the service, considered as starting point of the hierarchy
- **address** – The address of the Customer to be found, pass null if address is to be ignored
- **identification** – The identification of the Customer to be found, pass null if identification is to be ignored
- **blacklistReason** – The blacklistReason of the customers which are to be found. Pass null if this is to be ignored
- **displayName** – The displayName of the customers which are to be found
- **customerType** – The customer type the customer to be found has to have
- **minHierarchyLevel** – The min hierarchy level to search for. Pass null to ignore
- **maxHierarchyLevel** – The max hierarchy level to search for. Pass null to ignore
- **maxRecords** – The maximum number of records which is to be returned by the service

Returns

A map containing Customers (Key) with the corresponding hierarchy level (Value)

Usage

If the min or max HierarchyLevel are to be ignored set the corresponding value to null.

A map containing Customers (Key) with the corresponding hierarchylevel(Value)

findHierarchicalCustomerIncludeIdentifications(long, Address, Identification, boolean, Long, String, Integer, Integer, Integer, int, String) method

Yet another findHierarchicalCustomer method, but additionally to findHierarchicalCustomer includes the customers identifications in the result list.

Syntax

```
List< ExtendedCustomerSearchResult >
findHierarchicalCustomerIncludeIdentifications ( long
callingAgentId , Address address , Identification identification , boolean
anyIdOrgUnit , Long blacklistReason , String displayName , Integer
customerType , Integer minHierarchyLevel , Integer maxHierarchyLevel , int
maxRecords , String orgUnit )
```

Usage

So the returned list contains entries of Object[] with the following information:

- element 0: the Customer object
- element 1: a List<Identification> object
- element 2: an Integer object representing the hierarchy level)

see

findHierarchicalCustomer(long, Address, Identification, boolean, Long, String, Integer, Integer, Integer, int, String)

getActiveCredential(long, int) method

Get the active credential of a given customer and credential type.

Syntax

```
Credential getActiveCredential ( long customerId , int credentialType )
```

Parameters

- **customerId** –
- **credentialType** –

getActiveCredential(final long, final int, final int) method

Get the active credential of a given customer, credential type and credential status.

Syntax

`Credential getActiveCredential ( final long customerId , final int credentialType , final int credentialStatus )`

Parameters

- **customerId** –
- **credentialType** –
- **credentialStatus** –

getActiveIdentity(long, int) method

Get the active identification for a customer.

Syntax

`Identity getActiveIdentity ( long customerId , int identityType )`

Parameters

- **customerId** –
- **identityType** –

getCustomersLastLoginFallsBetween(Date, Date) method

Get customer ids of failed logins.

Syntax

`List< Long > getCustomersLastLoginFallsBetween ( Date fromDate , Date toDate )`

Parameters

- **fromDate** –
- **toDate** –

getCustomerWithActiveIdentification(String, int) method

`CustomerDAO#getCustomerWithActiveIdentification(String, int, String)`

Syntax

`Customer getCustomerWithActiveIdentification ( String identifier , int identificationType )`

getCustomerWithActiveIdentification(String, int, String) method

get a customer by its identification and org unit

Syntax

```
Customer getCustomerWithActiveIdentification ( String identifier ,  
int identificationType , String orgUnit )
```

Parameters

- **identifier** – the identification value
- **identificationType** – the identification type
- **orgUnit** – the org unit

getPrivileges(long) method

get privileges of a customer.

Syntax

```
List< String >getPrivileges ( long customerId )
```

Parameters

- **customerId** –

Usage

This is a pass through of

CustomerPrivilegeDAO#getAllPrivilegesByCustomerId(long, String, Date)

*CustomerNetworkDAO interface***Syntax**

```
public interface CustomerNetworkDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCustomerNetworks(long, boolean, Integer, Integer) method

Get the customer network of a given customer.

Syntax

```
List< CustomerNetwork > getCustomerNetworks ( long customerId ,  
boolean includeInactive , Integer status , Integer type )
```

Parameters

- **customerId** –
- **includeInactive** – whether or not include inactive
- **status** –
- **type** –

Usage

Optional filter on status and type

CustomerOtpDAO interface

Syntax

```
public interface CustomerOtpDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

cleanupExpired(long, Integer) method

Clean up all expired OTPs for a customer.

Syntax

```
void cleanupExpired ( long customerId , Integer otpType )
```

Parameters

- **customerId** –
- **otpType** –

Usage

Optional filter on otpType

CustomerPersistentLoginDAO interface

Dao contract for CustomerPersistentLogins.

Syntax

```
public interface CustomerPersistentLoginDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

deleteLoginsForUser(final String) method

Delete all remember me tokens for the given identification.

Syntax

```
int deleteLoginsForUser ( final String identification )
```

Parameters

- **identification** – the identification.

Returns

the number of logins which were removed

Usage

the number of logins which were removed

deleteLoginsOlderThan(final Date) method

Delete all persistent logins older than this date.

Syntax

```
int deleteLoginsOlderThan ( final Date cutoffDate )
```

Parameters

- **cutoffDate** – cutoff date

Returns

the number of logins which were removed

Usage

the number of logins which were removed

findByUser(final String) method

Find all logins for user.

Syntax

```
List< CustomerPersistentLogin > findByUser ( final String  
identification )
```

Parameters

- **identification** – the identification.

Returns

the logins which were removed

Usage

the logins which were removed

CustomerPrivilegeDAO interface

Syntax

```
public interface CustomerPrivilegeDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getAllPrivilegesByCustomerId(long, String, Date) method

Get all direct assigned privileges for a customer that are valid by a date.

Syntax

```
List< String > getAllPrivilegesByCustomerId ( long customerId ,  
String orgUnit , Date now )
```

Parameters

- **customerId** –
- **orgUnit** –

- **now** –

getCustomerPrivilege(long, String, String, Date) method

Get a single customer privileges for a customerId.

Syntax

```
CustomerPrivilege getCustomerPrivilege ( long customerId , String
privilege , String orgUnit , Date now )
```

Parameters

- **customerId** –
- **privilege** –
- **orgUnit** –
- **now** –

getPrivilegesByCustomerId(long, Date) method

Get all direct assigned privileges for a customer that are valid by a date.

Syntax

```
List< CustomerPrivilege > getPrivilegesByCustomerId ( long
customerId , Date now )
```

Parameters

- **customerId** –
- **now** –

CustomerRoleDAO interface

Syntax

```
public interface CustomerRoleDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCustomerRole(long, String, String, Date) method

get a single role assignment

Syntax

```
CustomerRole getCustomerRole ( long customerId , String role , String  
orgUnit , Date now )
```

Parameters

- **customerId** –
- **role** –
- **orgUnit** –
- **now** –

getRolesByCustomerId(long, Date) method

Get all roles of a customer that have been assigned directly (not via customer type)

Syntax

```
List< CustomerRole > getRolesByCustomerId ( long customerId , Date  
now )
```

Parameters

- **customerId** –
- **now** –

CustomerSessionDAO interface

Syntax

```
public interface CustomerSessionDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

findOpenByCustomerId(long) method

Get all open customer sessions for a customer.

Syntax

```
List< CustomerSession > findOpenByCustomerId ( long customerId )
```

Parameters

- **customerId** –

HistoryEntryDAO interface

Syntax

```
public interface HistoryEntryDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getAllEntriesForCustomer(long, Date, Date, Long) method

get all history entries for a customer

Syntax

```
List< HistoryEntry > getAllEntriesForCustomer ( long customerId ,  
Date from , Date to , Long creatorId )
```

Parameters

- **customerId** –
- **from** –
- **to** –
- **creatorId** –

getEntries(long) method

Get all history entries for a specific object.

Syntax

```
List< HistoryEntry > getEntries ( long affectedEntityId )
```

Parameters

- **affectedEntityId** –

obfuscateHistoryEntries(long, long) method

Obfuscate history entries for a customer.

Syntax

```
int obfuscateHistoryEntries ( long customerId , long callerId )
```

Parameters

- **customerId** –
- **callerId** –

IdentificationDAO interface

Syntax

```
public interface IdentificationDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getActiveByCustomerId(long) method

get all active identification of a customer

Syntax

```
List< Identification > getActiveByCustomerId ( long customerId )
```

Parameters

- **customerId** –

getActiveByCustomerId(long, int, String) method

Return a single active identification by customer and identification type.

Syntax

```
Identification getActiveByCustomerId ( long customerId , int  
identificationType , String orgUnit )
```

Parameters

- **customerId** –
- **identificationType** –

getActiveByCustomerId(long, int) method

Syntax

```
Identification getActiveByCustomerId ( long customerId , int  
identificationType )
```

getActiveIdentification(String, int, String) method

This method will always only return a single result since identification, identification Type in combination with the orgunit form a unique constraint.

Syntax

```
Identification getActiveIdentification ( String identification ,  int
identificationType ,  String orgUnit )
```

Parameters

- **identification** – The identification
- **identificationType** – The identification Type
- **orgUnit** – The orgunit, null means null is the value, as it is standard in money.

*getActiveIdentification(String, int) method***Syntax**

```
Identification getActiveIdentification ( String identification ,  int
identificationType )
```

getAllByCustomerId(long) method

Get all active and inactive identifications of a customer.

Syntax

```
List< Identification > getAllByCustomerId ( long customerId )
```

Parameters

- **customerId** –

getAnyActiveIdentifications(String, Integer) method

This may be used to find active identifications across all orgunits (including orgunit null)

Syntax

```
List< Identification > getAnyActiveIdentifications ( String
identification ,  Integer identificationType )
```

Parameters

- **identification** – The identification to look for
- **identificationType** – The optional Identification type

Returns

A list of identifications

Usage

A list of identifications

getIdentifications(long, Integer, String, boolean) method

This method will return all matching ids.

Syntax

```
List< Identification > getIdentifications ( long customerId ,  
Integer identificationType , String orgUnit , boolean includeInactive )
```

Parameters

- **customerId** – The customerId
- **identificationType** – The identification type
- **orgUnit** – The orgunit, null means null is the value, as it is standard in money
- **includeInactive** – Whether or not to include inactive

IdentityDAO interface

Syntax

```
public interface IdentityDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getIdentities(long, Integer, boolean) method

Get all identities for a customer.

Syntax

```
List< Identity > getIdentities ( long customerId , Integer  
identityType , boolean includeInactive )
```

Parameters

- **customerId** –
- **identityType** –

- **includeInactive** – whether or not include inactive

Usage

Optional filter on type

InvoiceAttributeDAO interface

Syntax

```
public interface InvoiceAttributeDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

findByInvoiceId(long) method

Get all invoice attributes by invoice ID.

Syntax

```
List< InvoiceAttribute > findByInvoiceId ( long invoiceId )
```

Parameters

- **invoiceId** –

InvoiceConfigurationAttributeDAO interface

Syntax

```
public interface InvoiceConfigurationAttributeDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

findByInvoiceConfigurationId(long) method

Find all InvoiceConfigurationAttributes by invoice configuration.

Syntax

```
List< InvoiceConfigurationAttribute >  
findByInvoiceConfigurationId ( long invoiceConfigurationId )
```

Parameters

- **invoiceConfigurationId** –

InvoiceConfigurationDAO interface

Syntax

```
public interface InvoiceConfigurationDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

findByCustomerIdAndActiveFlag(long, Boolean) method

Find Invoice Configurations by customer and active flag.

Syntax

```
List< InvoiceConfiguration > findByCustomerIdAndActiveFlag  
( long customerId , Boolean active )
```

Parameters

- **customerId** –
- **active** –

findByCustomerIdAndInvoiceTypeIdAndActiveFlag(long, long, Boolean) method

Find Invoice Configurations by customer, invoice type and active flag.

Syntax

```
List< InvoiceConfiguration >  
findByCustomerIdAndInvoiceTypeIdAndActiveFlag ( long customerId ,  
long invoiceTypeId , Boolean active )
```

Parameters

- **customerId** –
- **invoiceTypeId** –
- **active** –

findByGroupIdAndCurrentDate(int) method

Find Invoice Configuration by group ID and current date.

Syntax

```
List< InvoiceConfiguration > findByGroupIdAndCurrentDate ( int  
    groupId )
```

Parameters

- **groupId** –

findByReferenceAndActiveFlag(String, Boolean) method

Find Invoice Configuration by reference.

Syntax

```
InvoiceConfiguration findByReferenceAndActiveFlag ( String  
    reference , Boolean active )
```

Parameters

- **reference** –
- **active** –

InvoiceDAO interface

Syntax

```
public interface InvoiceDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

findByCustomerIdAndStatus(long, Integer) method

Find invoices by customerID and optional status.

Syntax

```
List< Invoice > findByCustomerIdAndStatus ( long customerId ,  
    Integer status )
```

Parameters

- **customerId** –
- **status** –

findByInvoiceConfigurationIdAndStatus(long, Integer) method

Find invoices by configurationID and optional status.

Syntax

```
List< Invoice > findByInvoiceConfigurationIdAndStatus ( long  
invoiceConfigurationId , Integer status )
```

Parameters

- **invoiceConfigurationId** –
- **status** –

findByReferenceAndActiveFlag(String, Boolean) method

Find invoice by references and active flag.

Syntax

```
Invoice findByReferenceAndActiveFlag ( String reference , Boolean  
active )
```

Parameters

- **reference** –
- **active** –

processDueInvoices(final IProcessDueInvoiceVisitor, final int, final int, final int, final Integer...) method

process all due invoices

Syntax

```
int processDueInvoices ( final IProcessDueInvoiceVisitor visitor ,  
final int maxWorkers , final int executionTimeout , final int batchSize ,  
final Integer... invoiceStatus )
```

Parameters

- **visitor** –
- **maxWorkers** –
- **executionTimeout** –

- **batchSize** –
- **invoiceStatus** –

LimitSumDAO interface

Syntax

```
public interface LimitSumDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

findByClassAndCustomer(final long, final long) method

find a Limit Sum by the customer id and limit class id

Syntax

```
LimitSum findByClassAndCustomer ( final long limitClassId , final  
long customerId )
```

Parameters

- **limitClassId** –
- **customerId** –

findByClassAndPaymentInstrument(final long, final long) method

Find the Limit Sum by limit class id and payment instrument.

Syntax

```
LimitSum findByClassAndPaymentInstrument ( final long limitClassId ,  
final long paymentInstrumentId )
```

Parameters

- **limitClassId** –
- **paymentInstrumentId** –

findByClassAndWalletEntry(final long, final long) method

Find the Limit Sum by limit class id and wallet ID.

Syntax

```
LimitSum findByClassAndWalletEntry ( final long limitClassId , final  
long walletEntryId )
```

Parameters

- **limitClassId** –
- **walletEntryId** –

NoteDAO interface

Syntax

```
public interface NoteDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCountedNotes(final Long, final Integer, final Integer, final Date, final Date, final String, final Long, final int) method

Get list of notes by flexible filter.

Syntax

```
List< Note > getCountedNotes ( final Long customerId , final Integer category , final Integer status , final Date fromDate , final Date toDate , final String subject , final Long creator , final int maxRecords )
```

Parameters

- **customerId** – filter by customer ID
- **category** – filter by category
- **status** – filter by status (open/closed)
- **fromDate** – filter from date
- **toDate** – filter to date
- **subject** – filter by subject
- **creator** – filter by creator
- **maxRecords** – limit result size

getNotes(final Long, final Integer, final Integer, final Date, final Date, final String, final Long) method

Syntax

```
List< Note > getNotes ( final Long customerId , final Integer category , final Integer status , final Date fromDate , final Date toDate , final String subject , final Long creator )
```

Parameters

- **customerId** –
- **category** –
- **status** –
- **fromDate** –
- **toDate** –
- **subject** –
- **creator** –

getNotesByCustomer(final long) method

Syntax

```
List< Note > getNotesByCustomer ( final long customerId )
```

Parameters

- **customerId** –

OtpDAO interface

Syntax

```
public interface OtpDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

WrkCustomerDAO interface

Syntax

```
public interface WrkCustomerDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

job package

Members

All public members of the job package.

- **JobDAO interface** –
- **JobHistoryDAO interface** –

JobDAO interface

Syntax

```
public interface JobDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getActiveJobsByHandlerName(String) method

Get active jobs by handler name.

Syntax

```
List< Job > getActiveJobsByHandlerName ( String handlerName )
```

Parameters

- **handlerName** –

JobHistoryDAO interface

Syntax

```
public interface JobHistoryDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

clearHistoryForJob(Job) method

clean the job history for a job

Syntax

```
int clearHistoryForJob ( Job job )
```

Parameters

- **job** –

getHistoryForJob(Job) method

Get the job history for a job.

Syntax

```
List< JobHistory > getHistoryForJob ( Job job )
```

Parameters

- **job** –

pi package*Members*

All public members of the pi package.

- **BalanceAlertDAO** interface –
- **BankAccountDAO** interface –
- **CardDAO** interface –
- **CardTypeDAO** interface –
- **ExternalAccountDAO** interface –
- **InternalVoucherDAO** interface –
- **MandateDAO** interface –
- **PaymentInstrumentDAO** interface –
- **PaymentInstrumentTypeDAO** interface –
- **StoredValueAccountDAO** interface –
- **WalletEntryDAO** interface –
- **WrkWalletEntryDAO** interface –

*BalanceAlertDAO interface***Syntax**

```
public interface BalanceAlertDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getBalanceAlertsByPaymentInstrumentId(long, boolean, Long) method

Get all balance alerts for a given payment instrument ID.

Syntax

```
List< BalanceAlert > getBalanceAlertsByPaymentInstrumentId  
( long paymentInstrumentId , boolean includeInactive , Long minThreshold )
```

Parameters

- **paymentInstrumentId** – the payment instrument ID
- **includeInactive** – indicates whether inactive balance alerts should also be returned
- **minThreshold** – the minimum threshold a returned balance alert must have

Returns

the balance alerts

Usage

the balance alerts

BankAccountDAO interface

Syntax

```
public interface BankAccountDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

save(PaymentInstrument, BankAccount, Long) method

save a bank account definition based on payment instrument and bank account (extension)

Syntax

```
void save ( PaymentInstrument paymentInstrument , BankAccount  
bankAccount , Long creator )
```

Parameters

- **paymentInstrument** –
- **bankAccount** –
- **creator** –

*CardDAO interface**Syntax*

```
public interface CardDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

save(PaymentInstrument, Card, Long) method

Save payment instrument and card extension to payment instrument.

Syntax

```
void save ( PaymentInstrument paymentInstrument , Card card , Long  
creator )
```

Parameters

- **paymentInstrument** –
- **card** –
- **creator** –

*CardTypeDAO interface**Syntax*

```
public interface CardTypeDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

*ExternalAccountDAO interface**Syntax*

```
public interface ExternalAccountDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

save(PaymentInstrument, ExternalAccount, Long) method

Save external account and embedded payment instrument.

Syntax

```
void save ( PaymentInstrument paymentInstrument , ExternalAccount  
externalAccount , Long creator )
```

Parameters

- **paymentInstrument** –
- **externalAccount** –
- **creator** –

InternalVoucherDAO interface

Syntax

```
public interface InternalVoucherDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

findExpiredVouchers() method

Returns a list of SubTransactions that are expired.

Syntax

```
List< SubTransaction > findExpiredVouchers ()
```

findInternalVoucher(long, String) method

find internal voucher by customer id and pickup code

Syntax

```
InternalVoucher findInternalVoucher ( long customerId , String  
pickupCode )
```

Parameters

- **customerId** –
- **pickupCode** –

save(PaymentInstrument, InternalVoucher, Long) method
Save internal voucher and embedded payment instrument.

Syntax

```
void save ( PaymentInstrument paymentInstrument ,   InternalVoucher  
internalVoucher ,   Long creator )
```

Parameters

- **paymentInstrument** –
- **internalVoucher** –
- **creator** –

MandateDAO interface

Syntax

```
public interface MandateDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getActiveByPiAndCreditor(final long, final String) method
Get an active mandate by pi and creditorId.

Syntax

```
Mandate getActiveByPiAndCreditor ( final long piId ,   final String  
creditorId )
```

Parameters

- **piId** –
- **creditorId** –

getMandates(final Long, final Long, final boolean) method

Find mandates by filter.

Syntax

```
List< Mandate > getMandates ( final Long customerId , final Long  
piId , final boolean includeInactive )
```

Parameters

- **customerId** – filter on customer with ID
- **piId** – filter on payment instrument with ID
- **includeInactive** – whether or not to include inactive mandates

PaymentInstrumentDAO interface

Syntax

```
public interface PaymentInstrumentDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getBestPaymentInstrument(final long, final PaymentInstrument.Class, final Set< PaymentInstrument.Type >, final Currency, final boolean) method

Similar to the getPaymentInstruments but only returns the first element in the list and has a slightly different approach (no joined fetching) to enhance performance.

Syntax

```
PaymentInstrument getBestPaymentInstrument ( final long  
customerId , final PaymentInstrument.Class clazz , final Set<  
PaymentInstrument.Type > types , final Currency currency , final  
boolean credit )
```

Parameters

- **customerId** –
- **clazz** –
- **types** –
- **currency** –
- **credit** –

Returns

the best payment instrument given the provided filters or null

Usage

the best payment instrument given the provided filters or null

getPaymentInstrumentsByCustomerId(long, Integer, Integer) method

Get all payment instruments for a given customer ID.

Syntax

```
List< PaymentInstrument > getPaymentInstrumentsByCustomerId  
( long customerId , Integer piClassFilter , Integer piTypeFilter )
```

Parameters

- **customerId** – the customer ID

Returns

the payment instruments

Usage

Please use

the payment instruments

getPaymentInstrumentsByCustomerId(long, Integer, Integer, boolean) method

Get all payment instruments for a given customer ID.

Syntax

```
List< PaymentInstrument > getPaymentInstrumentsByCustomerId  
( long customerId , Integer piClassFilter , Integer piTypeFilter , boolean  
includeInactive )
```

Parameters

- **customerId** – the customer ID

Returns

the payment instruments

Usage

the payment instruments

PaymentInstrumentTypeDAO interface

Syntax

```
public interface PaymentInstrumentTypeDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

StoredValueAccountDAO interface

Syntax

```
public interface StoredValueAccountDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

save(PaymentInstrument, StoredValueAccount, Long) method

Save SVA with embedded payment instrument.

Syntax

```
void save ( PaymentInstrument paymentInstrument , StoredValueAccount  
storedValueAccount , Long creator )
```

Parameters

- **paymentInstrument** –
- **storedValueAccount** –
- **creator** –

WalletEntryDAO interface

Syntax

```
public interface WalletEntryDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

getWalletEntriesByCustomerId(long, Integer, Integer) method

Get all wallet entries for a given customer ID.

Syntax

```
List< WalletEntry > getWalletEntriesByCustomerId ( long  
customerId , Integer piClassFilter , Integer piTypeFilter )
```

Parameters

- **customerId** – the customer ID

Returns

the wallet entries

Usage

the wallet entries

getWalletEntry(Customer, PaymentInstrument) method

Returns the wallet entry for the given customer and payment instrument.

Syntax

```
WalletEntry getWalletEntry ( Customer customer , PaymentInstrument  
paymentInstrument )
```

Parameters

- **customer** – the customer
- **paymentInstrument** – the payment instrument

Returns

the wallet entry

Usage

the wallet entry

isAliasUsedInCustomerWallet(final Customer, final PaymentInstrument, final String)
method

Returns whether a payment instrument exists with that alias in the customer's wallet.

Syntax

```
boolean isAliasUsedInCustomerWallet ( final Customer customer ,  
final PaymentInstrument paymentInstrument , final String alias )
```

Parameters

- **customer** – the customer
- **paymentInstrument** – the payment instrument
- **alias** – the alias

Returns

true if the pi is in the customer's wallet

Usage

true if the pi is in the customer's wallet

isDebitCreditPrioUsedInCustomerWallet(final long, final long, final Integer, final Integer) method

Returns whether the debit or credit priority is used in the customer's wallet.

Syntax

```
boolean isDebitCreditPrioUsedInCustomerWallet ( final long  
customerId , final long piId , final Integer debitPrio , final Integer  
creditPrio )
```

Parameters

- **customerId** – the customer ID
- **piId** – the pi ID
- **debitPrio** – the debit priority
- **creditPrio** – the credit priority

Returns

true if the pi is in the customer's wallet

Usage

true if the pi is in the customer's wallet

isPIInCustomerWallet(final Customer, final PaymentInstrument) method

Returns whether the given payment instrument is in the customer's wallet.

Syntax

```
boolean isPIInCustomerWallet ( final Customer customer , final  
PaymentInstrument paymentInstrument )
```

Parameters

- **customer** – the customer
- **paymentInstrument** – the payment instrument

Returns

true if the pi is in the customer's wallet

Usage

true if the pi is in the customer's wallet

WrkWalletEntryDAO interface

Syntax

```
public interface WrkWalletEntryDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

system package

Members

All public members of the system package.

- **AuthenticationMatrixDAO interface** –
- **AuthenticationMethodRuleDAO interface** –
- **AuthenticationMethodRuleStepDAO interface** –
- **BatchRunDAO interface** –
- **BulkProcessingFileDAO interface** –
- **BulkProcessingFileTypeDAO interface** –
- **BullyDAO interface** –

- **ClearingModeDAO interface** –
- **CountryDAO interface** –
- **CredentialPolicyDAO interface** –
- **CredentialPolicyRegexDAO interface** – 2013 SAP AG or an SAP affiliate company.
- **CustomerTypeCredentialPolicyDAO interface** –
- **CustomerTypeDAO interface** –
- **ErrorCodeDAO interface** –
- **ExchangeRateDAO interface** –
- **ExchangeRateHistoryDAO interface** –
- **FeeSetDAO interface** –
- **FeeTypeDAO interface** –
- **FeeTypeFeeSetScaleDAO interface** –
- **FileDAO interface** –
- **IndividualRestrictionDAO interface** –
- **InterestBalanceSnapshotDAO interface** –
- **InterestCalculationDAO interface** –
- **InterestConfigurationDAO interface** –
- **InterestPeriodDAO interface** –
- **InterestProductDAO interface** –
- **InterestProductMappingDAO interface** –
- **InvoiceTypeAttributeDAO interface** –
- **InvoiceTypeBatchConfigDAO interface** –
- **InvoiceTypeBatchInvoicesDAO interface** –
- **InvoiceTypeDAO interface** –
- **InvoiceTypeOnlineCmdQueueDAO interface** –
- **InvoiceTypeOnlineConfigDAO interface** –
- **LimitClassDAO interface** –
- **LimitSetClassDAO interface** –
- **LimitSetDAO interface** –
- **LookupQueryDAO interface** – LookupQueries are used to get a name/id list for lookup tables that are not mapped to a JPA bean/DAO.
- **OrderChannelDAO interface** – Simple DAO to lookup the name of an order channel.
- **OrgUnitDAO interface** –
- **OtpTypeDAO interface** –
- **RestrictionDAO interface** –
- **RestrictionForOrgUnitDAO interface** –
- **RestrictionForRiskCategoryDAO interface** –
- **RestrictionOnOrgUnitDAO interface** –
- **RestrictionOnRiskCategoryDAO interface** –

- **RestrictionOnUseCaseDAO** interface –
- **RestrictionRuleDAO** interface –
- **RestrictionsGroupDAO** interface –
- **RiskCategoryDAO** interface –
- **ScaleDAO** interface –
- **ScalePeriodDAO** interface –
- **ScaleStepDAO** interface –
- **SessionPolicyDAO** interface –
- **SubTransactionTypeDAO** interface –
- **TransactionTemplateMappingDAO** interface –
- **UmgrPrivilegeDAO** interface –
- **UmgrRoleDAO** interface –
- **UmgrRolePrivilegeDAO** interface –
- **UseCaseDAO** interface –
- **UseCaseFeeDAO** interface –
- **UseCasePaymentInstrumentTypeDAO** interface –
- **UseCasePrivilegeDAO** interface –
- **ValidationRuleDAO** interface –
- **VersionDAO** interface –
- **WrkBulkProcessingDAO** interface –

AuthenticationMatrixDAO interface

Syntax

```
public interface AuthenticationMatrixDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

findMatchingMatrices(final long, final long, final int, final int, final int, final String, final boolean) method

Find AuthenticationMatrix objects matching the given criteria.

Syntax

```
List< AuthenticationMatrix > findMatchingMatrices ( final long
payee , final long payer , final int useCase , final int orderChannel ,
final int piType , final String orgUnit , final boolean isPayer )
```

Parameters

- **payee** – payee
- **payer** – payer
- **useCase** – the use case
- **orderChannel** – the order channel
- **piType** – the pi type
- **orgUnit** – the org unit
- **isPayer** – whether we are searching for the payer or payee's matrix

Returns

list of matching matrices

Usage

list of matching matrices

AuthenticationMethodRuleDAO interface

Syntax

```
public interface AuthenticationMethodRuleDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

AuthenticationMethodRuleStepDAO interface

Syntax

```
public interface AuthenticationMethodRuleStepDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

BatchRunDAO interface

Syntax

```
public interface BatchRunDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getLastFinished(int) method

Get last finished batch run of a given type.

Syntax

```
BatchRun getLastFinished ( int type )
```

Parameters

- **type** –

getUnfinished(int, int) method

Returns all unfinished BatchRuns of the given type.

Syntax

```
List< BatchRun > getUnfinished ( int type , int maxAgeInSeconds )
```

Usage

This is used to check if a specific job has already finished.

BulkProcessingFileDAO interface***Syntax***

```
public interface BulkProcessingFileDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getBulkProcessingFiles(final Integer, final String, final Date, final Date, final List< Integer >, final Long) method

Get a list of bulk processing files based on filter.

Syntax

```
List< BulkProcessingFile > getBulkProcessingFiles ( final  
Integer bulkFileType , final String fileName , final Date startDate ,
```

```
final Date endDate , final List< Integer > status , final Long  
uploader )
```

Parameters

- **bulkFileType** – optional filter on file type
- **fileName** – optional file name
- **startDate** – optional start date
- **endDate** – optional end date
- **status** – optional status
- **uploader** – optional uploaders customer ID

BulkProcessingFileTypeDAO interface

Syntax

```
public interface BulkProcessingFileTypeDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

BullyDAO interface

Syntax

```
public interface BullyDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getActiveBullyByServiceName(String) method

Get active bully by service name.

Syntax

```
Bully getActiveBullyByServiceName ( String serviceName )
```

Parameters

- **serviceName** –

*ClearingModeDAO interface**Syntax*

```
public interface ClearingModeDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getClearingModesbyFrequency(String) method

Get clearing modes by clearing frequency.

Syntax

```
List< ClearingMode > getClearingModesbyFrequency ( String  
clearingFrequency )
```

Parameters

- **clearingFrequency** –

*CountryDAO interface**Syntax*

```
public interface CountryDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

*CredentialPolicyDAO interface**Syntax*

```
public interface CredentialPolicyDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CredentialPolicyRegexDAO interface

2013 SAP AG or an SAP affiliate company.

Syntax

```
public interface CredentialPolicyRegexDAO
```

Usage

All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

deleteAll(CredentialPolicy) method

Deletes all CredentialPolicyRegex for a CredentialPolicy.

Syntax

```
void deleteAll ( CredentialPolicy policy )
```

Parameters

- **policy** –

CustomerTypeCredentialPolicyDAO interface

Syntax

```
public interface CustomerTypeCredentialPolicyDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getByCustomerTypeAndCredentialType(int, int) method

Get credential policy for customer type and credential type.

Syntax

```
CredentialPolicy getByCustomerTypeAndCredentialType ( int  
customerTypeId , int credentialTypeId )
```

Parameters

- **customerTypeId** –

- **credentialTypeId** –

Returns

null if not defined

Usage

null if not defined

*CustomerTypeDAO interface**Syntax*

```
public interface CustomerTypeDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

*ErrorCodeDAO interface**Syntax*

```
public interface ErrorCodeDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getAll() method

Returns a map of all error codes with their descriptions.

Syntax

```
Map< Integer, String > getAll ()
```

Returns

the map

Usage

the map

getErrorCodeInformation(final int) method

Returns the information about an error code.

Syntax

```
String getErrorCodeInformation ( final int errorCode )
```

Parameters

- **errorCode** – the errorCode's id

Returns

the error code information

Usage

the error code information

ExchangeRateDAO interface

Syntax

```
public interface ExchangeRateDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getExchangeRate(final Currency, final Currency) method

Find the newest Exchange rate for the given two currencies.

Syntax

```
ExchangeRate getExchangeRate ( final Currency from , final  
Currency to )
```

Parameters

- **from** – the from currency
- **to** – the to currency

Returns

the exchange rate or null if none is found

Usage

the exchange rate or null if none is found

getExchangeRate(final Currency, final Currency, final Date) method

Find the newest Exchange rate for the given two currencies.

Syntax

```
ExchangeRate getExchangeRate ( final Currency from , final
Currency to , final Date validAt )
```

Parameters

- **from** – the from currency
- **to** – the to currency
- **validAt** – the rate must be of the same age or older than the given date.

Returns

the exchange rate or null if none is found

Usage

the exchange rate or null if none is found

getExchangeRates(final Currency, final Currency) method

exchange rates based on filter

Syntax

```
List< ExchangeRate > getExchangeRates ( final Currency from ,
final Currency to )
```

Parameters

- **from** – from currency (optional)
- **to** – to currency (optional)

ExchangeRateHistoryDAO interface

Syntax

```
public interface ExchangeRateHistoryDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

findByDate(Currency, Currency, Date) method

This method returns an exchange rate that was valid at a specific point in time.

Syntax

`ExchangeRateHistory findByDate ( Currency from , Currency to , Date validAt )`

Parameters

- **from** –
- **to** –
- **validAt** –

Returns

the history entry

Usage

the history entry

getExchangeRateHistory(Currency, Currency) method

Retrives a list of all former exchange rate configurations for the given from and to currencies.

Syntax

`List< ExchangeRateHistory > getExchangeRateHistory ( Currency from , Currency to )`

Parameters

- **from** –
- **to** –

FeeSetDAO interface

Syntax

```
public interface FeeSetDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

deleteCascading(long, long) method

Delete all fee configuration that references the given fee set id, including the fee set.

Syntax

```
void deleteCascading ( long feeSetId ,   long updater )
```

Parameters

- **feeSetId** –
- **updater** –

getAll(Boolean) method

Fetches all fee sets.

Syntax

```
List< FeeSet > getAll ( Boolean Individual )
```

Parameters

- **Individual** –

Usage

The individual flag behaves the following way:

- null = fetches all fee sets
- true = fetches only individual fee sets
- false = fetches only global fee sets

FeeTypeDAO interface

Syntax

```
public interface FeeTypeDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getFeeTypesByFeeSet(long) method
get fee types for a fee set

Syntax

```
List< FeeType > getFeeTypesByFeeSet ( long feeSetId )
```

Parameters

- **feeSetId** –

FeeTypeFeeSetScaleDAO interface

Syntax

```
public interface FeeTypeFeeSetScaleDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

FileDAO interface

Syntax

```
public interface FileDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

IndividualRestrictionDAO interface

Syntax

```
public interface IndividualRestrictionDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getRestriction(Customer, Customer, UseCase) method

Returns a matching individual restriction configuration.

Syntax

```
IndividualRestriction getRestriction ( Customer payer , Customer
payee , UseCase useCase )
```

Parameters

- **payer** –
- **payee** –
- **useCase** –

getRestrictions(Customer, Customer, UseCase) method

Returns all matching individual restriction configuration based on the given filter.

Syntax

```
List< IndividualRestriction > getRestrictions ( Customer payer ,
Customer payee , UseCase useCase )
```

Parameters

- **payer** – if set, it returns entries with this payer value
- **payee** – if set, it returns entries with this payee value
- **useCase** – if set, it returns entries with this use case value

*InterestBalanceSnapshotDAO interface***Syntax**

```
public interface InterestBalanceSnapshotDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

recordPiInterestSnaphots() method

Create interest snapshots.

Syntax

```
int recordPiInterestSnaphots ()
```

Returns

0 on success

Usage

0 on success

InterestCalculationDAO interface

Syntax

```
public interface InterestCalculationDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

updateClearedCalculations(Long, Long, Long, Long, boolean) method

See business logic documentation.

Syntax

```
int updateClearedCalculations ( Long piId , Long lock , Long  
clearingTxn , Long taxTxnId , boolean isCredit )
```

Parameters

- **piId** –
- **lock** –
- **clearingTxn** –
- **taxTxnId** –
- **isCredit** –

updatePreAllocationTxnForCalculations(Long, long) method

See business logic documentation.

Syntax

```
int updatePreAllocationTxnForCalculations ( Long lock , long  
txnId )
```

Parameters

- **lock** –

- **txnId** –

InterestConfigurationDAO interface

Syntax

```
public interface InterestConfigurationDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

UnmappedPaymentInstrumentProductPair class

Syntax

```
public class UnmappedPaymentInstrumentProductPair
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

UnmappedPaymentInstrumentProductPair(PaymentInstrument, InterestProduct) constructor

Syntax

```
UnmappedPaymentInstrumentProductPair ( PaymentInstrument  
paymentInstrument , InterestProduct prod )
```

getPaymentInstrument() method

Syntax

```
PaymentInstrument getPaymentInstrument ()
```

getProd() method

Syntax

```
InterestProduct getProd ()
```

setPaymentInstrument(PaymentInstrument) method

Syntax

```
void setPaymentInstrument ( PaymentInstrument paymentInstrument )
```

setProd(InterestProduct) method

Syntax

```
void setProd ( InterestProduct prod )
```

calculatePiInterest(IPiInterestCalculatingVisitor, Long, String, Long, Integer, Integer) method

Calculate the Pi Interest.

Syntax

```
int calculatePiInterest ( IPiInterestCalculatingVisitor visitor ,  
Long callerId , String calculationPeriod , Long lock , Integer maxworkers ,  
Integer batchSize )
```

Parameters

- **visitor** –
- **callerId** –
- **calculationPeriod** –
- **lock** –
- **maxworkers** –
- **batchSize** –

clearInterest(IPiInterestClearingVisitor, Long, String, Long, Integer, Integer) method

Performe interest rate clearing of calculated interest rates.

Syntax

```
int clearInterest ( IPiInterestClearingVisitor visitor , Long  
callerId , String calculationPeriod , Long lock , Integer maxworkers ,  
Integer batchSize )
```

Parameters

- **visitor** –
- **callerId** –
- **calculationPeriod** –

- **lock** –
- **maxworkers** –
- **batchSize** –

getPisWithoutConfigs() method

Syntax

```
List< UnmappedPaymentInstrumentProductPair >
getPisWithoutConfigs ()
```

Returns

An object array containing the unmapped pis and products

Usage

An object array containing the unmapped pis and products

InterestPeriodDAO interface

Syntax

```
public interface InterestPeriodDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getLastCutoffDate(InterestPeriod, Date) method

Get the last cutoff date of a interest period.

Syntax

```
Date getLastCutoffDate ( InterestPeriod period , Date d ) throws
ParseException
```

Parameters

- **period** –
- **d** –

Exceptions

- **ParseException** –

InterestProductDAO interface

Syntax

```
public interface InterestProductDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getInterestProductList() method

Loads a list of InterestProducts.

Syntax

```
List< InterestProduct > getInterestProductList ()
```

InterestProductMappingDAO interface

Syntax

```
public interface InterestProductMappingDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getInterestProductMappingList() method

Loads a list of InterestProductMapping.

Syntax

```
List< InterestProductMapping > getInterestProductMappingList  
( )
```

getMappingForCustomerTypePiTypeAndOrgUnit(CustomerType, Type, OrgUnit) method

Get the mapping for a customer type, pi type and org unit.

Syntax

```
InterestProductMapping  
getMappingForCustomerTypePiTypeAndOrgUnit ( CustomerType  
customerType , Type piType , OrgUnit orgUnit )
```

Parameters

- **customerType** –
- **piType** –
- **orgUnit** –

*InvoiceTypeAttributeDAO interface***Syntax**

```
public interface InvoiceTypeAttributeDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

findByInvoiceTypeId(int) method

Get all attributes for a invoice type.

Syntax

```
List< InvoiceTypeAttribute > findByInvoiceTypeId ( int  
invoiceTypeId )
```

Parameters

- **invoiceTypeId** –

*InvoiceTypeBatchConfigDAO interface***Syntax**

```
public interface InvoiceTypeBatchConfigDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

save(InvoiceType, InvoiceTypeBatchConfig, Long) method

save invoice type batch configuration and invoice type

Syntax

```
void save ( InvoiceType invoiceType , InvoiceTypeBatchConfig  
invoiceTypeBatchConfig , Long creator )
```

Parameters

- **invoiceType** –
- **invoiceTypeBatchConfig** –
- **creator** –

InvoiceTypeBatchInvoicesDAO interface

Syntax

```
public interface InvoiceTypeBatchInvoicesDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

InvoiceTypeDAO interface

Syntax

```
public interface InvoiceTypeDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

findActive() method

returns all active InvoiceTypes

Syntax

```
List< InvoiceType > findActive ()
```

findByIdAndActiveFlag(int, Boolean) method

Returns the InvoiceType with the given groupid.

Syntax

```
List< InvoiceType > findByIdAndActiveFlag ( int groupId ,  
Boolean active )
```

Parameters

- **reference** –

- **active** – if null all are returned, otherwise only the active / inactive.

findByIssuerIdAndActiveFlag(long, Boolean) method

Returns the list of InvoiceTypes for an issuer.

Syntax

```
List< InvoiceType > findByIssuerIdAndActiveFlag ( long issuerId ,  
Boolean active )
```

Parameters

- **issuerId** –
- **active** – if null all are returned, otherwise only the active / inactive.

findByReferenceAndActiveFlag(String, Boolean) method

Returns the InvoiceType with the specified reference.

Syntax

```
InvoiceType findByReferenceAndActiveFlag ( String reference ,  
Boolean active )
```

Parameters

- **reference** –
- **active** – if null all are returned, otherwise only the active / inactive.

InvoiceTypeOnlineCmdQueueDAO interface

Syntax

```
public interface InvoiceTypeOnlineCmdQueueDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getPendingInvoicesCmdQueue() method

Syntax

```
List< InvoiceTypeOnlineCmdQueue > getPendingInvoicesCmdQueue  
( )
```

Usage

Returns a list of

InvoiceTypeOnlineCmdQueue

s that are pending status.

InvoiceTypeOnlineConfigDAO interface

Syntax

```
public interface InvoiceTypeOnlineConfigDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

save(InvoiceType, InvoiceTypeOnlineConfig, Long) method

Safe invoiceTypeOnlineConfig with embedded invoiceType.

Syntax

```
void save ( InvoiceType invoiceType , InvoiceTypeOnlineConfig  
invoiceTypeOnlineConfig , Long creator )
```

Parameters

- **invoiceType** –
- **invoiceTypeOnlineConfig** –
- **creator** –

LimitClassDAO interface

Syntax

```
public interface LimitClassDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

findByLimitSetIdAndUseCase(final long, final int) method

Returns the LimitSetClass for the given set id and use case id.

Syntax

```
List< LimitClass > findByLimitSetIdAndUseCase ( final long
limitSetId , final int useCaseId )
```

Parameters

- **limitSetId** – the limit set id
- **useCaseId** – the user case id, may be null

Returns

the limit set class or null if no match

Usage

the limit set class or null if no match

getAll(boolean) method

Returns all limit classes The filterIndividual flag behaves the following way:

Syntax

```
List< LimitClass > getAll ( boolean filterIndividual )
```

Parameters

- **filterIndividual** –

Usage

- false - All limit classes will be returned
- true - Only limit classes which are linked with at least one "global" limit set will be returned

*LimitSetClassDAO interface***Syntax**

```
public interface LimitSetClassDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

clearLimitSetclass(long) method

Remove all LimitSetClass entries for the given limit set.

Syntax

```
void clearLimitSetclass ( long limitSetId )
```

Parameters

- **limitSetId** –

findByLimitSetIdAndUseCase(final long, final int) method

Find all LimitSetClasses based on limit set id and use case id.

Syntax

```
List< LimitSetClass > findByLimitSetIdAndUseCase ( final long  
limitSetId , final int useCaseId )
```

Parameters

- **limitSetId** –
- **useCaseId** –

getLimitSetClassByLimitClass(final long) method

Find all LimitSetClasses for a limit class.

Syntax

```
List< LimitSetClass > getLimitSetClassByLimitClass ( final long  
limitClassId )
```

Parameters

- **limitClassId** –

getLimitSetClassByLimitSet(final long) method

Find all LimitSetClasses for a limit set ID.

Syntax

```
List< LimitSetClass > getLimitSetClassByLimitSet ( final long  
limitSetId )
```

Parameters

- **limitSetId** –

getLimitSetClassFromCustomer(final long) method

Find all LimitSetClasses for a customer.

Syntax

```
List< LimitSetClass > getLimitSetClassFromCustomer ( final long
customerId )
```

Parameters

- **customerId** –

getLimitSetClassFromPi(final long) method

Find all LimitSetClasses for a payment instrument.

Syntax

```
List< LimitSetClass > getLimitSetClassFromPi ( final long piId )
```

Parameters

- **piId** –

getLimitSetClassFromWallet(final long) method

Find all LimitSetClasses for a wallet.

Syntax

```
List< LimitSetClass > getLimitSetClassFromWallet ( final long
walletId )
```

Parameters

- **walletId** –

LimitSetDAO interface

Syntax

```
public interface LimitSetDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

deleteCascading(long, long) method
delete a limit set and all referencing entries

Syntax

```
void deleteCascading ( long limitSetId , long updater )
```

Parameters

- **limitSetId** –
- **updater** –

getAll(Boolean) method
Fetches all limit sets.

Syntax

```
List< LimitSet > getAll ( Boolean Individual )
```

Parameters

- **Individual** –

Usage

The individual flag behaves the following way:

- null = fetches all limit sets
- true = fetches only individual limit sets
- false = fetches only global limit sets

getLimitSetByCustomer(long) method
Get a limit set for a customer.

Syntax

```
LimitSet getLimitSetByCustomer ( long customerId )
```

Parameters

- **customerId** –

getLimitSetByPi(long) method

Get a limit set for a payment instrument.

Syntax

```
LimitSet getLimitSetByPi ( long piId )
```

Parameters

- **piId** –

getLimitSetByWalletEntry(long) method

Get a limit set for a wallet entry.

Syntax

```
LimitSet getLimitSetByWalletEntry ( long weId )
```

Parameters

- **weId** –

LookupQueryDAO interface

LookupQueries are used to get a name/id list for lookup tables that are not mapped to a JPA bean/DAO.

Syntax

```
public interface LookupQueryDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

allowQueryExecution(final LookupQuery, final List< String >) method

Check if caller privileges contains privilege to execute the configured lookup query.

Syntax

```
boolean allowQueryExecution ( final LookupQuery lookupQuery ,  
final List< String > callerPrivileges )
```

Parameters

- **lookupQuery** –
- **callerPrivileges** –

executeQuery(final LookupQuery, final List< String >) method

Execute the lookup query after checking that the caller privileges contains the required privilege to execute the lookup query.

Syntax

```
List< LookupEntity > executeQuery ( final LookupQuery lookupQuery ,  
final List< String > callerPrivileges )
```

Parameters

- **lookupQuery** –
- **callerPrivileges** –

getByCaseInsensitiveId(String) method

Get id by name (case insensitive)

Syntax

```
LookupQuery getByCaseInsensitiveId ( String id )
```

Parameters

- **id** –

OrderChannelDAO interface

Simple DAO to lookup the name of an order channel.

Syntax

```
public interface OrderChannelDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getOrderChannelName(final int) method

Returns the name of the order channel.

Syntax

```
String getOrderChannelName ( final int orderChannel )
```


Parameters

- **orderChannel** – the order channel's id

Returns

the name of the channel

Usage

the name of the channel

OrgUnitDAO interface

Syntax

```
public interface OrgUnitDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

OtpTypeDAO interface

Syntax

```
public interface OtpTypeDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

RestrictionDAO interface

Syntax

```
public interface RestrictionDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getByRestrictionGroup(RestrictionsGroup) method

Get a list of Restrictions for a given Restrictions Group.

Syntax

```
List< Restriction > getByRestrictionGroup ( RestrictionsGroup  
group )
```

Parameters

- **group** –

getOnOrgUnitSet(long) method

Get the list of Org units IDs this restriction is applied on.

Syntax

```
List< String > getOnOrgUnitSet ( long restrictionId )
```

Parameters

- **restrictionId** –

getOnRiskSet(long) method

Get the list of Risk Categories IDs this restriction is applied on.

Syntax

```
List< Integer > getOnRiskSet ( long restrictionId )
```

Parameters

- **restrictionId** –

getOnUseCaseSet(long) method

Get the list of UseCase IDs this restriction is applied on.

Syntax

```
List< Integer > getOnUseCaseSet ( long restrictionId )
```

Parameters

- **restrictionId** –

getTransactionHistory(Customer, RestrictionRule) method

Get the transaction history values (number and amount sum) for a rule.

Syntax

```
Long[] getTransactionHistory ( Customer customer , RestrictionRule
rule )
```

Parameters

- **customer** –
- **rule** –

isPriorityUnique(final int, final long) method

Check if the priority does not exist in the current restrictions group.

Syntax

```
boolean isPriorityUnique ( final int restricitonsGroupId , final long
priorityId )
```

Parameters

- **restricitonsGroupId** –
- **priorityId** –

retrieveRestrictions(Customer, Customer, int) method

Find the list of restrictions for a given transaction.

Syntax

```
List< Restriction > retrieveRestrictions ( Customer payer ,
Customer payee , int useCase )
```

Parameters

- **payer** –
- **payee** –
- **useCase** –

*RestrictionForOrgUnitDAO interface***Syntax**

```
public interface RestrictionForOrgUnitDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getAll(Restriction) method

Get all for a given restriction.

Syntax

```
List< RestrictionForOrgUnit > getAll ( Restriction restriction )
```

Parameters

- **restriction** –

RestrictionForRiskCategoryDAO interface

Syntax

```
public interface RestrictionForRiskCategoryDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getAll(Restriction) method

Get all for a given restriction.

Syntax

```
List< RestrictionForRiskCategory > getAll ( Restriction  
restriction )
```

Parameters

- **restriction** –

RestrictionOnOrgUnitDAO interface

Syntax

```
public interface RestrictionOnOrgUnitDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getAll(Restriction) method

Get all for a given restriction.

Syntax

```
List< RestrictionOnOrgUnit > getAll ( Restriction restriction )
```

Parameters

- **restriction** –

RestrictionOnRiskCategoryDAO interface

Syntax

```
public interface RestrictionOnRiskCategoryDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getAll(Restriction) method

Get all for a given restriction.

Syntax

```
List< RestrictionOnRiskCategory > getAll ( Restriction restriction )
```

Parameters

- **restriction** –

RestrictionOnUseCaseDAO interface

Syntax

```
public interface RestrictionOnUseCaseDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

getAll(Restriction) method

Get all for a given restriction.

Syntax

```
List< RestrictionOnUseCase > getAll ( Restriction restriction )
```

Parameters

- **restriction** –

RestrictionRuleDAO interface

Syntax

```
public interface RestrictionRuleDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getAll(Restriction) method

Get all for a given restriction.

Syntax

```
List< RestrictionRule > getAll ( Restriction restriction )
```

Parameters

- **restriction** –

RestrictionsGroupDAO interface

Syntax

```
public interface RestrictionsGroupDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

*RiskCategoryDAO interface**Syntax*

```
public interface RiskCategoryDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

*ScaleDAO interface**Syntax*

```
public interface ScaleDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCurrentScalePeriods(long, String) method

get the list of scale periods for a scale and currency

Syntax

```
List< Scale.Period >getCurrentScalePeriods ( long scaleId, String currency )
```

Parameters

- **scaleId** –
- **currency** –

getScaleByFeeTypeAndFeeSet(int, long) method

get a scale for a fee type and set

Syntax

```
Scale getScaleByFeeTypeAndFeeSet ( int feeTypeId, long feeSetId )
```

Parameters

- **feeTypeId** –
- **feeSetId** –

getScalePeriodForDate(long, Date, String) method
get the scale period for a scale, date and currency

Syntax

Scale.Period getScalePeriodForDate (long *scaleId* , Date *date* ,
String *currency*)

Parameters

- **scaleId** –
- **date** –
- **currency** –

getScaleStepForDateAndAmount(long, Date, long, String) method
Get the scale step for a scale, date, amount and currency.

Syntax

Scale.Step getScaleStepForDateAndAmount (long *scaleId* , Date *date* ,
long *amount* , String *currency*)

Parameters

- **scaleId** –
- **date** –
- **amount** –
- **currency** –

getScaleStepsByFeeTypeAndFeeSet(long, Integer, String) method
get scale steps for a fee set, type and currency

Syntax

Map< FeeType, Map< String, List< Scale.Step > > >
getScaleStepsByFeeTypeAndFeeSet (long *feeSetId* , Integer *feeTypeId* ,
String *currency*)

Parameters

- **feeSetId** –

- **feeTypeId** –
- **currency** –

getScaleStepsByScaleId(Integer, String) method
get the scale steps for a scale and currency

Syntax

```
Map< String, List< Scale.Step > > getScaleStepsByScaleId
( Integer scaleId , String currency )
```

Parameters

- **scaleId** –
- **currency** –

ScalePeriodDAO interface

Syntax

```
public interface ScalePeriodDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

ScaleStepDAO interface

Syntax

```
public interface ScaleStepDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

save(Step, Scale.Period, long) method
Save a scale step and emedded period.

Syntax

```
void save ( Step step , Scale.Period period , long creator )
```

Parameters

- **step** –
- **period** –
- **creator** –

SessionPolicyDAO interface

Syntax

```
public interface SessionPolicyDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

SubTransactionTypeDAO interface

Syntax

```
public interface SubTransactionTypeDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

TransactionTemplateMappingDAO interface

Syntax

```
public interface TransactionTemplateMappingDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getAllTransactionTemplateMappings() method

See getAll().

Syntax

```
List< TransactionTemplateMapping >  
getAllTransactionTemplateMappings ()
```

getTemplate(int, int, int, String, boolean, boolean) method

Get a transaction messaging template for a use case, error code, status, and org unit.

Syntax

```
TransactionTemplateMapping getTemplate ( int useCase , int
errorCode , int status , String orgunit , boolean payer , boolean test )
```

Parameters

- **useCase** –
- **errorCode** –
- **status** –
- **orgunit** –
- **payer** – if the payer or payee template is requested
- **test** – if a test template is requested

getTransactionTemplateMappingsByOrgUnitAndUseCaseId(@MobiliserParam(name="orgunit", required=true, length=6, description="The orgunit to filter for") String, @MobiliserParam(name="useCaseId", required=true, length=5, description="The use case id to filter for") int) method

Returns all records for the given orgunit and use case id from the database.

Syntax

```
List< TransactionTemplateMapping >
getTransactionTemplateMappingsByOrgUnitAndUseCaseId
( @MobiliserParam(name="orgunit", required=true, length=6,
description="The orgunit to filter for") String orgunit ,
@MobiliserParam(name="useCaseId", required=true, length=5,
description="The use case id to filter for") int useCaseId )
```

Parameters

- **orgunit** –
- **useCaseId** –

getTransactionTemplateMappingsByUseCaseId(@MobiliserParam(name="useCaseId", required=true, length=5, description="The use case id to filter for") int) method

Get all transaction messaging templates for a use case id.

Syntax

```
List< TransactionTemplateMapping >
getTransactionTemplateMappingsByUseCaseId
```

```
( @MobiliserParam(name="useCaseId", required=true, length=5,
description="The use case id to filter for") int useCaseId)
```

Parameters

- **useCaseId** –

UmgrPrivilegeDAO interface

Syntax

```
public interface UmgrPrivilegeDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getUmgrPrivileges(String, String) method

Get all UmgrPrivileges based on given filter.

Syntax

```
List< UmgrPrivilege > getUmgrPrivileges ( String idFilter , String
descriptionFilter )
```

Parameters

- **idFilter** – optional filter on id
- **descriptionFilter** – optional filter on description

Usage

SQL wildcards are allowed in the filter value

UmgrRoleDAO interface

Syntax

```
public interface UmgrRoleDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getUmgrRoles(String, String) method

Get UmgrRoles based on filter.

Syntax

```
List< UmgrRole > getUmgrRoles ( String idFilter , String  
descriptionFilter )
```

Parameters

- **idFilter** – optional filter on id
- **descriptionFilter** – optional filter on description

Usage

SQL wildcards are allowed in the filter value

UmgrRolePrivilegeDAO interface

Syntax

```
public interface UmgrRolePrivilegeDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getPrivilegesForRole(String) method

Get assigned privileges for a role.

Syntax

```
Collection< String > getPrivilegesForRole ( String roleId )
```

Parameters

- **roleId** –

getUmgrRolePrivileges(String, String) method

get UmgrRolePrivileges for role and privilege id filter

Syntax

```
List< UmgrRolePrivilege > getUmgrRolePrivileges ( String roleId ,  
String privilegeId )
```

Parameters

- **roleId** – optional
- **privilegeId** – optional

UseCaseDAO interface

Syntax

```
public interface UseCaseDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

UseCaseFeeDAO interface

Syntax

```
public interface UseCaseFeeDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

isUnique(final UseCase, final boolean, final SubTransactionType, final PaymentInstrument.Type, final PaymentInstrument.Type, final OrgUnit, final OrgUnit, final FeeType) method

Check uniqueness of the given configuration.

Syntax

```
boolean isUnique ( final UseCase useCase , final boolean isPayee ,  
final SubTransactionType txnType , final PaymentInstrument.Type  
payeePiType , final PaymentInstrument.Type payerPiType , final  
OrgUnit payeeOrgUnit , final OrgUnit payerOrgUnit , final FeeType  
feeType )
```

Parameters

- **useCase** –
- **isPayee** –
- **txnType** –

- **payeePiType** –
- **payerPiType** –
- **payeeOrgUnit** –
- **payerOrgUnit** –
- **feeType** –

searchUseCaseFees(final UseCase, final SubTransactionType, final PaymentInstrument.Type, final PaymentInstrument.Type, final OrgUnit, final OrgUnit) method

Search for UseCaseFee based on the given criteria.

Syntax

```
List< UseCaseFee > searchUseCaseFees ( final UseCase useCase ,
final SubTransactionType txnType , final PaymentInstrument.Type
payeePiType , final PaymentInstrument.Type payerPiType , final
OrgUnit payeeOrgUnit , final OrgUnit payerOrgUnit )
```

Parameters

- **useCase** – the use case
- **txnType** – the txnType
- **payeePiType** – the payee payment instrument type
- **payerPiType** – the payer payment instrument type
- **payeeOrgUnit** – the payee org unit
- **payerOrgUnit** – the payer org unit

Returns

the list of use case fees

Usage

the list of use case fees

UseCasePaymentInstrumentTypeDAO interface

Syntax

```
public interface UseCasePaymentInstrumentTypeDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

findByUseCaseId(int) method

Returns a list of UseCasePaymentInstrumentType instances for the specified use case.

Syntax

```
List< UseCasePaymentInstrumentType > findByUseCaseId ( int  
useCaseId )
```

Parameters

- **useCaseId** –

findPiTypesforUseCaseId(int, boolean) method

Returns the list of supported payment instrument types for the given use case id and for either payer or payee.

Syntax

```
List< PaymentInstrument.Type > findPiTypesforUseCaseId ( int  
useCaseId , boolean payee )
```

Parameters

- **useCaseId** –
- **payee** –

UseCasePrivilegeDAO interface

Syntax

```
public interface UseCasePrivilegeDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

ValidationRuleDAO interface

Syntax

```
public interface ValidationRuleDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

getValidationExpressions(String, String, boolean) method

Returns an ordered list of validation expressions applicable to the passed Mobiliser service call.

Syntax

```
List< String > getValidationExpressions ( String endpoint , String method , boolean request )
```

Parameters

- **endpoint** – the endpoint interface name
- **method** – the method name
- **request** – indicated is request or response validation rules are required

Returns

an ordered list of applicable validation expressions

Usage

an ordered list of applicable validation expressions

VersionDAO interface

Syntax

```
public interface VersionDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

WrkBulkProcessingDAO interface

Syntax

```
public interface WrkBulkProcessingDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

transaction package

Members

All public members of the transaction package.

- **AuthorisationCancelDAO interface** –
- **AuthorisationCodeDAO interface** –
- **AuthorisationDAO interface** –
- **CaptureCancelDAO interface** –
- **CaptureDAO interface** –
- **CommissionClearingDAO interface** –
- **CommissionDAO interface** –
- **FeeDAO interface** –
- **InvoiceTransactionDAO interface** –
- **SubTransactionDAO interface** –
- **SvaBookingDAO interface** –
- **TraceableRequestDAO interface** – W.
- **TransactionAttributeDAO interface** –
- **TransactionDAO interface** –
- **XmlMessageDAO interface** –

AuthorisationCancelDAO interface

Syntax

```
public interface AuthorisationCancelDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

findById(long) method

Find by transaction id (includes previously failed ones)

Syntax

```
List< AuthorisationCancel > findById ( long  
transactionId )
```

Parameters

- **transactionId** –

findSuccessfulByTransactionId(long) method

Find successful authorization cancel for a transaction.

Syntax

```
AuthorisationCancel findSuccessfulByTransactionId ( long  
transactionId )
```

Parameters

- **transactionId** –

AuthorisationCodeDAO interface

Syntax

```
public interface AuthorisationCodeDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getNextAuthCode() method

Returns the next available auth code.

Syntax

```
String getNextAuthCode ()
```

Returns

the auth code

Usage

the auth code

AuthorisationDAO interface

Syntax

```
public interface AuthorisationDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

findAuthorisationsForTransaction(final long) method

Syntax

```
List< Authorisation > findAuthorisationsForTransaction ( final  
long transactionId )
```

Parameters

- **transactionId** – the money transaction id

Returns

the list of authorisation

Usage

Returns a list of Authorisations for the given transaction id, sorted by the authorisation's transaction id in descending order.

the list of authorisation

findByTransactionId(long) method

Find authorization by transaction.

Syntax

```
Authorisation findByTransactionId ( long transactionId )
```

Parameters

- **transactionId** –

getSuccessfulAuthorisation(final long) method

Syntax

```
Authorisation getSuccessfulAuthorisation ( final long  
transactionId )
```

Parameters

- **transactionId** – the id of the money transaction

Returns

the appropriate `Authorisation`; null if such a transaction doesn't exist

Usage

Returns the successful `Authorisation` belonging to the given

Money transaction

.

the appropriate `Authorisation`; null if such a transaction doesn't exist

CaptureCancelDAO interface

Syntax

```
public interface CaptureCancelDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

findByTransactionId(long) method

Find all capture cancel sub transaction for a transaction, including failed cancels.

Syntax

```
List< CaptureCancel > findByTransactionId ( long transactionId )
```

Parameters

- **transactionId** –

findSuccessfulByTransactionId(long) method

Returns the successful capture cancel sub transaction.

Syntax

```
CaptureCancel findSuccessfulByTransactionId ( long transactionId )
```

Parameters

- **transactionId** –

CaptureDAO interface

Syntax

```
public interface CaptureDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

findById(long) method

Find all capture sub transactions.

Syntax

```
List< Capture > findById ( long transactionId )
```

Parameters

- **transactionId** –

findSuccessfulByTransactionId(long) method

Find the successful capture for a transaction.

Syntax

```
Capture findSuccessfulByTransactionId ( long transactionId )
```

Parameters

- **transactionId** –

CommissionClearingDAO interface

Syntax

```
public interface CommissionClearingDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getClearingInfoForCustomer(long) method

Aggregate the amount per clearing type of all open commissions.

Syntax

```
List< Map< String, Number > > getClearingInfoForCustomer ( long  
customerId )
```

Parameters

- **customerId** – the beneficiary

getCustomersWithClearingFrequency(String) method

Get a list of customers (identified by its customer ID) for a given clearing frequency.

Syntax

```
List< Long > getCustomersWithClearingFrequency ( String  
paramClrFrq )
```

Parameters

- **paramClrFrq** –

Returns

customer IDs

Usage

customer IDs

setCommissionsToInProgress(long, Calendar) method

Update the pending commission clearing to status in progress

COMMISSION_STATUS_IN_PROGRESS.

Syntax

```
void setCommissionsToInProgress ( long customerId , Calendar  
calendar )
```

Parameters

- **customerId** – beneficiary
- **calendar** – clearing cutoff

updateCommissionsAndSettlementDateUponFailure(long, int) method

Set all commissions in status INPROGRESS to status SETTLED for the customer and the fee type of this iteration after failure.

Syntax

```
void updateCommissionsAndSettlementDateUponFailure ( long  
customerId , int feeType )
```

Parameters

- **customerId** –
- **feeType** –

*updateCommissionsAndSettlementDateUponSuccess(long,
CommissionConfiguration, int, long) method*

Set all commissions in status INPROGRESS to status SETTLED for the customer and the fee type of this iteration.

Syntax

```
void updateCommissionsAndSettlementDateUponSuccess ( long  
customerId , CommissionConfiguration cc , int feeType , long txnId )
```

Parameters

- **customerId** –
- **cc** –
- **feeType** –
- **txnId** –

COMMISSION_STATUS_IN_PROGRESS variable

Indicates that the commission is currently in progress.

Syntax

```
final int COMMISSION_STATUS_IN_PROGRESS
```

COMMISSION_STATUS_OPEN variable

Indicates that the commission has not been settled.

Syntax

```
final int COMMISSION_STATUS_OPEN
```


COMMISSION_STATUS_SETTLED variable

Indicates that the commission has been cleared and settled.

Syntax

```
final int COMMISSION_STATUS_SETTLED
```

CommissionDAO interface***Syntax***

```
public interface CommissionDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCommissionsByCustomer(final long, int) method

Get all commissions of a certain status.

Syntax

```
List< Commission > getCommissionsByCustomer ( final long  
customerId , int status )
```

Parameters

- **customerId** – beneficiary
- **status** – clearing status

getCommissionsByTxn(long) method

Get all commissions that applies for a given transaction.

Syntax

```
List< Commission > getCommissionsByTxn ( long txnId )
```

Parameters

- **txnId** – transaction ID

FeeDAO interface***Syntax***

```
public interface FeeDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

InvoiceTransactionDAO interface

Syntax

```
public interface InvoiceTransactionDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getInvoiceIDfromTransactionID(long) method

Get the invoice ID from a transaction.

Syntax

```
Long getInvoiceIDfromTransactionID ( long transactionID )
```

Parameters

- **transactionID** –

Returns

null if not found

Usage

null if not found

SubTransactionDAO interface

Syntax

```
public interface SubTransactionDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getByPaymentInstrumentIdAndErrorCodes(final long, List< Integer >, Date, Date)
method

Get Subtransactions by payment instrumentId and error code list.

Syntax

```
List< SubTransaction > getByPaymentInstrumentIdAndErrorCodes
( final long paymentInstrumentId , List< Integer > errorCodeList , Date
fromDate , Date toDate )
```

Parameters

- **paymentInstrumentId** –
- **errorCodeList** –
- **fromDate** –
- **toDate** –

getByTxnId(long) method

Get all sub transactions for a transaction.

Syntax

```
List< SubTransaction > getByTxnId ( long id )
```

Parameters

- **id** –

getLastBillableByPaymentInstrumentId(long, Date) method

Get last successful sub transaction (error code=0) that has a billable flag.

Syntax

```
SubTransaction getLastBillableByPaymentInstrumentId ( long
paymentInstrumentId , Date refDate )
```

Parameters

- **paymentInstrumentId** –
- **refDate** –

SvaBookingDAO interface

Syntax

```
public interface SvaBookingDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

TraceableRequestDAO interface

W.

Syntax

```
public interface TraceableRequestDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

findByOriginTraceNoAndAge(final String, final String, final int) method

Returns the response for the given system and trace number that has a maximum age of maxAgeHours (+/-1).

Syntax

```
TraceableRequest findByOriginTraceNoAndAge ( final String origin ,  
final String traceNo , final int maxAgeHours )
```

Parameters

- **origin** –
- **traceNo** –
- **maxAgeHours** –

Returns

the bean or null if no match is found

Usage

the bean or null if no match is found

TransactionAttributeDAO interface

Syntax

```
public interface TransactionAttributeDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getTransactionAttribute(long) method

Get all TransactionAttributes for a given transaction.

Syntax

```
List< TransactionAttribute > getTransactionAttribute ( long
trxID )
```

Parameters

- **trxID** – transaction ID

TransactionDAO interface

Syntax

```
public interface TransactionDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

countOpenTransactions(final long) method

Counts for open (initial) Transactions for a money consumer where the customer is the payer.

Syntax

```
long countOpenTransactions ( final long customerId )
```

Parameters

- **customer** – the customer whose transactions are requested.

Returns

count of open transactions

Usage

count of open transactions

expireAllInitial(final long, final boolean) method

Expire all initial transactions older than maxAge milliseconds initial transactions have transaction status 0 or 2 and error code 0.

Syntax

```
int expireAllInitial ( final long maxAge , final boolean  
includeLocked )
```

Parameters

- **maxAge** – in milliseconds
- **includeLocked** – also include locked transactions

Returns

number of expired transactions

Usage

number of expired transactions

findAuthorisedTransactionsForPayee(final long) method

Find all successful authorised transaction for a certain payee Returns only transaction in state 20 - authorised and code 0 - OK.

Syntax

```
List< Transaction > findAuthorisedTransactionsForPayee ( final  
long payee )
```

Parameters

- **payee** –

findByAuthId(final String) method

Find Transactions by their auth code, returning them order descending by transaction id.

Syntax

```
List< Transaction > findByAuthId ( final String authId )
```

Parameters

- **authId** – the auth code to search for

Returns

the list of transactions

Usage

the list of transactions

findExpiredTransactions() method

Syntax

```
List< Transaction > findExpiredTransactions ()
```

Usage

Returns a list of

Transaction

s that are expired, i.e. where the

Transaction#getExpiration()

lies in the past

findInitialTransactions(final long, final boolean, final int, final int) method

Find all initial transactions older than maxAge milliseconds initial transactions have transaction status [fromState..toState[and error code 0.

Syntax

```
List< Transaction > findInitialTransactions ( final long maxAge ,  
final boolean includeLocked , final int fromState , final int toState )
```

Parameters

- **maxAge** – milliseconds
- **includeLocked** – also include locked transactions
- **fromState** – minimum include transaction state
- **toState** – maximum exclude transaction state

findTransactions(Long, Long, Long, Date, Date, Boolean, Integer, String, Integer, int, Long, Boolean, Boolean) method

Syntax

```
List< FindTransactionsResultBean > findTransactions ( Long  
customerId , Long joinedCustomerId , Long caller , Date fromDate , Date  
toDate , Boolean showFaulty , Integer statusFilter , String orderIdFilter ,
```

Integer *maxHierarchyLevel* , int *maxRecords* , Long *paymentInstrumentId* ,
Boolean *customerIsPayer* , Boolean *showInitial*) throws
IllegalArgumentException

getOpenTransactions(final String, final long, final long) method
Returns the list of open transaction matching the given criteria.

Syntax

```
List< Transaction > getOpenTransactions ( final String orderId ,  
final long payeeId , final long amount )
```

Parameters

- **orderId** – the order id of the transaction
- **payeeId** – the payee's id
- **amount** – the amount of the transaction

Returns

the list of matching transactions

Usage

the list of matching transactions

unlockIdleTransactions(final long) method
Unlock idle locked transactions older than *maxAge* milliseconds.

Syntax

```
int unlockIdleTransactions ( final long maxAge )
```

Parameters

- **maxAge** – milliseconds

Returns

number of unlocked/updated transactions

Usage

Please note, it only unlocks transactions with error code 0

number of unlocked/updated transactions

*XmlMessageDAO interface**Syntax*

```
public interface XmlMessageDAO
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

*util package**Members*

All public members of the util package.

- **DAddress class** – Helper class that contains an Address bean and a distance.
- **ExtendedCustomerSearchResult class** – Returnd by the customer extended search method.
- **FindInternalVoucherResultBean class** – Complex bean returned by findInternalVoucher.
- **FindTransactionsResultBean class** – Complex result bean returned by findTransactions.
- **GLAccountBean class** – Complex type returned by GL Account methods.
- **GLAccountTreeElementBean class** – Represents a hierarchical GLAccount tree node.
- **IPiInterestCalculatingVisitor interface** –
- **IPiInterestClearingVisitor interface** –
- **IProcessDueInvoiceVisitor interface** –
- **LookupEntity class** –
- **PaginationInfo class** –

DAddress class

Helper class that contains an Address bean and a distance.

Syntax

```
public class DAddress
```

Usage

Comparable allows to sort list of DAddresses by distance

2012 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

DAddress(double, Address) constructor

Constructor with given field values.

Syntax

DAddress (double *distance* , Address *location*)

Parameters

- **distance** – in meter
- **location** – a address bean with logitude and latitude

compareTo(DAddress) method

Syntax

int compareTo (DAddress *arg0*)

getAddress() method

Get the address bean field.

Syntax

Address getAddress ()

getDistance() method

get the distance in meter

Syntax

double getDistance ()

ExtendedCustomerSearchResult class

Returnd by the customer extended search method.

Syntax

```
public class ExtendedCustomerSearchResult
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCustomer() method

Syntax

Customer getCustomer ()

getHierarchyLevel() method

Syntax

Integer getHierarchyLevel ()

getIdentifications() method

Syntax

List< Identification > getIdentifications ()

setCustomer(Customer) method

Syntax

void setCustomer (Customer *customer*)

setHierarchyLevel(Integer) method

Syntax

void setHierarchyLevel (Integer *hierarchyLevel*)

setIdentifications(List< Identification >) method

Syntax

void setIdentifications (List< Identification > *identifications*)

FindInternalVoucherResultBean class

Complex bean returned by findInternalVoucher.

Syntax

```
public class FindInternalVoucherResultBean
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

MoneyAmount class

Syntax

```
public class MoneyAmount
```

MoneyAmount(long, Currency) constructor

Syntax

```
MoneyAmount ( long amount , Currency currency )
```

getAmount() method

Syntax

```
long getAmount ()
```

getCurrency() method

Syntax

```
Currency getCurrency ()
```

setAmount(long) method

Syntax

```
void setAmount ( long amount )
```

setCurrency(Currency) method

Syntax

```
void setCurrency ( Currency currency )
```

amount variable

Syntax

```
long amount
```

currency variable

Syntax

```
Currency currency
```

FindInternalVoucherResultBean() constructor

Syntax

FindInternalVoucherResultBean ()

FindInternalVoucherResultBean(Date, MoneyAmount, long, long) constructor

Syntax

FindInternalVoucherResultBean (Date *dateTime* , MoneyAmount *originalAmount* , long *txnId* , long *voucherId*)

getDateTime() method

Syntax

Date getDateTime ()

getOriginalAmount() method

Syntax

MoneyAmount getOriginalAmount ()

getTxnId() method

Syntax

long getTxnId ()

getVoucherId() method

Syntax

long getVoucherId ()

setDateTime(Date) method

Syntax

void setDateTime (Date *dateTime*)

setOriginalAmount(MoneyAmount) method

Syntax

void setOriginalAmount (MoneyAmount *originalAmount*)

setTxnId(long) method

Syntax

```
void setTxnId ( long txnId )
```

setVoucherId(long) method

Syntax

```
void setVoucherId ( long voucherId )
```

FindTransactionsResultBean class

Complex result bean returned by findTransactions.

Syntax

```
public class FindTransactionsResultBean
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

FindTransactionsResultBean() constructor

Syntax

```
FindTransactionsResultBean ()
```

FindTransactionsResultBean(long, Date, int, int, int, String, long, String, long, int, long, long, int, String, Long, String, long, long, long, String, String, String) constructor

Syntax

```
FindTransactionsResultBean ( long id , Date creationDate , int useCase ,  
int errorCode , int status , String authCode , long payerId , String  
payerDisplayName , long payerPild , int payerPiType , long payeeId , long  
payeePild , int payeePiType , String payeeDisplayName , Long callerId ,  
String text , long amount , long payerAmount , long payeeAmount , String  
currency , String payerCurrency , String payeeCurrency )
```

getAmount() method

Syntax

```
long getAmount ()
```

getAuthCode() method

Syntax

String getAuthCode ()

getCallerId() method

Syntax

Long getCallerId ()

getCreationDate() method

Syntax

Date getCreationDate ()

getCurrency() method

Syntax

String getCurrency ()

getErrorCode() method

Syntax

int getErrorCode ()

getId() method

Syntax

long getId ()

getPayeeAmount() method

Syntax

long getPayeeAmount ()

getPayeeCurrency() method

Syntax

String getPayeeCurrency ()

getPayeeDisplayName() method

Syntax

String getPayeeDisplayName ()

getPayeeId() method

Syntax

long getPayeeId ()

getPayeePild() method

Syntax

long getPayeePiId ()

getPayeePiType() method

Syntax

int getPayeePiType ()

getPayerAmount() method

Syntax

long getPayerAmount ()

getPayerCurrency() method

Syntax

String getPayerCurrency ()

getPayerDisplayName() method

Syntax

String getPayerDisplayName ()

getPayerId() method

Syntax

long getPayerId ()

getPayerPild() method

Syntax

long getPayerPiId ()

getPayerPiType() method

Syntax

int getPayerPiType ()

getStatus() method

Syntax

int getStatus ()

getText() method

Syntax

String getText ()

getUseCase() method

Syntax

int getUseCase ()

setAmount(long) method

Syntax

void setAmount (long *amount*)

setAuthCode(String) method

Syntax

void setAuthCode (String *authCode*)

setCallerId(Long) method

Syntax

void setCallerId (Long *callerId*)

setCreationDate(Date) method

Syntax

void setCreationDate (Date *creationDate*)

setCurrency(String) method

Syntax

void setCurrency (String *currency*)

setErrorCode(int) method

Syntax

void setErrorCode (int *errorCode*)

setId(long) method

Syntax

void setId (long *id*)

setPayeeAmount(BigDecimal) method

Syntax

void setPayeeAmount (BigDecimal *payeeAmount*)

setPayeeCurrency(String) method

Syntax

void setPayeeCurrency (String *payeeCurrency*)

setPayeeDisplayName(String) method

Syntax

void setPayeeDisplayName (String *payeeDisplayName*)

setPayeeId(long) method

Syntax

void setPayeeId (long *payeeId*)

setPayeePild(long) method

Syntax

void setPayeePiId (long *payeePiId*)

setPayeePiType(int) method

Syntax

void setPayeePiType (int *payeePiType*)

setPayerAmount(BigDecimal) method

Syntax

void setPayerAmount (BigDecimal *payerAmount*)

setPayerCurrency(String) method

Syntax

void setPayerCurrency (String *payerCurrency*)

setPayerDisplayName(String) method

Syntax

void setPayerDisplayName (String *payerDisplayName*)

setPayerId(long) method

Syntax

void setPayerId (long *payerId*)

setPayerPild(long) method

Syntax

void setPayerPiId (long *payerPild*)

setPayerPiType(int) method

Syntax

void setPayerPiType (int *payerPiType*)

setStatus(int) method

Syntax

```
void setStatus ( int status )
```

setText(String) method

Syntax

```
void setText ( String text )
```

setUseCase(int) method

Syntax

```
void setUseCase ( int useCase )
```

GLAccountBean class

Complex type returned by GL Account methods.

Syntax

```
public class GLAccountBean
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

GLAccountBean(Long, long, int, boolean, String, String, String, String, String, String) constructor

Syntax

```
GLAccountBean ( Long parentAccountId , long id , int accountType ,  
boolean isActive , String glAccountName , String glAccountNumber ,  
String bankAccountNumber , String bankBranchCode , String bankCode ,  
String orgUnit )
```

getAccountType() method

Syntax

```
int getAccountType ()
```

getBankAccountNumber() method

Syntax

String getBankAccountNumber ()

getBankBranchCode() method

Syntax

String getBankBranchCode ()

getBankCode() method

Syntax

String getBankCode ()

getGlAccountName() method

Syntax

String getGlAccountName ()

getGlAccountNumber() method

Syntax

String getGlAccountNumber ()

getId() method

Syntax

long getId ()

getOrgUnit() method

Syntax

String getOrgUnit ()

getParentAccountId() method

Syntax

Long getParentAccountId ()

isActive() method

Syntax

boolean *isActive* ()

setAccountType(int) method

Syntax

void *setAccountType* (int *accountType*)

setActive(boolean) method

Syntax

void *setActive* (boolean *isActive*)

setBankAccountNumber(String) method

Syntax

void *setBankAccountNumber* (String *bankAccountNumber*)

setBankBranchCode(String) method

Syntax

void *setBankBranchCode* (String *bankBranchCode*)

setBankCode(String) method

Syntax

void *setBankCode* (String *bankCode*)

setGIAccountName(String) method

Syntax

void *setGIAccountName* (String *glAccountName*)

setGIAccountNumber(String) method

Syntax

void *setGIAccountNumber* (String *glAccountNumber*)

setId(long) method

Syntax

void setId (long *id*)

setOrgUnit(String) method

Syntax

void setOrgUnit (String *orgUnit*)

setParentAccountId(Long) method

Syntax

void setParentAccountId (Long *parentAccountId*)

accountType variable

Syntax

int accountType

bankAccountNumber variable

Syntax

String bankAccountNumber

bankBranchCode variable

Syntax

String bankBranchCode

bankCode variable

Syntax

String bankCode

glAccountName variable

Syntax

String glAccountName

glAccountNumber variable

Syntax

String glAccountNumber

id variable

Syntax

long id

isActive variable

Syntax

boolean isActive

orgUnit variable

Syntax

String orgUnit

parentAccountId variable

Syntax

Long parentAccountId

GLAccountTreeElementBean class

Represents a hierarchical GLAccount tree node.

Syntax

```
public class GLAccountTreeElementBean
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

GLAccountTreeElementBean(Long, long, String, String, int) constructor

Syntax

```
GLAccountTreeElementBean ( Long parentId , long id , String  
accountName , String accountNumber , int accountType )
```


getAccountName() method

Syntax

String getAccountName ()

getAccountNumber() method

Syntax

String getAccountNumber ()

getAccountType() method

Syntax

int getAccountType ()

getId() method

Syntax

long getId ()

getParentId() method

Syntax

Long getParentId ()

accountName variable

Syntax

String accountName

accountNumber variable

Syntax

String accountNumber

accountType variable

Syntax

int accountType

id variable

Syntax

long id

parentId variable

Syntax

Long parentId

IPiInterestCalculatingVisitor interface

Syntax

```
public interface IPiInterestCalculatingVisitor
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

calculateInterest(long, InterestProduct, List< InterestBalanceSnapshot >, Date, Long) method

Syntax

```
boolean calculateInterest ( long pild , InterestProduct product ,  
List< InterestBalanceSnapshot > balanceHistoryL , Date currentCutoff ,  
Long lock )
```

doInterestTransactions(InterestProduct, BigDecimal, BigDecimal, Long) method

Syntax

```
void doInterestTransactions ( InterestProduct product , BigDecimal  
totalDebitInterest , BigDecimal totalCreditInterest , Long lock )
```

IPiInterestClearingVisitor interface

Syntax

```
public interface IPiInterestClearingVisitor
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

clearInterest(long, long, Currency, InterestProduct, Date, Long, Long, Long) method

Syntax

```
boolean clearInterest ( long pild , long customerId , Currency currency ,
InterestProduct product , Date cutoff , Long totalCreditInterestAmount ,
Long totalDebitInterestAmount , Long lock )
```

IProcessDueInvoiceVisitor interface

Syntax

```
public interface IProcessDueInvoiceVisitor
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

processDueInvoice(long, Date, long) method

Syntax

```
boolean processDueInvoice ( long invoiceId , Date payDate , long
amount )
```

LookupEntity class

Syntax

```
public class LookupEntity
```

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

LookupEntity() constructor

Syntax

```
LookupEntity ()
```

LookupEntity(String, String) constructor

Syntax

LookupEntity (String *id*, String *name*)

getId() method

Syntax

String getId ()

getName() method

Syntax

String getName ()

setId(String) method

Syntax

void setId (String *id*)

setName(String) method

Syntax

void setName (String *name*)

PaginationInfo class

Syntax

public class PaginationInfo

Usage

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

PaginationInfo() constructor

Syntax

PaginationInfo ()

PaginationInfo(String, boolean, int, int) constructor

Syntax

```
PaginationInfo ( String orderColumn , boolean orderAsc , int firstRow ,
int maxResults )
```

getFirstRow() method

Syntax

```
final int getFirstRow ()
```

getMaxResults() method

Syntax

```
final int getMaxResults ()
```

getOrderColumn() method

Syntax

```
final String getOrderColumn ()
```

isOrderAsc() method

Syntax

```
final boolean isOrderAsc ()
```

setFirstRow(int) method

Syntax

```
final void setFirstRow ( int firstRow )
```

setMaxResults(int) method

Syntax

```
final void setMaxResults ( int maxResults )
```

setOrderAsc(boolean) method

Syntax

```
final void setOrderAsc ( boolean orderAsc )
```

setOrderColumn(String) method

Syntax

```
final void setOrderColumn ( String orderColumn )
```

model package

Members

All public members of the model package.

- **customer package** –
- **pi package** –
- **system package** –
- **transaction package** –

customer package

Members

All public members of the customer package.

- **Address class** –
- **Attachment class** –
- **AuditLog class** –
- **CommissionConfiguration class** –
- **Credential class** –
- **Customer class** –
- **CustomerAttribute class** –
- **CustomerNetwork class** –
- **CustomerOtp class** –
- **CustomerPersistentLogin class** –
- **CustomerPrivilege class** –
- **CustomerRole class** –
- **CustomerSession class** –
- **HistoryEntry class** –
- **Identification class** –
- **Identity class** –
- **Invoice class** –
- **InvoiceAttribute class** –
- **InvoiceConfiguration class** –
- **InvoiceConfigurationAttribute class** –

- **LimitSum class** –
- **Note class** –
- **Otp class** –
- **WrkCustomer class** –

Address class

Syntax

```
public class Address
```

Usage

The `Address` represents a customer's address. An address may be a postal address, an MSISDN, an email etc.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Address() constructor

Syntax

```
Address ()
```

Usage

Creates a new instance of

```
Address
```

.

getAccuracy() method

Syntax

```
Integer getAccuracy ()
```

getAddressSince() method

Syntax

```
Date getAddressSince ()
```

Returns

the `addressSince` value

Usage

Returns the

addressSince

of this instance.

the `addressSince` value

getAddressStatus() method

Syntax

```
int getAddressStatus ()
```

getAddressType() method

Syntax

```
int getAddressType ()
```

getAlias() method

Syntax

```
String getAlias ()
```

Returns

the `alias` value

Usage

Returns the

alias

of this instance.

the `alias` value

getCity() method

Syntax

```
String getCity ()
```

Returns

the `city` value

Usage

Returns the

city

of this instance.

the `city` value

getCompany1() method

Syntax

`String getCompany1 ()`

Returns

the `company1` value

Usage

Returns the

company1

of this instance.

the `company1` value

getCompany2() method

Syntax

`String getCompany2 ()`

Returns

the `company2` value

Usage

Returns the

company2

of this instance.

the `company2` value

getCompanyShortname() method

Syntax

String getCompanyShortname ()

Returns

the companyShortname value

Usage

Returns the

companyShortname

of this instance.

the companyShortname value

getCountryId() method

Syntax

String getCountryId ()

Returns

the address' Country

Usage

Returns the country of this address.

the address' Country

getCustomer() method

Syntax

Customer getCustomer ()

Returns

the address' owning Customer

Usage

Returns the customer this address belongs to.

the address' owning Customer

getEmail() method

Syntax

String getEmail ()

Returns

the email value

Usage

Returns the

email

of this instance.

the email value

getFax() method

Syntax

String getFax ()

Returns

the fax value

Usage

Returns the

fax

of this instance.

the fax value

getFirstName() method

Syntax

String getFirstName ()

Returns

the firstName value

Usage

Returns the

firstName

of this instance.

the `firstName` value

getGender() method

Syntax

Integer `getGender ()`

Returns

the address' resp. customer's Gender

Usage

Returns the gender of this instance.

the address' resp. customer's Gender

getHouseNumber() method

Syntax

String `getHouseNumber ()`

Returns

the `houseNumber` value

Usage

Returns the

houseNumber

of this instance.

the `houseNumber` value

getLastName() method

Syntax

String `getLastName ()`

Returns

the `lastName` value

Usage

Returns the

lastName

of this instance.

the `lastName` value

getLatitude() method

Syntax

`BigDecimal getLatitude ()`

getLocationType() method

Syntax

`Integer getLocationType ()`

getLongitude() method

Syntax

`BigDecimal getLongitude ()`

getMiddleName() method

Syntax

`String getMiddleName ()`

Returns

the `middleName` value

Usage

Returns the

middleName

of this instance.

the `middleName` value

getPhone1() method

Syntax

String getPhone1 ()

Returns

the phone1 value

Usage

Returns the

phone1

of this instance.

the phone1 value

getPhone2() method

Syntax

String getPhone2 ()

Returns

the phone2 value

Usage

Returns the

phone2

of this instance.

the phone2 value

getPosition() method

Syntax

String getPosition ()

Returns

the position value

Usage

Returns the

position

of this instance.

the position value

getState() method

Syntax

String getState ()

Returns

the state value

Usage

Returns the

state

of this instance.

the state value

getStreet1() method

Syntax

String getStreet1 ()

Returns

the street1 value

Usage

Returns the

street1

of this instance.

the street1 value

getStreet2() method

Syntax

String getStreet2 ()

Returns

the `street2` value

Usage

Returns the

street2

of this instance.

the `street2` value

getTitle() method

Syntax

String getTitle ()

Returns

the `title` value

Usage

Returns the

title

of this instance.

the `title` value

getUrl() method

Syntax

String getUrl ()

Returns

the `url` value

Usage

Returns the

url

of this instance.

the `url` value

getZip() method

Syntax

`String getZip ()`

Returns

the `zip` value

Usage

Returns the

zip

of this instance.

the `zip` value

isSetAccuracy() method

Syntax

`boolean isSetAccuracy ()`

isSetAddressSince() method

Syntax

`boolean isSetAddressSince ()`

Returns

`false` if the `addressSince` is `null`. Otherwise, it returns `true`.

Usage

Checks whether the

addressSince

of this instance is set.

false if the `addressSince` is null. Otherwise, it returns true.

isSetAddressStatus() method

Syntax

`boolean isSetAddressStatus ()`

Returns

false if the `addressStatus` is null. Otherwise, it returns true.

Usage

Checks whether the

addressStatus

of this instance is set.

false if the `addressStatus` is null. Otherwise, it returns true.

isSetAddressType() method

Syntax

`boolean isSetAddressType ()`

Returns

false if the `addressType` is null. Otherwise, it returns true.

Usage

Checks whether the

addressType

of this instance is set.

false if the `addressType` is null. Otherwise, it returns true.

isSetAlias() method

Syntax

`boolean isSetAlias ()`

Returns

false if the `alias` is null. Otherwise, it returns true.

Usage

Checks whether the

alias

of this instance is set.

false if the `alias` is null. Otherwise, it returns true.

isSetCity() method

Syntax

```
boolean isSetCity ()
```

Returns

false if the `city` is null. Otherwise, it returns true.

Usage

Checks whether the

city

of this instance is set.

false if the `city` is null. Otherwise, it returns true.

isSetCompany1() method

Syntax

```
boolean isSetCompany1 ()
```

Returns

false if the `company1` is null. Otherwise, it returns true.

Usage

Checks whether the

company1

of this instance is set.

false if the `company1` is null. Otherwise, it returns true.

isSetCompany2() method

Syntax

boolean isSetCompany2 ()

Returns

false if the company2 is null. Otherwise, it returns true.

Usage

Checks whether the

company2

of this instance is set.

false if the company2 is null. Otherwise, it returns true.

isSetCompanyShortname() method

Syntax

boolean isSetCompanyShortname ()

Returns

false if the companyShortname is null. Otherwise, it returns true.

Usage

Checks whether the

companyShortname

of this instance is set.

false if the companyShortname is null. Otherwise, it returns true.

isSetCountryId() method

Syntax

boolean isSetCountryId ()

Returns

false if the countryId is null . Otherwise, it returns true.

Usage

Checks whether the

countryId

of this instance is set.

false if the *countryId* is null . Otherwise, it returns true.

isSetCustomer() method

Syntax

```
boolean isSetCustomer ()
```

Returns

false if the customer is null. Otherwise, it returns true.

Usage

Checks whether the

customer

of this instance is set.

false if the customer is null. Otherwise, it returns true.

isSetEmail() method

Syntax

```
boolean isSetEmail ()
```

Returns

false if the email is null. Otherwise, it returns true.

Usage

Checks whether the

email

of this instance is set.

false if the email is null. Otherwise, it returns true.

isSetFax() method

Syntax

boolean isSetFax ()

Returns

false if the fax is null. Otherwise, it returns true.

Usage

Checks whether the

fax

of this instance is set.

false if the fax is null. Otherwise, it returns true.

isSetFirstName() method

Syntax

boolean isSetFirstName ()

Returns

false if the firstName is null . Otherwise, it returns true.

Usage

Checks whether the

firstName

of this instance is set.

false if the firstName is null . Otherwise, it returns true.

isSetGender() method

Syntax

boolean isSetGender ()

Returns

false if the gender is null. Otherwise, it returns true.

Usage

Checks whether the

gender

of this instance is set.

false if the gender is null. Otherwise, it returns true.

isSetHouseNumber() method

Syntax

```
boolean isSetHouseNumber ()
```

Returns

false if the houseNumber is null. Otherwise, it returns true.

Usage

Checks whether the

houseNumber

of this instance is set.

false if the houseNumber is null. Otherwise, it returns true.

isSetLastName() method

Syntax

```
boolean isSetLastName ()
```

Returns

false if the lastName is null. Otherwise, it returns true.

Usage

Checks whether the

lastName

of this instance is set.

false if the lastName is null. Otherwise, it returns true.

isSetLatitude() method

Syntax

boolean isSetLatitude ()

isSetLocationType() method

Syntax

boolean isSetLocationType ()

isSetLongitude() method

Syntax

boolean isSetLongitude ()

isSetMiddleName() method

Syntax

boolean isSetMiddleName ()

Returns

false if the middleName is null. Otherwise, it returns true.

Usage

Checks whether the

middleName

of this instance is set.

false if the middleName is null. Otherwise, it returns true.

isSetPhone1() method

Syntax

boolean isSetPhone1 ()

Returns

false if the phone1 is null. Otherwise, it returns true.

Usage

Checks whether the

phone1

of this instance is set.

false if the *phone1* is null. Otherwise, it returns true.

isSetPhone2() method

Syntax

```
boolean isSetPhone2 ()
```

Returns

false if the *phone2* is null. Otherwise, it returns true.

Usage

Checks whether the

phone2

of this instance is set.

false if the *phone2* is null. Otherwise, it returns true.

isSetPosition() method

Syntax

```
boolean isSetPosition ()
```

Returns

false if the *position* is null. Otherwise, it returns true.

Usage

Checks whether the

position

of this instance is set.

false if the *position* is null. Otherwise, it returns true.

isSetState() method

Syntax

```
boolean isSetState ()
```

Returns

false if the state is null. Otherwise, it returns true.

Usage

Checks whether the

state

of this instance is set.

false if the state is null. Otherwise, it returns true.

isSetStreet1() method

Syntax

```
boolean isSetStreet1 ()
```

Returns

false if the street1 is null. Otherwise, it returns true.

Usage

Checks whether the

street1

of this instance is set.

false if the street1 is null. Otherwise, it returns true.

isSetStreet2() method

Syntax

```
boolean isSetStreet2 ()
```

Returns

false if the street2 is null. Otherwise, it returns true.

Usage

Checks whether the

street2

of this instance is set.

false if the street2 is null. Otherwise, it returns true.

isSetTitle() method

Syntax

`boolean isSetTitle ()`

Returns

false if the title is null. Otherwise, it returns true.

Usage

Checks whether the

title

of this instance is set.

false if the title is null. Otherwise, it returns true.

isSetUrl() method

Syntax

`boolean isSetUrl ()`

Returns

false if the url is null. Otherwise, it returns true.

Usage

Checks whether the

url

of this instance is set.

false if the url is null. Otherwise, it returns true.

isSetZip() method

Syntax

`boolean isSetZip ()`

Returns

false if the zip is null. Otherwise, it returns true.

Usage

Checks whether the

zip

of this instance is set.

false if the zip is null. Otherwise, it returns true.

setAccuracy(Integer) method

Syntax

void setAccuracy (Integer *accuracy*)

setAddressSince(Date) method

Syntax

void setAddressSince (Date *addressSince*)

Parameters

- **addressSince** – the addressSince value to set

Usage

Sets the

addressSince

of this instance.

setAddressStatus(int) method

Syntax

void setAddressStatus (int *addressStatus*)

setAddressType(int) method

Syntax

void setAddressType (int *addressType*)

setAlias(String) method

Syntax

void setAlias (String *alias*)

Parameters

- **alias** – the `alias` value to set

Usage

Sets the

alias

of this instance.

setCity(String) method

Syntax

```
void setCity ( String city )
```

Parameters

- **city** – the `city` value to set

Usage

Sets the

city

of this instance.

setCompany1(String) method

Syntax

```
void setCompany1 ( String company1 )
```

Parameters

- **company1** – the `company1` value to set

Usage

Sets the

company1

of this instance.

setCompany2(String) method

Syntax

void setCompany2 (String *company2*)

Parameters

- **company2** – the `company2` value to set

Usage

Sets the

company2

of this instance.

setCompanyShortname(String) method

Syntax

void setCompanyShortname (String *companyShortname*)

Parameters

- **companyShortname** – the `companyShortname` value to set

Usage

Sets the

companyShortname

of this instance.

setCountry(String) method

Syntax

void setCountry (String *countryId*)

Parameters

- **country** – the `country` value to set

Usage

Sets the

country

of this instance.

setCustomer(Customer) method

Syntax

```
void setCustomer ( Customer customer )
```

Parameters

- **customer** – the `customer` value to set

Usage

Sets the

customer

of this instance.

setEmail(String) method

Syntax

```
void setEmail ( String email )
```

Parameters

- **email** – the `email` value to set

Usage

Sets the

email

of this instance.

setFax(String) method

Syntax

```
void setFax ( String fax )
```

Parameters

- **fax** – the `fax` value to set

Usage

Sets the

fax

of this instance.

setFirstName(String) method

Syntax

```
void setFirstName ( String firstName )
```

Parameters

- **firstName** – the `firstName` value to set

Usage

Sets the

firstName

of this instance.

setGender(Integer) method

Syntax

```
void setGender ( Integer gender )
```

Parameters

- **gender** – the `Gender` to set

Usage

Sets the gender of this instance.

setHouseNumber(String) method

Syntax

```
void setHouseNumber ( String houseNumber )
```

Parameters

- **houseNumber** – the `houseNumber` value to set

Usage

Sets the

houseNumber

of this instance.

setLastName(String) method

Syntax

```
void setLastName ( String lastName )
```

Parameters

- **lastName** – the `lastName` value to set

Usage

Sets the

lastName

of this instance.

setLatitude(BigDecimal) method

Syntax

```
void setLatitude ( BigDecimal latitude )
```

setLocationType(Integer) method

Syntax

```
void setLocationType ( Integer locationType )
```

setLongitude(BigDecimal) method

Syntax

```
void setLongitude ( BigDecimal longitude )
```

setMiddleName(String) method

Syntax

```
void setMiddleName ( String middleName )
```

Parameters

- **middleName** – the `middleName` value to set

Usage

Sets the

middleName

of this instance.

setPhone1(String) method

Syntax

```
void setPhone1 ( String phone1 )
```

Parameters

- **phone1** – the `phone1` value to set

Usage

Sets the

phone1

of this instance.

setPhone2(String) method

Syntax

```
void setPhone2 ( String phone2 )
```

Parameters

- **phone2** – the `phone2` value to set

Usage

Sets the

phone2

of this instance.

setPosition(String) method

Syntax

```
void setPosition ( String position )
```

Parameters

- **position** – the position value to set

Usage

Sets the

position

of this instance.

setState(String) method

Syntax

```
void setState ( String state )
```

Parameters

- **state** – the state value to set

Usage

Sets the

state

of this instance.

setStreet1(String) method

Syntax

```
void setStreet1 ( String street1 )
```

Parameters

- **street1** – the street1 value to set

Usage

Sets the

street1

of this instance.

setStreet2(String) method

Syntax

```
void setStreet2 ( String street2 )
```

Parameters

- **street2** – the `street2` value to set

Usage

Sets the

street2

of this instance.

setTitle(String) method

Syntax

```
void setTitle ( String title )
```

Parameters

- **title** – the `title` value to set

Usage

Sets the

title

of this instance.

setUrl(String) method

Syntax

```
void setUrl ( String url )
```

Parameters

- **url** – the `url` value to set

Usage

Sets the

url

of this instance.

setZip(String) method

Syntax

```
void setZip ( String zip )
```

Parameters

- **zip** – the zip value to set

Usage

Sets the

zip

of this instance.

Attachment class**Syntax**

```
public class Attachment
```

Usage

The `Attachment` represents a file attached to a customer.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Attachment() constructor

Syntax

```
Attachment ()
```

Usage

Creates a new instance of

`Attachment`

.

getContent() method

Syntax

byte[] getContent ()

Returns

The attachment's content as byte array.

Usage

Use this method to retrieve the attachment's content as a

byte

array.

The attachment's content as byte array.

getContentType() method

Syntax

String getContentType ()

Returns

the contentType

Usage

Returns the content type of this attachment.

the contentType

getCustomer() method

Syntax

Customer getCustomer ()

Returns

the Customer

Usage

Returns the customer this attachment is attached to.

the Customer

getName() method

Syntax

String getName ()

Returns

the name

Usage

Returns the name of this attachment.

the name

getStatus() method

Syntax

int getStatus ()

Returns

the status

Usage

the status

getTransaction() method

Syntax

Transaction getTransaction ()

Returns

the Transaction

Usage

Returns the transaction this attachment is attached to.

the Transaction

getType() method

Syntax

int getType ()

Returns

the AttachmentType

Usage

Returns the type of this attachment.

the AttachmentType

isSetContent() method

Syntax

boolean isSetContent ()

Returns

false if the content is null. Otherwise, it returns true.

Usage

Checks whether the

content

of this instance is set.

false if the content is null. Otherwise, it returns true.

isSetContentType() method

Syntax

boolean isSetContentType ()

Returns

false if the contentType is null. Otherwise, it returns true.

Usage

Checks whether the

contentType

of this instance is set.

false if the contentType is null. Otherwise, it returns true.

isSetCustomer() method

Syntax

```
boolean isSetCustomer ()
```

Returns

false if the customer is null. Otherwise, it returns true.

Usage

Checks whether the

customer

of this instance is set.

false if the customer is null. Otherwise, it returns true.

isSetName() method

Syntax

```
boolean isSetName ()
```

Returns

false if the name is null. Otherwise, it returns true.

Usage

Checks whether the

name

of this instance is set.

false if the name is null. Otherwise, it returns true.

isSetStatus() method

Syntax

```
boolean isSetStatus ()
```

Returns

false if the status is null. Otherwise, it returns true.

Usage

Checks whether the

status

of this instance is set.

false if the status is null. Otherwise, it returns true.

isSetTransaction() method

Syntax

```
boolean isSetTransaction ()
```

Returns

false if the transaction is null. Otherwise, it returns true.

Usage

Checks whether the

transaction

of this instance is set.

false if the transaction is null. Otherwise, it returns true.

isSetType() method

Syntax

```
boolean isSetType ()
```

Returns

false if the type is null. Otherwise, it returns true.

Usage

Checks whether the

type

of this instance is set.

false if the type is null. Otherwise, it returns true.

setContent(byte[]) method

Use this method to set the attachment's content.

Syntax

```
void setContent ( byte[] content )
```

Parameters

- **content** – The attachment's content as a `byte` array.

Usage

The

`Blob`

object which is saved to the database is generated automatically, when this method is called.

*setContentType(String) method***Syntax**

```
void setContentType ( String contentType )
```

Parameters

- **contentType** – the `content` type to set

Usage

Sets the

content type

of this attachment.

*setCustomer(Customer) method***Syntax**

```
void setCustomer ( Customer customer )
```

Parameters

- **customer** – the `customer` to set

Usage

Sets the

customer

this attachment is to be attached to.

setName(String) method

Syntax

```
void setName ( String name )
```

Parameters

- **name** – the name to set

Usage

Sets the

name

of this attachment.

setStatus(int) method

Syntax

```
void setStatus ( int status )
```

Parameters

- **status** – the status to set

setTransaction(Transaction) method

Syntax

```
void setTransaction ( Transaction transaction )
```

Parameters

- **transaction** – the transaction to set

Usage

Sets the

transaction

this attachment is to be attached to.

setType(int) method

Syntax

```
void setType ( int type )
```

Parameters

- **type** – the type value to set

Usage

Sets the

type

of this attachment.

AuditLog class

Syntax

```
public class AuditLog
```

Usage

The AuditLog represents data that is either stored explicitly or stored when calling an endpoint service.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

AuditLog() constructor

Syntax

```
AuditLog ()
```

Usage

Creates a new instance of

```
AuditLog
```

.

getAction() method

Syntax

String getAction ()

getActionResultCode() method

Syntax

String getActionResultCode ()

getApplication() method

Syntax

String getApplication ()

getApplicationVersion() method

Syntax

String getApplicationVersion ()

getCaller() method

Syntax

Customer getCaller ()

getCategory() method

Syntax

String getCategory ()

getCustomerId() method

Syntax

Long getCustomerId ()

getDevice() method

Syntax

String getDevice ()

getDeviceId() method

Syntax

String getDeviceId ()

getDurationMilliseconds() method

Syntax

Long getDurationMilliseconds ()

getInstance() method

Syntax

String getInstance ()

getOrigin() method

Syntax

String getOrigin ()

getOtherDeviceId() method

Syntax

String getOtherDeviceId ()

getParameter1() method

Syntax

String getParameter1 ()

getParameter2() method

Syntax

String getParameter2 ()

getParameter3() method

Syntax

String getParameter3 ()

getParameter4() method

Syntax

String getParameter4 ()

getParameter5() method

Syntax

String getParameter5 ()

getParameter6() method

Syntax

String getParameter6 ()

getParameter7() method

Syntax

String getParameter7 ()

getParameter8() method

Syntax

String getParameter8 ()

getParameter9() method

Syntax

String getParameter9 ()

getSessionId() method

Syntax

String getSessionId ()

Returns

the sessionId

Usage

the sessionId

getTraceNo() method

Syntax

String getTraceNo ()

getTransactionId() method

Syntax

Long getTransactionId ()

isSetAction() method

Syntax

boolean isSetAction ()

Returns

false if the action is null. Otherwise, it returns true.

Usage

Checks whether the

action

of this instance is set.

false if the action is null. Otherwise, it returns true.

isSetActionResultCode() method

Syntax

boolean isSetActionResultCode ()

Returns

false if the actionResultCode is null. Otherwise, it returns true.

Usage

Checks whether the

actionResultCode

of this instance is set.

false if the actionResultCode is null. Otherwise, it returns true.

isSetApplication() method

Syntax

```
boolean isSetApplication ()
```

Returns

false if the application is null. Otherwise, it returns true.

Usage

Checks whether the

application

of this instance is set.

false if the application is null. Otherwise, it returns true.

isSetApplicationVersion() method

Syntax

```
boolean isSetApplicationVersion ()
```

Returns

false if the applicationVersion is null. Otherwise, it returns true.

Usage

Checks whether the

applicationVersion

of this instance is set.

false if the applicationVersion is null. Otherwise, it returns true.

isSetCaller() method

Syntax

```
boolean isSetCaller ()
```

Returns

false if the caller is null. Otherwise, it returns true.

Usage

Checks whether the

caller

of this instance is set.

false if the caller is null. Otherwise, it returns true.

isSetCategory() method

Syntax

```
boolean isSetCategory ()
```

isSetCustomerId() method

Syntax

```
boolean isSetCustomerId ()
```

Returns

false if the customer is null. Otherwise, it returns true.

Usage

Checks whether the

customer

of this instance is set.

false if the customer is null. Otherwise, it returns true.

isSetDevice() method

Syntax

```
boolean isSetDevice ()
```

Returns

false if the device is null. Otherwise, it returns true.

Usage

Checks whether the

device

of this instance is set.

false if the device is null. Otherwise, it returns true.

isSetDeviceId() method

Syntax

boolean isSetDeviceId ()

Returns

false if the deviceId is null. Otherwise, it returns true.

Usage

Checks whether the

deviceId

of this instance is set.

false if the deviceId is null. Otherwise, it returns true.

isSetDurationMilliseconds() method

Syntax

boolean isSetDurationMilliseconds ()

Returns

false if the durationMilliseconds is null. Otherwise, it returns true.

Usage

Checks whether the

durationMilliseconds

of this instance is set.

false if the durationMilliseconds is null. Otherwise, it returns true.

isSetInstance() method

Syntax

boolean isSetInstance ()

Returns

false if the details is null. Otherwise, it returns true.

Usage

Checks whether the

details

of this instance is set.

false if the details is null. Otherwise, it returns true.

isSetOrigin() method

Syntax

```
boolean isSetOrigin ()
```

Returns

false if the origin is null. Otherwise, it returns true.

Usage

Checks whether the

origin

of this instance is set.

false if the origin is null. Otherwise, it returns true.

isSetOtherDeviceId() method

Syntax

```
boolean isSetOtherDeviceId ()
```

Returns

false if the otherDeviceId is null. Otherwise, it returns true.

Usage

Checks whether the

otherDeviceId

of this instance is set.

false if the otherDeviceId is null. Otherwise, it returns true.

isSetParameter1() method

Syntax

```
boolean isSetParameter1 ()
```

Returns

false if the parameter1 is null. Otherwise, it returns true.

Usage

Checks whether the

parameter1

of this instance is set.

false if the parameter1 is null. Otherwise, it returns true.

isSetParameter2() method

Syntax

```
boolean isSetParameter2 ()
```

Returns

false if the parameter2 is null. Otherwise, it returns true.

Usage

Checks whether the

parameter2

of this instance is set.

false if the parameter2 is null. Otherwise, it returns true.

isSetParameter3() method

Syntax

```
boolean isSetParameter3 ()
```

Returns

false if the parameter3 is null. Otherwise, it returns true.

Usage

Checks whether the

parameter3

of this instance is set.

false if the parameter3 is null. Otherwise, it returns true.

isSetParameter4() method

Syntax

```
boolean isSetParameter4 ()
```

Returns

false if the parameter4 is null. Otherwise, it returns true.

Usage

Checks whether the

parameter4

of this instance is set.

false if the parameter4 is null. Otherwise, it returns true.

isSetParameter5() method

Syntax

```
boolean isSetParameter5 ()
```

Returns

false if the parameter5 is null. Otherwise, it returns true.

Usage

Checks whether the

parameter5

of this instance is set.

false if the parameter5 is null. Otherwise, it returns true.

isSetParameter6() method

Syntax

```
boolean isSetParameter6 ()
```

Returns

false if the parameter6 is null. Otherwise, it returns true.

Usage

Checks whether the

parameter6

of this instance is set.

false if the parameter6 is null. Otherwise, it returns true.

isSetParameter7() method

Syntax

```
boolean isSetParameter7 ()
```

Returns

false if the parameter7 is null. Otherwise, it returns true.

Usage

Checks whether the

parameter7

of this instance is set.

false if the parameter7 is null. Otherwise, it returns true.

isSetParameter8() method

Syntax

```
boolean isSetParameter8 ()
```

Returns

false if the parameter8 is null. Otherwise, it returns true.

Usage

Checks whether the

parameter8

of this instance is set.

false if the parameter8 is null. Otherwise, it returns true.

isSetParameter9() method

Syntax

```
boolean isSetParameter9 ()
```

Returns

false if the parameter9 is null. Otherwise, it returns true.

Usage

Checks whether the

parameter9

of this instance is set.

false if the parameter9 is null. Otherwise, it returns true.

isSetSessionId() method

Syntax

```
boolean isSetSessionId ()
```

Returns

false if the sessionId is null . Otherwise, it returns true.

Usage

Checks whether the

sessionId

of this instance is set.

false if the sessionId is null . Otherwise, it returns true.

isSetTraceNo() method

Syntax

boolean isSetTraceNo ()

Returns

false if the traceNo is null. Otherwise, it returns true.

Usage

Checks whether the

traceNo

of this instance is set.

false if the traceNo is null. Otherwise, it returns true.

isSetTransactionId() method

Syntax

boolean isSetTransactionId ()

Returns

false if the transaction is null. Otherwise, it returns true.

Usage

Checks whether the

transaction

of this instance is set.

false if the transaction is null. Otherwise, it returns true.

setAction(String) method

Syntax

void setAction (String *action*)

setActionResultCode(String) method

Syntax

void setActionResultCode (String *actionResultCode*)

setApplication(String) method

Syntax

void setApplication (String *application*)

setApplicationVersion(String) method

Syntax

void setApplicationVersion (String *applicationVersion*)

setCaller(Customer) method

Syntax

void setCaller (Customer *caller*)

setCategory(String) method

Syntax

void setCategory (String *category*)

setCustomerId(Long) method

Syntax

void setCustomerId (Long *customerId*)

setDevice(String) method

Syntax

void setDevice (String *device*)

setDeviceId(String) method

Syntax

void setDeviceId (String *deviceId*)

setDurationMilliseconds(Long) method

Syntax

void setDurationMilliseconds (Long *durationMilliseconds*)

setInstance(String) method

Syntax

void setInstance (String *details*)

setOrigin(String) method

Syntax

void setOrigin (String *origin*)

setOtherDeviceId(String) method

Syntax

void setOtherDeviceId (String *otherDeviceId*)

setParameter1(String) method

Syntax

void setParameter1 (String *parameter1*)

setParameter2(String) method

Syntax

void setParameter2 (String *parameter2*)

setParameter3(String) method

Syntax

void setParameter3 (String *parameter3*)

setParameter4(String) method

Syntax

void setParameter4 (String *parameter4*)

setParameter5(String) method

Syntax

void setParameter5 (String *parameter5*)

setParameter6(String) method

Syntax

void setParameter6 (String *parameter6*)

setParameter7(String) method

Syntax

void setParameter7 (String *parameter7*)

setParameter8(String) method

Syntax

void setParameter8 (String *parameter8*)

setParameter9(String) method

Syntax

void setParameter9 (String *parameter9*)

setSessionId(String) method

Syntax

void setSessionId (String *sessionId*)

Parameters

- **sessionId** – the sessionId to set

setTraceNo(String) method

Syntax

void setTraceNo (String *traceNo*)

setTransactionId(Long) method

Syntax

void setTransactionId (Long *transactionId*)

CommissionConfiguration class

Syntax

```
public class CommissionConfiguration
```

Usage

The `CommissionConfiguration` model holds the individual commission configuration for the Customer.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CommissionConfiguration() constructor

Syntax

```
CommissionConfiguration ()
```

Usage

Creates a new instance of

```
CommissionConfiguration
```

.

getClearingFrequency() method

Syntax

```
String getClearingFrequency ()
```

Returns

the clearingFrequency

Usage

the clearingFrequency

getCustomer() method

Syntax

```
Customer getCustomer ()
```

Returns

the affected Customer

Usage

Returns the customer by this commission.

the affected Customer

getPaymentInstrument() method

Syntax

```
PaymentInstrument getPaymentInstrument ()
```

Returns

the associated PaymentInstrument

Usage

Returns the payment instrument associated to the wallet entry.

the associated PaymentInstrument

getPercentage() method

Syntax

```
BigDecimal getPercentage ()
```

Returns

the percentage

Usage

Returns the percentage to apply for this step.

the percentage

getPercentageSelf() method

Syntax

```
BigDecimal getPercentageSelf ()
```

Returns

the percentage

Usage

Returns the percentage to apply for this step.
the percentage

getSettlementDate() method

Syntax

Date getSettlementDate ()

Returns

the Date representing the settlement time

Usage

Returns the settlement cut-off time.
the Date representing the settlement time

isSetClearingFrequency() method

Syntax

boolean isSetClearingFrequency ()

Returns

false if the clearingFrequency is null. Otherwise, it returns true.

Usage

Checks whether the
clearingFrequency
of this instance is set.
false if the clearingFrequency is null. Otherwise, it returns true.

isSetCustomer() method

Syntax

boolean isSetCustomer ()

Returns

false if the customer is null. Otherwise, it returns true.

Usage

Checks whether the

customer

of this instance is set.

false if the customer is null. Otherwise, it returns true.

isSetPaymentInstrument() method

Syntax

```
boolean isSetPaymentInstrument ()
```

Returns

false if the paymentInstrument is null. Otherwise, it returns true.

Usage

Checks whether the

paymentInstrument

of this instance is set.

false if the paymentInstrument is null. Otherwise, it returns true.

isSetPercentage() method

Syntax

```
boolean isSetPercentage ()
```

Returns

false if the percentage is null. Otherwise, it returns true.

Usage

Checks whether the

percentage

of this instance is set.

false if the percentage is null. Otherwise, it returns true.

isSetPercentageSelf() method

Syntax

boolean isSetPercentageSelf ()

Returns

false if the `percentageSelf` is null. Otherwise, it returns true.

Usage

Checks whether the

percentageSelf

of this instance is set.

false if the `percentageSelf` is null. Otherwise, it returns true.

isSetSettlementDate() method

Syntax

boolean isSetSettlementDate ()

Returns

false if the `settlementDate` is null. Otherwise, it returns true.

Usage

Checks whether the

settlementDate

of this instance is set.

false if the `settlementDate` is null. Otherwise, it returns true.

setClearingFrequency(String) method

Syntax

void setClearingFrequency (String *clearingFrequency*)

Parameters

- **clearingFrequency** – the `clearingFrequency` to set

setCustomer(Customer) method

Syntax

`void setCustomer ( Customer customer )`

Parameters

- **customer** – the customer value to set

Usage

Sets the

customer

of this instance.

setPaymentInstrument(PaymentInstrument) method

Syntax

`void setPaymentInstrument ( PaymentInstrument paymentInstrument )`

Parameters

- **paymentInstrument** – the paymentInstrument value to set

Usage

Sets the

paymentInstrument

of this instance.

setPercentage(BigDecimal) method

Syntax

`void setPercentage ( BigDecimal percentage )`

Parameters

- **percentage** – the percentage value to set

Usage

Sets the

percentage

of this instance.

setPercentageSelf(BigDecimal) method

Syntax

```
void setPercentageSelf ( BigDecimal percentageSelf )
```

Parameters

- **percentage** – the percentage value to set

Usage

Sets the

percentage

of this instance.

setSettlementDate(Date) method

Syntax

```
void setSettlementDate ( Date settlementDate )
```

Parameters

- **settlementDate** – the settlementDate to set

Usage

Sets the settlement cut-off time.

Credential class

Syntax

```
public class Credential
```

Usage

The `Credential` models the customers' credentials like PIN, password etc. Credentials are stored secure: The value of `getCredential()` is stored as the concatenation of the customer ID, the pipe character (|) and the secret (PIN, password) - encrypted with the SHA algorithm and Base64 encoding as described here.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

Credential() constructor

Syntax

```
Credential ()
```

Usage

Creates a new instance of

```
Credential
```

.

getBlockedUntil() method

Syntax

```
Date getBlockedUntil ()
```

getCredential() method

Syntax

```
String getCredential ()
```

Returns

the credential `String` (may be null)

Usage

Returns the credential string the customer may be identified with.

the credential `String` (may be null)

getCredentialStatus() method

Syntax

```
int getCredentialStatus ()
```

getCredentialType() method

Syntax

```
int getCredentialType ()
```

getCustomer() method

Syntax

Customer getCustomer ()

Returns

the Customer

Usage

Returns the customer associated to this identification.

the Customer

getLastUsed() method

Syntax

Date getLastUsed ()

getWrongCredentialCount() method

Syntax

int getWrongCredentialCount ()

isActive() method

Syntax

boolean isActive ()

Returns

true if the credential could be used; false otherwise

Usage

Indicates if this credential may be used to authenticate a customer.

true if the credential could be used; false otherwise

isSetActive() method

Syntax

boolean isSetActive ()

Returns

false if the dbActive is null. Otherwise, it returns true.

Usage

Checks whether the

dbActive

of this instance is set.

false if the dbActive is null. Otherwise, it returns true.

isSetBlockedUntil() method

Syntax

```
boolean isSetBlockedUntil ()
```

Returns

false if the blockedUntil is null. Otherwise, it returns true.

Usage

Checks whether the

blockedUntil

of this instance is set.

false if the blockedUntil is null. Otherwise, it returns true.

isSetCredential() method

Syntax

```
boolean isSetCredential ()
```

Returns

false if the credential is null. Otherwise, it returns true.

Usage

Checks whether the

credential

of this instance is set.

false if the credential is null. Otherwise, it returns true.

isSetCredentialStatus() method

Syntax

boolean isSetCredentialStatus ()

Returns

false if the credentialStatus is null. Otherwise, it returns true.

Usage

Checks whether the

credentialStatus

of this instance is set.

false if the credentialStatus is null. Otherwise, it returns true.

isSetCredentialType() method

Syntax

boolean isSetCredentialType ()

Returns

false if the credentialType is null. Otherwise, it returns true.

Usage

Checks whether the

credentialType

of this instance is set.

false if the credentialType is null. Otherwise, it returns true.

isSetCustomer() method

Syntax

boolean isSetCustomer ()

Returns

false if the customer is null. Otherwise, it returns true.

Usage

Checks whether the

customer

of this instance is set.

false if the customer is null. Otherwise, it returns true.

isSetLastUsed() method

Syntax

```
boolean isSetLastUsed ()
```

isSetWrongCredentialCount() method

Syntax

```
boolean isSetWrongCredentialCount ()
```

Returns

false if the wrongCredentialCount is null. Otherwise, it returns true.

Usage

Checks whether the

wrongCredentialCount

of this instance is set.

false if the wrongCredentialCount is null. Otherwise, it returns true.

setActive(boolean) method

Syntax

```
void setActive ( boolean active )
```

Parameters

- **active** – the active value to set

Usage

Sets the

active

of this instance.

setBlockedUntil(Date) method

Syntax

void setBlockedUntil (Date *blockedUntil*)

setCredential(String) method

Syntax

void setCredential (String *credential*)

Parameters

- **credential** – the `credential` value to set

Usage

Sets the

credential

of this instance.

setCredentialStatus(int) method

Syntax

void setCredentialStatus (int *credentialStatus*)

setCredentialType(int) method

Syntax

void setCredentialType (int *credentialType*)

setCustomer(Customer) method

Syntax

void setCustomer (Customer *customer*)

Parameters

- **customer** – the `customer` value to set

Usage

Sets the

customer

of this instance.

setLastUsed(Date) method

Syntax

```
void setLastUsed ( Date lastUsed )
```

setWrongCredentialCount(int) method

Syntax

```
void setWrongCredentialCount ( int wrongCredentialCount )
```

Customer class

Syntax

```
public class Customer
```

Usage

The `Customer` represents any person interacting with the paybox system.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Customer() constructor

Syntax

```
Customer ( )
```

Usage

Creates a new instance of

```
Customer
```

```
.
```

getBlacklistReason() method

Syntax

```
int getBlacklistReason ( )
```

Returns

an `int` representing the blacklist reason

Usage

Returns the blacklist reason of this customer.

0

means that the customer is not on the blacklist.

an `int` representing the blacklist reason

getCancellationReason() method

Syntax

`int getCancellationReason ()`

getCustomerType() method

Syntax

`CustomerType getCustomerType ()`

getDateOfBirth() method

Syntax

`Date getDateOfBirth ()`

getDisplayName() method

Syntax

`String getDisplayName ()`

Returns

the display name

Usage

Returns the display name of this customer.

the display name

getFeeSetId() method

Syntax

`Long getFeeSetId ()`

*getLimitSetId() method***Syntax**

```
Long getLimitSetId ()
```

*getLocale() method***Syntax**

```
Locale getLocale ()
```

Returns

the customer's `Locale`

Usage

Returns the locale of the customer.

the customer's `Locale`

*getNotificationModeId() method***Syntax**

```
int getNotificationModeId ()
```

*getOrgUnit() method***Syntax**

```
OrgUnit getOrgUnit ()
```

Returns

the customer's `OrgUnit`

Usage

Returns the `OrgUnit` this customer is assigned to.

the customer's `OrgUnit`

*getParent() method***Syntax**

```
Customer getParent ()
```

Returns

the superordinate Customer

Usage

Returns the parent customer of this customer.

the superordinate Customer

getReferralCode() method

Syntax

String getReferralCode ()

getRiskCategory() method

Syntax

RiskCategory getRiskCategory ()

getSecurityAnswer() method

Syntax

String getSecurityAnswer ()

Returns

the customer's security answer

Usage

Returns the security answer of this customer.

the customer's security answer

getSecurityQuestion() method

Syntax

String getSecurityQuestion ()

Returns

the customer's security question

Usage

Returns the security question of this customer.

the customer's security question

getTimeZone() method

Syntax

`TimeZone getTimeZone ()`

Returns

the customer's `TimeZone`

Usage

Returns the time zone the customer lives in.

the customer's `TimeZone`

getTxnText() method

Syntax

`String getTxnText ()`

getVat() method

Syntax

`BigDecimal getVat ()`

isActive() method

Syntax

`boolean isActive ()`

Returns

`true` if this instance is active; `false` if it is inactive

Usage

Indicates if this customer is active.

`true` if this instance is active; `false` if it is inactive

isAncestor(Customer) method

Syntax

`boolean isAncestor ( Customer customer )`

Parameters

- **customer** – the Customer to examine

Returns

true if this instance is customer's ancestor; false otherwise

Usage

Determines if this instance is a either a direct parent or an ancestor (of whatever degree) of the specified customer.

true if this instance is customer's ancestor; false otherwise

isActive() method

Syntax

```
boolean isActive ()
```

Returns

false if the dbActive is null. Otherwise, it returns true.

Usage

Checks whether the

dbActive

of this instance is set.

false if the dbActive is null. Otherwise, it returns true.

isSetBlacklistReason() method

Syntax

```
boolean isSetBlacklistReason ()
```

Returns

false if the blacklistReason is null. Otherwise, it returns true.

Usage

Checks whether the

blacklistReason

of this instance is set.

false if the blacklistReason is null. Otherwise, it returns true.

isSetCancellationReason() method

Syntax

boolean isSetCancellationReason ()

Returns

false if the cancellationReason is null. Otherwise, it returns true.

Usage

Checks whether the

cancellationReason

of this instance is set.

false if the cancellationReason is null. Otherwise, it returns true.

isSetCustomerType() method

Syntax

boolean isSetCustomerType ()

Returns

false if the customerType is null. Otherwise, it returns true.

Usage

Checks whether the

customerType

of this instance is set.

false if the customerType is null. Otherwise, it returns true.

isSetDateOfBirth() method

Syntax

boolean isSetDateOfBirth ()

Returns

false if the dateOfBirth is null. Otherwise, it returns true.

Usage

Checks whether the

dateOfBirth

of this instance is set.

false if the *dateOfBirth* is null. Otherwise, it returns true.

isSetDisplayName() method

Syntax

```
boolean isSetDisplayName ()
```

Returns

false if the *displayName* is null. Otherwise, it returns true.

Usage

Checks whether the

displayName

of this instance is set.

false if the *displayName* is null. Otherwise, it returns true.

isSetFeeSetId() method

Syntax

```
boolean isSetFeeSetId ()
```

Returns

false if the *feeSetId* is null. Otherwise, it returns true.

Usage

Checks whether the

feeSetId

of this instance is set.

false if the *feeSetId* is null. Otherwise, it returns true.

*isSetLimitSetId() method***Syntax**

```
boolean isSetLimitSetId ()
```

Returns

false if the limitSetId is null. Otherwise, it returns true.

Usage

Checks whether the

limitSetId

of this instance is set.

false if the limitSetId is null. Otherwise, it returns true.

*isSetLocale() method***Syntax**

```
boolean isSetLocale ()
```

Returns

false if the dbLanguage and dbCountry is null. Otherwise, it returns true.

Usage

Checks whether the

dbLanguage

or

dbCountry

of this instance is set.

false if the dbLanguage and dbCountry is null. Otherwise, it returns true.

*isSetNotificationModeId() method***Syntax**

```
boolean isSetNotificationModeId ()
```

Returns

false if the txnReceiptModeId is null. Otherwise, it returns true.

Usage

Checks whether the

txnReceiptModeId

of this instance is set.

false if the `txnReceiptModeId` is null. Otherwise, it returns true.

isSetOrgUnit() method

Syntax

```
boolean isSetOrgUnit ()
```

Returns

false if the `orgUnit` is null. Otherwise, it returns true.

Usage

Checks whether the

orgUnit

of this instance is set.

false if the `orgUnit` is null. Otherwise, it returns true.

isSetParent() method

Syntax

```
boolean isSetParent ()
```

Returns

false if the `parent` is null. Otherwise, it returns true.

Usage

Checks whether the

parent

of this instance is set.

false if the `parent` is null. Otherwise, it returns true.

*isSetReferralCode() method***Syntax**

```
boolean isSetReferralCode ()
```

Returns

false if the referralCode is null. Otherwise, it returns true.

Usage

Checks whether the

referralCode

of this instance is set.

false if the referralCode is null. Otherwise, it returns true.

*isSetRiskCategory() method***Syntax**

```
boolean isSetRiskCategory ()
```

Returns

false if the riskCategory is null. Otherwise, it returns true.

Usage

Checks whether the

riskCategory

of this instance is set.

false if the riskCategory is null. Otherwise, it returns true.

*isSetSecurityAnswer() method***Syntax**

```
boolean isSetSecurityAnswer ()
```

Returns

false if the securityAnswer is null. Otherwise, it returns true.

Usage

Checks whether the

securityAnswer

of this instance is set.

false if the *securityAnswer* is null. Otherwise, it returns true.

isSetSecurityQuestion() method

Syntax

```
boolean isSetSecurityQuestion ()
```

Returns

false if the *securityQuestion* is null. Otherwise, it returns true.

Usage

Checks whether the

securityQuestion

of this instance is set.

false if the *securityQuestion* is null. Otherwise, it returns true.

isSetTest() method

Syntax

```
boolean isSetTest ()
```

Returns

false if the *dbTest* is null. Otherwise, it returns true.

Usage

Checks whether the

dbTest

of this instance is set.

false if the *dbTest* is null. Otherwise, it returns true.

*isSetTimeZone() method***Syntax**

```
boolean isSetTimeZone ()
```

Returns

false if the `dbTimeZone` is null. Otherwise, it returns true.

Usage

Checks whether the

dbTimeZone

of this instance is set.

false if the `dbTimeZone` is null. Otherwise, it returns true.

*isSetTxnText() method***Syntax**

```
boolean isSetTxnText ()
```

Returns

false if the `txnText` is null. Otherwise, it returns true.

Usage

Checks whether the

txnText

of this instance is set.

false if the `txnText` is null. Otherwise, it returns true.

*isSetVat() method***Syntax**

```
boolean isSetVat ()
```

Returns

false if the `vat` is null. Otherwise, it returns true.

Usage

Checks whether the

vat

of this instance is set.

false if the vat is null. Otherwise, it returns true.

isTest() method

Syntax

```
boolean isTest ()
```

Returns

true if this instance is a test user; false otherwise

Usage

Indicates if this customer is a test user.

true if this instance is a test user; false otherwise

setActive(boolean) method

Syntax

```
void setActive ( boolean active )
```

Parameters

- **active** – the active value to set

Usage

Sets the

active

of this instance.

setBlacklistReason(int) method

Syntax

```
void setBlacklistReason ( int blacklistReason )
```


Parameters

- **blacklistReason** – the `blacklistReason` value to set

Usage

Sets the

blacklistReason

of this instance.

setCancellationReason(int) method

Syntax

`void setCancellationReason ( int cancellationReason )`

setCustomerType(CustomerType) method

Syntax

`void setCustomerType ( CustomerType customerType )`

setDateOfBirth(Date) method

Syntax

`void setDateOfBirth ( Date dateOfBirth )`

setDisplayName(String) method

Syntax

`void setDisplayName ( String displayName )`

Parameters

- **displayName** – the `displayName` value to set

Usage

Sets the

displayName

of this instance.

setFeeSetId(Long) method

Syntax

void setFeeSetId (Long *feeSetId*)

setLimitSetId(Long) method

Syntax

void setLimitSetId (Long *limitSetId*)

setLocale(Locale) method

Syntax

void setLocale (Locale *locale*)

Parameters

- **locale** – the locale value to set

Usage

Sets the

locale

of this instance.

setNotificationModeId(int) method

Syntax

void setNotificationModeId (int *txnReceiptModeId*)

setOrgUnit(OrgUnit) method

Syntax

void setOrgUnit (OrgUnit *orgUnit*)

Parameters

- **orgUnit** – the orgUnit value to set

Usage

Sets the

orgUnit

of this instance.

setParent(Customer) method

Syntax

```
void setParent ( Customer parent )
```

Parameters

- **parent** – the `parent` value to set

Usage

Sets the

parent

of this instance.

setReferralCode(String) method

Syntax

```
void setReferralCode ( String referralCode )
```

setRiskCategory(RiskCategory) method

Syntax

```
void setRiskCategory ( RiskCategory riskCategory )
```

setSecurityAnswer(String) method

Syntax

```
void setSecurityAnswer ( String securityAnswer )
```

Parameters

- **securityAnswer** – the `securityAnswer` value to set

Usage

Sets the

securityAnswer

of this instance.

setSecurityQuestion(String) method

Syntax

`void setSecurityQuestion ( String securityQuestion )`

Parameters

- **securityQuestion** – the `securityQuestion` value to set

Usage

Sets the

securityQuestion

of this instance.

setTest(boolean) method

Syntax

`void setTest ( boolean test )`

Parameters

- **test** – the `test` value to set

Usage

Sets the

test

of this instance.

setTimeZone(TimeZone) method

Syntax

`void setTimeZone ( TimeZone timeZone )`

Parameters

- **timeZone** – the `timeZone` value to set

Usage

Sets the

timeZone

of this instance.

setTxnText(String) method

Syntax

```
void setTxnText ( String txnText )
```

setVat(BigDecimal) method

Syntax

```
void setVat ( BigDecimal vat )
```

CustomerAttribute class

Syntax

```
public class CustomerAttribute
```

Usage

The CustomerNetwork class represents the relations between customers.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CustomerAttributePk class

Syntax

```
public class CustomerAttributePk
```

Usage

The CustomerAttributePk class is the embeddable primary key object for a customer Attribute.

© 2000-2007 by paybox solutions AG

Revision:1.3Name:not supported by cvs2svn

CustomerAttributePk() constructor

Syntax

```
CustomerAttributePk ()
```

Usage

Creates a new instance of

CustomerOtpPk

.

CustomerAttributePk(Customer, int) constructor

Syntax

CustomerAttributePk (Customer *customer* , int *customerAttributeTypeId*)

Parameters

- **customer** –
- **customerAttributeTypeId** –

Usage

Creates a new instance of

CustomerOtpPk

.

equals(Object) method

Syntax

boolean equals (Object *o*)

Parameters

- **o** – the Object to compare with

Returns

true if this class is assignable from o's class and has the same PK values as this instance

Usage

Indicates if another object is

equal to

this one.

true if this class is assignable from o's class and has the same PK values as this instance

getCustomer() method

Syntax

```
Customer getCustomer ()
```

Returns

the customer

Usage

the customer

getCustomerAttributeTypeId() method

Syntax

```
int getCustomerAttributeTypeId ()
```

Returns

the attribute

Usage

the attribute

hashCode() method

Syntax

```
int hashCode ()
```

Returns

a hash value for this instance

Usage

Returns a hash value.

a hash value for this instance

isSetCustomer() method

Syntax

```
boolean isSetCustomer ()
```

Returns

false if the customer is null. Otherwise, it returns true.

Usage

Checks whether the

customer

of this instance is set.

false if the customer is null. Otherwise, it returns true.

isSetCustomerAttributeTypeId() method

Syntax

```
boolean isSetCustomerAttributeTypeId ()
```

Returns

false if the customerAttributeTypeId is null. Otherwise, it returns true.

Usage

Checks whether the

customerAttributeTypeId

of this instance is set.

false if the customerAttributeTypeId is null. Otherwise, it returns true.

toString() method

Syntax

```
String toString ()
```

Returns

a *String* representing this instance

Usage

Returns a string representation of this object.

a *String* representing this instance

CustomerAttribute() constructor

Syntax

```
CustomerAttribute ()
```

Usage

Creates a new instance of

```
CustomerAttribute
```

.

CustomerAttribute(CustomerAttributePk) constructor

Syntax

```
CustomerAttribute ( CustomerAttributePk id )
```

Parameters

- **id** – the PK value

Usage

Creates a new instance of

```
CustomerAttribute
```

with a preset primary key.

getCustomer() method

Syntax

```
Customer getCustomer ()
```

Returns

the customer

Usage

the customer

getCustomerAttributeTypeId() method

Syntax

```
int getCustomerAttributeTypeId ()
```

Returns

the attribute

Usage

the attribute

getValue() method

Syntax

String getValue ()

Returns

the value

Usage

the value

isSetCustomer() method

Syntax

boolean isSetCustomer ()

Returns

false if the id or the id's customer is null. Otherwise, it returns true.

Usage

Checks whether the

customer

of this instance's PK is set.

false if the id or the id's customer is null. Otherwise, it returns true.

isSetCustomerAttributeTypeId() method

Syntax

boolean isSetCustomerAttributeTypeId ()

Returns

false if the id or the id's customerAttributeTypeId is null. Otherwise, it returns true.

Usage

Checks whether the

customerAttributeTypeId

of this instance's PK is set.

false if the id or the id's customerAttributeTypeId is null. Otherwise, it returns true.

isSetValue() method

Syntax

```
boolean isSetValue ()
```

Returns

false if the value is null. Otherwise, it returns true.

Usage

Checks whether the

value

of this instance is set.

false if the value is null. Otherwise, it returns true.

setValue(String) method

Syntax

```
void setValue ( String value )
```

Parameters

- **value** – the value to set

CustomerNetwork class**Syntax**

```
public class CustomerNetwork
```

Usage

The CustomerNetwork class represents the relations between customers.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

CustomerNetwork() constructor

Syntax

```
CustomerNetwork ()
```

Usage

Creates a new instance of

```
CustomerNetwork
```

.

getAlias() method

Syntax

```
String getAlias ()
```

Returns

the CustomerNetwork's alias

Usage

Returns the type of this instance.

the CustomerNetwork's alias

getChild() method

Syntax

```
Customer getChild ()
```

Returns

the CustomerNetwork's child - Customer

Usage

Returns the type of this instance.

the CustomerNetwork's child - Customer

getFather() method

Syntax

Customer getFather ()

Returns

the affected Customer

Usage

Returns the father this

CustomerNetwork

is assigned to.

the affected Customer

getNetworkStatusId() method

Syntax

int getNetworkStatusId ()

Returns

the networkStatusId

Usage

the networkStatusId

getNetworkTypeId() method

Syntax

int getNetworkTypeId ()

Returns

the networkTypeId

Usage

the networkTypeId

isActive() method

Syntax

`boolean isActive ()`

Returns

true if the CustomerNetwork could be used; false otherwise

Usage

Returns the active status of this instance

true if the CustomerNetwork could be used; false otherwise

isSetActive() method

Syntax

`boolean isSetActive ()`

Returns

false if the dbActive is null. Otherwise, it returns true.

Usage

Checks whether the

dbActive

of this instance is set.

false if the dbActive is null. Otherwise, it returns true.

isSetAlias() method

Syntax

`boolean isSetAlias ()`

Returns

false if the alias is null. Otherwise, it returns true.

Usage

Checks whether the

alias

of this instance is set.

false if the alias is null. Otherwise, it returns true.

isSetChild() method

Syntax

```
boolean isSetChild ()
```

Returns

false if the child is null. Otherwise, it returns true.

Usage

Checks whether the

child

of this instance is set.

false if the child is null. Otherwise, it returns true.

isSetFather() method

Syntax

```
boolean isSetFather ()
```

Returns

false if the father is null. Otherwise, it returns true.

Usage

Checks whether the

father

of this instance is set.

false if the father is null. Otherwise, it returns true.

isSetNetworkStatusId() method

Syntax

```
boolean isSetNetworkStatusId ()
```

Returns

false if the networkStatusId is null. Otherwise, it returns true.

Usage

Checks whether the

networkStatusId

of this instance is set.

false if the networkStatusId is null. Otherwise, it returns true.

isSetNetworkTypeId() method

Syntax

```
boolean isSetNetworkTypeId ()
```

Returns

false if the networkTypeId is null. Otherwise, it returns true.

Usage

Checks whether the

networkTypeId

of this instance is set.

false if the networkTypeId is null. Otherwise, it returns true.

setActive(boolean) method

Syntax

```
void setActive ( boolean active )
```

Parameters

- **active** – the active value to set

Usage

Sets the

active

of this instance.

setAlias(String) method

Syntax

```
void setAlias ( String alias )
```


Parameters

- **alias** – the alias to set

Usage

Sets the type of this instance.

setChild(Customer) method

Syntax

```
void setChild ( Customer child )
```

Parameters

- **child** – the child to set

Usage

Sets the child of this instance.

setFather(Customer) method

Syntax

```
void setFather ( Customer father )
```

Parameters

- **father** – the father value to set

Usage

Sets the

father

of this instance.

setNetworkStatusId(int) method

Syntax

```
void setNetworkStatusId ( int networkStatusId )
```

Parameters

- **networkStatusId** – the networkStatusId to set

setNetworkTypeId(int) method

Syntax

```
void setNetworkTypeId ( int networkTypeId )
```

Parameters

- **networkTypeId** – the networkTypeId to set

CustomerOtp class

Syntax

```
public class CustomerOtp
```

Usage

The CustomerOtp models a one-time password assigned to a customer. The OTP is intended to be used for a specific purpose indicated by its type.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CustomerOtpPk class

Syntax

```
public class CustomerOtpPk
```

Usage

The Type is a lookup for the customer OTP type.

© 2000-2007 by paybox solutions AG

Revision:1.5Name:not supported by cvs2svnThe CustomerOtpPk class is the embeddable primary key object for a customer OTP.

© 2000-2007 by paybox solutions AG © 2011 by Sybase Inc., an SAP company

Revision:1.5Name:not supported by cvs2svn

CustomerOtpPk() constructor

Syntax

```
CustomerOtpPk ()
```

Usage

Creates a new instance of

CustomerOtpPk

.

CustomerOtpPk(Customer, OtpType, Long) constructor

Syntax

CustomerOtpPk (Customer *customer* , OtpType *otpType* , Long *referenceID*)

Parameters

- **customer** –
- **otpType** –
- **referenceID** –

Usage

Creates a new instance of

CustomerOtpPk

.

equals(Object) method

Syntax

boolean equals (Object *o*)

Parameters

- **o** – the Object to compare with

Returns

true if this class is assignable from o's class and has the same PK values as this instance

Usage

Indicates if another object is

equal to

this one.

true if this class is assignable from o's class and has the same PK values as this instance

getCustomer() method

Syntax

Customer getCustomer ()

Returns

the customer

Usage

the customer

getOtpType() method

Syntax

OtpType getOtpType ()

Returns

the otpType

Usage

the otpType

getReferenceID() method

Syntax

Long getReferenceID ()

hashCode() method

Syntax

int hashCode ()

Returns

a hash value for this instance

Usage

Returns a hash value.

a hash value for this instance

*isReferenceID() method***Syntax**

```
boolean isReferenceID ()
```

Returns

false if the referenceID is null. Otherwise, it returns true.

Usage

Checks whether the

referenceID

of this instance is set.

false if the referenceID is null. Otherwise, it returns true.

*isSetCustomer() method***Syntax**

```
boolean isSetCustomer ()
```

Returns

false if the customer is null. Otherwise, it returns true.

Usage

Checks whether the

customer

of this instance is set.

false if the customer is null. Otherwise, it returns true.

*isSetOtpType() method***Syntax**

```
boolean isSetOtpType ()
```

Returns

false if the otpType is null. Otherwise, it returns true.

Usage

Checks whether the

otpType

of this instance is set.

false if the otpType is null. Otherwise, it returns true.

toString() method

Syntax

String toString ()

Returns

a String representing this instance

Usage

Returns a string representation of this object.

a String representing this instance

CustomerOtp() constructor

Syntax

CustomerOtp ()

Usage

Creates a new instance of

CustomerOtp

.

CustomerOtp(CustomerOtpPk) constructor

Syntax

CustomerOtp (CustomerOtpPk *id*)

Parameters

- **id** – the PK value

Usage

Creates a new instance of

CustomerOtp

with a preset primary key.

getCustomer() method

Syntax

Customer getCustomer ()

Returns

the associated Customer

Usage

Returns the customer this instance is assigned to.

the associated Customer

getExpiration() method

Syntax

Date getExpiration ()

Returns

the expiry Date; null if the OTP does not expire at all

Usage

Returns the point in time this one-time password expires.

the expiry Date; null if the OTP does not expire at all

getMailJobId() method

Syntax

Long getMailJobId ()

getOtp() method

Syntax

String getOtp ()

Returns

the otp

Usage

the otp

getOtpType() method

Syntax

OtpType getOtpType ()

Returns

the associated OtpType

Usage

the associated OtpType

getSentCount() method

Syntax

Integer getSentCount ()

getWrongEntryCount() method

Syntax

int getWrongEntryCount ()

Returns

the wrongEntryCount

Usage

the wrongEntryCount

isSetCustomer() method

Syntax

boolean isSetCustomer ()

Returns

false if the id or the id's customer is null. Otherwise, it returns true.

Usage

Checks whether the

customer

of this instance's PK is set.

false if the `id` or the `id's customer` is null. Otherwise, it returns true.

isSetExpiration() method

Syntax

```
boolean isSetExpiration ()
```

Returns

false if the expiration is null. Otherwise, it returns true.

Usage

Checks whether the

expiration

of this instance is set.

false if the `expiration` is null. Otherwise, it returns true.

isSetMailJobId() method

Syntax

```
boolean isSetMailJobId ()
```

isSetOtp() method

Syntax

```
boolean isSetOtp ()
```

Returns

false if the `otp` is null. Otherwise, it returns true.

Usage

Checks whether the

otp

of this instance is set.

false if the otp is null. Otherwise, it returns true.

isSetOtpType() method

Syntax

boolean isSetOtpType ()

Returns

false if the id or the id's otpType is null. Otherwise, it returns true.

Usage

Checks whether the

otpType

of this instance's PK is set.

false if the id or the id's otpType is null. Otherwise, it returns true.

isSetSentCount() method

Syntax

boolean isSetSentCount ()

isSetWrongEntryCount() method

Syntax

boolean isSetWrongEntryCount ()

Returns

false if the wrongEntryCount is null. Otherwise, it returns true.

Usage

Checks whether the

wrongEntryCount

of this instance is set.

false if the wrongEntryCount is null. Otherwise, it returns true.

setExpiration(Date) method

Syntax

void setExpiration (Date *expiration*)

Parameters

- **expiration** – the expiration value to set

Usage

Sets the

expiration

of this instance.

setMailJobId(Long) method

Syntax

void setMailJobId (Long *mailJobId*)

setOtp(String) method

Syntax

void setOtp (String *otp*)

Parameters

- **otp** – the otp to set

setSentCount(Integer) method

Syntax

void setSentCount (Integer *sentCount*)

setWrongEntryCount(int) method

Syntax

void setWrongEntryCount (int *wrongEntryCount*)

Parameters

- **wrongEntryCount** – the wrongEntryCount to set

CustomerPersistentLogin class

Syntax

```
public class CustomerPersistentLogin
```

Usage

Models persistent logins in mobiliser (RememberMe Tokens).

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CustomerPersistentLogin() constructor

Syntax

```
CustomerPersistentLogin ()
```

Usage

Creates a new instance of

```
CustomerPersistentLogin
```

.

CustomerPersistentLogin(final String) constructor

Syntax

```
CustomerPersistentLogin ( final String series )
```

Parameters

- **series** – the series to use as PK value

Usage

Creates a new instance of

```
CustomerPersistentLogin
```

with a preset primary key.

getIdentification() method

Syntax

```
String getIdentification ()
```

Returns

the identification

Usage

the identification

getLastUsed() method

Syntax

Date getLastUsed ()

Returns

the lastUsed

Usage

the lastUsed

getToken() method

Syntax

String getToken ()

Returns

the token

Usage

the token

setIdentification(final String) method

Syntax

void setIdentification (final String *identification*)

Parameters

- **identification** – the identification to set

setLastUsed(final Date) method

Syntax

void setLastUsed (final Date *lastUsed*)

Parameters

- **lastUsed** – the lastUsed to set

setToken(final String) method

Syntax

```
void setToken ( final String token )
```

Parameters

- **token** – the token to set

CustomerPrivilege class

Syntax

```
public class CustomerPrivilege
```

Usage

The UmgrUserPrivilege is used for the mapping of privileges to customers.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CustomerPrivilege() constructor

Syntax

```
CustomerPrivilege ()
```

Usage

Creates a new instance of

```
CustomerPrivilege
```

.

getCustomer() method

Syntax

```
Customer getCustomer ()
```

Returns

the affected Customer

Usage

Returns the customer possessing the privilege.

the affected Customer

getOrgUnit() method

Syntax

```
String getOrgUnit ()
```

Returns

the orgunit

Usage

the orgunit

getPrivilege() method

Syntax

```
String getPrivilege ()
```

Returns

the associated UmgrPrivilege

Usage

Returns the privilege the customer possesses.

the associated UmgrPrivilege

getValidFrom() method

Syntax

```
Date getValidFrom ()
```

Returns

the start Date; null if there is no restriction

Usage

Returns the begin of the privilege's validity period.

the start Date; null if there is no restriction

getValidTo() method

Syntax

Date getValidTo ()

Returns

the end Date; null if there is no restriction

Usage

Returns the end of the privilege's validity period.

the end Date; null if there is no restriction

isGrantOption() method

Syntax

boolean isGrantOption ()

Returns

true if the customer has the grant option; false if not

Usage

Indicates if the associated customer possesses the grant option for this privilege.

true if the customer has the grant option; false if not

isSetCustomer() method

Syntax

boolean isSetCustomer ()

Returns

false if the customer is null. Otherwise, it returns true.

Usage

Checks whether the

customer

of this instance is set.

false if the customer is null. Otherwise, it returns true.

isSetGrantOption() method

Syntax

```
boolean isSetGrantOption ()
```

Returns

false if the dbGrantOption is null. Otherwise, it returns true.

Usage

Checks whether the

dbGrantOption

of this instance is set.

false if the dbGrantOption is null. Otherwise, it returns true.

isSetOrgUnit() method

Syntax

```
boolean isSetOrgUnit ()
```

Returns

false if the orgUnit is null. Otherwise, it returns true.

Usage

Checks whether the

orgUnit

of this instance is set.

false if the orgUnit is null. Otherwise, it returns true.

isSetPrivilege() method

Syntax

```
boolean isSetPrivilege ()
```

Returns

false if the privilege is null . Otherwise, it returns true.

Usage

Checks whether the

privilege

of this instance is set.

false if the `privilege` is null . Otherwise, it returns true.

isValidFrom() method

Syntax

```
boolean isValidFrom ()
```

Returns

false if the `validFrom` is null . Otherwise, it returns true.

Usage

Checks whether the

validFrom

of this instance is set.

false if the `validFrom` is null . Otherwise, it returns true.

isValidTo() method

Syntax

```
boolean isValidTo ()
```

Returns

false if the `validTo` is null. Otherwise, it returns true.

Usage

Checks whether the

validTo

of this instance is set.

false if the `validTo` is null. Otherwise, it returns true.

setCustomer(final Customer) method

Syntax

```
void setCustomer ( final Customer customer )
```

Parameters

- **customer** – the customer value to set

Usage

Sets the

customer

of this instance.

setGrantOption(final boolean) method

Syntax

```
void setGrantOption ( final boolean grantOption )
```

Parameters

- **grantOption** – the grantOption value to set

Usage

Sets the

grantOption

of this instance.

setOrgUnit(final String) method

Syntax

```
void setOrgUnit ( final String orgUnit )
```

Parameters

- **orgUnit** –

setPrivilege(final String) method

Syntax

```
void setPrivilege ( final String privilege )
```

Parameters

- **privilege** – the privilege value to set

Usage

Sets the

privilege

of this instance.

setValidFrom(final Date) method

Syntax

```
void setValidFrom ( final Date validFrom )
```

Parameters

- **validFrom** – the validFrom value to set

Usage

Sets the

validFrom

of this instance.

setValidTo(final Date) method

Syntax

```
void setValidTo ( final Date validTo )
```

Parameters

- **validTo** – the validTo value to set

Usage

Sets the

validTo

of this instance.

CustomerRole class

Syntax

```
public class CustomerRole
```

Usage

The `UmgrUserRole` is used for the mapping of roles to customers.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CustomerRole() constructor

Syntax

```
CustomerRole ()
```

Usage

Creates a new instance of

```
CustomerRole
```

```
.
```

getCustomer() method

Syntax

```
Customer getCustomer ()
```

Returns

the affected `Customer`

Usage

Returns the customer possessing the role.

the affected `Customer`

getOrgUnit() method

Syntax

String getOrgUnit ()

getRole() method

Syntax

String getRole ()

Returns

the associated UmgrRole

Usage

Returns the role the customer possesses.

the associated UmgrRole

getValidFrom() method

Syntax

Date getValidFrom ()

Returns

the start Date; null if there is no restriction

Usage

Returns the begin of the role's validity period.

the start Date; null if there is no restriction

getValidTo() method

Syntax

Date getValidTo ()

Returns

the end Date; null if there is no restriction

Usage

Returns the end of the role's validity period.
the end Date; null if there is no restriction

isGrantOption() method

Syntax

`boolean isGrantOption ()`

Returns

true if the customer has the grant option; false if not

Usage

Indicates if the associated customer possesses the grant option for this role.
true if the customer has the grant option; false if not

isSetCustomer() method

Syntax

`boolean isSetCustomer ()`

Returns

false if the customer is null. Otherwise, it returns true.

Usage

Checks whether the

customer

of this instance is set.

false if the customer is null. Otherwise, it returns true.

isSetGrantOption() method

Syntax

`boolean isSetGrantOption ()`

Returns

false if the dbGrantOption is null. Otherwise, it returns true.

Usage

Checks whether the

dbGrantOption

of this instance is set.

false if the dbGrantOption is null. Otherwise, it returns true.

isSetOrgUnit() method

Syntax

```
boolean isSetOrgUnit ()
```

Returns

false if the orgUnit is null. Otherwise, it returns true.

Usage

Checks whether the

orgUnit

of this instance is set.

false if the orgUnit is null. Otherwise, it returns true.

isSetRole() method

Syntax

```
boolean isSetRole ()
```

Returns

false if the role is null. Otherwise, it returns true.

Usage

Checks whether the

role

of this instance is set.

false if the role is null. Otherwise, it returns true.

*isValidFrom() method***Syntax**

```
boolean isValidFrom ()
```

Returns

false if the validFrom is null . Otherwise, it returns true.

Usage

Checks whether the

validFrom

of this instance is set.

false if the validFrom is null . Otherwise, it returns true.

*isValidTo() method***Syntax**

```
boolean isValidTo ()
```

Returns

false if the validTo is null. Otherwise, it returns true.

Usage

Checks whether the

validTo

of this instance is set.

false if the validTo is null. Otherwise, it returns true.

*setCustomer(final Customer) method***Syntax**

```
void setCustomer ( final Customer customer )
```

Parameters

- **customer** – the customer value to set

Usage

Sets the

customer

of this instance.

setGrantOption(final boolean) method

Syntax

```
void setGrantOption ( final boolean grantOption )
```

Parameters

- **grantOption** – the `grantOption` value to set

Usage

Sets the

grantOption

of this instance.

setOrgUnit(final String) method

Syntax

```
void setOrgUnit ( final String orgUnit )
```

Parameters

- **orgUnit** –

setRole(final String) method

Syntax

```
void setRole ( final String role )
```

Parameters

- **role** – the `role` value to set

Usage

Sets the

role

of this instance.

setValidFrom(final Date) method

Syntax

```
void setValidFrom ( final Date validFrom )
```

Parameters

- **validFrom** – the validFrom value to set

Usage

Sets the

validFrom

of this instance.

setValidTo(final Date) method

Syntax

```
void setValidTo ( final Date validTo )
```

Parameters

- **validTo** – the validTo value to set

Usage

Sets the

validTo

of this instance.

CustomerSession class

Syntax

```
public class CustomerSession
```

Usage

The CustomerSession traces the activities of a customer.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

CustomerSession() constructor

Syntax

```
CustomerSession ()
```

Usage

Creates a new instance of

```
CustomerSession
```

.

CustomerSession(String) constructor

Syntax

```
CustomerSession ( String sessionId )
```

Parameters

- **sessionId** – the sessionId to use as PK value

Usage

Creates a new instance of

```
CustomerSession
```

with a preset primary key.

getCustomer() method

Syntax

```
Customer getCustomer ()
```

Returns

the affected Customer

Usage

Returns the customer this role is assigned to.

the affected Customer

getIdentification() method

Syntax

String getIdentification ()

Returns

the identification

Usage

the identification

getLastActivity() method

Syntax

Date getLastActivity ()

Returns

the last activity Date

Usage

Returns the point in time the customer was last active.

the last activity Date

getLogin() method

Syntax

Date getLogin ()

Returns

the login Date

Usage

Returns the point in time of the last login.

the login Date

getLogoff() method

Syntax

Date getLogoff ()

Returns

the logoff Date; null if the session is yet active (though it may still be timed out)

Usage

Returns the point in time of the last logoff. If a user logged in, this date should be cleared to indicate that the session is active.

the logoff Date; null if the session is yet active (though it may still be timed out)

getOrigin() method

Syntax

String getOrigin ()

Returns

the session's origin

Usage

Returns the origin of the session; this is usually the IP of the calling web client, a terminal ID or something similar.

the session's origin

isSetCustomer() method

Syntax

boolean isSetCustomer ()

Returns

false if the customer is null. Otherwise, it returns true.

Usage

Checks whether the

customer

of this instance is set.

false if the customer is null. Otherwise, it returns true.

isSetLastActivity() method

Syntax

boolean isSetLastActivity ()

Returns

false if the `lastActivity` is null. Otherwise, it returns true.

Usage

Checks whether the

lastActivity

of this instance is set.

false if the `lastActivity` is null. Otherwise, it returns true.

isSetLogin() method

Syntax

```
boolean isSetLogin ()
```

Returns

false if the `login` is null. Otherwise, it returns true.

Usage

Checks whether the

login

of this instance is set.

false if the `login` is null. Otherwise, it returns true.

isSetLogoff() method

Syntax

```
boolean isSetLogoff ()
```

Returns

false if the `logoff` is null. Otherwise, it returns true.

Usage

Checks whether the

logoff

of this instance is set.

false if the `logoff` is null. Otherwise, it returns true.

isSetOrigin() method

Syntax

`boolean isSetOrigin ()`

Returns

false if the origin is null. Otherwise, it returns true.

Usage

Checks whether the

origin

of this instance is set.

false if the origin is null. Otherwise, it returns true.

setCustomer(Customer) method

Syntax

`void setCustomer ( Customer customer )`

Parameters

- **customer** – the `customer` value to set

Usage

Sets the

customer

of this instance.

setIdentification(final String) method

Syntax

`void setIdentification ( final String identification )`

Parameters

- **identification** – the identification to set

setLastActivity(Date) method

Syntax

void setLastActivity (Date *lastActivity*)

Parameters

- **lastActivity** – the lastActivity value to set

Usage

Sets the

lastActivity

of this instance.

setLogin(Date) method

Syntax

void setLogin (Date *login*)

Parameters

- **login** – the login value to set

Usage

Sets the

login

of this instance.

setLogoff(Date) method

Syntax

void setLogoff (Date *logoff*)

Parameters

- **logoff** – the logoff value to set

Usage

Sets the

logout

of this instance.

setOrigin(String) method

Syntax

```
void setOrigin ( String origin )
```

Parameters

- **origin** – the *origin* value to set

Usage

Sets the

origin

of this instance.

HistoryEntry class

Syntax

```
public class HistoryEntry
```

Usage

The `HistoryEntry` entity represents a change of customer or transaction data, be it a change to the customer entry or his address, identification, identity, etc.

It tracks the affected entity along with the property that was changed, the old and the new value that were set.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

HistoryEntry() constructor

Syntax

```
HistoryEntry ()
```

Usage

Creates a new instance of

```
HistoryEntry
```

.

getAffectedEntityId() method

Syntax

long getAffectedEntityId ()

getComment() method

Syntax

String getComment ()

getFieldName() method

Syntax

String getFieldName ()

getNewValue() method

Syntax

String getNewValue ()

getOldValue() method

Syntax

String getOldValue ()

getTableName() method

Syntax

String getTableName ()

isSetAffectedEntityId() method

Syntax

boolean isSetAffectedEntityId ()

Returns

false if the affectedEntityId is null. Otherwise, it returns true.

Usage

Checks whether the

affectedEntityId

of this instance is set.

false if the affectedEntityId is null. Otherwise, it returns true.

isSetComment() method

Syntax

```
boolean isSetComment ()
```

Returns

false if the comment is null. Otherwise, it returns true.

Usage

Checks whether the

comment

of this instance is set.

false if the comment is null. Otherwise, it returns true.

isSetFieldName() method

Syntax

```
boolean isSetFieldName ()
```

Returns

false if the fieldName is null . Otherwise, it returns true.

Usage

Checks whether the

fieldName

of this instance is set.

false if the fieldName is null . Otherwise, it returns true.

isSetNewValue() method

Syntax

```
boolean isSetNewValue ()
```

Returns

false if the *newValue* is null. Otherwise, it returns true.

Usage

Checks whether the

newValue

of this instance is set.

false if the *newValue* is null. Otherwise, it returns true.

isSetOldValue() method

Syntax

```
boolean isSetOldValue ()
```

Returns

false if the *oldValue* is null. Otherwise, it returns true.

Usage

Checks whether the

oldValue

of this instance is set.

false if the *oldValue* is null. Otherwise, it returns true.

isSetTableName() method

Syntax

```
boolean isSetTableName ()
```

Returns

false if the *tableName* is null . Otherwise, it returns true.

Usage

Checks whether the

tableName

of this instance is set.

false if the *tableName* is null . Otherwise, it returns true.

setAffectedEntityId(long) method

Syntax

void setAffectedEntityId (long *affectedEntityId*)

setComment(String) method

Syntax

void setComment (String *comment*)

setFieldName(String) method

Syntax

void setFieldName (String *fieldName*)

setNewValue(String) method

Syntax

void setNewValue (String *newValue*)

setOldValue(String) method

Syntax

void setOldValue (String *oldValue*)

setTableName(String) method

Syntax

void setTableName (String *tableName*)

Identification class

Syntax

public class Identification

Usage

The `Identification` models the customers' identifiers like CID, MSISDN etc.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

*Identification() constructor***Syntax**

```
Identification ()
```

Usage

Creates a new instance of

```
Identification
```

```
.
```

*getCustomer() method***Syntax**

```
Customer getCustomer ()
```

Returns

the Customer

Usage

Returns the customer associated to this identification.

```
the Customer
```

*getIdentificationStatus() method***Syntax**

```
int getIdentificationStatus ()
```

*getIdentificationType() method***Syntax**

```
int getIdentificationType ()
```

*getIdentifier() method***Syntax**

```
String getIdentifier ()
```

Returns

the identifier String (may be null)

Usage

Returns the identifier string the customer may be identified with.

the identifier `String` (may be `null`)

getOrgUnit() method

Gets the `OrgUnit` of this instance.

Syntax

```
OrgUnit getOrgUnit ()
```

Returns

the `orgUnit` of the Identification, may be `NULL` and is `NULL` by default!

Usage

the `orgUnit` of the Identification, may be `NULL` and is `NULL` by default!

getProvider() method

Syntax

```
String getProvider ()
```

Returns

the network provider `String` (may be `null`)

Usage

Returns the network provider ID for the consumer

the network provider `String` (may be `null`)

isActive() method

Syntax

```
boolean isActive ()
```

Returns

`true` if the identification could be used; `false` otherwise

Usage

Indicates if this identification may be used to identify a customer with. E.g. An IMEI identification should be inactivated if the mobile device was stolen/lost.

true if the identification could be used; false otherwise

isSetActive() method

Syntax

```
boolean isSetActive ()
```

Returns

false if the dbActive is null. Otherwise, it returns true.

Usage

Checks whether the

dbActive

of this instance is set.

false if the dbActive is null. Otherwise, it returns true.

isSetCustomer() method

Syntax

```
boolean isSetCustomer ()
```

Returns

false if the customer is null. Otherwise, it returns true.

Usage

Checks whether the

customer

of this instance is set.

false if the customer is null. Otherwise, it returns true.

isSetIdentificationStatus() method

Syntax

```
boolean isSetIdentificationStatus ()
```

Returns

false if the identificationStatus is null. Otherwise, it returns true.

Usage

Checks whether the

identificationStatus

of this instance is set.

false if the *identificationStatus* is null. Otherwise, it returns true.

isSetIdentificationType() method

Syntax

```
boolean isSetIdentificationType ()
```

Returns

false if the *identificationType* is null. Otherwise, it returns true.

Usage

Checks whether the

identificationType

of this instance is set.

false if the *identificationType* is null. Otherwise, it returns true.

isSetIdentifier() method

Syntax

```
boolean isSetIdentifier ()
```

Returns

false if the *identifier* is null. Otherwise, it returns true.

Usage

Checks whether the

identifier

of this instance is set.

false if the *identifier* is null. Otherwise, it returns true.

*isSetProvider() method***Syntax**

```
boolean isSetProvider ()
```

Returns

false if the provider is null. Otherwise, it returns true.

Usage

Checks whether the

provider

of this instance is set.

false if the provider is null. Otherwise, it returns true.

*setActive(boolean) method***Syntax**

```
void setActive ( boolean active )
```

Parameters

- **active** – the active value to set

Usage

Sets the

active

of this instance.

*setCustomer(Customer) method***Syntax**

```
void setCustomer ( Customer customer )
```

Parameters

- **customer** – the customer value to set

Usage

Sets the

customer

of this instance.

setIdentificationStatus(int) method

Syntax

`void setIdentificationStatus ( int identificationStatus )`

setIdentificationType(int) method

Syntax

`void setIdentificationType ( int identificationType )`

setIdentifier(String) method

Syntax

`void setIdentifier ( String identifier )`

Parameters

- **identifier** – the `identifier` value to set

Usage

Sets the

identifier

of this instance.

setOrgUnit(OrgUnit) method

Sets the `OrgUnit` of this instance.

Syntax

`void setOrgUnit ( OrgUnit orgUnit )`

Parameters

- **orgUnit** – the `orgUnit` to set

setProvider(String) method

Syntax

```
void setProvider ( String provider )
```

Parameters

- **provider** – the provider value to set

Usage

Sets the

network provider ID

of this instance.

Identity class

Syntax

```
public class Identity
```

Usage

The `Identity` models the customers' identities like passports, etc.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Identity() constructor

Syntax

```
Identity ()
```

Usage

Creates a new instance of

`Identity`

.

getCustomer() method

Syntax

Customer getCustomer ()

Returns

the Customer

Usage

Returns the customer associated to this identification.

the Customer

getExpirationDate() method

Syntax

Date getExpirationDate ()

Returns

the expirationDate

Usage

the expirationDate

getIdentity() method

Syntax

String getIdentity ()

Returns

the identifier String (may be null)

Usage

Returns the identifier string the customer may be identified with.

the identifier String (may be null)

getIssueCountry() method

Syntax

String getIssueCountry ()

Returns

the issueCountry

Usage

the issueCountry

getIssueDate() method

Syntax

Date getIssueDate ()

Returns

the issueDate

Usage

the issueDate

getIssuePlace() method

Syntax

String getIssuePlace ()

Returns

the issuePlace

Usage

the issuePlace

getIssuer() method

Syntax

String getIssuer ()

Returns

the issuer String (may be null)

Usage

Returns the issuer of the

Identity

the issuer `String` (may be `null`)

getStatus() method

Syntax

`int getStatus ()`

getType() method

Syntax

`int getType ()`

Returns

the identification's `Type`

Usage

Returns the type of this instance.

the identification's `Type`

isActive() method

Syntax

`boolean isActive ()`

Returns

`true` if the identification could be used; `false` otherwise

Usage

Indicates if this identification may be used to identify a customer with. E.g. An IMEI identification should be inactivated if the mobile device was stolen/lost.

`true` if the identification could be used; `false` otherwise

isSetActive() method

Syntax

`boolean isSetActive ()`

Returns

`false` if the `dbActive` is `null`. Otherwise, it returns `true`.

Usage

Checks whether the

dbActive

of this instance is set.

false if the dbActive is null. Otherwise, it returns true.

isSetCustomer() method

Syntax

```
boolean isSetCustomer ()
```

Returns

false if the customer is null. Otherwise, it returns true.

Usage

Checks whether the

customer

of this instance is set.

false if the customer is null. Otherwise, it returns true.

isSetExpirationDate() method

Syntax

```
boolean isSetExpirationDate ()
```

Returns

false if the expirationDate is null. Otherwise, it returns true.

Usage

Checks whether the

expirationDate

of this instance is set.

false if the expirationDate is null. Otherwise, it returns true.

isSetIdentity() method

Syntax

boolean isSetIdentity ()

Returns

false if the identity is null. Otherwise, it returns true.

Usage

Checks whether the

identity

of this instance is set.

false if the identity is null. Otherwise, it returns true.

isSetIssueCountry() method

Syntax

boolean isSetIssueCountry ()

Returns

false if the issueCountry is null. Otherwise, it returns true.

Usage

Checks whether the

issueCountry

of this instance is set.

false if the issueCountry is null. Otherwise, it returns true.

isSetIssueDate() method

Syntax

boolean isSetIssueDate ()

Returns

false if the issueDate is null . Otherwise, it returns true.

Usage

Checks whether the

issueDate

of this instance is set.

false if the `issueDate` is null. Otherwise, it returns true.

isSetIssuePlace() method

Syntax

```
boolean isSetIssuePlace ()
```

Returns

false if the `issuePlace` is null. Otherwise, it returns true.

Usage

Checks whether the

issuePlace

of this instance is set.

false if the `issuePlace` is null. Otherwise, it returns true.

isSetIssuer() method

Syntax

```
boolean isSetIssuer ()
```

Returns

false if the `issuer` is null. Otherwise, it returns true.

Usage

Checks whether the

issuer

of this instance is set.

false if the `issuer` is null. Otherwise, it returns true.

isSetStatus() method

Syntax

```
boolean isSetStatus ()
```

Returns

false if the status is null. Otherwise, it returns true.

Usage

Checks whether the

status

of this instance is set.

false if the status is null. Otherwise, it returns true.

isSetType() method

Syntax

```
boolean isSetType ()
```

Returns

false if the type is null. Otherwise, it returns true.

Usage

Checks whether the

type

of this instance is set.

false if the type is null. Otherwise, it returns true.

setActive(boolean) method

Syntax

```
void setActive ( boolean active )
```

Parameters

- **active** – the active value to set

Usage

Sets the

active

of this instance.

setCustomer(Customer) method

Syntax

```
void setCustomer ( Customer customer )
```

Parameters

- **customer** – the `customer` value to set

Usage

Sets the

customer

of this instance.

setExpirationDate(Date) method

Syntax

```
void setExpirationDate ( Date expirationDate )
```

Parameters

- **expirationDate** – the `expirationDate` to set

setIdentity(String) method

Syntax

```
void setIdentity ( String identity )
```

Parameters

- **identity** – the `identity` value to set

Usage

Sets the

identity

com.sybase365.mobiliser.money package

of this instance.

setIssueCountry(String) method

Syntax

```
void setIssueCountry ( String issueCountry )
```

Parameters

- **issueCountry** – the issueCountry to set

setIssueDate(Date) method

Syntax

```
void setIssueDate ( Date issueDate )
```

Parameters

- **issueDate** – the issueDate to set

setIssuePlace(String) method

Syntax

```
void setIssuePlace ( String issuePlace )
```

Parameters

- **issuePlace** – the issuePlace to set

setIssuer(String) method

Syntax

```
void setIssuer ( String issuer )
```

Parameters

- **issuer** – the issuer value to set

Usage

Sets the

network provider ID

of this instance.

setStatus(int) method

Syntax

```
void setStatus ( int status )
```

setType(int) method

Syntax

```
void setType ( int type )
```

Parameters

- **type** – the Type to set

Usage

Sets the type of this instance.

Invoice class

Syntax

```
public class Invoice
```

Usage

The Invoice represents a invoice.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Invoice() constructor

Syntax

```
Invoice ()
```

Usage

Creates a new instance of

```
Invoice
```

.

getAmount() method

Syntax

long getAmount ()

getCurrency() method

Syntax

Currency getCurrency ()

Returns

the transaction's Currency

Usage

Returns the dbCurrency this transaction is executed with.
the transaction's Currency

getDate() method

Syntax

Date getDate ()

Returns

the date

Usage

the date

getDueDate() method

Syntax

Date getDueDate ()

Returns

the dueDate

Usage

the dueDate

getExpirationDate() method

Syntax

Date getExpirationDate ()

Returns

the payDate

Usage

the payDate

getInvoiceAttributes() method

Syntax

Set< InvoiceAttribute > getInvoiceAttributes ()

Returns

the attributes

Usage

the attributes

getInvoiceConfiguration() method

Syntax

InvoiceConfiguration getInvoiceConfiguration ()

Returns

the invoiceConfiguration

Usage

the invoiceConfiguration

getPayDate() method

Syntax

Date getPayDate ()

Returns

the payDate

Usage

the payDate

getReference() method

Syntax

```
String getReference ()
```

Returns

the reference

Usage

the reference

getStatus() method

Syntax

```
int getStatus ()
```

isSetAmount() method

Syntax

```
boolean isSetAmount ()
```

Returns

false if the amount is null. Otherwise, it returns true.

Usage

Checks whether the

amount

of this instance is set.

false if the amount is null. Otherwise, it returns true.

isSetCurrency() method

Syntax

```
boolean isSetCurrency ()
```

Returns

false if the `dbCurrency` is null. Otherwise, it returns true.

Usage

Checks whether the

dbCurrency

of this instance is set.

false if the `dbCurrency` is null. Otherwise, it returns true.

isSetDate() method

Syntax

```
boolean isSetDate ()
```

Returns

false if the `date` is null. Otherwise, it returns true.

Usage

Checks whether the

date

of this instance is set.

false if the `date` is null. Otherwise, it returns true.

isSetDueDate() method

Syntax

```
boolean isSetDueDate ()
```

Returns

false if the `dueDate` is null. Otherwise, it returns true.

Usage

Checks whether the

dueDate

of this instance is set.

false if the `dueDate` is null. Otherwise, it returns true.

isSetExpirationDate() method

Syntax

```
boolean isSetExpirationDate ()
```

Returns

false if the expirationDate is null. Otherwise, it returns true.

Usage

Checks whether the

expirationDate

of this instance is set.

false if the expirationDate is null. Otherwise, it returns true.

isSetInvoiceAttributes() method

Syntax

```
boolean isSetInvoiceAttributes ()
```

Returns

false if the invoiceAttributes is null. Otherwise, it returns true.

Usage

Checks whether the

invoiceAttributes

of this instance is set.

false if the invoiceAttributes is null. Otherwise, it returns true.

isSetInvoiceConfiguration() method

Syntax

```
boolean isSetInvoiceConfiguration ()
```

Returns

false if the invoiceConfiguration is null. Otherwise, it returns true.

Usage

Checks whether the

invoiceConfiguration

of this instance is set.

false if the *invoiceConfiguration* is null. Otherwise, it returns true.

isSetPayDate() method

Syntax

```
boolean isSetPayDate ()
```

Returns

false if the *payDate* is null. Otherwise, it returns true.

Usage

Checks whether the

payDate

of this instance is set.

false if the *payDate* is null. Otherwise, it returns true.

isSetReference() method

Syntax

```
boolean isSetReference ()
```

Returns

false if the *reference* is null . Otherwise, it returns true.

Usage

Checks whether the

reference

of this instance is set.

false if the *reference* is null . Otherwise, it returns true.

isSetStatus() method

Syntax

boolean isSetStatus ()

Returns

false if the status is null. Otherwise, it returns true.

Usage

Checks whether the

status

of this instance is set.

false if the status is null. Otherwise, it returns true.

setAmount(long) method

Syntax

void setAmount (long *amount*)

setCurrency(Currency) method

Syntax

void setCurrency (Currency *dbCurrency*)

Parameters

- **dbCurrency** – the *dbCurrency* value to set

Usage

Sets the

dbCurrency

of this instance.

setDate(Date) method

Syntax

void setDate (Date *date*)

Parameters

- **date** – the date to set

setDueDate(Date) method

Syntax

```
void setDueDate ( Date dueDate )
```

Parameters

- **dueDate** – the dueDate to set

setExpirationDate(Date) method

Syntax

```
void setExpirationDate ( Date expirationDate )
```

Parameters

- **expirationDate** – the expirationDate to set

setInvoiceAttributes(Set< InvoiceAttribute >) method

Syntax

```
void setInvoiceAttributes ( Set< InvoiceAttribute >  
invoiceAttributes )
```

Parameters

- **attributes** – the attributes to set

setInvoiceConfiguration(InvoiceConfiguration) method

Syntax

```
void setInvoiceConfiguration ( InvoiceConfiguration  
invoiceConfiguration )
```

Parameters

- **invoiceConfiguration** – the invoiceConfiguration to set

setPayDate(Date) method

Syntax

```
void setPayDate ( Date payDate )
```

Parameters

- **payDate** – the payDate to set

setReference(String) method

Syntax

```
void setReference ( String reference )
```

Parameters

- **reference** – the reference to set

setStatus(int) method

Syntax

```
void setStatus ( int status )
```

InvoiceAttribute class

Syntax

```
public class InvoiceAttribute
```

Usage

The `Attribute` is used for the invoice attributes.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

InvoiceAttributePK class

Syntax

```
public class InvoiceAttributePK
```

Usage

This is the embedded Primary Key

© 2000-2009 by paybox solutions AG

Revision:1.14Name: not supported by cvs2svn

InvoiceAttributePK(Invoice, int) constructor

Constructs a new mapping PK.

Syntax

`InvoiceAttributePK ( Invoice invoice , int attributeTypeId )`

Parameters

- **invoice** –
- **attribute** –

InvoiceAttributePK() constructor

The empty constructor.

Syntax

`InvoiceAttributePK ()`

equals(Object) method

Syntax

`boolean equals ( Object o )`

Parameters

- **o** – the Object to compare with

Returns

true if this class is assignable from o's class and has the same PK values as this instance

Usage

Indicates if another object is

equal to

this one.

true if this class is assignable from o's class and has the same PK values as this instance

getAttributeTypeId() method

Syntax

int getAttributeTypeId ()

Returns

the attributeTypeId

Usage

the attributeTypeId

getInvoice() method

Syntax

Invoice getInvoice ()

Returns

the invoice

Usage

the invoice

hashCode() method

Syntax

int hashCode ()

Returns

a hash value for this instance

Usage

Returns a hash value.

a hash value for this instance

isSetAttributeTypeId() method

Syntax

boolean isSetAttributeTypeId ()

Returns

false if the `attributeTypeId` is null. Otherwise, it returns true.

Usage

Checks whether the

attributeTypeId

of this instance is set.

false if the `attributeTypeId` is null. Otherwise, it returns true.

isSetInvoice() method

Syntax

```
boolean isSetInvoice ()
```

Returns

false if the `invoice` is null. Otherwise, it returns true.

Usage

Checks whether the

invoice

of this instance is set.

false if the `invoice` is null. Otherwise, it returns true.

toString() method

Syntax

```
String toString ()
```

Returns

a `String` representing this instance

Usage

Returns a string representation of this object.

a `String` representing this instance

InvoiceAttribute() constructor

Syntax

InvoiceAttribute ()

Usage

Creates a new instance of

InvoiceAttribute

.

InvoiceAttribute(InvoiceAttributePK) constructor

Syntax

InvoiceAttribute (InvoiceAttributePK *id*)

Parameters

- **id** – the PK value

Usage

Creates a new instance of

InvoiceAttribute

with a preset primary key.

getAttributeTypeId() method

Syntax

int getAttributeTypeId ()

Returns

the attributeTypeId

Usage

the attributeTypeId

getInvoice() method

Syntax

Invoice getInvoice ()

Returns

the invoice

Usage

the invoice

getStrValue() method

Syntax

```
String getStrValue ()
```

Returns

the strValue

Usage

the strValue

isSetAttributeTypeId() method

Syntax

```
boolean isSetAttributeTypeId ()
```

Returns

false if the id or the id's attributeTypeId is null. Otherwise, it returns true.

Usage

Checks whether the

attributeTypeId

of this instance's PK is set.

false if the id or the id's attributeTypeId is null. Otherwise, it returns true.

isSetInvoice() method

Syntax

```
boolean isSetInvoice ()
```

Returns

false if the id or the id's invoice is null. Otherwise, it returns true.

Usage

Checks whether the

invoice

of this instance's PK is set.

false if the *id* or the *id*'s *invoice* is null. Otherwise, it returns true.

isSetStrValue() method

Syntax

```
boolean isSetStrValue ()
```

Returns

false if the *strValue* is null. Otherwise, it returns true.

Usage

Checks whether the

strValue

of this instance is set.

false if the *strValue* is null. Otherwise, it returns true.

setStrValue(String) method

Syntax

```
void setStrValue ( String strValue )
```

Parameters

- **strValue** – the *strValue* to set

InvoiceConfiguration class

Syntax

```
public class InvoiceConfiguration
```

Usage

The *Invoice* represents an invoice configuration.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

InvoiceConfiguration() constructor

Syntax

```
InvoiceConfiguration ()
```

Usage

Creates a new instance of

```
InvoiceConfiguration
```

.

getAlias() method

Syntax

```
String getAlias ()
```

Returns

the alias

Usage

the alias

getAmount() method

Syntax

```
Long getAmount ()
```

Returns

the amount

Usage

the amount

getAuthorisationThreshold() method

Syntax

```
Long getAuthorisationThreshold ()
```

Returns

the authorisationThreshold

Usage

the authorisationThreshold

getAutoPayDelay() method

Returns the timeframe period or the invoice configuratio's auto-pay delay according to the W3C XSD timeframe data type, if empty there is no reoccurrence.

Syntax

```
Duration getAutoPayDelay ()
```

Returns

the autoPayDelay

Usage

the autoPayDelay

getBillingCycle() method

Returns the timeframe period or the subscription According to the W3C XSD timeframe data type, if empty there is no reoccurrence.

Syntax

```
Duration getBillingCycle ()
```

Returns

the billingCycle

Usage

the billingCycle

getCustomer() method

Syntax

```
Customer getCustomer ()
```

Returns

the customer

Usage

the customer

getDefaultPi() method

Syntax

```
PaymentInstrument getDefaultPi ()
```

Returns

the defaultPi

Usage

the defaultPi

getEndDate() method

Syntax

```
Date getEndDate ()
```

Returns

the endDate

Usage

the endDate

getInvoiceConfigurationAttributes() method

Syntax

```
Set< InvoiceConfigurationAttribute >  
getInvoiceConfigurationAttributes ()
```

Returns

the attributes

Usage

the attributes

getInvoices() method

Syntax

Set< Invoice >getInvoices ()

Returns

the invoices

Usage

the invoices

getInvoiceType() method

Syntax

InvoiceType getInvoiceType ()

Returns

the invType

Usage

the invType

getLastInvoice() method

Syntax

Date getLastInvoice ()

Returns

the lastInvoice

Usage

the lastInvoice

getMaxDelay() method

Returns the maximum delay to still process an open invoice that was generated from this invoice configuration.

Syntax

Duration getMaxDelay ()

Returns

the billingCycle

Usage

XSD duration format.

the billingCycle

getNextInvoice() method

Syntax

Date getNextInvoice ()

Returns

the nextInvoice

Usage

the nextInvoice

getReference() method

Syntax

String getReference ()

Returns

the reference

Usage

the reference

getStartDate() method

Syntax

Date getStartDate ()

Returns

the startDate

Usage

the startDate

getStatus() method

Syntax

`int getStatus ()`

Returns

the status

Usage

the status

isActive() method

Syntax

`boolean isActive ()`

Returns

true if this instance is active; false if it is inactive

Usage

Indicates if this invoice configuration is active.

true if this instance is active; false if it is inactive

isSetActive() method

Syntax

`boolean isSetActive ()`

Returns

false if the dbActive is null. Otherwise, it returns true.

Usage

Checks whether the

dbActive

of this instance is set.

false if the dbActive is null. Otherwise, it returns true.

isSetAlias() method

Syntax

```
boolean isSetAlias ()
```

Returns

false if the alias is null. Otherwise, it returns true.

Usage

Checks whether the

alias

of this instance is set.

false if the alias is null. Otherwise, it returns true.

isSetAmount() method

Syntax

```
boolean isSetAmount ()
```

Returns

false if the amount is null. Otherwise, it returns true.

Usage

Checks whether the

amount

of this instance is set.

false if the amount is null. Otherwise, it returns true.

isSetAuthorisationThreshold() method

Syntax

```
boolean isSetAuthorisationThreshold ()
```

Returns

false if the authorisationThreshold is null. Otherwise, it returns true.

Usage

Checks whether the

authorisationThreshold

of this instance is set.

false if the `authorisationThreshold` is null. Otherwise, it returns true.

isSetAutoPayDelay() method

Syntax

```
boolean isSetAutoPayDelay ()
```

Returns

false if the `autoPayDelay` is null. Otherwise, it returns true.

Usage

Checks whether the

autoPayDelay

of this instance is set.

false if the `autoPayDelay` is null. Otherwise, it returns true.

isSetBillingCycle() method

Syntax

```
boolean isSetBillingCycle ()
```

Returns

false if the `billingCycle` is null. Otherwise, it returns true.

Usage

Checks whether the

billingCycle

of this instance is set.

false if the `billingCycle` is null. Otherwise, it returns true.

isSetCustomer() method

Syntax

```
boolean isSetCustomer ()
```

Returns

false if the customer is null. Otherwise, it returns true.

Usage

Checks whether the

customer

of this instance is set.

false if the customer is null. Otherwise, it returns true.

isSetDefaultPi() method

Syntax

```
boolean isSetDefaultPi ()
```

Returns

false if the defaultPi is null . Otherwise, it returns true.

Usage

Checks whether the

defaultPi

of this instance is set.

false if the defaultPi is null . Otherwise, it returns true.

isSetEndDate() method

Syntax

```
boolean isSetEndDate ()
```

Returns

false if the endDate is null. Otherwise, it returns true.

Usage

Checks whether the

endDate

of this instance is set.

false if the *endDate* is null. Otherwise, it returns true.

isSetInvoiceConfigurationAttributes() method

Syntax

```
boolean isSetInvoiceConfigurationAttributes ()
```

Returns

false if the *attributes* is null. Otherwise, it returns true.

Usage

Checks whether the

attributes

of this instance is set.

false if the *attributes* is null. Otherwise, it returns true.

isSetInvoices() method

Syntax

```
boolean isSetInvoices ()
```

Returns

false if the *invoices* is null. Otherwise, it returns true.

Usage

Checks whether the

invoices

of this instance is set.

false if the *invoices* is null. Otherwise, it returns true.

isSetInvoiceType() method

Syntax

```
boolean isSetInvoiceType ()
```

Returns

false if the `invoiceType` is null. Otherwise, it returns true.

Usage

Checks whether the

invoiceType

of this instance is set.

false if the `invoiceType` is null. Otherwise, it returns true.

isSetLastInvoice() method

Syntax

```
boolean isSetLastInvoice ()
```

Returns

false if the `lastInvoice` is null. Otherwise, it returns true.

Usage

Checks whether the

lastInvoice

of this instance is set.

false if the `lastInvoice` is null. Otherwise, it returns true.

isSetMaxDelay() method

Syntax

```
boolean isSetMaxDelay ()
```

Returns

false if the `maxDelay` is null. Otherwise, it returns true.

Usage

Checks whether the

maxDelay

of this instance is set.

false if the maxDelay is null. Otherwise, it returns true.

isSetNextInvoice() method

Syntax

```
boolean isSetNextInvoice ()
```

Returns

false if the nextInvoice is null. Otherwise, it returns true.

Usage

Checks whether the

nextInvoice

of this instance is set.

false if the nextInvoice is null. Otherwise, it returns true.

isSetReference() method

Syntax

```
boolean isSetReference ()
```

Returns

the reference

Usage

the reference

isSetStartDate() method

Syntax

```
boolean isSetStartDate ()
```

Returns

false if the startDate is null . Otherwise, it returns true.

Usage

Checks whether the

startDate

of this instance is set.

false if the startDate is null . Otherwise, it returns true.

isSetStatus() method

Syntax

```
boolean isSetStatus ()
```

Returns

false if the status is null. Otherwise, it returns true.

Usage

Checks whether the

status

of this instance is set.

false if the status is null. Otherwise, it returns true.

setActive(boolean) method

Syntax

```
void setActive ( boolean active )
```

Parameters

- **active** – the active value to set

Usage

Sets the

active

of this instance.

setAlias(String) method

Syntax

```
void setAlias ( String alias )
```

Parameters

- **alias** – the alias to set

setAmntAuthorisationThreshold(Long) method

Syntax

void setAmntAuthorisationThreshold (Long *authorisationThreshold*)

Parameters

- **authorisationThreshold** – the authorisationThreshold to set

setAmount(Long) method

Syntax

void setAmount (Long *amount*)

Parameters

- **amount** – the amount to set

setAutoPayDelay(Duration) method

Syntax

void setAutoPayDelay (Duration *autoPayDelay*)

Parameters

- **autoPayDelay** – the autoPayDelay to set

setBillingCycle(Duration) method

Syntax

void setBillingCycle (Duration *billingCycle*)

Parameters

- **billingCycle** – the billingCycle to set

setCustomer(Customer) method

Syntax

void setCustomer (Customer *customer*)

Parameters

- **customer** – the customer to set

setDefaultPi(PaymentInstrument) method

Syntax

```
void setDefaultPi ( PaymentInstrument defaultPi )
```

Parameters

- **defaultPi** – the defaultPi to set

setEndDate(Date) method

Syntax

```
void setEndDate ( Date endDate )
```

Parameters

- **endDate** – the endDate to set

setInvoiceConfigurationAttributes(Set< InvoiceConfigurationAttribute >) method

Syntax

```
void setInvoiceConfigurationAttributes ( Set<  
InvoiceConfigurationAttribute > attributes )
```

Parameters

- **attributes** – the invoice configuration attributes to set

setInvoices(Set< Invoice >) method

Syntax

```
void setInvoices ( Set< Invoice > invoices )
```

Parameters

- **invoices** – the invoices to set

setInvoiceType(InvoiceType) method

Syntax

void setInvoiceType (InvoiceType *invoiceType*)

Parameters

- **invoiceType** – the invType to set

setLastInvoice(Date) method

Syntax

void setLastInvoice (Date *lastInvoice*)

Parameters

- **lastInvoice** – the lastInvoice to set

setMaxDelay(Duration) method

Syntax

void setMaxDelay (Duration *maxDelay*)

Parameters

- **maxDelay** – the maxDelay to set

setNextInvoice(Date) method

Syntax

void setNextInvoice (Date *nextInvoice*)

Parameters

- **nextInvoice** – the nextInvoice to set

setReference(String) method

Syntax

void setReference (String *reference*)

Parameters

- **reference** – the reference to set

setStartDate(Date) method

Syntax

```
void setStartDate ( Date startDate )
```

Parameters

- **startDate** – the startDate to set

setStatus(int) method

Syntax

```
void setStatus ( int status )
```

Parameters

- **status** – the status to set

InvoiceConfigurationAttribute class

Syntax

```
public class InvoiceConfigurationAttribute
```

Usage

The `Attribute` is used for the invoice configuration attributes.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

InvoiceConfigurationAttributesPK class

Syntax

```
public class InvoiceConfigurationAttributesPK
```

Usage

This is the embedded Primary Key

© 2000-2009 by paybox solutions AG

Revision:1.23Name:not supported by cvs2svn

InvoiceConfigurationAttributesPK(InvoiceConfiguration, int) constructor
Constructs a new mapping PK.

Syntax

*InvoiceConfigurationAttributesPK (InvoiceConfiguration
invoiceConfiguration , int attributeTypeId)*

Parameters

- **invoiceConfiguration** –
- **attribute** –

InvoiceConfigurationAttributesPK() constructor
The empty constructor.

Syntax

InvoiceConfigurationAttributesPK ()

equals(Object) method

Syntax

boolean equals (Object o)

Parameters

- **o** – the Object to compare with

Returns

true if this class is assignable from o's class and has the same PK values as this instance

Usage

Indicates if another object is

equal to

this one.

true if this class is assignable from o's class and has the same PK values as this instance

getAttributeTypeId() method

Syntax

int getAttributeTypeId ()

Returns

the attributeTypeId

Usage

the attributeTypeId

getInvoiceConfiguration() method

Syntax

```
InvoiceConfiguration getInvoiceConfiguration ()
```

Returns

the invoice configuration

Usage

the invoice configuration

hashCode() method

Syntax

```
int hashCode ()
```

Returns

a hash value for this instance

Usage

Returns a hash value.

a hash value for this instance

isSetAttributeTypeId() method

Syntax

```
boolean isSetAttributeTypeId ()
```

Returns

false if the attributeTypeId is null. Otherwise, it returns true.

Usage

Checks whether the

attributeTypeId

of this instance is set.

false if the *attributeTypeId* is null. Otherwise, it returns true.

isSetInvoiceConfiguration() method

Syntax

boolean isSetInvoiceConfiguration ()

Returns

false if the *invoiceConfiguration* is null. Otherwise, it returns true.

Usage

Checks whether the

invoiceConfiguration

of this instance is set.

false if the *invoiceConfiguration* is null. Otherwise, it returns true.

toString() method

Syntax

String toString ()

Returns

a String representing this instance

Usage

Returns a string representation of this object.

a String representing this instance

InvoiceConfigurationAttribute() constructor

Syntax

InvoiceConfigurationAttribute ()

Usage

Creates a new instance of

InvoiceConfigurationAttribute

.

InvoiceConfigurationAttribute(InvoiceConfigurationAttributesPK) constructor

Syntax

```
InvoiceConfigurationAttribute  
( InvoiceConfigurationAttributesPK id )
```

Parameters

- **id** – the PK value

Usage

Creates a new instance of
`InvoiceConfigurationAttribute`
with a preset primary key.

getAttributeTypeId() method

Syntax

```
int getAttributeTypeId ()
```

Returns

the attributeTypeId

Usage

the attributeTypeId

getInvoiceConfiguration() method

Syntax

```
InvoiceConfiguration getInvoiceConfiguration ()
```

Returns

the invoice configuration

Usage

the invoice configuration

getStrValue() method

Syntax

String getStrValue ()

Returns

the strValue

Usage

the strValue

isSetAttributeTypeId() method

Syntax

boolean isSetAttributeTypeId ()

Returns

false if the id or the id's attributeTypeId is null. Otherwise, it returns true.

Usage

Checks whether the

attributeTypeId

of this instance's PK is set.

false if the id or the id's attributeTypeId is null. Otherwise, it returns true.

isSetInvoiceConfiguration() method

Syntax

boolean isSetInvoiceConfiguration ()

Returns

false if the id or the id's invoiceConfiguration is null. Otherwise, it returns true.

Usage

Checks whether the

invoiceConfiguration

of this instance's PK is set.

false if the id or the id's invoiceConfiguration is null. Otherwise, it returns true.

isSetStrValue() method

Syntax

```
boolean isSetStrValue ()
```

Returns

false if the strValue is null. Otherwise, it returns true.

Usage

Checks whether the

strValue

of this instance is set.

false if the strValue is null. Otherwise, it returns true.

setStrValue(String) method

Syntax

```
void setStrValue ( String strValue )
```

Parameters

- **strValue** – the strValue to set

LimitSum class

Syntax

```
public class LimitSum
```

Usage

The `Limit` tracks the limits and turnover for different entities in the mobiliser platform like customers, payment instruments, wallet entries, etc.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

LimitSum() constructor

Syntax

LimitSum ()

Usage

Creates a new instance of

LimitSum

.

compareTo(LimitSum) method

This is mainly used to sort the LimitSum objects before they get locked to make sure no dead-lock occurs.

Syntax

int compareTo (LimitSum o)

getAbsoluteCreditCount() method

Syntax

long getAbsoluteCreditCount ()

Returns

the absoluteCreditCount

Usage

the absoluteCreditCount

getAbsoluteCreditSum() method

Syntax

long getAbsoluteCreditSum ()

Returns

the absoluteCreditSum

Usage

the absoluteCreditSum

getAbsoluteDebitCount() method

Syntax

long getAbsoluteDebitCount ()

Returns

the absoluteDebitCount

Usage

the absoluteDebitCount

getAbsoluteDebitSum() method

Syntax

long getAbsoluteDebitSum ()

Returns

the absoluteDebitSum

Usage

the absoluteDebitSum

getCustomer() method

Syntax

Customer getCustomer ()

getDailyCreditCount() method

Syntax

long getDailyCreditCount ()

Returns

the dailyCreditCount

Usage

the dailyCreditCount

getDailyCreditSum() method

Syntax

long getDailyCreditSum ()

Returns

the dailyCreditSum

Usage

the dailyCreditSum

getDailyDebitCount() method

Syntax

long getDailyDebitCount ()

Returns

the dailyDebitCount

Usage

the dailyDebitCount

getDailyDebitSum() method

Syntax

long getDailyDebitSum ()

Returns

the dailyDebitSum

Usage

the dailyDebitSum

getLastLimitUpdate() method

Syntax

Date getLastLimitUpdate ()

Returns

the Date representing the timestamp of the last limit update

Usage

Returns the point in time when the limit sums and counts have been updated. Used to evaluate if a weekly sum etc. should be reset.

the `Date` representing the timestamp of the last limit update

getLimitClass() method

Syntax

```
LimitClass getLimitClass ()
```

Returns

the `LimitClass`

Usage

the `LimitClass`

getMonthlyCreditCount() method

Syntax

```
long getMonthlyCreditCount ()
```

Returns

the `monthlyCreditCount`

Usage

the `monthlyCreditCount`

getMonthlyCreditSum() method

Syntax

```
long getMonthlyCreditSum ()
```

Returns

the `monthlyCreditSum`

Usage

the `monthlyCreditSum`

getMonthlyDebitCount() method

Syntax

long getMonthlyDebitCount ()

Returns

the monthlyDebitCount

Usage

the monthlyDebitCount

getMonthlyDebitSum() method

Syntax

long getMonthlyDebitSum ()

Returns

the monthlyDebitSum

Usage

the monthlyDebitSum

getPaymentInstrument() method

Syntax

PaymentInstrument getPaymentInstrument ()

getSingleCreditAmount() method

Syntax

Long getSingleCreditAmount ()

getSingleDebitAmount() method

Syntax

Long getSingleDebitAmount ()

getWalletEntry() method

Syntax

WalletEntry getWalletEntry ()

getWeeklyCreditCount() method

Syntax

long getWeeklyCreditCount ()

Returns

the weeklyCreditCount

Usage

the weeklyCreditCount

getWeeklyCreditSum() method

Syntax

long getWeeklyCreditSum ()

Returns

the weeklyCreditSum

Usage

the weeklyCreditSum

getWeeklyDebitCount() method

Syntax

long getWeeklyDebitCount ()

Returns

the weeklyDebitCount

Usage

the weeklyDebitCount

getWeeklyDebitSum() method

Syntax

long getWeeklyDebitSum ()

Returns

the weeklyDebitSum

Usage

the weeklyDebitSum

isSetAbsoluteCreditCount() method

Syntax

boolean isSetAbsoluteCreditCount ()

Returns

false if the absoluteCreditCount is null. Otherwise, it returns true.

Usage

Checks whether the

absoluteCreditCount

of this instance is set.

false if the absoluteCreditCount is null. Otherwise, it returns true.

isSetAbsoluteCreditSum() method

Syntax

boolean isSetAbsoluteCreditSum ()

Returns

false if the absoluteCreditSum is null. Otherwise, it returns true.

Usage

Checks whether the

absoluteCreditSum

of this instance is set.

false if the `absoluteCreditSum` is null. Otherwise, it returns true.

isSetAbsoluteDebitCount() method

Syntax

```
boolean isSetAbsoluteDebitCount ()
```

Returns

false if the `absoluteDebitCount` is null. Otherwise, it returns true.

Usage

Checks whether the

absoluteDebitCount

of this instance is set.

false if the `absoluteDebitCount` is null. Otherwise, it returns true.

isSetAbsoluteDebitSum() method

Syntax

```
boolean isSetAbsoluteDebitSum ()
```

Returns

false if the `absoluteDebitSum` is null. Otherwise, it returns true.

Usage

Checks whether the

absoluteDebitSum

of this instance is set.

false if the `absoluteDebitSum` is null. Otherwise, it returns true.

isSetCustomer() method

Syntax

```
boolean isSetCustomer ()
```

Returns

false if the `customer` is null. Otherwise, it returns true.

Usage

Checks whether the

customer

of this instance is set.

false if the customer is null. Otherwise, it returns true.

isSetDailyCreditCount() method

Syntax

```
boolean isSetDailyCreditCount ()
```

Returns

false if the dailyCreditCount is null. Otherwise, it returns true.

Usage

Checks whether the

dailyCreditCount

of this instance is set.

false if the dailyCreditCount is null. Otherwise, it returns true.

isSetDailyCreditSum() method

Syntax

```
boolean isSetDailyCreditSum ()
```

Returns

false if the dailyCreditSum is null. Otherwise, it returns true.

Usage

Checks whether the

dailyCreditSum

of this instance is set.

false if the dailyCreditSum is null. Otherwise, it returns true.

isSetDailyDebitCount() method

Syntax

`boolean isSetDailyDebitCount ()`

Returns

false if the `dailyDebitCount` is null. Otherwise, it returns true.

Usage

Checks whether the

dailyDebitCount

of this instance is set.

false if the `dailyDebitCount` is null. Otherwise, it returns true.

isSetDailyDebitSum() method

Syntax

`boolean isSetDailyDebitSum ()`

Returns

false if the `dailyDebitSum` is null. Otherwise, it returns true.

Usage

Checks whether the

dailyDebitSum

of this instance is set.

false if the `dailyDebitSum` is null. Otherwise, it returns true.

isSetLastLimitUpdate() method

Syntax

`boolean isSetLastLimitUpdate ()`

Returns

false if the `lastLimitUpdate` is null. Otherwise, it returns true.

Usage

Checks whether the

lastLimitUpdate

of this instance is set.

false if the lastLimitUpdate is null. Otherwise, it returns true.

isSetLimitClass() method

Syntax

```
boolean isSetLimitClass ()
```

Returns

false if the limitClass is null. Otherwise, it returns true.

Usage

Checks whether the

limitClass

of this instance is set.

false if the limitClass is null. Otherwise, it returns true.

isSetMonthlyCreditCount() method

Syntax

```
boolean isSetMonthlyCreditCount ()
```

Returns

false if the monthlyCreditCount is null. Otherwise, it returns true.

Usage

Checks whether the

monthlyCreditCount

of this instance is set.

false if the monthlyCreditCount is null. Otherwise, it returns true.

isSetMonthlyCreditSum() method

Syntax

boolean isSetMonthlyCreditSum ()

Returns

false if the monthlyCreditSum is null. Otherwise, it returns true.

Usage

Checks whether the

monthlyCreditSum

of this instance is set.

false if the monthlyCreditSum is null. Otherwise, it returns true.

isSetMonthlyDebitCount() method

Syntax

boolean isSetMonthlyDebitCount ()

Returns

false if the monthlyDebitCount is null. Otherwise, it returns true.

Usage

Checks whether the

monthlyDebitCount

of this instance is set.

false if the monthlyDebitCount is null. Otherwise, it returns true.

isSetMonthlyDebitSum() method

Syntax

boolean isSetMonthlyDebitSum ()

Returns

false if the monthlyDebitSum is null. Otherwise, it returns true.

Usage

Checks whether the

monthlyDebitSum

of this instance is set.

false if the `monthlyDebitSum` is null. Otherwise, it returns true.

isSetPaymentInstrument() method

Syntax

```
boolean isSetPaymentInstrument ()
```

Returns

false if the `paymentInstrument` is null. Otherwise, it returns true.

Usage

Checks whether the

paymentInstrument

of this instance is set.

false if the `paymentInstrument` is null. Otherwise, it returns true.

isSetSingleCreditAmount() method

Syntax

```
boolean isSetSingleCreditAmount ()
```

Returns

false if the `singleCreditAmount` is null. Otherwise, it returns true.

Usage

Checks whether the

singleCreditAmount

of this instance is set.

false if the `singleCreditAmount` is null. Otherwise, it returns true.

*isSetSingleDebitAmount() method***Syntax**

```
boolean isSetSingleDebitAmount ()
```

Returns

false if the `singleDebitAmount` is null. Otherwise, it returns true.

Usage

Checks whether the

singleDebitAmount

of this instance is set.

false if the `singleDebitAmount` is null. Otherwise, it returns true.

*isSetWalletEntry() method***Syntax**

```
boolean isSetWalletEntry ()
```

Returns

false if the `walletEntry` is null. Otherwise, it returns true.

Usage

Checks whether the

walletEntry

of this instance is set.

false if the `walletEntry` is null. Otherwise, it returns true.

*isSetWeeklyCreditCount() method***Syntax**

```
boolean isSetWeeklyCreditCount ()
```

Returns

false if the `weeklyCreditCount` is null. Otherwise, it returns true.

Usage

Checks whether the

weeklyCreditCount

of this instance is set.

false if the weeklyCreditCount is null. Otherwise, it returns true.

isSetWeeklyCreditSum() method

Syntax

```
boolean isSetWeeklyCreditSum ()
```

Returns

false if the weeklyCreditSum is null. Otherwise, it returns true.

Usage

Checks whether the

weeklyCreditSum

of this instance is set.

false if the weeklyCreditSum is null. Otherwise, it returns true.

isSetWeeklyDebitCount() method

Syntax

```
boolean isSetWeeklyDebitCount ()
```

Returns

false if the weeklyDebitCount is null. Otherwise, it returns true.

Usage

Checks whether the

weeklyDebitCount

of this instance is set.

false if the weeklyDebitCount is null. Otherwise, it returns true.

isSetWeeklyDebitSum() method

Syntax

```
boolean isSetWeeklyDebitSum ()
```

Returns

false if the weeklyDebitSum is null. Otherwise, it returns true.

Usage

Checks whether the

weeklyDebitSum

of this instance is set.

false if the weeklyDebitSum is null. Otherwise, it returns true.

setAbsoluteCreditCount(long) method

Syntax

```
void setAbsoluteCreditCount ( long absoluteCreditCount )
```

Parameters

- **absoluteCreditCount** – the absoluteCreditCount to set

setAbsoluteCreditSum(long) method

Syntax

```
void setAbsoluteCreditSum ( long absoluteCreditSum )
```

Parameters

- **absoluteCreditSum** – the absoluteCreditSum to set

setAbsoluteDebitCount(long) method

Syntax

```
void setAbsoluteDebitCount ( long absoluteDebitCount )
```

Parameters

- **absoluteDebitCount** – the absoluteDebitCount to set

setAbsoluteDebitSum(long) method

Syntax

void setAbsoluteDebitSum (long *absoluteDebitSum*)

Parameters

- **absoluteDebitSum** – the absoluteDebitSum to set

setCustomer(Customer) method

Syntax

void setCustomer (Customer *customer*)

setDailyCreditCount(long) method

Syntax

void setDailyCreditCount (long *dailyCreditCount*)

Parameters

- **dailyCreditCount** – the dailyCreditCount to set

setDailyCreditSum(long) method

Syntax

void setDailyCreditSum (long *dailyCreditSum*)

Parameters

- **dailyCreditSum** – the dailyCreditSum to set

setDailyDebitCount(long) method

Syntax

void setDailyDebitCount (long *dailyDebitCount*)

Parameters

- **dailyDebitCount** – the dailyDebitCount to set

setDailyDebitSum(long) method

Syntax

void setDailyDebitSum (long *dailyDebitSum*)

Parameters

- **dailyDebitSum** – the dailyDebitSum to set

setLastLimitUpdate(Date) method

Syntax

void setLastLimitUpdate (Date *lastLimitUpdate*)

Parameters

- **lastLimitUpdate** – the lastLimitUpdate to set

Usage

Sets the lastLimitUpdate value of this instance.

setLimitClass(LimitClass) method

Syntax

void setLimitClass (LimitClass *limitClass*)

Parameters

- **limitClass** – the limitClass to set

setMonthlyCreditCount(long) method

Syntax

void setMonthlyCreditCount (long *monthlyCreditCount*)

Parameters

- **monthlyCreditCount** – the monthlyCreditCount to set

setMonthlyCreditSum(long) method

Syntax

`void setMonthlyCreditSum ( long monthlyCreditSum )`

Parameters

- **monthlyCreditSum** – the monthlyCreditSum to set

setMonthlyDebitCount(long) method

Syntax

`void setMonthlyDebitCount ( long monthlyDebitCount )`

Parameters

- **monthlyDebitCount** – the monthlyDebitCount to set

setMonthlyDebitSum(long) method

Syntax

`void setMonthlyDebitSum ( long monthlyDebitSum )`

Parameters

- **monthlyDebitSum** – the monthlyDebitSum to set

setPaymentInstrument(PaymentInstrument) method

Syntax

`void setPaymentInstrument ( PaymentInstrument paymentInstrument )`

setSingleCreditAmount(Long) method

Syntax

`void setSingleCreditAmount ( Long singleCreditAmount )`

setSingleDebitAmount(Long) method

Syntax

`void setSingleDebitAmount ( Long singleDebitAmount )`

setWalletEntry(WalletEntry) method

Syntax

void setWalletEntry (WalletEntry *walletEntry*)

setWeeklyCreditCount(long) method

Syntax

void setWeeklyCreditCount (long *weeklyCreditCount*)

Parameters

- **weeklyCreditCount** – the weeklyCreditCount to set

setWeeklyCreditSum(long) method

Syntax

void setWeeklyCreditSum (long *weeklyCreditSum*)

Parameters

- **weeklyCreditSum** – the weeklyCreditSum to set

setWeeklyDebitCount(long) method

Syntax

void setWeeklyDebitCount (long *weeklyDebitCount*)

Parameters

- **weeklyDebitCount** – the weeklyDebitCount to set

setWeeklyDebitSum(long) method

Syntax

void setWeeklyDebitSum (long *weeklyDebitSum*)

Parameters

- **weeklyDebitSum** – the weeklyDebitSum to set

Note class

Syntax

```
public class Note
```

Usage

The `Note` represents a message/notice/piece of information attached to a customer.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Note() constructor

Syntax

```
Note ()
```

Usage

Creates a new instance of

```
Note
```

```
.
```

getCategory() method

Syntax

```
int getCategory ()
```

Returns

the category

Usage

the category

getCustomer() method

Syntax

```
Customer getCustomer ()
```

Returns

the affected Customer

Usage

Returns the affected customer of this note.

the affected Customer

getStatus() method

Syntax

```
int getStatus ()
```

Returns

the status

Usage

the status

getSubject() method

Syntax

```
String getSubject ()
```

Returns

the subject String

Usage

Returns the subject of this note.

the subject String

getText() method

Syntax

```
String getText ()
```

Returns

the message String

Usage

Returns the message/notice/data of this note.

the message `String`

isSetCategory() method

Syntax

`boolean isSetCategory ()`

Returns

false if the category is null. Otherwise, it returns true.

Usage

Checks whether the

category

of this instance is set.

false if the category is null. Otherwise, it returns true.

isSetCustomer() method

Syntax

`boolean isSetCustomer ()`

Returns

false if the customer is null. Otherwise, it returns true.

Usage

Checks whether the

customer

of this instance is set.

false if the customer is null. Otherwise, it returns true.

isSetStatus() method

Syntax

`boolean isSetStatus ()`

Returns

false if the status is null. Otherwise, it returns true.

Usage

Checks whether the

status

of this instance is set.

false if the status is null. Otherwise, it returns true.

isSetSubject() method

Syntax

```
boolean isSetSubject ()
```

Returns

false if the subject is null. Otherwise, it returns true.

Usage

Checks whether the

subject

of this instance is set.

false if the subject is null. Otherwise, it returns true.

isSetText() method

Syntax

```
boolean isSetText ()
```

Returns

false if the text is null. Otherwise, it returns true.

Usage

Checks whether the

text

of this instance is set.

false if the text is null. Otherwise, it returns true.

setCategory(int) method

Syntax

void setCategory (int *category*)

Parameters

- **category** – the category to set

setCustomer(Customer) method

Syntax

void setCustomer (Customer *customer*)

Parameters

- **customer** – the customer value to set

Usage

Sets the

customer

of this instance.

setStatus(int) method

Syntax

void setStatus (int *status*)

Parameters

- **status** – the status to set

setSubject(String) method

Syntax

void setSubject (String *subject*)

Parameters

- **subject** – the subject value to set

Usage

Sets the

subject

of this instance.

setText(String) method

Syntax

```
void setText ( String text )
```

Parameters

- **text** – the text value to set

Usage

Sets the

text

of this instance.

Otp class**Syntax**

```
public class Otp
```

Usage

The Otp represents one-time-passwords that are usually used for the registration process.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Otp() constructor

Syntax

```
Otp ()
```

Usage

Creates a new instance of

AuditLog

.

getExpiration() method

Syntax

Date getExpiration ()

getMailJobId() method

Syntax

Long getMailJobId ()

getOtp() method

Syntax

String getOtp ()

getOtpType() method

Syntax

OtpType getOtpType ()

getReceiver() method

Syntax

String getReceiver ()

getReferenceID() method

Syntax

String getReferenceID ()

getSentCount() method

Syntax

Integer getSentCount ()

getWrongEntryCount() method

Syntax

int getWrongEntryCount ()

isSetExpiration() method

Syntax

```
boolean isSetExpiration ()
```

isSetMailJobId() method

Syntax

```
boolean isSetMailJobId ()
```

isSetOtp() method

Syntax

```
boolean isSetOtp ()
```

isSetOtpType() method

Syntax

```
boolean isSetOtpType ()
```

isSetReceiver() method

Syntax

```
boolean isSetReceiver ()
```

isSetReferenceID() method

Syntax

```
boolean isSetReferenceID ()
```

isSetSentCount() method

Syntax

```
boolean isSetSentCount ()
```

isSetWrongEntryCount() method

Syntax

```
boolean isSetWrongEntryCount ()
```

setExpiration(Date) method

Syntax

void setExpiration (Date *expiration*)

setMailJobId(Long) method

Syntax

void setMailJobId (Long *mailJobId*)

setOtp(String) method

Syntax

void setOtp (String *otp*)

setOtpType(OtpType) method

Syntax

void setOtpType (OtpType *otpType*)

setReceiver(String) method

Syntax

void setReceiver (String *receiver*)

setReferenceID(String) method

Syntax

void setReferenceID (String *referenceID*)

setSentCount(Integer) method

Syntax

void setSentCount (Integer *sentCount*)

setWrongEntryCount(int) method

Syntax

void setWrongEntryCount (int *wrongEntryCount*)

WrkCustomer class**Syntax**

```
public class WrkCustomer
```

Usage

The `WrkCustomer` represents a maker/checker configuration for customer types. Allowing to define `UmgrPrivileges` for each action all tied to a specific customer type.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

WrkCustomer() constructor**Syntax**

```
WrkCustomer ()
```

WrkCustomer(Integer) constructor**Syntax**

```
WrkCustomer ( Integer id )
```

Parameters

- `id` –

Usage

Please use the

```
newInstance
```

method of the corresponding DAO.

getCheckerPrivilege() method**Syntax**

```
String getCheckerPrivilege ()
```

getCheckerPrivilegeDeact() method**Syntax**

```
String getCheckerPrivilegeDeact ()
```

getDaysObfuscationDelay() method

Syntax

int getDaysObfuscationDelay ()

getExecutePrivilege() method

Syntax

String getExecutePrivilege ()

getExecutePrivilegeDeact() method

Syntax

String getExecutePrivilegeDeact ()

getMakerPrivilege() method

Syntax

String getMakerPrivilege ()

getMakerPrivilegeDeact() method

Syntax

String getMakerPrivilegeDeact ()

isSetCheckerPrivilege() method

Syntax

boolean isSetCheckerPrivilege ()

isSetCheckerPrivilegeDeact() method

Syntax

boolean isSetCheckerPrivilegeDeact ()

isSetdaysObfuscationDelay() method

Syntax

boolean isSetdaysObfuscationDelay ()

isSetExecutePrivilege() method

Syntax

`boolean isSetExecutePrivilege ()`

isSetExecutePrivilegeDeact() method

Syntax

`boolean isSetExecutePrivilegeDeact ()`

isSetMakerPrivilege() method

Syntax

`boolean isSetMakerPrivilege ()`

isSetMakerPrivilegeDeact() method

Syntax

`boolean isSetMakerPrivilegeDeact ()`

setCheckerPrivilege(String) method

Syntax

`void setCheckerPrivilege ( String checkerPrivilege )`

setCheckerPrivilegeDeact(String) method

Syntax

`void setCheckerPrivilegeDeact ( String checkerPrivilegeDeact )`

setDaysObfuscationDelay(int) method

Syntax

`void setDaysObfuscationDelay ( int daysObfuscationDelay )`

setExecutePrivilege(String) method

Syntax

`void setExecutePrivilege ( String executePrivilege )`

setExecutePrivilegeDeact(String) method

Syntax

void setExecutePrivilegeDeact (String *executePrivilegeDeact*)

setMakerPrivilege(String) method

Syntax

void setMakerPrivilege (String *makerPrivilege*)

setMakerPrivilegeDeact(String) method

Syntax

void setMakerPrivilegeDeact (String *makerPrivilegeDeact*)

pi package

Members

All public members of the pi package.

- **BalanceAlert** class –
- **BankAccount** class –
- **Card** class –
- **ExternalAccount** class –
- **InternalVoucher** class –
- **Mandate** class –
- **PaymentInstrument** class –
- **StoredValueAccount** class –
- **WalletEntry** class –
- **WrkWalletEntry** class –

BalanceAlert class

Syntax

public class BalanceAlert

Usage

The `BalanceAlert` represents a balance-based alert which is configured for a payment instrument.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

BalanceAlert() constructor

Syntax

```
BalanceAlert ()
```

Usage

Creates a new instance of

```
BalanceAlert
```

.

getEmails() method

Gets the comma separated list of E-Mail addresses used to send alert E-Mails to.

Syntax

```
String getEmails ()
```

Returns

a list of comma separated E-Mail addresses

Usage

a list of comma separated E-Mail addresses

getLocale() method

Syntax

```
Locale getLocale ()
```

Returns

the balance alert's Locale

Usage

Returns the locale of the balance alert.

the balance alert's Locale

getMsisdns() method

Gets the comma separated list of MSISDNs used to send alert SMS to.

Syntax

String getMsisdns ()

Returns

a list of comma separated msisdns

Usage

a list of comma separated msisdns

getPaymentInstrument() method

Gets the payment instrument whose balance is observed with the configuration of this BalanceAlert.

Syntax

PaymentInstrument getPaymentInstrument ()

Returns

the payment instrument whose balance is observed

Usage

the payment instrument whose balance is observed

getTemplateName() method

Gets the name of the template which is used to send the messages via E-Mail and SMS.

Syntax

String getTemplateName ()

Returns

the name of the template which is used to send the messages via E-Mail and SMS

Usage

the name of the template which is used to send the messages via E-Mail and SMS

getThreshold() method

Gets the threshold amount determining if this alert configuration has to be applied.

Syntax

```
long getThreshold ()
```

Returns

the threshold amount determining if this alert configuration has to be applied

Usage

the threshold amount determining if this alert configuration has to be applied

isActive() method

Syntax

```
boolean isActive ()
```

Returns

true if this instance is active; false if it is inactive

Usage

Indicates if this balance alert is active.

true if this instance is active; false if it is inactive

isOnlyTransition() method

Indicates whether an alert should only be sent when the balance has been currently fallen below the threshold amount.

Syntax

```
boolean isOnlyTransition ()
```

Returns

the onlyTransition false if the alert is sent every time the balance is below the threshold amount. true if the alert is only sent when the balance has been currently fallen below the threshold amount.

Usage

the onlyTransition `false` if the alert is sent every time the balance is below the threshold amount. `true` if the alert is only sent when the balance has been currently fallen below the threshold amount.

isOwnerEmail() method

Indicates whether an alert E-Mail should be sent to the owner of the payment instrument.

Syntax

```
Boolean isOwnerEmail ()
```

Returns

the ownerEmail `false` if no E-Mail alert should be sent to the payment instrument's owner. `true` if an E-Mail alert should be sent to the payment instrument's owner.

Usage

the ownerEmail `false` if no E-Mail alert should be sent to the payment instrument's owner. `true` if an E-Mail alert should be sent to the payment instrument's owner.

isOwnerSms() method

Indicates whether an alert SMS should be sent to the owner of the payment instrument.

Syntax

```
Boolean isOwnerSms ()
```

Returns

the ownerEmail `false` if no SMS alert should be sent to the payment instrument's owner. `true` if an SMS alert should be sent to the payment instrument's owner.

Usage

the ownerEmail `false` if no SMS alert should be sent to the payment instrument's owner. `true` if an SMS alert should be sent to the payment instrument's owner.

isSetActive() method

Syntax

```
boolean isSetActive ()
```

Returns

`false` if the dbActive is null. Otherwise, it returns `true`.

Usage

Checks whether the

dbActive

of this instance is set.

false if the dbActive is null. Otherwise, it returns true.

isSetEmails() method

Syntax

```
boolean isSetEmails ()
```

Returns

false if the emails is null. Otherwise, it returns true.

Usage

Checks whether the

emails

of this instance is set.

false if the emails is null. Otherwise, it returns true.

isSetLocale() method

Syntax

```
boolean isSetLocale ()
```

Returns

false if the dbLanguage and dbCountry is null. Otherwise, it returns true.

Usage

Checks whether the

dbLanguage

or

dbCountry

of this instance is set.

false if the dbLanguage and dbCountry is null. Otherwise, it returns true.

isSetMsisdns() method

Syntax

```
boolean isSetMsisdns ()
```

Returns

false if the msisdns is null. Otherwise, it returns true.

Usage

Checks whether the

msisdns

of this instance is set.

false if the msisdns is null. Otherwise, it returns true.

isSetOnlyTransition() method

Syntax

```
boolean isSetOnlyTransition ()
```

Returns

false if the onlyTransition is null. Otherwise, it returns true.

Usage

Checks whether the

onlyTransition

of this instance is set.

false if the onlyTransition is null. Otherwise, it returns true.

isSetOwnerEmail() method

Syntax

```
boolean isSetOwnerEmail ()
```

Returns

false if the ownerEmail is null. Otherwise, it returns true.

Usage

Checks whether the

ownerEmail

of this instance is set.

false if the ownerEmail is null. Otherwise, it returns true.

isSetOwnerSms() method

Syntax

```
boolean isSetOwnerSms ()
```

Returns

false if the ownerSms is null. Otherwise, it returns true.

Usage

Checks whether the

ownerSms

of this instance is set.

false if the ownerSms is null. Otherwise, it returns true.

isSetPaymentInstrument() method

Syntax

```
boolean isSetPaymentInstrument ()
```

Returns

false if the paymentInstrument is null. Otherwise, it returns true.

Usage

Checks whether the

paymentInstrument

of this instance is set.

false if the paymentInstrument is null. Otherwise, it returns true.

isSetTemplateName() method

Syntax

```
boolean isSetTemplateName ()
```

Returns

false if the `templateName` is null. Otherwise, it returns true.

Usage

Checks whether the

templateName

of this instance is set.

false if the `templateName` is null. Otherwise, it returns true.

isSetThreshold() method

Syntax

```
boolean isSetThreshold ()
```

Returns

false if the `threshold` is null . Otherwise, it returns true.

Usage

Checks whether the

threshold

of this instance is set.

false if the `threshold` is null . Otherwise, it returns true.

setActive(boolean) method

Syntax

```
void setActive ( boolean active )
```

Parameters

- **active** – the active value to set

Usage

Sets the

active

of this instance.

setEmails(String) method

Sets the given comma separated list of E-Mail addresses used to send alert E-Mails to.

Syntax

```
void setEmails ( String emails )
```

Parameters

- **emails** – a list of comma separated E-Mail addresses

setLocale(Locale) method

Syntax

```
void setLocale ( Locale locale )
```

Parameters

- **locale** – the `locale` value to set

Usage

Sets the

locale

of this instance.

setMsisdns(String) method

Sets the given comma separated list of MSISDNs used to send alert SMS to.

Syntax

```
void setMsisdns ( String msisdns )
```

Parameters

- **msisdns** – a list of comma separated msisdns

setOnlyTransition(boolean) method

Sets the flag which indicates whether an alert should only be sent when the balance has been currently fallen below the threshold amount.

Syntax

```
void setOnlyTransition ( boolean onlyTransition )
```

Parameters

- **onlyTransition** – `false` if the alert should be sent every time the balance is below the threshold amount. `true` if the alert should only be sent when the balance has been currently fallen below the threshold amount.

setOwnerEmail(Boolean) method

Sets the flag which indicates whether an alert E-Mail should be sent to the owner of the payment instrument.

Syntax

```
void setOwnerEmail ( Boolean ownerEmail )
```

Parameters

- **onlyTransition** – `false` if no E-Mail alert should be sent to the payment instrument's owner. `true` if an E-Mail alert should be sent to the payment instrument's owner.

setOwnerSms(Boolean) method

Sets the flag which indicates whether an alert SMS should be sent to the owner of the payment instrument.

Syntax

```
void setOwnerSms ( Boolean ownerSms )
```

Parameters

- **onlyTransition** – `false` if no SMS alert should be sent to the payment instrument's owner. `true` if an SMS alert should be sent to the payment instrument's owner.

setPaymentInstrument(PaymentInstrument) method

Sets the payment instrument whose balance should be observed with the configuration of this BalanceAlert.

Syntax

```
void setPaymentInstrument ( PaymentInstrument paymentInstrument )
```


Parameters

- **paymentInstrument** – the payment instrument which should be observed

setTemplateName(String) method

Sets the name of the template which is used to send the messages via.

Syntax

```
void setTemplateName ( String templateName )
```

Parameters

- **templateName** – the name of the template which is used to send the messages via E-Mail and SMS

setThreshold(long) method

Sets the threshold amount determining if this alert configuration has to be applied.

Syntax

```
void setThreshold ( long threshold )
```

Parameters

- **threshold** – the threshold amount which shall be set to this balance alert

BankAccount class*Syntax*

```
public class BankAccount
```

Usage

The `BankAccount` is a payment instrument extension that contains the ID of the account.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

*BankAccount() constructor***Syntax**

```
BankAccount ()
```

Usage

Creates a new instance of

BankAccount

.

BankAccount(long) constructor

Syntax

BankAccount (long *pid*)

Parameters

- **pid** – the PK value

Usage

Creates a new instance of

BankAccount

with a preset primary key.

getAccountNumber() method

Syntax

String getAccountNumber ()

Returns

the account number of this bank account.

Usage

Returns the account number of this bank account.

the account number of this bank account.

getBankCity() method

Syntax

String getBankCity ()

Returns

the city of the bank.

Usage

Returns the city of the bank.
the city of the bank.

getBankCountryId() method

Syntax

String getBankCountryId ()

Returns

the country (id) of the bank.

Usage

Returns the country (id) of the bank.
the country (id) of the bank.

getBankName() method

Syntax

String getBankName ()

Returns

the name of the bank.

Usage

Returns the name of the bank.
the name of the bank.

getBranchCode() method

Syntax

String getBranchCode ()

Returns

the branch code of the bank.

Usage

Returns the branch code of the bank.
the branch code of the bank.

getDisplayNumber() method

Syntax

String getDisplayNumber ()

Returns

the account number of this bank account.

Usage

Returns the masked display account number of this bank account.

the account number of this bank account.

getInstitutionCode() method

Syntax

String getInstitutionCode ()

Returns

the institution code of the bank.

Usage

Returns the institution code of the bank.

the institution code of the bank.

getOwnerName() method

Syntax

String getOwnerName ()

Returns

the name of the account owner.

Usage

Returns the name of the account owner.

the name of the account owner.

getSepaCreditorId() method

Syntax

String getSepaCreditorId ()

Returns

the sepaCreditorId

Usage

the sepaCreditorId

isSetAccountNumber() method

Syntax

boolean isSetAccountNumber ()

Returns

false if the accountNo is null . Otherwise, it returns true.

Usage

Checks whether the

accountNo

of this instance is set.

false if the accountNo is null . Otherwise, it returns true.

isSetBankCity() method

Syntax

boolean isSetBankCity ()

Returns

false if the bankCity is null. Otherwise, it returns true.

Usage

Checks whether the

bankCity

of this instance is set.

false if the bankCity is null. Otherwise, it returns true.

isSetBankCountryId() method

Syntax

boolean isSetBankCountryId ()

Returns

false if the bankCountryId is null. Otherwise, it returns true.

Usage

Checks whether the

bankCountryId

of this instance is set.

false if the bankCountryId is null. Otherwise, it returns true.

isSetBankName() method

Syntax

boolean isSetBankName ()

Returns

false if the bankName is null. Otherwise, it returns true.

Usage

Checks whether the

bankName

of this instance is set.

false if the bankName is null. Otherwise, it returns true.

isSetBranchCode() method

Syntax

boolean isSetBranchCode ()

Returns

false if the branchCode is null. Otherwise, it returns true.

Usage

Checks whether the

branchCode

of this instance is set.

false if the branchCode is null. Otherwise, it returns true.

isSetDisplayNumber() method

Syntax

```
boolean isSetDisplayNumber ()
```

Returns

false if the displayNumber is null . Otherwise, it returns true.

Usage

Checks whether the

displayNumber

of this instance is set.

false if the displayNumber is null . Otherwise, it returns true.

isSetInstitutionCode() method

Syntax

```
boolean isSetInstitutionCode ()
```

Returns

false if the institutionCode is null. Otherwise, it returns true.

Usage

Checks whether the

institutionCode

of this instance is set.

false if the institutionCode is null. Otherwise, it returns true.

isSetOwnerName() method

Syntax

```
boolean isSetOwnerName ()
```

Returns

false if the ownerName is null . Otherwise, it returns true.

Usage

Checks whether the

ownerName

of this instance is set.

false if the ownerName is null . Otherwise, it returns true.

isSetSepaCreditorId() method

Syntax

```
boolean isSetSepaCreditorId ()
```

setAccountNumber(String) method

Syntax

```
void setAccountNumber ( String accountNumber )
```

Parameters

- **accountNo** – The account number of this bank account.

Usage

Sets the account number of this bank account.

setBankCity(String) method

Syntax

```
void setBankCity ( String bankCity )
```

Parameters

- **bankCity** – The city of the bank.

Usage

Sets the city of the bank.

setBankCountryId(String) method

Syntax

```
void setBankCountryId ( String bankCountryId )
```

Parameters

- **bankCountryId** – The country (id) of the bank.

Usage

Sets the country (id) of the bank.

setBankName(String) method

Syntax

```
void setBankName ( String bankName )
```

Parameters

- **bankName** – The name of the bank.

Usage

Sets the name of the bank.

setBranchCode(String) method

Syntax

```
void setBranchCode ( String branchCode )
```

Parameters

- **branchCode** – The branch code of the bank.

Usage

Sets the branch code of the bank.

setDisplayNumber(String) method

Syntax

void setDisplayNumber (String *displayNumber*)

Parameters

- **accountNo** – The account number of this bank account.

Usage

Sets the masked display number of this bank account.

setInstitutionCode(String) method

Syntax

void setInstitutionCode (String *institutionCode*)

Parameters

- **institutionCode** – The institution code of the bank.

Usage

Sets the institution code of the bank.

setOwnerName(String) method

Syntax

void setOwnerName (String *ownerName*)

Parameters

- **ownerName** – The name of the account owner.

Usage

Sets the name of the account owner.

setSepaCreditorId(String) method

Syntax

void setSepaCreditorId (String *sepaCreditorId*)

Parameters

- **sepaCreditorId** – the sepaCreditorId to set

Card class**Syntax**

```
public class Card
```

Usage

The `CreditCard` is a payment instrument extension modelling a credit card.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CardType class**Syntax**

```
public class CardType
```

Usage

The `CardType` is a lookup for the card's type (e.g. "VISA").

© 2000-2007 by paybox solutions AG

Revision:1.7Name:not supported by cvs2svn

CardType() constructor**Syntax**

```
CardType ()
```

Usage

Creates a new instance of

```
CardType
```

```
.
```

CardType(int) constructor**Syntax**

```
CardType ( int id )
```

Parameters

- **id** – the PK value

Usage

Creates a new instance of

`CardType`

with a preset primary key.

getPiType() method

Syntax

```
PaymentInstrument.Type getPiType ()
```

Returns

the associated `PaymentInstrument.Type`

Usage

Returns the paymentinstrument type this card type is categorized as.

the associated `PaymentInstrument.Type`

isSetPiType() method

Syntax

```
boolean isSetPiType ()
```

Returns

false if the `piType` is null. Otherwise, it returns true.

Usage

Checks whether the

piType

of this instance is set.

false if the `piType` is null. Otherwise, it returns true.

setPiType(PaymentInstrument.Type) method

Syntax

```
void setPiType ( PaymentInstrument.Type piType )
```

Parameters

- **piType** – the piType value to set

Usage

Sets the

piType

of this instance.

Card() constructor

Syntax

Card ()

Usage

Creates a new instance of

Card

.

Card(long) constructor

Syntax

Card (long *piId*)

Parameters

- **piId** – the PK value

Usage

Creates a new instance of

Card

with a preset primary key.

getCardHolder() method

Syntax

String getCardHolder ()

Returns

the card holder's name

Usage

Returns the name of the holder of this card.

the card holder's name

getCardNumber() method

Syntax

String getCardNumber ()

Returns

the card's number

Usage

Returns the number of this card.

the card's number

getCardType() method

Syntax

CardType getCardType ()

Returns

the CardType

Usage

Returns the

card type

of this instance.

the CardType

getDisplayNumber() method

Syntax

String getDisplayNumber ()

Returns

the card's display number. May be `null`.

Usage

Returns the display number of this card. May be

`null`

.

the card's display number. May be `null`.

getIssueNumber() method

Syntax

Integer getIssueNumber ()

Returns

the card's issue number; may be `null`

Usage

Returns the issue number of this card.

the card's issue number; may be `null`

getSecurityNumber() method

Syntax

String getSecurityNumber ()

Returns

the card's security number

Usage

Returns the security number of this card.

the card's security number

getValidTo() method

Syntax

Date getValidTo ()

Returns

the card's *valid thru* Date

Usage

Returns the date determining how long this card is valid;

note

that only the month is relevant: 2006/11/01 is in this context the same as 2006/11/05 or any other day in November 2006.

the card's *valid thru* Date

isSetCardHolder() method

Syntax

```
boolean isSetCardHolder ()
```

Returns

false if the `cardHolder` is null. Otherwise, it returns true.

Usage

Checks whether the

cardHolder

of this instance is set.

false if the `cardHolder` is null. Otherwise, it returns true.

isSetCardNumber() method

Syntax

```
boolean isSetCardNumber ()
```

Returns

false if the `cardNumber` is null. Otherwise, it returns true.

Usage

Checks whether the

cardNumber

of this instance is set.

false if the `cardNumber` is null. Otherwise, it returns true.

isSetCardType() method

Syntax

`boolean isSetCardType ()`

Returns

false if the `cardType` is null. Otherwise, it returns true.

Usage

Checks whether the

cardType

of this instance is set.

false if the `cardType` is null. Otherwise, it returns true.

isSetDisplayNumber() method

Syntax

`boolean isSetDisplayNumber ()`

Returns

false if the `displayNumber` is null. Otherwise, it returns true.

Usage

Checks whether the

displayNumber

of this instance is set.

false if the `displayNumber` is null. Otherwise, it returns true.

isSetIssueNumber() method

Syntax

`boolean isSetIssueNumber ()`

Returns

false if the `issueNumber` is null. Otherwise, it returns true.

Usage

Checks whether the

issueNumber

of this instance is set.

false if the `issueNumber` is null. Otherwise, it returns true.

isSetSecurityNumber() method

Syntax

```
boolean isSetSecurityNumber ()
```

Returns

false if the `securityNumber` is null. Otherwise, it returns true.

Usage

Checks whether the

securityNumber

of this instance is set.

false if the `securityNumber` is null. Otherwise, it returns true.

isSetValidTo() method

Syntax

```
boolean isSetValidTo ()
```

Returns

false if the `dbValidTo` is null . Otherwise, it returns true.

Usage

Checks whether the

dbValidTo

of this instance is set.

false if the `dbValidTo` is null . Otherwise, it returns true.

*isValid(Date) method***Syntax**

```
boolean isValid ( Date date )
```

Parameters

- **date** – the Date to check the validity for

Returns

true if this card is valid at date; false otherwise

Usage

Computes if this credit card is still valid at the specified date.

true if this card is valid at date; false otherwise

*setCardHolder(String) method***Syntax**

```
void setCardHolder ( String cardHolder )
```

Parameters

- **cardHolder** – the cardHolder value to set

Usage

Sets the

cardHolder

of this instance.

*setCardNumber(String) method***Syntax**

```
void setCardNumber ( String cardNumber )
```

Parameters

- **cardNumber** – the cardNumber value to set

Usage

Sets the

cardNumber

of this instance.

setCardType(CardType) method

Syntax

```
void setCardType ( CardType cardType )
```

Parameters

- **cardType** – the *cardType* value to set

Usage

Sets the

cardType

of this instance.

setDisplayNumber(String) method

Syntax

```
void setDisplayNumber ( String displayNumber )
```

Parameters

- **displayNumber** – the *displayNumber* value to set.

Usage

Sets the

displayNumber

of this instance.

setIssueNumber(Integer) method

Syntax

```
void setIssueNumber ( Integer issueNumber )
```

Parameters

- **issueNumber** – the `issueNumber` value to set

Usage

Sets the

issueNumber

of this instance.

setSecurityNumber(String) method

Syntax

```
void setSecurityNumber ( String securityNumber )
```

Parameters

- **securityNumber** – the `securityNumber` value to set

Usage

Sets the

securityNumber

of this instance.

setValidTo(Date) method

Syntax

```
void setValidTo ( Date validTo )
```

Parameters

- **validTo** – the `validTo` value to set

Usage

Sets the

validTo

of this instance.

ExternalAccount class

Syntax

```
public class ExternalAccount
```

Usage

The `ExternalAccount` is a payment instrument extension that contains four possible ID's for the account.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

ExternalAccount() constructor

Syntax

```
ExternalAccount ()
```

Usage

Creates a new instance of

```
ExternalAccount
```

.

ExternalAccount(long) constructor

Syntax

```
ExternalAccount ( long piId )
```

Parameters

- **piId** – the PK value

Usage

Creates a new instance of

```
ExternalAccount
```

with a preset primary key.

getId1() method

Syntax

String getId1 ()

Returns

the ID1

Usage

Returns the ID1 of the External Account.

the ID1

getId2() method

Syntax

String getId2 ()

Returns

the ID2

Usage

Returns the ID2 of the External Account.

the ID2

getId3() method

Syntax

String getId3 ()

Returns

the ID3

Usage

Returns the ID3 of the External Account.

the ID3

getId4() method

Syntax

String getId4 ()

Returns

the ID4

Usage

Returns the ID4 of the External Account.

the ID4

getId5() method

Syntax

String getId5 ()

Returns

the ID5

Usage

Returns the ID5 of the External Account.

the ID5

getId6() method

Syntax

String getId6 ()

Returns

the ID6

Usage

Returns the ID6 of the External Account.

the ID6

getId7() method

Syntax

String getId7 ()

Returns

the ID7

Usage

Returns the ID7 of the External Account.

the ID7

getId8() method

Syntax

String getId8 ()

Returns

the ID8

Usage

Returns the ID8 of the External Account.

the ID8

isSetId1() method

Syntax

boolean isSetId1 ()

Returns

false if the id1 is null. Otherwise, it returns true.

Usage

Checks whether the

id1

of this instance is set.

false if the id1 is null. Otherwise, it returns true.

isSetId2() method

Syntax

boolean isSetId2 ()

Returns

false if the id2 is null. Otherwise, it returns true.

Usage

Checks whether the

id2

of this instance is set.

false if the id2 is null. Otherwise, it returns true.

isSetId3() method

Syntax

boolean isSetId3 ()

Returns

false if the id3 is null. Otherwise, it returns true.

Usage

Checks whether the

id3

of this instance is set.

false if the id3 is null. Otherwise, it returns true.

isSetId4() method

Syntax

boolean isSetId4 ()

Returns

false if the id4 is null. Otherwise, it returns true.

Usage

Checks whether the

id4

of this instance is set.

false if the *id4* is null. Otherwise, it returns true.

isSetId5() method

Syntax

```
boolean isSetId5 ()
```

Returns

false if the *id5* is null. Otherwise, it returns true.

Usage

Checks whether the

id5

of this instance is set.

false if the *id5* is null. Otherwise, it returns true.

isSetId6() method

Syntax

```
boolean isSetId6 ()
```

Returns

false if the *id6* is null. Otherwise, it returns true.

Usage

Checks whether the

id6

of this instance is set.

false if the *id6* is null. Otherwise, it returns true.

isSetId7() method

Syntax

boolean isSetId7 ()

Returns

false if the id7 is null. Otherwise, it returns true.

Usage

Checks whether the

id7

of this instance is set.

false if the id7 is null. Otherwise, it returns true.

isSetId8() method

Syntax

boolean isSetId8 ()

Returns

false if the id8 is null. Otherwise, it returns true.

Usage

Checks whether the

id8

of this instance is set.

false if the id8 is null. Otherwise, it returns true.

setId1(String) method

Syntax

void setId1 (String *id1*)

Parameters

- **id1** – the ID1 value to set

Usage

Sets the

ID1

of this instance.

setId2(String) method

Syntax

```
void setId2 ( String id2 )
```

Parameters

- **id2** – the ID2 value to set

Usage

Sets the

ID2

of this instance.

setId3(String) method

Syntax

```
void setId3 ( String id3 )
```

Parameters

- **id3** – the ID3 value to set

Usage

Sets the

ID3

of this instance.

setId4(String) method

Syntax

```
void setId4 ( String id4 )
```

Parameters

- **id4** – the ID4 value to set

Usage

Sets the

ID4

of this instance.

setId5(String) method

Syntax

```
void setId5 ( String id5 )
```

Parameters

- **id5** – the ID5 value to set

Usage

Sets the

ID5

of this instance.

setId6(String) method

Syntax

```
void setId6 ( String id6 )
```

Parameters

- **id6** – the ID6 value to set

Usage

Sets the

ID6

of this instance.

setId7(String) method

Syntax

```
void setId7 ( String id7 )
```

Parameters

- **id4** – the ID7 value to set

Usage

Sets the

ID7

of this instance.

setId8(String) method

Syntax

```
void setId8 ( String id8 )
```

Parameters

- **id8** – the ID8 value to set

Usage

Sets the

ID8

of this instance.

InternalVoucher class

Syntax

```
public class InternalVoucher
```

Usage

The `InternalVoucher` represents a voucher of the internal *Mobiliser Voucher Management System*.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

InternalVoucher() constructor

Syntax

`InternalVoucher ()`

Usage

Creates a new instance of

`InternalVoucher`

.

InternalVoucher(long) constructor

Syntax

`InternalVoucher ( long piId )`

Parameters

- **piId** – the PK value

Usage

Creates a new instance of

`InternalVoucher`

with a preset primary key.

getCurrentValue() method

Syntax

`long getCurrentValue ()`

Returns

the current value

Usage

Returns the current value of this voucher.

the current value

getExpiration() method

Syntax

Date getExpiration ()

Returns

the Date this voucher expires

Usage

Returns the point in time this voucher expires.

the Date this voucher expires

getInitialValue() method

Syntax

long getInitialValue ()

Returns

the initial value

Usage

Returns the initial value of this voucher.

the initial value

getPickupCode() method

Syntax

String getPickupCode ()

Returns

the name of the bank.

Usage

Returns the pickup code

the name of the bank.

getTxnCreatedBy() method

Syntax

SubTransaction getTxnCreatedBy ()

Returns

the transactionID

Usage

Returns the transactionID while the transaction is started.

the transactionID

getTxnPaidOutBy() method

Syntax

SubTransaction getTxnPaidOutBy ()

Returns

the transactionID

Usage

Returns the transactionID while the transaction is paid out.

the transactionID

isActive() method

Syntax

boolean isActive ()

Returns

true if this instance is active; false if it is inactive

Usage

Indicates if this voucher is active.

true if this instance is active; false if it is inactive

*isSetActive() method***Syntax**

```
boolean isSetActive ()
```

Returns

false if the dbActive is null. Otherwise, it returns true.

Usage

Checks whether the

dbActive

of this instance is set.

false if the dbActive is null. Otherwise, it returns true.

*isSetCurrentValue() method***Syntax**

```
boolean isSetCurrentValue ()
```

Returns

false if the currentValue is null. Otherwise, it returns true.

Usage

Checks whether the

currentValue

of this instance is set.

false if the currentValue is null. Otherwise, it returns true.

*isSetExpiration() method***Syntax**

```
boolean isSetExpiration ()
```

Returns

false if the expiration is null. Otherwise, it returns true.

Usage

Checks whether the

expiration

of this instance is set.

false if the expiration is null. Otherwise, it returns true.

isSetInitialValue() method

Syntax

```
boolean isSetInitialValue ()
```

Returns

false if the initialValue is null. Otherwise, it returns true.

Usage

Checks whether the

initialValue

of this instance is set.

false if the initialValue is null. Otherwise, it returns true.

isSetPickupCode() method

Syntax

```
boolean isSetPickupCode ()
```

Returns

false if the pickupCode is null. Otherwise, it returns true.

Usage

Checks whether the

pickupCode

of this instance is set.

false if the pickupCode is null. Otherwise, it returns true.

*isSetTxnCreatedBy() method***Syntax**

```
boolean isSetTxnCreatedBy ()
```

Returns

false if the `txnCreatedBy` is null. Otherwise, it returns true.

Usage

Checks whether the

txnCreatedBy

of this instance is set.

false if the `txnCreatedBy` is null. Otherwise, it returns true.

*isSetTxnPaidOutBy() method***Syntax**

```
boolean isSetTxnPaidOutBy ()
```

Returns

false if the `txnPaidOutBy` is null. Otherwise, it returns true.

Usage

Checks whether the

txnPayerOutBy

of this instance is set.

false if the `txnPaidOutBy` is null. Otherwise, it returns true.

*setActive(boolean) method***Syntax**

```
void setActive ( boolean active )
```

Parameters

- **active** – the active value to set

Usage

Sets the

active

of this instance.

setCurrentValue(long) method

Syntax

```
void setCurrentValue ( long currentValue )
```

Parameters

- **currentValue** – the `currentValue` value to set

Usage

Sets the

currentValue

of this instance.

setExpiration(Date) method

Syntax

```
void setExpiration ( Date expiration )
```

Parameters

- **expiration** – the `expiration` value to set

Usage

Sets the

expiration

of this instance.

setInitialValue(long) method

Syntax

```
void setInitialValue ( long initialValue )
```

Parameters

- **initialValue** – the `initialValue` value to set

Usage

Sets the

initialValue

of this instance.

setPickupCode(String) method

Syntax

```
void setPickupCode ( String pickupCode )
```

Parameters

- **pickupCode** – The value of the pickup code.

Usage

Sets the pickup code

setTxnCreatedBy(SubTransaction) method

Syntax

```
void setTxnCreatedBy ( SubTransaction txnCreatedBy )
```

Parameters

- **txnCreatedBy** – the `txnCreatedBy` value to set

Usage

Sets the

txnCreatedBy

of this instance.

setTxnPaidOutBy(SubTransaction) method

Syntax

```
void setTxnPaidOutBy ( SubTransaction txnPaidOutBy )
```

Parameters

- **txnPayedOutBy** – the txnPayedOutBy value to set

Usage

Sets the

txnPayedOutBy

of this instance.

Mandate class

Syntax

```
public class Mandate
```

Usage

The `Mandate` represents a customers agreement to perform monetary transactions and the settlement of his account(s)

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Mandate() constructor

Syntax

```
Mandate ()
```

Usage

Creates a new instance of

`WalletEntry`

.

getCancellationReason() method

Syntax

```
int getCancellationReason ()
```

Returns

the cancellationReason

Usage

the cancellationReason

getCreditorId() method

Syntax

String getCreditorId ()

Returns

the creditorId

Usage

the creditorId

getExpirationDate() method

Syntax

Date getExpirationDate ()

Returns

the expirationDate

Usage

the expirationDate

getIssueDate() method

Syntax

Date getIssueDate ()

Returns

the issueDate

Usage

the issueDate

getLastTransactionDate() method

Syntax

Date getLastTransactionDate ()

Returns

the lastTransactionDate

Usage

the lastTransactionDate

getMandate() method

Syntax

byte[] getMandate ()

Returns

the mandate

Usage

the mandate

getPaymentInstrument() method

Syntax

PaymentInstrument getPaymentInstrument ()

Returns

the paymentInstrument

Usage

the paymentInstrument

getType() method

Syntax

int getType ()

Returns

the type

Usage

the type

getValidFromDate() method

Syntax

Date getValidFromDate ()

Returns

the validFromDate

Usage

the validFromDate

isActive() method

Syntax

boolean isActive ()

Returns

the dbActive

Usage

the dbActive

isSetActive() method

Syntax

boolean isSetActive ()

isSetCancellationReason() method

Syntax

boolean isSetCancellationReason ()

isSetCreditorId() method

Syntax

boolean isSetCreditorId ()

isSetExpirationDate() method

Syntax

boolean isSetExpirationDate ()

isSetIssueDate() method

Syntax

boolean isSetIssueDate ()

isSetLastTransactionDate() method

Syntax

boolean isSetLastTransactionDate ()

isSetMandate() method

Syntax

boolean isSetMandate ()

isSetPaymentInstrument() method

Syntax

boolean isSetPaymentInstrument ()

isSetType() method

Syntax

boolean isSetType ()

isSetValidFromDate() method

Syntax

boolean isSetValidFromDate ()

setActive(boolean) method

Syntax

void setActive (boolean *active*)

Parameters

- **dbActive** – the dbActive to set

setCancellationReason(int) method

Syntax

```
void setCancellationReason ( int cancellationReason )
```

Parameters

- **cancellationReason** – the cancellationReason to set

setCreditorId(String) method

Syntax

```
void setCreditorId ( String creditorId )
```

Parameters

- **creditorId** – the creditorId to set

setExpirationDate(Date) method

Syntax

```
void setExpirationDate ( Date expirationDate )
```

Parameters

- **expirationDate** – the expirationDate to set

setIssueDate(Date) method

Syntax

```
void setIssueDate ( Date issueDate )
```

Parameters

- **issueDate** – the issueDate to set

setLastTransactionDate(Date) method

Syntax

```
void setLastTransactionDate ( Date lastTransactionDate )
```

Parameters

- **lastTransactionDate** – the lastTransactionDate to set

setMandate(byte[]) method

Syntax

```
void setMandate ( byte[] mandate )
```

Parameters

- **mandate** – the mandate to set

setPaymentInstrument(PaymentInstrument) method

Syntax

```
void setPaymentInstrument ( PaymentInstrument paymentInstrument )
```

Parameters

- **paymentInstrument** – the paymentInstrument to set

setType(int) method

Syntax

```
void setType ( int type )
```

Parameters

- **type** – the type to set

setValidFromDate(Date) method

Syntax

```
void setValidFromDate ( Date validFromDate )
```

Parameters

- **validFromDate** – the validFromDate to set

PaymentInstrument class

Syntax

```
public class PaymentInstrument
```

Usage

The `PaymentInstrument` represents any payment instrument available to a customer.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Type class**Syntax**

```
public class Type
```

Usage

The `Type` is a lookup for the payment instrument type.

© 2000-2008 by paybox solutions AG

Revision:1.27Name:not supported by cvs2svn

Type() constructor**Syntax**

```
Type ()
```

Usage

Creates a new instance of

```
Type
```

```
.
```

Type(int) constructor**Syntax**

```
Type ( int id )
```

Parameters

- **id** – the PK value

Usage

Creates a new instance of

```
Type
```

com.sybase365.mobiliser.money package

with a preset primary key.

getBookingMode() method

Syntax

```
int getBookingMode ()
```

Returns

the bookingMode

Usage

the bookingMode

getDefaultClearingMode() method

Syntax

```
ClearingMode getDefaultClearingMode ()
```

Returns

the payment instrument's default clearing mode

Usage

Returns the default clearing mode of this payment instrument.

the payment instrument's default clearing mode

getDefaultFeeClearingMode() method

Syntax

```
ClearingMode getDefaultFeeClearingMode ()
```

Returns

the defaultFeeClearingMode

Usage

the defaultFeeClearingMode

getFeeSettlementMode() method

Syntax

```
String getFeeSettlementMode ()
```


Returns

the feeSettlementMode

Usage

the feeSettlementMode

getGeneralLedgerAccount() method

Syntax

PaymentInstrument getGeneralLedgerAccount ()

getIssuer() method

Syntax

Customer getIssuer ()

Returns

the issuing Customer; may be null

Usage

Returns the issuer of this payment instrument type.

the issuing Customer; may be null

getPiClass() method

Syntax

Class getPiClass ()

Returns

the type's PaymentInstrument.Class

Usage

Returns the class of this instance.

the type's PaymentInstrument.Class

getSettlementMode() method

Syntax

String getSettlementMode ()

Returns

the settlementMode

Usage

the settlementMode

getSettlementPaymentInstrument() method

Syntax

```
PaymentInstrument getSettlementPaymentInstrument ()
```

Returns

the settlementPaymentInstrument

Usage

the settlementPaymentInstrument

isCreditAllowed() method

Syntax

```
boolean isCreditAllowed ()
```

Returns

true if this instance can be used for credits; false otherwise

Usage

Indicates if this payment instrument may be used for credits.

true if this instance can be used for credits; false otherwise

isDebitAllowed() method

Syntax

```
boolean isDebitAllowed ()
```

Returns

true if this instance can be used for debits; false otherwise

Usage

Indicates if this payment instrument may be used for debits.

true if this instance can be used for debits; false otherwise

isExternalClearingControl() method

Syntax

boolean isExternalClearingControl ()

isMandateRequired() method

Syntax

boolean isMandateRequired ()

Returns

the dbMandateRequired

Usage

the dbMandateRequired

isSetBookingMode() method

Syntax

boolean isSetBookingMode ()

Returns

false if the dbBookingMode is null. Otherwise, it returns true.

Usage

Checks whether the

dbBookingMode

of this instance is set.

false if the dbBookingMode is null. Otherwise, it returns true.

isSetCreditAllowed() method

Syntax

boolean isSetCreditAllowed ()

Returns

false if the dbCreditAllowed is null. Otherwise, it returns true.

Usage

Checks whether the

dbCreditAllowed

of this instance is set.

false if the dbCreditAllowed is null. Otherwise, it returns true.

isSetDebitAllowed() method

Syntax

```
boolean isSetDebitAllowed ()
```

Returns

false if the dbDebitAllowed is null. Otherwise, it returns true.

Usage

Checks whether the

dbDebitAllowed

of this instance is set.

false if the dbDebitAllowed is null. Otherwise, it returns true.

isSetDefaultClearingMode() method

Syntax

```
boolean isSetDefaultClearingMode ()
```

Returns

false if the defaultClearingMode is null. Otherwise, it returns true.

Usage

Checks whether the

defaultClearingMode

of this instance is set.

false if the defaultClearingMode is null. Otherwise, it returns true.

isSetDefaultFeeClearingMode() method

Syntax

boolean isSetDefaultFeeClearingMode ()

isSetExternalClearingControl() method

Syntax

boolean isSetExternalClearingControl ()

Returns

false if the dbExternalClearingControl is null. Otherwise, it returns true

Usage

Checks whether the

dbExternalClearingControl

of this instance is set.

false if the dbExternalClearingControl is null. Otherwise, it returns true

isSetFeeSettlementMode() method

Syntax

boolean isSetFeeSettlementMode ()

isSetGeneralLedgerAccount() method

Syntax

boolean isSetGeneralLedgerAccount ()

isSetIssuer() method

Syntax

boolean isSetIssuer ()

Returns

false if the issuer is null. Otherwise, it returns true.

Usage

Checks whether the

issuer

of this instance is set.

false if the issuer is null. Otherwise, it returns true.

isSetMandaterequired() method

Syntax

```
boolean isSetMandaterequired ()
```

isSetPiClass() method

Syntax

```
boolean isSetPiClass ()
```

Returns

false if the dbClass is null. Otherwise, it returns true.

Usage

Checks whether the

dbClass

of this instance is set.

false if the dbClass is null. Otherwise, it returns true.

isSetSettlementMode() method

Syntax

```
boolean isSetSettlementMode ()
```

Returns

false if the settlementMode is null. Otherwise, it returns true.

Usage

Checks whether the

settlementMode

of this instance is set.

false if the settlementMode is null. Otherwise, it returns true.

isSetSettlementPaymentInstrument() method

Syntax

```
boolean isSetSettlementPaymentInstrument ()
```

Returns

false if the `settlementPaymentInstrument` is null. Otherwise, it returns true.

Usage

Checks whether the

settlementPaymentInstrument

of this instance is set.

false if the `settlementPaymentInstrument` is null. Otherwise, it returns true.

setBookingMode(int) method

Syntax

```
void setBookingMode ( int bookingMode )
```

Parameters

- **bookingMode** – the `bookingMode` to set

setCreditAllowed(boolean) method

Syntax

```
void setCreditAllowed ( boolean creditAllowed )
```

Parameters

- **creditAllowed** – the `creditAllowed` value to set

Usage

Sets the

creditAllowed

of this instance.

setDebitAllowed(boolean) method

Syntax

void setDebitAllowed (boolean *debitAllowed*)

Parameters

- **debitAllowed** – the debitAllowed value to set

Usage

Sets the

debitAllowed

of this instance.

setDefaultClearingMode(ClearingMode) method

Syntax

void setDefaultClearingMode (ClearingMode *defaultClearingMode*)

Parameters

- **dbDefaultClearingMode** – the default clearingMode value to set

Usage

Sets the

default clearing mode

of this instance.

setDefaultFeeClearingMode(ClearingMode) method

Syntax

void setDefaultFeeClearingMode (ClearingMode *defaultFeeClearingMode*)

Parameters

- **defaultFeeClearingMode** – the defaultFeeClearingMode to set

setExternalClearingControl(boolean) method

Syntax

void setExternalClearingControl (boolean *externalClearingControl*)

setFeeSettlementMode(String) method

Syntax

void setFeeSettlementMode (String *feeSettlementMode*)

Parameters

- **feeSettlementMode** – the feeSettlementMode to set

setGeneralLedgerAccount(PaymentInstrument) method

Syntax

void setGeneralLedgerAccount (PaymentInstrument *generalLedgerAccount*)

setIssuer(Customer) method

Syntax

void setIssuer (Customer *issuer*)

Parameters

- **issuer** – the issuer value to set

Usage

Sets the

issuer

of this instance.

setMandaterequired(boolean) method

Syntax

void setMandaterequired (boolean *mandateRequired*)

Parameters

- **dbMandateRequired** – the dbMandateRequired to set

setPiClass(final Class) method

Syntax

```
void setPiClass ( final Class piClass )
```

Parameters

- **piClass** – the Class to set

Usage

Sets the piClass of this instance.

setSettlementMode(String) method

Syntax

```
void setSettlementMode ( String settlementMode )
```

Parameters

- **settlementMode** – the settlementMode to set

setSettlementPaymentInstrument(PaymentInstrument) method

Syntax

```
void setSettlementPaymentInstrument ( PaymentInstrument  
settlementPaymentInstrument )
```

Parameters

- **settlementPaymentInstrument** – the settlementPaymentInstrument to set

PaymentInstrument() constructor

Syntax

```
PaymentInstrument ( )
```

Usage

Creates a new instance of

```
PaymentInstrument
```

.

getClearingMode() method

Syntax

ClearingMode getClearingMode ()

Returns

the payment instrument's clearing mode

Usage

Returns the clearing mode of this payment instrument.

the payment instrument's clearing mode

getCurrency() method

Syntax

Currency getCurrency ()

Returns

the transaction's Currency

Usage

Returns the currency this transaction is executed with.

the transaction's Currency

getCustomer() method

Syntax

Customer getCustomer ()

Returns

the Customer

Usage

Returns the customer associated to this payment instrument.

the Customer

getFeeClearingMode() method

Syntax

ClearingMode getFeeClearingMode ()

Returns

the feeClearingMode

Usage

the feeClearingMode

getHash() method

Syntax

String getHash ()

getLimitSetId() method

Syntax

Long getLimitSetId ()

Returns

the limitSetId

Usage

the limitSetId

getStatus() method

Syntax

int getStatus ()

Returns

the payment instrument's status

Usage

Returns the status of this payment instrument.

the payment instrument's status

*getType() method***Syntax**`Type getType ()`**Returns**

the payment instrument's Type

Usage

Returns the type of this payment instrument.

the payment instrument's Type

*isActive() method***Syntax**`boolean isActive ()`**Returns**

true if this instance is active; false if it is inactive

Usage

Indicates if this payment instrument is active.

true if this instance is active; false if it is inactive

*isCreditAllowed() method***Syntax**`Boolean isCreditAllowed ()`**Returns**

- null if the superordinate type's values should be considered
- `Boolean.TRUE` if this instance can be used for credits
- `Boolean.FALSE` otherwise

Usage

Indicates if this payment instrument may be used for credits.

- null if the superordinate type's values should be considered
- `Boolean.TRUE` if this instance can be used for credits
- `Boolean.FALSE` otherwise

isDebitAllowed() method

Syntax

`Boolean isDebitAllowed ()`

Returns

`true` if this instance can be used for debits; `false` otherwise

Usage

Indicates if this payment instrument may be used for debits.

`true` if this instance can be used for debits; `false` otherwise

isMultiCurrency() method

Syntax

`boolean isMultiCurrency ()`

Returns

`true` if this instance has multi currencies; `false` if it has not multi currencies

Usage

Indicates if this payment instrument supports multi currency payments.

`true` if this instance has multi currencies; `false` if it has not multi currencies

isSetActive() method

Syntax

`boolean isSetActive ()`

Returns

`false` if the `dbActive` is null. Otherwise, it returns `true`.

Usage

Checks whether the

dbActive

of this instance is set.

false if the dbActive is null. Otherwise, it returns true.

isSetClearingMode() method

Syntax

```
boolean isSetClearingMode ()
```

Returns

false if the clearingMode is null. Otherwise, it returns true.

Usage

Checks whether the

clearingMode

of this instance is set.

false if the clearingMode is null. Otherwise, it returns true.

isSetCreditAllowed() method

Syntax

```
boolean isSetCreditAllowed ()
```

Returns

false if the dbCreditAllowed is null. Otherwise, it returns true.

Usage

Checks whether the

dbCreditAllowed

of this instance is set.

false if the dbCreditAllowed is null. Otherwise, it returns true.

isSetCurrency() method

Syntax

```
boolean isSetCurrency ()
```

Returns

false if the dbCurrency is null. Otherwise, it returns true.

Usage

Checks whether the

dbCurrency

of this instance is set.

false if the dbCurrency is null. Otherwise, it returns true.

isSetCustomer() method

Syntax

```
boolean isSetCustomer ()
```

Returns

false if the customer is null. Otherwise, it returns true.

Usage

Checks whether the

customer

of this instance is set.

false if the customer is null. Otherwise, it returns true.

isSetDebitAllowed() method

Syntax

```
boolean isSetDebitAllowed ()
```

Returns

false if the dbDebitAllowed is null. Otherwise, it returns true.

Usage

Checks whether the

dbDebitAllowed

of this instance is set.

false if the dbDebitAllowed is null. Otherwise, it returns true.

*isSetFeeClearingMode() method***Syntax**

```
boolean isSetFeeClearingMode ()
```

Returns

false if the `feeClearingMode` is null. Otherwise, it returns true.

Usage

Checks whether the

feeClearingMode

of this instance is set.

false if the `feeClearingMode` is null. Otherwise, it returns true.

*isSetHash() method***Syntax**

```
boolean isSetHash ()
```

*isSetLimitSetId() method***Syntax**

```
boolean isSetLimitSetId ()
```

Returns

false if the `limitSetId` is null. Otherwise, it returns true.

Usage

Checks whether the

limitSetId

of this instance is set.

false if the `limitSetId` is null. Otherwise, it returns true.

*isSetMultiCurrency() method***Syntax**

```
boolean isSetMultiCurrency ()
```

Returns

false if the `multiCurrency` is null. Otherwise, it returns true.

Usage

Checks whether the

multiCurrency

of this instance is set.

false if the `multiCurrency` is null. Otherwise, it returns true.

isSetStatus() method

Syntax

```
boolean isSetStatus ()
```

Returns

false if the `status` is null. Otherwise, it returns true.

Usage

Checks whether the

status

of this instance is set.

false if the `status` is null. Otherwise, it returns true.

isSetType() method

Syntax

```
boolean isSetType ()
```

Returns

false if the `type` is null. Otherwise, it returns true.

Usage

Checks whether the

type

of this instance is set.

false if the `type` is null. Otherwise, it returns true.

setActive(boolean) method

Syntax

void setActive (boolean *active*)

Parameters

- **active** – the active value to set

Usage

Sets the

active

of this instance.

setClearingMode(ClearingMode) method

Syntax

void setClearingMode (ClearingMode *clearingMode*)

Parameters

- **dbClearingMode** – the clearingMode value to set

Usage

Sets the

clearing mode

of this instance.

setCreditAllowed(Boolean) method

Syntax

void setCreditAllowed (Boolean *creditAllowed*)

Parameters

- **creditAllowed** – the creditAllowed value to set

Usage

Sets the

creditAllowed

of this instance.

setCurrency(Currency) method

Syntax

```
void setCurrency ( Currency currency )
```

Parameters

- **currency** – the `currency` value to set

Usage

Sets the

currency

of this instance.

setCurrency(String) method

Syntax

```
void setCurrency ( String currency )
```

Parameters

- **currency** – the `currency` value to set

Usage

Sets the

currency

of this instance.

setCustomer(Customer) method

Syntax

```
void setCustomer ( Customer customer )
```

Parameters

- **customer** – the `customer` value to set

Usage

Sets the

customer

of this instance.

setDebitAllowed(Boolean) method

Syntax

void setDebitAllowed (Boolean *debitAllowed*)

Parameters

- **debitAllowed** – the debitAllowed value to set

Usage

Sets the

debitAllowed

of this instance.

setFeeClearingMode(ClearingMode) method

Syntax

void setFeeClearingMode (ClearingMode *feeClearingMode*)

Parameters

- **feeClearingMode** – the feeClearingMode to set

setHash(String) method

Syntax

void setHash (String *hash*)

setLimitSetId(Long) method

Syntax

void setLimitSetId (Long *limitSetId*)

Parameters

- **limitSetId** – the limitSetId to set

setMultiCurrency(boolean) method

Syntax

```
void setMultiCurrency ( boolean multiCurrency )
```

Parameters

- **multiCurrency** – the multiCurrency value to set

Usage

Sets the

multiCurrency

of this instance.

setStatus(int) method

Syntax

```
void setStatus ( int status )
```

Parameters

- **status** – the status value to set

Usage

Sets the

status

of this instance.

setType(Type) method

Syntax

```
void setType ( Type type )
```

Parameters

- **type** – the type value to set

Usage

Sets the

type

of this instance.

Class() enumeration

Enum Constant Summary

- **BANK** – Payment by bank account.
- **CREDITCARD** – Payment by credit card.
- **DEBITCARD** – Payment by debit card.
- **EXTERNAL** – Payment by external account.
- **INTERNALVOUCHER** – Payment by internal voucher.
- **PHONEBILL** – Payment by phone bill.
- **PRIVATELABELCARD** – Payment by private label (credit or debit) card.
- **SVA** – Payment by SVA.
- **OFFLINESVA** – Offline SVA.

StoredValueAccount class***Syntax***

```
public class StoredValueAccount
```

Usage

The `StoredValueAccount` is a payment instrument extension that contains the ID of the account.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

StoredValueAccount() constructor**Syntax**

```
StoredValueAccount ()
```

Usage

Creates a new instance of

```
StoredValueAccount
```

.

StoredValueAccount(long) constructor

Syntax

`StoredValueAccount ( long piId )`

Parameters

- **piId** – the PK value

Usage

Creates a new instance of
`StoredValueAccount`
with a preset primary key.

compareTo(StoredValueAccount) method

This is mainly used to sort the `StoredValueAccount` objects before they get locked to make sure no dead-lock occurs.

Syntax

`int compareTo ( StoredValueAccount o )`

getCreditBalance() method

Syntax

`long getCreditBalance ()`

Returns

the `creditBalance`

Usage

the `creditBalance`

getCreditReserved() method

Syntax

`long getCreditReserved ()`

Returns

the `creditReserved`

Usage

the creditReserved

getDebitBalance() method

Syntax

long getDebitBalance ()

Returns

the debitBalance

Usage

the debitBalance

getDebitReserved() method

Syntax

long getDebitReserved ()

Returns

the debitReserved

Usage

the debitReserved

getLastBookingDate() method

Syntax

Date getLastBookingDate ()

Returns

the lastBookingDate

Usage

the lastBookingDate

getLastSvaBooking() method

Syntax

SvaBooking getLastSvaBooking ()

Returns

the lastSvaBooking

Usage

the lastSvaBooking

getMaxBalance() method

Syntax

Long getMaxBalance ()

Returns

the maxBalance

Usage

the maxBalance

getMinBalance() method

Syntax

long getMinBalance ()

Returns

the minBalance

Usage

the minBalance

getRefundAbsoluteAmount() method

Syntax

Long getRefundAbsoluteAmount ()

Returns

the absolute amount to refund this SVA with.

Usage

Returns the absolute amount to refund this SVA with.

the absolute amount to refund this SVA with.

getRefundPaymentInstrument() method

Syntax

PaymentInstrument getRefundPaymentInstrument ()

Returns

the refund PI.

Usage

Returns the PI to use for refunding.

the refund PI.

getRefundTargetAmount() method

Syntax

Long getRefundTargetAmount ()

Returns

the refund target amount for this SVA.

Usage

Returns the refund target amount for this SVA.

the refund target amount for this SVA.

getRefundThreshold() method

Syntax

Long getRefundThreshold ()

Returns

the refund threshold for this SVA.

Usage

Returns the refund threshold for this SVA.

the refund threshold for this SVA.

isRefundInTransaction() method

Syntax

Boolean isRefundInTransaction ()

Returns

true if this SVA should be refunded instantly during transactions; false otherwise

Usage

Indicates if this SVA should be refunded instantly during transactions

true if this SVA should be refunded instantly during transactions; false otherwise

isSetCreditBalance() method

Syntax

boolean isSetCreditBalance ()

Returns

false if the creditBalance is null. Otherwise, it returns true.

Usage

Checks whether the

creditBalance

of this instance is set.

false if the creditBalance is null. Otherwise, it returns true.

isSetCreditReserved() method

Syntax

boolean isSetCreditReserved ()

Returns

false if the creditReserved is null. Otherwise, it returns true.

Usage

Checks whether the

creditReserved

of this instance is set.

false if the creditReserved is null. Otherwise, it returns true.

isSetDebitBalance() method

Syntax

```
boolean isSetDebitBalance ()
```

Returns

false if the debitBalance is null. Otherwise, it returns true.

Usage

Checks whether the

debitBalance

of this instance is set.

false if the debitBalance is null. Otherwise, it returns true.

isSetDebitReserved() method

Syntax

```
boolean isSetDebitReserved ()
```

Returns

false if the debitReserved is null. Otherwise, it returns true.

Usage

Checks whether the

debitReserved

of this instance is set.

false if the debitReserved is null. Otherwise, it returns true.

isSetLastBookingDate() method

Syntax

```
boolean isSetLastBookingDate ()
```

Returns

false if the lastBookingDate is null. Otherwise, it returns true.

Usage

Checks whether the

lastBookingDate

of this instance is set.

false if the lastBookingDate is null. Otherwise, it returns true.

isSetLastSvaBooking() method

Syntax

```
boolean isSetLastSvaBooking ()
```

Returns

false if the lastSvaBooking is null. Otherwise, it returns true.

Usage

Checks whether the

lastSvaBooking

of this instance is set.

false if the lastSvaBooking is null. Otherwise, it returns true.

isSetMaxBalance() method

Syntax

```
boolean isSetMaxBalance ()
```

Returns

false if the maxBalance is null. Otherwise, it returns true.

Usage

Checks whether the

maxBalance

of this instance is set.

false if the maxBalance is null. Otherwise, it returns true.

*isSetMinBalance() method***Syntax**

```
boolean isSetMinBalance ()
```

Returns

false if the minBalance is null. Otherwise, it returns true.

Usage

Checks whether the

minBalance

of this instance is set.

false if the minBalance is null. Otherwise, it returns true.

*isSetRefundAbsoluteAmount() method***Syntax**

```
boolean isSetRefundAbsoluteAmount ()
```

Returns

false if the refundAbsoluteAmount is null. Otherwise, it returns true.

Usage

Checks whether the

refundAbsoluteAmount

of this instance is set.

false if the refundAbsoluteAmount is null. Otherwise, it returns true.

*isSetRefundInTransaction() method***Syntax**

```
boolean isSetRefundInTransaction ()
```

Returns

false if the dbRefundInTransaction is null. Otherwise, it returns true.

Usage

Checks whether the

dbRefundInTransaction

of this instance is set.

false if the dbRefundInTransaction is null. Otherwise, it returns true.

isSetRefundPaymentInstrument() method

Syntax

```
boolean isSetRefundPaymentInstrument ()
```

Returns

false if the refundPaymentInstrument is null. Otherwise, it returns true.

Usage

Checks whether the

refundPaymentInstrument

of this instance is set.

false if the refundPaymentInstrument is null. Otherwise, it returns true.

isSetRefundTargetAmount() method

Syntax

```
boolean isSetRefundTargetAmount ()
```

Returns

false if the refundTargetAmount is null. Otherwise, it returns true.

Usage

Checks whether the

refundTargetAmount

of this instance is set.

false if the refundTargetAmount is null. Otherwise, it returns true.

isSetRefundThreshold() method

Syntax

`boolean isSetRefundThreshold ()`

Returns

false if the refundThreshold is null. Otherwise, it returns true.

Usage

Checks whether the

refundThreshold

of this instance is set.

false if the refundThreshold is null. Otherwise, it returns true.

setCreditBalance(long) method

Syntax

`void setCreditBalance ( long creditBalance )`

Parameters

- **creditBalance** – the creditBalance to set

setCreditReserved(long) method

Syntax

`void setCreditReserved ( long creditReserved )`

Parameters

- **creditReserved** – the creditReserved to set

setDebitBalance(long) method

Syntax

`void setDebitBalance ( long debitBalance )`

Parameters

- **debitBalance** – the debitBalance to set

setDebitReserved(long) method

Syntax

void setDebitReserved (long *debitReserved*)

Parameters

- **debitReserved** – the debitReserved to set

setLastBookingDate(Date) method

Syntax

void setLastBookingDate (Date *lastBookingDate*)

Parameters

- **lastBookingDate** – the lastBookingDate to set

setLastSvaBooking(SvaBooking) method

Syntax

void setLastSvaBooking (SvaBooking *lastSvaBooking*)

Parameters

- **lastSvaBooking** – the lastSvaBooking to set

setMaxBalance(Long) method

Syntax

void setMaxBalance (Long *maxBalance*)

Parameters

- **maxBalance** – the maxBalance to set

setMinBalance(long) method

Syntax

void setMinBalance (long *minBalance*)

Parameters

- **minBalance** – the minBalance to set

setRefundAbsoluteAmount(Long) method

Syntax

`void setRefundAbsoluteAmount ( Long refundAbsoluteAmount )`

Parameters

- **refundAbsoluteAmount** – the refundAbsoluteAmount value to set

Usage

Sets the

refundAbsoluteAmount

of this instance.

setRefundInTransaction(Boolean) method

Syntax

`void setRefundInTransaction ( Boolean refundInTransaction )`

Parameters

- **refundInTransaction** – the refundInTransaction value to set

Usage

Sets the

refundInTransaction

of this instance.

setRefundPaymentInstrument(PaymentInstrument) method

Syntax

`void setRefundPaymentInstrument ( PaymentInstrument refundPaymentInstrument )`

Parameters

- **refundPaymentInstrument** – the `refundPaymentInstrument` value to set

Usage

Sets the

refundPaymentInstrument

of this instance.

setRefundTargetAmount(Long) method

Syntax

```
void setRefundTargetAmount ( Long refundTargetAmount )
```

Parameters

- **refundTargetAmount** – the `refundTargetAmount` value to set

Usage

Sets the

refundTargetAmount

of this instance.

setRefundThreshold(Long) method

Syntax

```
void setRefundThreshold ( Long refundThreshold )
```

Parameters

- **refundThreshold** – the `refundThreshold` value to set

Usage

Sets the

refundThreshold

of this instance.

WalletEntry class**Syntax**

```
public class WalletEntry
```

Usage

The `WalletEntry` links payment instruments to customers. Note that a payment instrument may be used by several users in parallel.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

WalletEntry() constructor**Syntax**

```
WalletEntry ()
```

Usage

Creates a new instance of

```
WalletEntry
```

```
.
```

getAlias() method**Syntax**

```
String getAlias ()
```

Returns

the associated alias

Usage

Returns the alias associated to the wallet entry.

the associated alias

getCreditPriority() method**Syntax**

```
Integer getCreditPriority ()
```

Returns

the wallet entry's *credit priority*; `null` if the payment instrument cannot or should not be used for credit transactions

Usage

Returns the priority of the wallet entry for credit transactions.

the wallet entry's *credit priority*; `null` if the payment instrument cannot or should not be used for credit transactions

getCustomer() method

Syntax

```
Customer getCustomer ()
```

Returns

the associated `Customer`

Usage

Returns the customer the wallet entry is associated with.

the associated `Customer`

getDebitPriority() method

Syntax

```
Integer getDebitPriority ()
```

Returns

the wallet entry's *debit priority*; `null` if the payment instrument cannot or should not be used for debit transactions

Usage

Returns the priority of the wallet entry for debit transactions.

the wallet entry's *debit priority*; `null` if the payment instrument cannot or should not be used for debit transactions

getLimitSetId() method

Syntax

```
Long getLimitSetId ()
```

Returns

the limitSetId

Usage

the limitSetId

getPaymentInstrument() method

Syntax

```
PaymentInstrument getPaymentInstrument ()
```

Returns

the associated PaymentInstrument

Usage

Returns the payment instrument associated to the wallet entry.

the associated PaymentInstrument

isSetAlias() method

Syntax

```
boolean isSetAlias ()
```

Returns

false if the alias is null. Otherwise, it returns true.

Usage

Checks whether the

alias

of this instance is set.

false if the alias is null. Otherwise, it returns true.

isSetCreditPriority() method

Syntax

```
boolean isSetCreditPriority ()
```

Returns

false if the creditPriority is null. Otherwise, it returns true.

Usage

Checks whether the

creditPriority

of this instance is set.

false if the *creditPriority* is null. Otherwise, it returns true.

isSetCustomer() method

Syntax

```
boolean isSetCustomer ()
```

Returns

false if the customer is null. Otherwise, it returns true.

Usage

Checks whether the

customer

of this instance is set.

false if the customer is null. Otherwise, it returns true.

isSetDebitPriority() method

Syntax

```
boolean isSetDebitPriority ()
```

Returns

false if the *debitPriority* is null. Otherwise, it returns true.

Usage

Checks whether the

debitPriority

of this instance is set.

false if the *debitPriority* is null. Otherwise, it returns true.

*isSetLimitSetId() method***Syntax**

```
boolean isSetLimitSetId ()
```

Returns

false if the `limitSetId` is null. Otherwise, it returns true.

Usage

Checks whether the

limitSetId

of this instance is set.

false if the `limitSetId` is null. Otherwise, it returns true.

*isSetPaymentInstrument() method***Syntax**

```
boolean isSetPaymentInstrument ()
```

Returns

false if the `paymentInstrument` is null. Otherwise, it returns true.

Usage

Checks whether the

paymentInstrument

of this instance is set.

false if the `paymentInstrument` is null. Otherwise, it returns true.

*setAlias(String) method***Syntax**

```
void setAlias ( String alias )
```

Parameters

- **alias** – the `alias` value to set

Usage

Sets the

alias

of this instance.

setCreditPriority(Integer) method

Syntax

```
void setCreditPriority ( Integer creditPriority )
```

Parameters

- **creditPriority** – the `creditPriority` value to set

Usage

Sets the

creditPriority

of this instance.

setCustomer(Customer) method

Syntax

```
void setCustomer ( Customer customer )
```

Parameters

- **customer** – the `customer` value to set

Usage

Sets the

customer

of this instance.

setDebitPriority(Integer) method

Syntax

```
void setDebitPriority ( Integer debitPriority )
```

Parameters

- **debitPriority** – the `debitPriority` value to set

Usage

Sets the

debitPriority

of this instance.

setLimitSetId(Long) method

Syntax

```
void setLimitSetId ( Long limitSetId )
```

Parameters

- **limitSetId** – the `limitSetId` to set

setPaymentInstrument(PaymentInstrument) method

Syntax

```
void setPaymentInstrument ( PaymentInstrument paymentInstrument )
```

Parameters

- **paymentInstrument** – the `paymentInstrument` value to set

Usage

Sets the

paymentInstrument

of this instance.

WrkWalletEntry class

Syntax

```
public class WrkWalletEntry
```

Usage

The `WrkWalletEntry` represents a maker/checker configuration for payment instrument types. Allowing to define `UmgrPrivileges` for each action all tied to a specific payment instrument type.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

WrkWalletEntry() constructor

Syntax

```
WrkWalletEntry ()
```

WrkWalletEntry(Integer) constructor

Syntax

```
WrkWalletEntry ( Integer id )
```

Parameters

- id –

Usage

Please use the

```
newInstance
```

method of the corresponding DAO.

getCheckerPrivilege() method

Syntax

```
String getCheckerPrivilege ()
```

getExecutePrivilege() method

Syntax

```
String getExecutePrivilege ()
```

getMakerPrivilege() method

Syntax

```
String getMakerPrivilege ()
```

isSetCheckerPrivilege() method

Syntax

`boolean isSetCheckerPrivilege ()`

isSetExecutePrivilege() method

Syntax

`boolean isSetExecutePrivilege ()`

isSetMakerPrivilege() method

Syntax

`boolean isSetMakerPrivilege ()`

setCheckerPrivilege(String) method

Syntax

`void setCheckerPrivilege ( String checkerPrivilege )`

setExecutePrivilege(String) method

Syntax

`void setExecutePrivilege ( String executePrivilege )`

setMakerPrivilege(String) method

Syntax

`void setMakerPrivilege ( String makerPrivilege )`

system package

Members

All public members of the system package.

- **AuthenticationMatrix** class –
- **AuthenticationMethodRule** class –
- **BatchRun** class –
- **BulkProcessingFile** class –
- **BulkProcessingFileType** class –
- **Bully** class –

- **ClearingMode** class –
- **Country** class –
- **CredentialPolicy** class –
- **CredentialPolicyRegex** class –
- **CustomerType** class –
- **CustomerTypeCredentialPolicy** class –
- **ExchangeRate** class –
- **ExchangeRateHistory** class –
- **FeeSet** class –
- **FeeType** class –
- **FeeTypeFeeSetScale** class –
- **File** class –
- **IndividualRestriction** class –
- **InterestBalanceSnapshot** class –
- **InterestCalculation** class –
- **InterestConfiguration** class –
- **InterestPeriod** class –
- **InterestProduct** class –
- **InterestProductMapping** class –
- **InvoiceType** class –
- **InvoiceTypeAttribute** class –
- **InvoiceTypeBatchConfig** class –
- **InvoiceTypeBatchInvoices** class –
- **InvoiceTypeOnlineCmdQueue** class –
- **InvoiceTypeOnlineConfig** class –
- **Job** class –
- **JobHistory** class –
- **LimitClass** class –
- **LimitSet** class –
- **LimitSetClass** class –
- **LookupQuery** class –
- **OrgUnit** class –
- **OtpType** class –
- **Restriction** class –
- **RestrictionForOrgUnit** class –
- **RestrictionForRiskCategory** class –
- **RestrictionOnOrgUnit** class –
- **RestrictionOnRiskCategory** class –
- **RestrictionOnUseCase** class –

- **RestrictionRule** class –
- **RestrictionsGroup** class –
- **RiskCategory** class –
- **Scale** class –
- **SessionPolicy** class –
- **SubTransactionType** class –
- **TransactionTemplateMapping** class –
- **UmgrPrivilege** class –
- **UmgrRole** class –
- **UmgrRolePrivilege** class –
- **UseCase** class –
- **UseCaseFee** class –
- **UseCasePaymentInstrumentType** class –
- **UseCasePrivilege** class –
- **ValidationRule** class –
- **Version** class –
- **WrkBulkProcessing** class –

AuthenticationMatrix class

Syntax

```
public class AuthenticationMatrix
```

Usage

The *AuthenticationMatrix* represents an entry of the *Authentication Matrix* which is determining which authentication method (rule) is to be applied for a transaction.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

AuthenticationMatrix() constructor

Syntax

```
AuthenticationMatrix ()
```

Usage

Creates a new instance of

```
AuthenticationMatrix
```

.

getOrderChannelId() method

Syntax

Integer getOrderChannelId ()

Returns

the orderChannelId

Usage

the orderChannelId

getOrgUnit() method

Syntax

OrgUnit getOrgUnit ()

Returns

the associated OrgUnit

Usage

Returns the associated orgunit

the associated OrgUnit

getPayee() method

Syntax

Customer getPayee ()

Returns

the associated payee Customer

Usage

Returns the associated payee.

the associated payee Customer

getPayer() method

Syntax

Customer getPayer ()

Returns

the associated payer Customer

Usage

Returns the associated payer.

the associated payer Customer

getPiType() method

Syntax

```
PaymentInstrument.Type getPiType ()
```

Returns

the associated PaymentInstrument.Type

Usage

Returns the associated payment instrument type

the associated PaymentInstrument.Type

getRule() method

Syntax

```
AuthenticationMethodRule getRule ()
```

Returns

the AuthenticationMethodRule to apply

Usage

Returns the associated authentication method rule.

the AuthenticationMethodRule to apply

getUseCase() method

Syntax

```
UseCase getUseCase ()
```

Returns

the associated UseCase

Usage

Returns the associated use case
the associated UseCase

isPayer() method

Syntax

`boolean isPayer ()`

Returns

if the rule applies for payer or payee of the transaction

Usage

Returns if the rule applies for payer or payee of the transaction.
if the rule applies for payer or payee of the transaction

isSetGetPayer() method

Syntax

`boolean isSetGetPayer ()`

Returns

false if the payer is null. Otherwise, it returns true.

Usage

Checks whether the

payer

of this instance is set.

false if the payer is null. Otherwise, it returns true.

isSetIsPayer() method

Syntax

`boolean isSetIsPayer ()`

Returns

false if the dbIsPayer is null . Otherwise, it returns true.

Usage

Checks whether the

dbIsPayer

of this instance is set.

false if the dbIsPayer is null . Otherwise, it returns true.

isSetOrderChannelId() method

Syntax

```
boolean isSetOrderChannelId ()
```

Returns

false if the orderChannelId is null. Otherwise, it returns true.

Usage

Checks whether the

orderChannelId

of this instance is set.

false if the orderChannelId is null. Otherwise, it returns true.

isSetOrgUnit() method

Syntax

```
boolean isSetOrgUnit ()
```

Returns

false if the orgUnit is null. Otherwise, it returns true.

Usage

Checks whether the

orgUnit

of this instance is set.

false if the orgUnit is null. Otherwise, it returns true.

isSetPayee() method

Syntax

boolean isSetPayee ()

Returns

false if the payee is null. Otherwise, it returns true.

Usage

Checks whether the

payee

of this instance is set.

false if the payee is null. Otherwise, it returns true.

isSetPiType() method

Syntax

boolean isSetPiType ()

Returns

false if the piType is null. Otherwise, it returns true.

Usage

Checks whether the

piType

of this instance is set.

false if the piType is null. Otherwise, it returns true.

isSetRule() method

Syntax

boolean isSetRule ()

Returns

false if the rule is null. Otherwise, it returns true.

Usage

Checks whether the

rule

of this instance is set.

false if the rule is null. Otherwise, it returns true.

isSetUseCase() method

Syntax

```
boolean isSetUseCase ()
```

Returns

false if the useCase is null. Otherwise, it returns true.

Usage

Checks whether the

useCase

of this instance is set.

false if the useCase is null. Otherwise, it returns true.

setOrderChannelId(Integer) method

Syntax

```
void setOrderChannelId ( Integer orderChannelId )
```

Parameters

- **orderChannelId** – the orderChannelId to set

setOrgUnit(OrgUnit) method

Syntax

```
void setOrgUnit ( OrgUnit orgUnit )
```

Parameters

- **orgUnit** – the orgUnit value to set

Usage

Sets the

orgUnit

of this instance.

setPayee(Customer) method

Syntax

void setPayee (Customer *payee*)

Parameters

- **payee** – the payee value to set

Usage

Sets the

payee

of this instance.

setPayer(boolean) method

Syntax

void setPayer (boolean *isPayer*)

Parameters

- **isPayer** – the isPayer value to set

Usage

Sets the

isPayer

of this instance.

setPayer(Customer) method

Syntax

void setPayer (Customer *payer*)

Parameters

- **payer** – the `payer` value to set

Usage

Sets the

payer

of this instance.

setPiType(PaymentInstrument.Type) method

Syntax

```
void setPiType ( PaymentInstrument.Type piType )
```

Parameters

- **piType** – the `piType` value to set

Usage

Sets the

piType

of this instance.

setRule(AuthenticationMethodRule) method

Syntax

```
void setRule ( AuthenticationMethodRule rule )
```

Parameters

- **rule** – the `rule` value to set

Usage

Sets the

rule

of this instance.

setUseCase(UseCase) method

Syntax

```
void setUseCase ( UseCase useCase )
```

Parameters

- **useCase** – the `useCase` value to set

Usage

Sets the

useCase

of this instance.

AuthenticationMethodRule class

Syntax

```
public class AuthenticationMethodRule
```

Usage

The `AuthenticationMethodRule` models a rule set for the authentication mechanism.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Step class

Syntax

```
public class Step
```

Usage

The `Steps` are used to define the various authentication methods to apply for specific threshold values.

© 2000-2007 by paybox solutions AG

Revision:1.10Name:not supported by cvs2svn

Step() constructor

Syntax

Step ()

Usage

Creates a new instance of

Step

.

Step(StepPk) constructor

Syntax

Step (StepPk *id*)

Parameters

- **id** – the PK value

Usage

Creates a new instance of

Step

with a preset primary key.

getAuthMethodId() method

Syntax

int getAuthMethodId ()

getCurrency() method

Syntax

Currency getCurrency ()

Returns

the currency

Usage

the currency

getMaxAmount() method

Syntax

Long getMaxAmount ()

Returns

the maxAmount

Usage

the maxAmount

getOrder() method

Syntax

int getOrder ()

Returns

the order

Usage

the order

getRule() method

Syntax

Integer getRule ()

Returns

the rule

Usage

the rule

isSetAuthMethodId() method

Syntax

boolean isSetAuthMethodId ()

Returns

false if the `authMethodId` is null. Otherwise, it returns true.

Usage

Checks whether the

authMethodId

of this instance is set.

false if the `authMethodId` is null. Otherwise, it returns true.

isSetCurrency() method

Syntax

```
boolean isSetCurrency ()
```

Returns

false if the `id` or the `id`'s `dbCurrency` is null. Otherwise, it returns true.

Usage

Checks whether the

dbCurrency

of this instance's PK is set.

false if the `id` or the `id`'s `dbCurrency` is null. Otherwise, it returns true.

isSetMaxAmount() method

Syntax

```
boolean isSetMaxAmount ()
```

Returns

false if the `maxAmount` is null. Otherwise, it returns true.

Usage

Checks whether the

maxAmount

of this instance is set.

false if the `maxAmount` is null. Otherwise, it returns true.

isSetOrder() method

Syntax

boolean isSetOrder ()

Returns

false if the id or the id's order is null. Otherwise, it returns true.

Usage

Checks whether the

order

of this instance's PK is set.

false if the id or the id's order is null. Otherwise, it returns true.

isSetRule() method

Syntax

boolean isSetRule ()

Returns

false if the id or the id's rule is null. Otherwise, it returns true.

Usage

Checks whether the

rule

of this instance's PK is set.

false if the id or the id's rule is null. Otherwise, it returns true.

setAuthMethodId(int) method

Syntax

void setAuthMethodId (int *authMethodId*)

setMaxAmount(Long) method

Syntax

void setMaxAmount (Long *maxAmount*)

Parameters

- **maxAmount** – the `maxAmount` value to set

Usage

Sets the

maxAmount

of this instance.

StepPk class

Syntax

```
public class StepPk
```

Usage

The `StepPk` class is the embeddable primary key object for an authentication rule's step.

© 2000-2007 by paybox solutions AG

Revision:1.10Name:not supported by cvs2svn

StepPk() constructor

Syntax

```
StepPk ()
```

Usage

Creates a new instance of

`StepPk`

.

StepPk(Integer, int, Currency) constructor

Syntax

```
StepPk ( Integer rule , int order , Currency currency )
```

Parameters

- **rule** – the associated `AuthenticationMethodRule`
- **order** – the priority/order of this step
- **currency** – the currency of this step

Usage

Creates a new instance of

StepPk

.

equals(Object) method

Syntax

boolean equals (Object *o*)

Parameters

- **o** – the Object to compare with

Returns

true if this class is assignable from o's class and has the same PK values as this instance

Usage

Indicates if another object is

equal to

this one.

true if this class is assignable from o's class and has the same PK values as this instance

getCurrency() method

Syntax

Currency getCurrency ()

Returns

the dbCurrency

Usage

the dbCurrency

getOrder() method

Syntax

int getOrder ()

Returns

the order

Usage

the order

getRule() method

Syntax

Integer getRule ()

Returns

the rule

Usage

the rule

hashCode() method

Syntax

int hashCode ()

Returns

a hash value for this instance

Usage

Returns a hash value.

a hash value for this instance

isSetCurrency() method

Syntax

boolean isSetCurrency ()

Returns

false if the dbCurrency is null. Otherwise, it returns true.

Usage

Checks whether the

dbCurrency

of this instance is set.

false if the dbCurrency is null. Otherwise, it returns true.

isSetOrder() method

Syntax

boolean isSetOrder ()

Returns

false if the order is null . Otherwise, it returns true.

Usage

Checks whether the

order

of this instance is set.

false if the order is null . Otherwise, it returns true.

isSetRule() method

Syntax

boolean isSetRule ()

Returns

false if the rule is null. Otherwise, it returns true.

Usage

Checks whether the

rule

of this instance is set.

false if the rule is null. Otherwise, it returns true.

toString() method

Syntax

String toString ()

Returns

a `String` representing this instance

Usage

Returns a string representation of this object.

a `String` representing this instance

AuthenticationMethodRule() constructor

Syntax

```
AuthenticationMethodRule ()
```

Usage

Creates a new instance of

```
AuthenticationMethodRule
```

.

AuthenticationMethodRule(int) constructor

Syntax

```
AuthenticationMethodRule ( int id )
```

Parameters

- `id` – the PK value

Usage

Creates a new instance of

```
AuthenticationMethodRule
```

with a preset primary key.

getSteps() method

Syntax

```
List< Step > getSteps ()
```

Returns

an ordered `List` of the assigned `Steps`

Usage

Returns the steps defined for this rule.
an ordered List of the assigned Steps

isSetSteps() method

Syntax

```
boolean isSetSteps ()
```

Returns

false if the steps is null. Otherwise, it returns true.

Usage

Checks whether the
steps
of this instance is set.
false if the steps is null. Otherwise, it returns true.

setSteps(List< Step >) method

Syntax

```
void setSteps ( List< Step > steps )
```

Parameters

- **steps** – the steps value to set

Usage

Sets the
steps
of this instance.

BatchRun class

Syntax

```
public class BatchRun
```

Usage

A Batchrun is typically used to synchronize runs among (cron-)jobs. A new Batchrun searches for records of the same type, start date younger than X hours, and no finish date. In such a case

the Batchrun usually exits with an error. It can also be see as historical information of Batchrun execution.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

BatchRun() constructor

Syntax

BatchRun ()

Usage

Creates a new instance of

BatchRun

.

getFinished() method

Syntax

Date getFinished ()

Returns

the finish date of the instance

Usage

the finish date of the instance

getStarted() method

Syntax

Date getStarted ()

Returns

the start date of the instance

Usage

the start date of the instance

getType() method

Syntax

`int getType ()`

Returns

the type of the instance

Usage

the type of the instance

isSetFinished() method

Syntax

`boolean isSetFinished ()`

Returns

false if the finished is null. Otherwise, it returns true.

Usage

Checks whether the

finished

of this instance is set.

false if the finished is null. Otherwise, it returns true.

isSetStarted() method

Syntax

`boolean isSetStarted ()`

Returns

false if the started is null. Otherwise, it returns true.

Usage

Checks whether the

started

of this instance is set.

false if the started is null. Otherwise, it returns true.

isSetType() method

Syntax

boolean isSetType ()

Returns

false if the type is null. Otherwise, it returns true.

Usage

Checks whether the

type

of this instance is set.

false if the type is null. Otherwise, it returns true.

setFinished(Date) method

Syntax

void setFinished (Date *finished*)

Parameters

- **finished** – the finish date to set

setStarted(Date) method

Syntax

void setStarted (Date *started*)

Parameters

- **started** – the start date to set

setType(int) method

Syntax

void setType (int *type*)

Parameters

- **type** – the type to set

BulkProcessingFile class

Syntax

```
public class BulkProcessingFile
```

Usage

The `BulkProcessingFile` represents a physical file whichs content resembles operations that need to be executed (e.g. Service calls). Typicall one line of the file contains all information to carry out the desired operation.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

BulkProcessingFile() constructor

Syntax

```
BulkProcessingFile ()
```

Usage

Creates a new instance of

Customer

.

getFileName() method

Syntax

```
String getFileName ()
```

getFileType() method

Syntax

```
Integer getFileType ()
```

getProcessingStatus() method

Syntax

```
Integer getProcessingStatus ()
```

getRequestFile() method

Syntax

byte[] getRequestFile ()

getResponseFile() method

Syntax

byte[] getResponseFile ()

isSeRequestFile() method

Syntax

boolean isSeRequestFile ()

isSetFileType() method

Syntax

boolean isSetFileType ()

isSetProcessingStatus() method

Syntax

boolean isSetProcessingStatus ()

isSetResponseFile() method

Syntax

boolean isSetResponseFile ()

setFileName(String) method

Syntax

void setFileName (String *fileName*)

setFileType(Integer) method

Syntax

void setFileType (Integer *fileType*)

setProcessingStatus(Integer) method

Syntax

void setProcessingStatus (Integer *processingStatus*)

setRequestFile(byte[]) method

Syntax

void setRequestFile (byte[] *requestFile*)

setResponseFile(byte[]) method

Syntax

void setResponseFile (byte[] *responseFile*)

BulkProcessingFileType class

Syntax

public class BulkProcessingFileType

Usage

The BulkProcessingFileType defines a file type used in conjunction with BulkProcessingFile entities.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

BulkProcessingFileType() constructor

Syntax

BulkProcessingFileType ()

BulkProcessingFileType(Integer) constructor

Syntax

BulkProcessingFileType (Integer *id*)

Parameters

- `id` –

Usage

Please use the

`newInstance`

method of the corresponding DAO.

getContentType() method

Syntax

`String getContentType ()`

getFileTypeDescription() method

Syntax

`String getFileTypeDescription ()`

isSetContentType() method

Syntax

`boolean isSetContentType ()`

isSetFileTypeDescription() method

Syntax

`boolean isSetFileTypeDescription ()`

setContentType(String) method

Syntax

`void setContentType ( String contentType )`

setFileTypeDescription(String) method

Syntax

`void setFileTypeDescription ( String fileTypeDescription )`

Bully class

Syntax

```
public class Bully
```

Usage

The `Bully` serves as a cross-JVM-semaphore

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Bully() constructor

Syntax

```
Bully ()
```

Usage

Creates a new instance of

```
Bully
```

.

Bully(String) constructor

Syntax

```
Bully ( String serviceName )
```

Parameters

- **serviceName** – the name of the service bullied by this row

Usage

Creates a new instance of

```
Bully
```

with a preset primary key.

*getLeaseEnd() method***Syntax**

```
Date getLeaseEnd ()
```

Returns

the end of the current worker's lease.

Usage

Returns the end of the current worker's lease.

the end of the current worker's lease.

*getLeaseTerm() method***Syntax**

```
int getLeaseTerm ()
```

Returns

the lease extension term in seconds.

Usage

Returns the lease extension term in seconds.

the lease extension term in seconds.

*getWorker() method***Syntax**

```
String getWorker ()
```

Returns

the worker's unique ID.

Usage

Returns the worker's unique ID.

the worker's unique ID.

isActive() method

Syntax

boolean isActive ()

Returns

if the service is active.

Usage

Returns if the service is active.

if the service is active.

isSetActive() method

Syntax

boolean isSetActive ()

Returns

false if the dbIsActive is null. Otherwise, it returns true.

Usage

Checks whether the

dbIsActive

of this instance is set.

false if the dbIsActive is null. Otherwise, it returns true.

isSetLeaseEnd() method

Syntax

boolean isSetLeaseEnd ()

Returns

false if the leaseEnd is null. Otherwise, it returns true.

Usage

Checks whether the

leaseEnd

of this instance is set.

false if the leaseEnd is null. Otherwise, it returns true.

isSetLeaseTerm() method

Syntax

```
boolean isSetLeaseTerm ()
```

Returns

false if the leaseTerm is null . Otherwise, it returns true.

Usage

Checks whether the

leaseTerm

of this instance is set.

false if the leaseTerm is null . Otherwise, it returns true.

isSetWorker() method

Syntax

```
boolean isSetWorker ()
```

Returns

false if the worker is null. Otherwise, it returns true.

Usage

Checks whether the

worker

of this instance is set.

false if the worker is null. Otherwise, it returns true.

setAffectedEntity(boolean) method

Syntax

```
void setAffectedEntity ( boolean active )
```

Parameters

- **active** – the active value to set

Usage

Sets the

dbIsActive

of this instance.

setLeaseEnd(Date) method

Syntax

```
void setLeaseEnd ( Date leaseEnd )
```

Parameters

- **leaseEnd** – the leaseEnd value to set

Usage

Sets the

leaseEnd

of this instance.

setLeaseTerm(int) method

Syntax

```
void setLeaseTerm ( int leaseTerm )
```

Parameters

- **leaseTerm** – the leaseTerm value to set

Usage

Sets the

leaseTerm

of this instance.

setWorker(String) method

Syntax

```
void setWorker ( String worker )
```

Parameters

- **worker** – the `worker` value to set

Usage

Sets the

worker

of this instance.

ClearingMode class

Syntax

```
public class ClearingMode
```

Usage

This entity represents a clearing mode. The clearing modes control the way accounts are cleared in terms of frequency and other parameters.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

ClearingMode() constructor

Syntax

```
ClearingMode ()
```

Usage

Creates a new instance of

`ClearingMode`

.

ClearingMode(String) constructor

Syntax

ClearingMode (String *id*)

Parameters

- **id** – the PK value

Usage

Creates a new instance of

ClearingMode

with a preset primary key.

getClearingFrequency() method

Syntax

String getClearingFrequency ()

Returns

the clearingFrequency

Usage

the clearingFrequency

getClearingMode() method

Syntax

String getClearingMode ()

Returns

the clearingMode

Usage

the clearingMode

getMpExpirationTime() method

Syntax

Integer getMpExpirationTime ()

Returns

the mpExpirationTime

Usage

the mpExpirationTime

getMpUpperLimit() method

Syntax

Long getMpUpperLimit ()

Returns

the mpUpperLimit

Usage

the mpUpperLimit

isAccumulated() method

Syntax

boolean isAccumulated ()

Returns

the dbAccumulated

Usage

the dbAccumulated

isActive() method

Syntax

boolean isActive ()

Returns

the dbIsActive

Usage

the dbIsActive

isInvoice() method

Syntax

```
boolean isInvoice ()
```

Returns

the dbInvoice

Usage

the dbInvoice

isMicroPayment() method

Syntax

```
boolean isMicroPayment ()
```

Returns

the dbMicropayment

Usage

the dbMicropayment

isSetAccumulated() method

Syntax

```
boolean isSetAccumulated ()
```

Returns

false if the dbAccumulated is null. Otherwise, it returns true.

Usage

Checks whether the

dbAccumulated

of this instance is set.

false if the dbAccumulated is null. Otherwise, it returns true.

isActive() method

Syntax

`boolean isActive ()`

Returns

false if the `isActive` is null. Otherwise, it returns true.

Usage

Checks whether the

isActive

of this instance is set.

false if the `isActive` is null. Otherwise, it returns true.

isClearingFrequency() method

Syntax

`boolean isClearingFrequency ()`

Returns

false if the `clearingFrequency` is null. Otherwise, it returns true.

Usage

Checks whether the

clearingFrequency

of this instance is set.

false if the `clearingFrequency` is null. Otherwise, it returns true.

isClearingMode() method

Syntax

`boolean isClearingMode ()`

Returns

false if the `clearingMode` is null. Otherwise, it returns true.

Usage

Checks whether the

clearingMode

of this instance is set.

false if the clearingMode is null. Otherwise, it returns true.

isSetInvoice() method

Syntax

```
boolean isSetInvoice ()
```

Returns

false if the dbInvoice is null . Otherwise, it returns true.

Usage

Checks whether the

dbInvoice

of this instance is set.

false if the dbInvoice is null . Otherwise, it returns true.

isSetMicroPayment() method

Syntax

```
boolean isSetMicroPayment ()
```

Returns

false if the dbMicropayment is null. Otherwise, it returns true.

Usage

Checks whether the

dbMicropayment

of this instance is set.

false if the dbMicropayment is null. Otherwise, it returns true.

*isSetMpExpirationTime() method***Syntax**

```
boolean isSetMpExpirationTime ()
```

Returns

false if the mpExpirationTime is null. Otherwise, it returns true.

Usage

Checks whether the

mpExpirationTime

of this instance is set.

false if the mpExpirationTime is null. Otherwise, it returns true.

*isSetMpUpperLimit() method***Syntax**

```
boolean isSetMpUpperLimit ()
```

Returns

false if the mpUpperLimit is null. Otherwise, it returns true.

Usage

Checks whether the

mpUpperLimit

of this instance is set.

false if the mpUpperLimit is null. Otherwise, it returns true.

*setAccumulated(boolean) method***Syntax**

```
void setAccumulated ( boolean dbAccumulated )
```

Parameters

- **dbAccumulated** – the dbAccumulated to set

setActive(boolean) method

Syntax

void setActive (boolean *dbIsActive*)

Parameters

- **dbIsActive** – the dbIsActive to set

setClearingFrequency(String) method

Syntax

void setClearingFrequency (String *clearingFrequency*)

Parameters

- **clearingFrequency** – the clearingFrequency to set

setClearingMode(String) method

Syntax

void setClearingMode (String *clearingMode*)

Parameters

- **clearingMode** – the clearingMode to set

setInvoice(boolean) method

Syntax

void setInvoice (boolean *dbInvoice*)

Parameters

- **dbInvoice** – the dbInvoice to set

setMicroPayment(boolean) method

Syntax

void setMicroPayment (boolean *dbMicropayment*)

Parameters

- **dbMicropayment** – the dbMicropayment to set

setMpExpirationTime(Integer) method

Syntax

```
void setMpExpirationTime ( Integer mpExpirationTime )
```

Parameters

- **mpExpirationTime** – the mpExpirationTime to set

setMpUpperLimit(Long) method

Syntax

```
void setMpUpperLimit ( Long mpUpperLimit )
```

Parameters

- **mpUpperLimit** – the mpUpperLimit to set

Country class**Syntax**

```
public class Country
```

Usage

The `Country` represents a country.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Country() constructor

Syntax

```
Country ()
```

Usage

Creates a new instance of

```
Country
```

.

Country(String) constructor

Syntax

Country (String *id*)

Parameters

- **id** – the PK value

Usage

Creates a new instance of

Country

with a preset primary key.

getCurrency() method

Syntax

Currency getCurrency ()

Returns

the country's Currency

Usage

Returns the currency officially used in this country.

the country's Currency

getIso3166a3() method

Syntax

String getIso3166a3 ()

getNumericCode() method

Syntax

int getNumericCode ()

getPhonePrefix() method

Syntax

String getPhonePrefix ()

isSetCurrency() method

Syntax

boolean isSetCurrency ()

Returns

false if the dbCurrency is null. Otherwise, it returns true.

Usage

Checks whether the

dbCurrency

of this instance is set.

false if the dbCurrency is null. Otherwise, it returns true.

isSetIso3166a3() method

Syntax

boolean isSetIso3166a3 ()

Returns

false if the iso3166a3 is null . Otherwise, it returns true.

Usage

Checks whether the

iso3166a3

of this instance is set.

false if the iso3166a3 is null . Otherwise, it returns true.

isSetNumericCode() method

Syntax

boolean isSetNumericCode ()

Returns

false if the `numericCode` is null. Otherwise, it returns true.

Usage

Checks whether the

numericCode

of this instance is set.

false if the `numericCode` is null. Otherwise, it returns true.

isSetPhonePrefix() method

Syntax

```
boolean isSetPhonePrefix ()
```

Returns

false if the `phonePrefix` is null. Otherwise, it returns true.

Usage

Checks whether the

phonePrefix

of this instance is set.

false if the `phonePrefix` is null. Otherwise, it returns true.

setCurrency(Currency) method

Syntax

```
void setCurrency ( Currency currency )
```

Parameters

- **currency** – the `currency` value to set

Usage

Sets the

currency

of this instance.

setIso3166a3(String) method

Syntax

```
void setIso3166a3 ( String iso3166a3 )
```

setNumericCode(int) method

Syntax

```
void setNumericCode ( int numericCode )
```

setPhonePrefix(String) method

Syntax

```
void setPhonePrefix ( String phonePrefix )
```

CredentialPolicy class

Syntax

```
public class CredentialPolicy
```

Usage

The `SecurityPolicy` models a security policy for the authentication processes, including credential length, block threshold and time, credential expiration and retention.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CredentialPolicy() constructor

Syntax

```
CredentialPolicy ()
```

Usage

Creates a new instance of

```
CredentialPolicy
```

.

getBlockMinutes() method

Syntax

Integer getBlockMinutes ()

Returns

an Integer representing the number of minutes to block a Customer; null to blacklist him

Usage

Returns the number of minutes to block a customer after he exceeded the block threshold. If this value is

null

, the customer should be blacklisted instead.

an Integer representing the number of minutes to block a Customer; null to blacklist him

getBlockSize() method

Syntax

int getBlockSize ()

Returns

the block size to examine

Usage

Returns the the number of chars to check for ascending ('1234'), descending or equal ('aaa') blocks. If, for example, the block size is 3, a credential like '9

123

' or 'a

bbb

c' would be fine, whereas for a

char block size

of 2, they would be invalid.

the block size to examine

*getBlockThreshold() method***Syntax**

```
Integer getBlockThreshold ()
```

Returns

an `Integer` representing the threshold; `null` if there are no consequences

Usage

Returns the number of times after which specifying wrong credentials causes a customer to be blocked. If this value is

`null`

, there should be no consequences for specifying wrong credentials.

an `Integer` representing the threshold; `null` if there are no consequences

*getCredentialExpiration() method***Syntax**

```
Integer getCredentialExpiration ()
```

Returns

an `Integer` being the number of expiration days; `null` if the credentials never expire

Usage

Returns the number of days after which a credential's validity expires.

an `Integer` being the number of expiration days; `null` if the credentials never expire

*getCredentialRetention() method***Syntax**

```
int getCredentialRetention ()
```

Returns

the number of archived credentials

Usage

Returns the number of credentials that should be archived.

the number of archived credentials

getMaximumCredentialLength() method

Syntax

```
int getMaximumCredentialLength ()
```

Returns

the maximum number of required characters

Usage

Returns the maximum (char) length of a new credential.

the maximum number of required characters

getMinimumCredentialLength() method

Syntax

```
int getMinimumCredentialLength ()
```

Returns

the minimum number of required characters

Usage

Returns the minimum (char) length of a new credential.

the minimum number of required characters

getName() method

Syntax

```
String getName ()
```

getRegex() method

Syntax

```
Set< CredentialPolicyRegex > getRegex ()
```

Returns

a Set containing the policy's SecurityPolicyRegex

Usage

Returns the associated regular expressions for this policy.
 a Set containing the policy's SecurityPolicyRegex

getTempExpiration() method

Syntax

Integer getTempExpiration ()

getUnusedTimeout() method

Syntax

Integer getUnusedTimeout ()

isSetBlockMinutes() method

Syntax

boolean isSetBlockMinutes ()

Returns

false if the blockMinutes is null. Otherwise, it returns true.

Usage

Checks whether the

blockMinutes

of this instance is set.

false if the blockMinutes is null. Otherwise, it returns true.

isSetBlockSize() method

Syntax

boolean isSetBlockSize ()

Returns

false if the blockSize is null . Otherwise, it returns true.

Usage

Checks whether the

blockSize

of this instance is set.

false if the blockSize is null . Otherwise, it returns true.

isSetBlockThreshold() method

Syntax

```
boolean isSetBlockThreshold ()
```

Returns

false if the blockThreshold is null. Otherwise, it returns true.

Usage

Checks whether the

blockThreshold

of this instance is set.

false if the blockThreshold is null. Otherwise, it returns true.

isSetCredentialExpiration() method

Syntax

```
boolean isSetCredentialExpiration ()
```

Returns

false if the credentialExpiration is null. Otherwise, it returns true.

Usage

Checks whether the

credentialExpiration

of this instance is set.

false if the credentialExpiration is null. Otherwise, it returns true.

isSetCredentialRetention() method

Syntax

```
boolean isSetCredentialRetention ()
```

Returns

false if the credentialRetention is null. Otherwise, it returns true.

Usage

Checks whether the

credentialRetention

of this instance is set.

false if the *credentialRetention* is null. Otherwise, it returns true.

isSetMaximumCredentialLength() method

Syntax

```
boolean isSetMaximumCredentialLength ()
```

Returns

false if the *maximumCredentialLength* is null. Otherwise, it returns true.

Usage

Checks whether the

maximumCredentialLength

of this instance is set.

false if the *maximumCredentialLength* is null. Otherwise, it returns true.

isSetMinimumCredentialLength() method

Syntax

```
boolean isSetMinimumCredentialLength ()
```

Returns

false if the *minimumCredentialLength* is null. Otherwise, it returns true.

Usage

Checks whether the

minimumCredentialLength

of this instance is set.

false if the *minimumCredentialLength* is null. Otherwise, it returns true.

isSetName() method

Syntax

boolean isSetName ()

isSetRegex() method

Syntax

boolean isSetRegex ()

Returns

false if the regex is null. Otherwise, it returns true.

Usage

Checks whether the

regex

of this instance is set.

false if the regex is null. Otherwise, it returns true.

isSetTempExpiration() method

Syntax

boolean isSetTempExpiration ()

isSetUnusedTimeout() method

Syntax

boolean isSetUnusedTimeout ()

setBlockMinutes(Integer) method

Syntax

void setBlockMinutes (Integer *blockMinutes*)

Parameters

- **blockMinutes** – the blockMinutes value to set

Usage

Sets the

blockMinutes

of this instance.

setBlockSize(int) method

Syntax

```
void setBlockSize ( int blockSize )
```

Parameters

- **blockSize** – the `blockSize` value to set

Usage

Sets the

blockSize

of this instance.

setBlockThreshold(Integer) method

Syntax

```
void setBlockThreshold ( Integer blockThreshold )
```

Parameters

- **blockThreshold** – the `blockThreshold` value to set

Usage

Sets the

blockThreshold

of this instance.

setCredentialExpiration(Integer) method

Syntax

```
void setCredentialExpiration ( Integer credentialExpiration )
```

Parameters

- **credentialExpiration** – the `credentialExpiration` value to set

Usage

Sets the

credentialExpiration

of this instance.

setCredentialRetention(int) method

Syntax

```
void setCredentialRetention ( int credentialRetention )
```

Parameters

- **credentialRetention** – the `credentialRetention` value to set

Usage

Sets the

credentialRetention

of this instance.

setMaximumCredentialLength(int) method

Syntax

```
void setMaximumCredentialLength ( int maximumCredentialLength )
```

Parameters

- **maximumCredentialLength** – the `maximumCredentialLength` value to set

Usage

Sets the

maximumCredentialLength

of this instance.

setMinimumCredentialLength(int) method

Syntax

void setMinimumCredentialLength (int *minimumCredentialLength*)

Parameters

- **minimumCredentialLength** – the minimumCredentialLength value to set

Usage

Sets the

minimumCredentialLength

of this instance.

setName(String) method

Syntax

void setName (String *name*)

setRegex(Set< CredentialPolicyRegex >) method

Syntax

void setRegex (Set< CredentialPolicyRegex > *regex*)

Parameters

- **regex** – the regex value to set

Usage

Sets the

regex

of this instance.

setTempExpiration(Integer) method

Syntax

void setTempExpiration (Integer *tempExpiration*)

setUnusedTimeout(Integer) method

Syntax

void setUnusedTimeout (Integer *unusedTimeout*)

CredentialPolicyRegex class

Syntax

public class CredentialPolicyRegex

Usage

The SecurityPolicyRegex represents a regular expression that is either to be matched or not matched b new credentials.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CredentialPolicyRegex() constructor

Syntax

CredentialPolicyRegex ()

Usage

Creates a new instance of

CredentialPolicyRegex

.

getI18nKey() method

Syntax

String getI18nKey ()

Returns

the i18n key

Usage

Returns the

i18n

key to be used in the error message if this key cannot be or is matched. The key will be looked up in a resource bundle.

the `i18n` key

getPolicy() method

Syntax

`CredentialPolicy` `getPolicy ()`

Returns

the `SecurityPolicy`

Usage

Returns the assigned security policy.

the `SecurityPolicy`

getRegex() method

Syntax

`Pattern` `getRegex ()`

Returns

the regex `Pattern`

Usage

Returns the regular expression pattern.

the regex `Pattern`

isMatch() method

Syntax

`boolean` `isMatch ()`

Returns

`true` if the regular expression needs to be matched; `false` if it must not be matched

Usage

Indicates if this regular expression should be matched or not.

`true` if the regular expression needs to be matched; `false` if it must not be matched

isSetI18nKey() method

Syntax

boolean isSetI18nKey ()

Returns

false if the i18nKey is null. Otherwise, it returns true.

Usage

Checks whether the

i18nKey

of this instance is set.

false if the i18nKey is null. Otherwise, it returns true.

isSetMatch() method

Syntax

boolean isSetMatch ()

Returns

false if the dbMatch is null. Otherwise, it returns true.

Usage

Checks whether the

dbMatch

of this instance is set.

false if the dbMatch is null. Otherwise, it returns true.

isSetPolicy() method

Syntax

boolean isSetPolicy ()

Returns

false if the policy is null. Otherwise, it returns true.

Usage

Checks whether the

policy

of this instance is set.

false if the *policy* is null. Otherwise, it returns true.

isSetRegex() method

Syntax

```
boolean isSetRegex ()
```

Returns

false if the *dbRegex* is null. Otherwise, it returns true.

Usage

Checks whether the

dbRegex

of this instance is set.

false if the *dbRegex* is null. Otherwise, it returns true.

setI18nKey(String) method

Syntax

```
void setI18nKey ( String key )
```

Parameters

- **key** – the *i18nKey* value to set

Usage

Sets the

i18nKey

of this instance.

setMatch(boolean) method

Syntax

```
void setMatch ( boolean match )
```

Parameters

- **match** – the `match` value to set

Usage

Sets the

match

of this instance.

setPolicy(CredentialPolicy) method

Syntax

```
void setPolicy ( CredentialPolicy policy )
```

Parameters

- **policy** – the `policy` value to set

Usage

Sets the

policy

of this instance.

setRegex(Pattern) method

Syntax

```
void setRegex ( Pattern regex )
```

Parameters

- **regex** – the `regex` value to set

Usage

Sets the

regex

of this instance.

*CustomerType class**Syntax*

```
public class CustomerType
```

Usage

The Customer Type is used to differentiate groups of Customers. Customer Types represent business roles in Mobiliser. Common Customer Types are "Consumer", "Merchant", "CSR", "Agent". There is no hard-coded logic on Customer Types. Customer Types can be defined for each installation separately. The Customer Type is mandatory for a Customer and is set at registration time and usually not changed over the lifetime of a Customer.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

*CustomerType() constructor***Syntax**

```
CustomerType ()
```

Usage

Creates a new instance of

```
CustomerType
```

```
.
```

*CustomerType(int) constructor***Syntax**

```
CustomerType ( int id )
```

Parameters

- **id** – the PK value

Usage

Creates a new instance of

```
CustomerType
```

with a preset primary key.

getFeeSetId() method

Syntax

Long getFeeSetId ()

getInternal() method

Syntax

boolean getInternal ()

getRiskCategory() method

Syntax

RiskCategory getRiskCategory ()

getRole() method

Syntax

String getRole ()

getSessionPolicy() method

Syntax

SessionPolicy getSessionPolicy ()

isSetFeeSetId() method

Syntax

boolean isSetFeeSetId ()

Returns

false if the feeSetId is null. Otherwise, it returns true.

Usage

Checks whether the

feeSetId

of this instance is set.

false if the feeSetId is null. Otherwise, it returns true.

*isSetInternal() method***Syntax**

```
boolean isSetInternal ()
```

Returns

false if the dbInternal is null. Otherwise, it returns true.

Usage

Checks whether the

dbInternal

of this instance is set.

false if the dbInternal is null. Otherwise, it returns true.

*isSetRiskCategory() method***Syntax**

```
boolean isSetRiskCategory ()
```

Returns

false if the riskCategory is null. Otherwise, it returns true.

Usage

Checks whether the

riskCategory

of this instance is set.

false if the riskCategory is null. Otherwise, it returns true.

*isSetRole() method***Syntax**

```
boolean isSetRole ()
```

Returns

false if the role is null. Otherwise, it returns true.

Usage

Checks whether the

role

of this instance is set.

false if the role is null. Otherwise, it returns true.

isSetSessionPolicy() method

Syntax

boolean isSetSessionPolicy ()

Returns

false if the sessionPolicy is null. Otherwise, it returns true.

Usage

Checks whether the

sessionPolicy

of this instance is set.

false if the sessionPolicy is null. Otherwise, it returns true.

setFeeSetId(Long) method

Syntax

void setFeeSetId (Long *feeSetId*)

setInternal(boolean) method

Syntax

void setInternal (boolean *internal*)

setRiskCategory(RiskCategory) method

Syntax

void setRiskCategory (RiskCategory *riskCategory*)

setRole(String) method

Syntax

void setRole (String *role*)

setSessionPolicy(SessionPolicy) method

Syntax

```
void setSessionPolicy ( SessionPolicy sessionPolicy )
```

CustomerTypeCredentialPolicy class

Syntax

```
public class CustomerTypeCredentialPolicy
```

Usage

Maps credential policies to customer types and credential types.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CustomerTypeCredentialPolicy() constructor

Syntax

```
CustomerTypeCredentialPolicy ()
```

Usage

Creates a new instance of

```
CustomerTypeCredentialPolicy
```

.

getCredentialPolicy() method

Syntax

```
CredentialPolicy getCredentialPolicy ()
```

getCredentialType() method

Syntax

```
int getCredentialType ()
```

Returns

the credentialType

Usage

the credentialType

getCustomerId() method

Syntax

```
int getCustomerId ()
```

Returns

the customerId

Usage

the customerId

isSetCredentialPolicy() method

Syntax

```
boolean isSetCredentialPolicy ()
```

Returns

false if the credentialPolicy is null. Otherwise, it returns true.

Usage

Checks whether the

credentialPolicy

of this instance is set.

false if the credentialPolicy is null. Otherwise, it returns true.

isSetCredentialType() method

Syntax

```
boolean isSetCredentialType ()
```

Returns

false if the credentialType is null. Otherwise, it returns true.

Usage

Checks whether the

credentialType

of this instance is set.

false if the *credentialType* is null. Otherwise, it returns true.

isSetCustomerId() method

Syntax

```
boolean isSetCustomerId ()
```

Returns

false if the *customerId* is null. Otherwise, it returns true.

Usage

Checks whether the

customerId

of this instance is set.

false if the *customerId* is null. Otherwise, it returns true.

setCredentialPolicy(CredentialPolicy) method

Syntax

```
void setCredentialPolicy ( CredentialPolicy credentialPolicy )
```

setCredentialType(int) method

Syntax

```
void setCredentialType ( int credentialType )
```

setCustomerId(int) method

Syntax

```
void setCustomerId ( int customerId )
```

ExchangeRate class

Syntax

```
public class ExchangeRate
```

Usage

The `ExchangeRate` is a simple lookup table to retrieve the exchange rate between to currencies. The conversion is like: For `fromAmount` of `fromCurrency` you get `rate` to `Amount` of `toCurrency` Implementor's note

If subclassing `ExchangeRate`, make sure that every calculation is based on `getAdjustedRate()`. This class attempts to ensure this by declaring all `convert` and `reverseCalculation` methods `final`.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

ExchangeRatePk class

Syntax

```
public class ExchangeRatePk
```

Usage

The `ExchangeRatePk` class is the embeddable primary key object for an exchange rate.

© 2000-2008 by paybox solutions AG

Revision:1.16.2.1Name:not supported by cvs2svn

ExchangeRatePk() constructor

Syntax

```
ExchangeRatePk ()
```

Usage

Creates a new instance of

```
ExchangeRatePk
```

.

ExchangeRatePk(String, String) constructor

Syntax

```
ExchangeRatePk ( String fromCurr , String toCurr )
```

Parameters

- **fromCurr** – the *from* currency
- **toCurr** – the *to* currency

Usage

Creates a new instance of

ExchangeRatePk

.

equals(Object) method

Syntax

boolean equals (Object *o*)

Parameters

- **o** – the Object to compare with

Returns

true if this class is assignable from o's class and has the same PK values as this instance

Usage

Indicates if another object is

equal to

this one.

true if this class is assignable from o's class and has the same PK values as this instance

getFromCurrency() method

Syntax

Currency getFromCurrency ()

Returns

the source Currency

Usage

Returns the currency to convert money from.

the source Currency

getToCurrency() method

Syntax

Currency getToCurrency ()

Returns

the target Currency

Usage

Returns the currency to convert money into.

the target Currency

hashCode() method

Syntax

int hashCode ()

Returns

a hash value for this instance

Usage

Returns a hash value.

a hash value for this instance

isSetFromCurrency() method

Syntax

boolean isSetFromCurrency ()

Returns

false if the dbFromCurrency is null. Otherwise, it returns true.

Usage

Checks whether the

dbFromCurrency

of this instance is set.

false if the dbFromCurrency is null. Otherwise, it returns true.

*isSetToCurrency() method***Syntax**

```
boolean isSetToCurrency ()
```

Returns

false if the dbToCurrency is null. Otherwise, it returns true.

Usage

Checks whether the

dbToCurrency

of this instance is set.

false if the dbToCurrency is null. Otherwise, it returns true.

*toString() method***Syntax**

```
String toString ()
```

Returns

a String representing this instance

Usage

Returns a string representation of this object.

a String representing this instance

*ExchangeRate() constructor***Syntax**

```
ExchangeRate ()
```

Usage

Creates a new instance of

ExchangeRate

.

ExchangeRate(ExchangeRatePk) constructor

Syntax

`ExchangeRate ( ExchangeRatePk id )`

Parameters

- **id** – the PK value

Usage

Creates a new instance of

`ExchangeRate`

with a preset primary key.

getFromAmount() method

Syntax

`long getFromAmount ()`

Returns

the *from* currency amount.

Usage

Returns the

from

currency amount.

the *from* currency amount.

getFromCurrency() method

Syntax

`Currency getFromCurrency ()`

Returns

the source `Currency`

Usage

Returns the currency to convert money from.

the source Currency

getRate() method

Syntax

BigDecimal getRate ()

Returns

the current rate.

Usage

Returns the current rate.

the current rate.

getToAmount() method

Syntax

long getToAmount ()

Returns

the *to* currency amount.

Usage

Returns the

to

currency amount.

the *to* currency amount.

getToCurrency() method

Syntax

Currency getToCurrency ()

Returns

the target Currency

Usage

Returns the currency to convert money into.

the target Currency

isSetFromAmount() method

Syntax

boolean isSetFromAmount ()

Returns

false if the fromAmount is null. Otherwise, it returns true.

Usage

Checks whether the

fromAmount

of this instance is set.

false if the fromAmount is null. Otherwise, it returns true.

isSetFromCurrency() method

Syntax

boolean isSetFromCurrency ()

Returns

false if the id or the id's dbFromCurrency is null. Otherwise, it returns true.

Usage

Checks whether the

dbFromCurrency

of this instance's PK is set.

false if the id or the id's dbFromCurrency is null. Otherwise, it returns true.

isSetRate() method

Syntax

boolean isSetRate ()

Returns

false if the rate is null. Otherwise, it returns true.

Usage

Checks whether the

rate

of this instance is set.

false if the rate is null. Otherwise, it returns true.

isSetToAmount() method

Syntax

```
boolean isSetToAmount ()
```

Returns

false if the toAmount is null. Otherwise, it returns true.

Usage

Checks whether the

toAmount

of this instance is set.

false if the toAmount is null. Otherwise, it returns true.

isSetToCurrency() method

Syntax

```
boolean isSetToCurrency ()
```

Returns

false if the id or the id's dbToCurrency is null. Otherwise, it returns true.

Usage

Checks whether the

dbToCurrency

of this instance's PK is set.

false if the id or the id's dbToCurrency is null. Otherwise, it returns true.

setFromAmount(long) method

Syntax

void setFromAmount (long *fromAmount*)

Parameters

- **fromAmount** – the *fromAmount* value to set

Usage

Sets the

fromAmount

of this instance.

setRate(BigDecimal) method

Syntax

void setRate (BigDecimal *rate*)

Parameters

- **rate** – the *rate* value to set

Usage

Sets the

rate

of this instance.

setToAmount(long) method

Syntax

void setToAmount (long *toAmount*)

Parameters

- **toAmount** – the *toAmount* value to set

Usage

Sets the

toAmount

of this instance.

ExchangeRateHistory class

Syntax

```
public class ExchangeRateHistory
```

Usage

The `ExchangeRateHistory` stores the history of exchange rates between currencies and their valid date.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

ExchangeRateHistory() constructor

Syntax

```
ExchangeRateHistory ()
```

Usage

Creates a new instance of

```
ExchangeRateHistory
```

.

getFromAmount() method

Syntax

```
long getFromAmount ()
```

getFromCurrency() method

Syntax

```
java.util.Currency getFromCurrency ()
```

getRate() method

Syntax

```
BigDecimal getRate ()
```

getToAmount() method

Syntax

```
long getToAmount ()
```

getToCurrency() method

Syntax

```
java.util.Currency getToCurrency ()
```

isSetFromAmount() method

Syntax

```
boolean isSetFromAmount ()
```

Returns

false if the fromAmount is null. Otherwise, it returns true.

Usage

Checks whether the

fromAmount

of this instance is set.

false if the fromAmount is null. Otherwise, it returns true.

isSetFromCurrency() method

Syntax

```
boolean isSetFromCurrency ()
```

Returns

false if the dbFromCurrency is null. Otherwise, it returns true.

Usage

Checks whether the

dbFromCurrency

of this instance is set.

false if the dbFromCurrency is null. Otherwise, it returns true.

*isSetRate() method***Syntax**

```
boolean isSetRate ()
```

Returns

false if the rate is null. Otherwise, it returns true.

Usage

Checks whether the

rate

of this instance is set.

false if the rate is null. Otherwise, it returns true.

*isSetToAmount() method***Syntax**

```
boolean isSetToAmount ()
```

Returns

false if the toAmount is null. Otherwise, it returns true.

Usage

Checks whether the

toAmount

of this instance is set.

false if the toAmount is null. Otherwise, it returns true.

*isSetToCurrency() method***Syntax**

```
boolean isSetToCurrency ()
```

Returns

false if the dbToCurrency is null. Otherwise, it returns true.

Usage

Checks whether the

dbToCurrency

of this instance is set.

false if the dbToCurrency is null. Otherwise, it returns true.

setFromAmount(long) method

Syntax

```
void setFromAmount ( long fromAmount )
```

setFromCurrency(Currency) method

Syntax

```
void setFromCurrency ( java.util.Currency fromCurrency )
```

setRate(BigDecimal) method

Syntax

```
void setRate ( BigDecimal rate )
```

setToAmount(long) method

Syntax

```
void setToAmount ( long toAmount )
```

setToCurrency(Currency) method

Syntax

```
void setToCurrency ( java.util.Currency toCurrency )
```

FeeSet class

Syntax

```
public class FeeSet
```

Usage

The representation of fee sets that can be applied to single customers, customer types or orgunits. Fee Sets are incremental, meaning that up to three fee sets are considered for a customer (from Customers, Customer Type, and OrgUnit), with the one from Customres

having the highest priority and overwriting settings for individual fee types from other Fee Sets.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

FeeSet() constructor

Syntax

```
FeeSet ()
```

Usage

Creates a new instance of

```
FeeSet
```

.

getName() method

Syntax

```
String getName ()
```

isIndividual() method

Syntax

```
boolean isIndividual ()
```

isSetIndividual() method

Syntax

```
boolean isSetIndividual ()
```

Returns

false if the dbIndividual is null. Otherwise, it returns true.

Usage

Checks whether the

dbIndividual

of this instance is set.

false if the `dbIndividual` is null. Otherwise, it returns true.

isSetName() method

Syntax

```
boolean isSetName ()
```

Returns

false if the name is null. Otherwise, it returns true.

Usage

Checks whether the

name

of this instance is set.

false if the name is null. Otherwise, it returns true.

setIndividual(boolean) method

Syntax

```
void setIndividual ( boolean individual )
```

setName(String) method

Syntax

```
void setName ( String name )
```

FeeType class

Syntax

```
public class FeeType
```

Usage

This entity represents a fee type allowing to configure fees that become due for a transaction.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

FeeType() constructor

Syntax

`FeeType ()`

Usage

Creates a new instance of

`FeeType`

.

FeeType(int) constructor

Syntax

`FeeType ( int id )`

Parameters

- **id** – the PK value

Usage

Creates a new instance of

`FeeType`

with a preset primary key.

getBeneficiary() method

Syntax

`Customer getBeneficiary ()`

Returns

the beneficiary value

Usage

Returns the

Money merchant

that will receive (or pay) the fee amount.

the beneficiary value

isCommissionFee() method

Syntax

```
boolean isCommissionFee ()
```

Returns

true if fees of this type should be considered as commission; false otherwise

Usage

Indicates if the transaction processing should consider fees as commission

true if fees of this type should be considered as commission; false otherwise

isIncludeVat() method

Syntax

```
boolean isIncludeVat ()
```

Returns

true if the VAT should be included; false if it should be added on-top

Usage

Indicates if the VAT for this fee type should be included or calculated on top.

true if the VAT should be included; false if it should be added on-top

isProcessWithTransaction() method

Syntax

```
boolean isProcessWithTransaction ()
```

Returns

true if fees of this type should be considered by transaction processing; false otherwise

Usage

Indicates if the transaction processing should consider fees of this type.

true if fees of this type should be considered by transaction processing; false otherwise

isSetBeneficiary() method

Syntax

```
boolean isSetBeneficiary ()
```

Returns

false if the beneficiary is null. Otherwise, it returns true.

Usage

Checks whether the

beneficiary

of this instance is set.

false if the beneficiary is null. Otherwise, it returns true.

isSetCommissionFee() method

Syntax

```
boolean isSetCommissionFee ()
```

Returns

false if the dbcommission is null. Otherwise, it returns true.

Usage

Checks whether the

dbcommission

of this instance is set.

false if the dbcommission is null. Otherwise, it returns true.

isSetIncludeVat() method

Syntax

```
boolean isSetIncludeVat ()
```

Returns

false if the dbincludeVat is null. Otherwise, it returns true.

Usage

Checks whether the

dbincludeVat

of this instance is set.

false if the dbincludeVat is null. Otherwise, it returns true.

isSetProcessWithTransaction() method

Syntax

boolean isSetProcessWithTransaction ()

Returns

false if the dbprocessWithTxn is null. Otherwise, it returns true.

Usage

Checks whether the

dbprocessWithTxn

of this instance is set.

false if the dbprocessWithTxn is null. Otherwise, it returns true.

setBeneficiary(Customer) method

Syntax

void setBeneficiary (Customer *beneficiary*)

Parameters

- **beneficiary** – the beneficiary value to set

Usage

Sets the

beneficiary

of this instance.

setCommissionFee(boolean) method

Syntax

void setCommissionFee (boolean *commission*)

Parameters

- **commission** – the `commission` value to set

Usage

Sets the

commission

of this instance.

setIncludeVat(boolean) method

Syntax

```
void setIncludeVat ( boolean includeVat )
```

Parameters

- **includeVat** – the `includeVat` value to set

Usage

Sets the

includeVat

of this instance.

setProcessWithTransaction(boolean) method

Syntax

```
void setProcessWithTransaction ( boolean processWithTransaction )
```

Parameters

- **processWithTransaction** – the `processWithTransaction` value to set

Usage

Sets the

processWithTransaction

of this instance.

FeeTypeFeeSetScale class

Syntax

```
public class FeeTypeFeeSetScale
```

Usage

The ExchangeRate is a simple lookup table to retrieve the exchange rate between to currencies. The conversion is like: For fromAmount of fromCurrency you get rate toAmount of toCurrency Implementor's note

If subclassing ExchangeRate, make sure that every calculation is based on getAdjustedRate(). This class attempts to ensure this by declaring all convert and reverseCalculation methods final.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

FeeTypeFeeSetScalePk class

Syntax

```
public class FeeTypeFeeSetScalePk
```

Usage

The ExchangeRatePk class is the embeddable primary key object for an exchange rate.

© 2000-2008 by paybox solutions AG

Revision:36127Name:not supported by cvs2svn

FeeTypeFeeSetScalePk() constructor

Syntax

```
FeeTypeFeeSetScalePk ()
```

Usage

Creates a new instance of

ExchangeRatePk

.

FeeTypeFeeSetScalePk(FeeType, FeeSet) constructor

Syntax

FeeTypeFeeSetScalePk (FeeType *feeType* , FeeSet *feeSet*)

Parameters

- **fromCurr** – the *from* currency
- **toCurr** – the *to* currency

Usage

Creates a new instance of

ExchangeRatePk

.

equals(Object) method

Syntax

boolean equals (Object *o*)

Parameters

- **o** – the Object to compare with

Returns

true if this class is assignable from o's class and has the same PK values as this instance

Usage

Indicates if another object is

equal to

this one.

true if this class is assignable from o's class and has the same PK values as this instance

getFeeSet() method

Syntax

FeeSet getFeeSet ()

getFeeType() method

Syntax

FeeType getFeeType ()

hashCode() method

Syntax

int hashCode ()

Returns

a hash value for this instance

Usage

Returns a hash value.

a hash value for this instance

isSetFeeSet() method

Syntax

boolean isSetFeeSet ()

Returns

false if the feeSet is null. Otherwise, it returns true.

Usage

Checks whether the

feeSet

of this instance is set.

false if the feeSet is null. Otherwise, it returns true.

isSetFeeType() method

Syntax

boolean isSetFeeType ()

Returns

false if the feeType is null. Otherwise, it returns true.

Usage

Checks whether the

feeType

of this instance is set.

false if the *feeType* is null. Otherwise, it returns true.

toString() method

Syntax

String toString ()

Returns

a String representing this instance

Usage

Returns a string representation of this object.

a String representing this instance

FeeTypeFeeSetScale() constructor

Syntax

FeeTypeFeeSetScale ()

Usage

Creates a new instance of

FeeTypeFeeSetScale

.

FeeTypeFeeSetScale(FeeTypeFeeSetScalePk) constructor

Syntax

FeeTypeFeeSetScale (FeeTypeFeeSetScalePk *id*)

Parameters

- **id** – the PK value

Usage

Creates a new instance of
`FeeTypeFeeSetScale`
with a preset primary key.

getFeeSet() method

Syntax

`FeeSet getFeeSet ()`

getFeeType() method

Syntax

`FeeType getFeeType ()`

getScale() method

Syntax

`Scale getScale ()`

isSetFeeSet() method

Syntax

`boolean isSetFeeSet ()`

Returns

false if the id or the id's feeSet is null. Otherwise, it returns true.

Usage

Checks whether the

feeSet

of this instance's PK is set.

false if the id or the id's feeSet is null. Otherwise, it returns true.

isSetFeeType() method

Syntax

`boolean isSetFeeType ()`

Returns

false if the id or the id's feeType is null. Otherwise, it returns true.

Usage

Checks whether the

feeType

of this instance's PK is set.

false if the id or the id's feeType is null. Otherwise, it returns true.

isSetScale() method

Syntax

```
boolean isSetScale ()
```

Returns

false if the scale is null. Otherwise, it returns true.

Usage

Checks whether the

scale

of this instance is set.

false if the scale is null. Otherwise, it returns true.

setScale(Scale) method

Syntax

```
void setScale ( Scale scale )
```

File class**Syntax**

```
public class File
```

Usage

The File entity represents a file either generated by mobiliser process, like the AMS file export or a file imported into the mobiliser system (e.g. for reconciliation purposes).

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

File() constructor

Syntax

File ()

Usage

Creates a new instance of

TcmDtinFile

.

getBatchRun() method

Syntax

BatchRun getBatchRun ()

getFileName() method

Syntax

String getFileName ()

getFileStatus() method

Syntax

int getFileStatus ()

getFileType() method

Syntax

int getFileType ()

getLocation() method

Syntax

String getLocation ()

getNumRecords() method

Syntax

Long getNumRecords ()

isSetBatchRun() method

Syntax

boolean isSetBatchRun ()

isSetFileName() method

Syntax

boolean isSetFileName ()

isSetFileStatus() method

Syntax

boolean isSetFileStatus ()

isSetFileType() method

Syntax

boolean isSetFileType ()

isSetLocation() method

Syntax

boolean isSetLocation ()

isSetNumRecords() method

Syntax

boolean isSetNumRecords ()

setBatchRun(BatchRun) method

Syntax

void setBatchRun (BatchRun *batchRun*)

setFileName(String) method

Syntax

```
void setFileName ( String fileName )
```

setStatus(int) method

Syntax

```
void setStatus ( int fileStatus )
```

setFileType(int) method

Syntax

```
void setFileType ( int fileType )
```

setLocation(String) method

Syntax

```
void setLocation ( String location )
```

setNumRecords(Long) method

Syntax

```
void setNumRecords ( Long numRecords )
```

IndividualRestriction class

Syntax

```
public class IndividualRestriction
```

Usage

The IndividualRestriction entity represents an allowance configuration for a certain payer/payee/use case combination (whether the defined combination of the three is allowed in a transaction or not).

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

IndividualRestriction() constructor

Syntax

IndividualRestriction ()

Usage

Creates a new instance of

IndividualRestriction

.

getPayee() method

Syntax

Customer getPayee ()

Returns

the *payee* Customer

Usage

Returns the associated payee.

the *payee* Customer

getPayer() method

Syntax

Customer getPayer ()

Returns

the *payer* Customer

Usage

Returns the associated payer.

the *payer* Customer

getUseCase() method

Syntax

UseCase getUseCase ()

Returns

the affected UseCase

Usage

Returns the use case affected by this restriction.

the affected UseCase

isAllowed() method

Syntax

`boolean isAllowed ()`

Returns

`true` if this rule is a permission; `false` if it is a prohibition

Usage

Indicates if this is a prohibition or a permission.

`true` if this rule is a permission; `false` if it is a prohibition

isSetAllowed() method

Syntax

`boolean isSetAllowed ()`

Returns

`false` if the `dbAllowed` is `null` . Otherwise, it returns `true`.

Usage

Checks whether the

dbAllowed

of this instance is set.

`false` if the `dbAllowed` is `null` . Otherwise, it returns `true`.

isSetPayee() method

Syntax

`boolean isSetPayee ()`

Returns

false if the payee is null. Otherwise, it returns true.

Usage

Checks whether the

payee

of this instance is set.

false if the payee is null. Otherwise, it returns true.

isSetPayer() method

Syntax

```
boolean isSetPayer ()
```

Returns

false if the payer is null. Otherwise, it returns true.

Usage

Checks whether the

payer

of this instance is set.

false if the payer is null. Otherwise, it returns true.

isSetUseCase() method

Syntax

```
boolean isSetUseCase ()
```

Returns

false if the useCase is null. Otherwise, it returns true.

Usage

Checks whether the

useCase

of this instance is set.

false if the useCase is null. Otherwise, it returns true.

setAllowed(boolean) method

Syntax

void setAllowed (boolean *allowed*)

Parameters

- **allowed** – the allowed value to set

Usage

Sets the

allowed

of this instance.

setPayee(Customer) method

Syntax

void setPayee (Customer *payee*)

Parameters

- **payee** – the payee value to set

Usage

Sets the

payee

of this instance.

setPayer(Customer) method

Syntax

void setPayer (Customer *payer*)

Parameters

- **payer** – the payer value to set

Usage

Sets the

payer

of this instance.

setUseCase(UseCase) method

Syntax

```
void setUseCase ( UseCase useCase )
```

Parameters

- **useCase** – the useCase value to set

Usage

Sets the

useCase

of this instance.

InterestBalanceSnapshot class

Syntax

```
public class InterestBalanceSnapshot
```

Usage

The InterestBalanceSnapshot entity is used to track the balance of a payment instrument.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

InterestBalanceSnapshotPk class

Syntax

```
public class InterestBalanceSnapshotPk
```

InterestBalanceSnapshotPk() constructor

Syntax

```
InterestBalanceSnapshotPk ()
```

Usage

Creates a new instance of

InterestBalanceSnapshotPk

.

InterestBalanceSnapshotPk(PaymentInstrument, Date) constructor

Syntax

InterestBalanceSnapshotPk (PaymentInstrument *pi* , Date
balanceDay)

Parameters

- **pi** –
- **balanceDay** –

Usage

Creates a new instance of

InterestBalanceSnapshotPk

.

equals(Object) method

Syntax

boolean equals (Object *o*)

Parameters

- **o** – the Object to compare with

Returns

true if this class is assignable from o's class and has the same PK values as this instance

Usage

Indicates if another object is

equal to

this one.

true if this class is assignable from o's class and has the same PK values as this instance

getBalanceDay() method

Syntax

Date getBalanceDay ()

getPi() method

Syntax

PaymentInstrument getPi ()

hashCode() method

Syntax

int hashCode ()

Returns

a hash value for this instance

Usage

Returns a hash value.

a hash value for this instance

setBalanceDay(Date) method

Syntax

void setBalanceDay (Date *balanceDay*)

setPi(PaymentInstrument) method

Syntax

void setPi (PaymentInstrument *pi*)

toString() method

Syntax

String toString ()

Returns

a String representing this instance

Usage

Returns a string representation of this object.

a `String` representing this instance

InterestBalanceSnapshot(InterestBalanceSnapshotPk) constructor

Syntax

`InterestBalanceSnapshot ( InterestBalanceSnapshotPk id )`

Parameters

- **id** – the PK value

Usage

Creates a new instance of

`InterestBalanceSnapshot`

with a preset primary key.

InterestBalanceSnapshot() constructor

Syntax

`InterestBalanceSnapshot ()`

Usage

Creates a new instance of

`InterestBalanceSnapshot`

.

getBalance() method

Syntax

`long getBalance ()`

getBatchRunId() method

Syntax

`Long getBatchRunId ()`

getInterest() method

Syntax

BigDecimal getInterest ()

getInterestCalculationStatus() method

Syntax

InterestCalculationStatus getInterestCalculationStatus ()

setBalance(long) method

Syntax

void setBalance (long *balance*)

setBatchRunId(Long) method

Syntax

void setBatchRunId (Long *batchRunId*)

setInterest(BigDecimal) method

Syntax

void setInterest (BigDecimal *interest*)

setInterestCalculationStatus(InterestCalculationStatus) method

Syntax

void setInterestCalculationStatus (InterestCalculationStatus *interestCalculationStatus*)

InterestCalculationStatus() enumeration

Enum Constant Summary

- **UNUSED** –
- **IN_PROGRESS** –
- **USED** –

InterestCalculation class

Syntax

```
public class InterestCalculation
```

Usage

The InterestCalculation entity is used to track the calculated interest of a payment instrument.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getBatchRunId() method

Syntax

```
Long getBatchRunId ()
```

getClearingStatus() method

Syntax

```
InterestClearingStatus getClearingStatus ()
```

getClearingTxnId() method

Syntax

```
Long getClearingTxnId ()
```

getCutoffDate() method

Syntax

```
java.util.Date getCutoffDate ()
```

getIdPi() method

Syntax

```
long getIdPi ()
```

getInterestAmount() method

Syntax

```
long getInterestAmount ()
```


getPreAllocationTxnId() method

Syntax

Long getPreAllocationTxnId ()

getTaxTxnId() method

Syntax

Long getTaxTxnId ()

isCredit() method

Syntax

boolean isCredit ()

setBatchRunId(Long) method

Syntax

void setBatchRunId (Long *batchRunId*)

setClearingStatus(InterestClearingStatus) method

Syntax

void setClearingStatus (InterestClearingStatus *clearingStatus*)

setClearingTxnId(Long) method

Syntax

void setClearingTxnId (Long *clearingTxnId*)

setCredit(boolean) method

Syntax

void setCredit (boolean *credit*)

setCutoffDate(Date) method

Syntax

void setCutoffDate (java.util.Date *cutoffDate*)

setIdPi(long) method

Syntax

void setIdPi (long *idPi*)

setInterestAmount(long) method

Syntax

void setInterestAmount (long *interestAmount*)

setPreAllocationTxnId(Long) method

Syntax

void setPreAllocationTxnId (Long *preAllocationTxnId*)

setTaxTxnId(Long) method

Syntax

void setTaxTxnId (Long *taxTxnId*)

InterestClearingStatus() enumeration

Enum Constant Summary

- **PENDING_PREBOOKING** –
- **UNCLEARED** –
- **IN_PROGRESS** –
- **CLEARED** –

clearingTxnId variable

Syntax

Long clearingTxnId

preAllocationTxnId variable

Syntax

Long preAllocationTxnId

taxTxnId variable

Syntax

Long taxTxnId

InterestConfiguration class

Syntax

```
public class InterestConfiguration
```

Usage

The `InterestConfiguration` entity represents the interest configuration of a payment instrument.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

InterestConfiguration() constructor

Syntax

```
InterestConfiguration ()
```

Usage

Creates a new instance of

```
InterestConfiguration
```

.

InterestConfiguration(long) constructor

Syntax

```
InterestConfiguration ( long piId )
```

Parameters

- **piId** – the PK value

Usage

Creates a new instance of

```
InterestConfiguration
```

with a preset primary key.

getLastInterestCalculationCutoffDate() method

Syntax

Date getLastInterestCalculationCutoffDate ()

getLastInterestClearingCutoffDate() method

Syntax

Date getLastInterestClearingCutoffDate ()

getPi() method

Syntax

PaymentInstrument getPi ()

getProduct() method

Syntax

InterestProduct getProduct ()

setLastInterestCalculationCutoffDate(Date) method

Syntax

void setLastInterestCalculationCutoffDate (Date
lastInterestCalculationCutoffDate)

setLastInterestClearingCutoffDate(Date) method

Syntax

void setLastInterestClearingCutoffDate (Date
lastInterestClearingCutoffDate)

setPi(PaymentInstrument) method

Syntax

void setPi (PaymentInstrument *pi*)

setProduct(InterestProduct) method

Syntax

void setProduct (InterestProduct *product*)

lastInterestCalculationCutoffDate variable

Syntax

Date lastInterestCalculationCutoffDate

lastInterestClearingCutoffDate variable

Syntax

Date lastInterestClearingCutoffDate

pi variable

Syntax

PaymentInstrument pi

product variable

Syntax

InterestProduct product

InterestPeriod class

Syntax

```
public class InterestPeriod
```

Usage

The InterestPeriod entity represents an interest period.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

InterestPeriod(String) constructor

Syntax

```
InterestPeriod ( String id )
```

Parameters

- **id** – the PK value

Usage

Creates a new instance of

`InterestPeriod`

with a preset primary key.

`InterestPeriod()` constructor

Syntax

`InterestPeriod ()`

Usage

Creates a new instance of

`InterestPeriod`

.

`getCutoffCron()` method

Syntax

`String getCutoffCron ()`

`getDescription()` method

Syntax

`String getDescription ()`

`setCutoffCron(String)` method

Syntax

`void setCutoffCron ( String cutoffCron )`

`setDescription(String)` method

Syntax

`void setDescription ( String description )`

InterestProduct class

Syntax

`public class InterestProduct`

Usage

The InterestProduct entity represents an interest product.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getCalculationInterestPeriod() method**Syntax**

```
InterestPeriod getCalculationInterestPeriod ()
```

getClearingInterestPeriod() method**Syntax**

```
InterestPeriod getClearingInterestPeriod ()
```

getCreditCalculationMethod() method**Syntax**

```
String getCreditCalculationMethod ()
```

getCreditClearingAccountPi() method**Syntax**

```
PaymentInstrument getCreditClearingAccountPi ()
```

getCreditScale() method**Syntax**

```
Scale getCreditScale ()
```

getCreditWithholdingTaxAccountPi() method**Syntax**

```
PaymentInstrument getCreditWithholdingTaxAccountPi ()
```

getDebitClearingAccountPi() method**Syntax**

```
PaymentInstrument getDebitClearingAccountPi ()
```

getDebitScale() method

Syntax

Scale getDebitScale ()

getExpenseAccountPi() method

Syntax

PaymentInstrument getExpenseAccountPi ()

getIncomeAccountPi() method

Syntax

PaymentInstrument getIncomeAccountPi ()

getProductName() method

Syntax

String getProductName ()

getSnapshotMethod() method

Syntax

String getSnapshotMethod ()

getWithholdingTaxScale() method

Syntax

Scale getWithholdingTaxScale ()

isActive() method

Syntax

boolean isActive ()

Returns

true if the product is active; false otherwise

Usage

Indicates if this product is active

true if the product is active; false otherwise

setActive(boolean) method

Syntax

```
void setActive ( boolean active )
```

Parameters

- **active** – the active value to set

Usage

Sets the

active

of this instance.

setCalculationInterestPeriod(InterestPeriod) method

Syntax

```
void setCalculationInterestPeriod ( InterestPeriod  
calculationInterestPeriod )
```

setClearingInterestPeriod(InterestPeriod) method

Syntax

```
void setClearingInterestPeriod ( InterestPeriod  
clearingInterestPeriod )
```

setCreditCalculationMethod(String) method

Syntax

```
void setCreditCalculationMethod ( String creditCalculationMethod )
```

setCreditClearingAccountPi(PaymentInstrument) method

Syntax

```
void setCreditClearingAccountPi ( PaymentInstrument  
creditClearingAccountPi )
```

setCreditScale(Scale) method

Syntax

void setCreditScale (Scale *creditScale*)

setCreditWithholdingTaxAccountPi(PaymentInstrument) method

Syntax

void setCreditWithholdingTaxAccountPi (PaymentInstrument
creditWithholdingTaxAccountPi)

setDebitClearingAccountPi(PaymentInstrument) method

Syntax

void setDebitClearingAccountPi (PaymentInstrument
debitClearingAccountPi)

setDebitScale(Scale) method

Syntax

void setDebitScale (Scale *debitScale*)

setExpenseAccountPi(PaymentInstrument) method

Syntax

void setExpenseAccountPi (PaymentInstrument *expenseAccountPi*)

setIncomeAccountPi(PaymentInstrument) method

Syntax

void setIncomeAccountPi (PaymentInstrument *incomeAccountPi*)

setProductName(String) method

Syntax

void setProductName (String *productName*)

setSnapshotMethod(String) method

Syntax

void setSnapshotMethod (String *snapshotMethod*)

*setWithholdingTaxScale(Scale) method***Syntax**

```
void setWithholdingTaxScale ( Scale withholdingTaxScale )
```

InterestProductMapping class**Syntax**

```
public class InterestProductMapping
```

Usage

The InterestProductMapping entity maps a customer type and a payment instrument type to an interest product.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

*getCustomerType() method***Syntax**

```
CustomerType getCustomerType ()
```

*getOrgUnit() method***Syntax**

```
OrgUnit getOrgUnit ()
```

*getPiType() method***Syntax**

```
Type getPiType ()
```

*getProduct() method***Syntax**

```
InterestProduct getProduct ()
```

setCustomerType(CustomerType) method

Syntax

```
void setCustomerType ( CustomerType customerType )
```

setOrgUnit(OrgUnit) method

Syntax

```
void setOrgUnit ( OrgUnit orgUnit )
```

setPiType(Type) method

Syntax

```
void setPiType ( Type piType )
```

setProduct(InterestProduct) method

Syntax

```
void setProduct ( InterestProduct product )
```

InvoiceType class

Syntax

```
public class InvoiceType
```

Usage

The InvoiceType represents an invoice type.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

InvoiceType() constructor

Syntax

```
InvoiceType ()
```

Usage

Creates a new instance of

```
InvoiceType
```

getAmount() method

Syntax

```
Long getAmount ()
```

Returns

the invoice type's default amount

Usage

the invoice type's default amount

getAttributes() method

Syntax

```
Set< InvoiceTypeAttribute > getAttributes ()
```

Returns

the attributes

Usage

the attributes

getAutoPayDelay() method

Returns the timeframe period or the invoice type's auto-pay delay according to the W3C XSD timeframe data type, if empty there is no reoccurrence.

Syntax

```
Duration getAutoPayDelay ()
```

Returns

the autoPayDelay

Usage

the autoPayDelay

getBillingCycle() method

Returns the timeframe period or the invoice type's billing cycle according to the W3C XSD timeframe data type, if empty there is no reoccurrence.

Syntax

Duration getBillingCycle ()

Returns

the billingCycle

Usage

the billingCycle

getCurrency() method

Syntax

Currency getCurrency ()

Returns

the invoice type's default Currency

Usage

the invoice type's default Currency

getDefaultPi() method

Syntax

PaymentInstrument getDefaultPi ()

Returns

the defaultPi

Usage

the defaultPi

getFetchCycle() method

Syntax

Duration getFetchCycle ()

getGroupId() method

Syntax

```
int getGroupId ()
```

Returns

the groupId

Usage

the groupId

getHandlerType() method

Syntax

```
Integer getHandlerType ()
```

Returns

the handlerType

Usage

the handlerType

getInitialDelay() method

Returns the timeframe period or the invoice type's initial delay according to the W3C XSD timeframe data type, if empty there is no reoccurrence.

Syntax

```
Duration getInitialDelay ()
```

Returns

the initialDelay

Usage

the initialDelay

getInvoiceConfigurationReferencePattern() method

Syntax

```
String getInvoiceConfigurationReferencePattern ()
```

com.sybase365.mobiliser.money package

Returns

the invoiceConfigurationReferencePattern

Usage

the invoiceConfigurationReferencePattern

getInvoiceReferencePattern() method

Syntax

String getInvoiceReferencePattern ()

Returns

the invoiceReferencePattern

Usage

the invoiceReferencePattern

getIssuer() method

Syntax

Customer getIssuer ()

Returns

the issuer

Usage

the issuer

getLastFetch() method

Syntax

Date getLastFetch ()

getName() method

Syntax

String getName ()

getReference() method

Syntax

String getReference ()

Returns

the reference

Usage

the reference

getUseCase() method

Syntax

UseCase getUseCase ()

Returns

the useCase

Usage

the useCase

isActive() method

Syntax

boolean isActive ()

isAmountFixed() method

Syntax

boolean isAmountFixed ()

Returns

the dbAmountFixed

Usage

the dbAmountFixed

isCustomerAddsInvoice() method

Syntax

boolean isCustomerAddsInvoice ()

Returns

the dbCustomerAddsInvoice

Usage

the dbCustomerAddsInvoice

isCustomerAddsInvoiceConfiguration() method

Syntax

boolean isCustomerAddsInvoiceConfiguration ()

Returns

the dbCustomerAddsInvoiceConfiguration

Usage

the dbCustomerAddsInvoiceConfiguration

isIssuerAddsInvoice() method

Syntax

boolean isIssuerAddsInvoice ()

Returns

the dbIssuerAddsInvoice

Usage

the dbIssuerAddsInvoice

isIssuerAddsInvoiceConfiguration() method

Syntax

boolean isIssuerAddsInvoiceConfiguration ()

Returns

the dbIssuerAddsInvoiceConfiguration

Usage

the dbIssuerAddsInvoiceConfiguration

isSetActive() method

Syntax

```
boolean isSetActive ()
```

Returns

false if the dbActive is null. Otherwise, it returns true.

Usage

Checks whether the

dbActive

of this instance is set.

false if the dbActive is null. Otherwise, it returns true.

isSetAmount() method

Syntax

```
boolean isSetAmount ()
```

Returns

false if the amount is null. Otherwise, it returns true.

Usage

Checks whether the

amount

of this instance is set.

false if the amount is null. Otherwise, it returns true.

isSetAttributes() method

Syntax

```
boolean isSetAttributes ()
```

Returns

false if the attributes is null. Otherwise, it returns true.

Usage

Checks whether the

attributes

of this instance is set.

false if the attributes is null. Otherwise, it returns true.

isSetAutoPayDelay() method

Syntax

boolean isSetAutoPayDelay ()

Returns

false if the autoPayDelay is null. Otherwise, it returns true.

Usage

Checks whether the

autoPayDelay

of this instance is set.

false if the autoPayDelay is null. Otherwise, it returns true.

isSetBillingCycle() method

Syntax

boolean isSetBillingCycle ()

Returns

false if the billingCycle is null. Otherwise, it returns true.

Usage

Checks whether the

billingCycle

of this instance is set.

false if the billingCycle is null. Otherwise, it returns true.

*isSetCurrency() method***Syntax**

```
boolean isSetCurrency ()
```

Returns

false if the dbCurrency is null. Otherwise, it returns true.

Usage

Checks whether the

dbCurrency

of this instance is set.

false if the dbCurrency is null. Otherwise, it returns true.

*isSetDefaultPi() method***Syntax**

```
boolean isSetDefaultPi ()
```

Returns

false if the defaultPi is null . Otherwise, it returns true.

Usage

Checks whether the

defaultPi

of this instance is set.

false if the defaultPi is null . Otherwise, it returns true.

*isSetFetchCycle() method***Syntax**

```
boolean isSetFetchCycle ()
```

Returns

false if the fetchCycle is null. Otherwise, it returns true.

Usage

Checks whether the

fetchCycle

of this instance is set.

false if the `fetchCycle` is null. Otherwise, it returns true.

isSetGroupId() method

Syntax

```
boolean isSetGroupId ()
```

Returns

false if the `groupId` is null. Otherwise, it returns true.

Usage

Checks whether the

groupId

of this instance is set.

false if the `groupId` is null. Otherwise, it returns true.

isSetHandlerType() method

Syntax

```
boolean isSetHandlerType ()
```

isSetInitialDelay() method

Syntax

```
boolean isSetInitialDelay ()
```

Returns

false if the `initialDelay` is null. Otherwise, it returns true.

Usage

Checks whether the

initialDelay

of this instance is set.

false if the `initialDelay` is null. Otherwise, it returns true.

isSetInvoiceConfigurationReferencePattern() method

Syntax

```
boolean isSetInvoiceConfigurationReferencePattern ()
```

Returns

the `invoiceConfigurationReferencePattern`

Usage

the `invoiceConfigurationReferencePattern`

isSetInvoiceReferencePattern() method

Syntax

```
boolean isSetInvoiceReferencePattern ()
```

Returns

the `invoiceReferencePattern`

Usage

the `invoiceReferencePattern`

isSetIssuer() method

Syntax

```
boolean isSetIssuer ()
```

Returns

false if the `issuer` is null. Otherwise, it returns true.

Usage

Checks whether the

issuer

of this instance is set.

false if the `issuer` is null. Otherwise, it returns true.

isSetLastFetch() method

Syntax

boolean isSetLastFetch ()

Returns

false if the lastFetch is null . Otherwise, it returns true.

Usage

Checks whether the

lastFetch

of this instance is set.

false if the lastFetch is null . Otherwise, it returns true.

isSetName() method

Syntax

boolean isSetName ()

Returns

false if the name is null. Otherwise, it returns true.

Usage

Checks whether the

name

of this instance is set.

false if the name is null. Otherwise, it returns true.

isSetReference() method

Syntax

boolean isSetReference ()

Returns

the reference

Usage

the reference

isSetUseCase() method

Syntax

```
boolean isSetUseCase ()
```

Returns

false if the useCase is null. Otherwise, it returns true.

Usage

Checks whether the

useCase

of this instance is set.

false if the useCase is null. Otherwise, it returns true.

isSystemAddsInvoice() method

Syntax

```
boolean isSystemAddsInvoice ()
```

Returns

the dbSystemAddsInvoice

Usage

the dbSystemAddsInvoice

setActive(boolean) method

Syntax

```
void setActive ( boolean active )
```

setAmount(Long) method

Syntax

```
void setAmount ( Long amount )
```

Parameters

- **amount** – the default amount to set

setAmountFixed(boolean) method

Syntax

```
void setAmountFixed ( boolean isAmountFixed )
```

Parameters

- **dbSystemAddsInvoice** – the dbSystemAddsInvoice to set

setAttributes(Set< InvoiceTypeAttribute >) method

Syntax

```
void setAttributes ( Set< InvoiceTypeAttribute > attributes )
```

Parameters

- **attributes** – the attributes to set

setAutoPayDelay(Duration) method

Syntax

```
void setAutoPayDelay ( Duration autoPayDelay )
```

Parameters

- **autoPayDelay** – the autoPayDelay to set

setBillingCycle(Duration) method

Syntax

```
void setBillingCycle ( Duration billingCycle )
```

Parameters

- **billingCycle** – the billingCycle to set

setCurrency(Currency) method

Syntax

```
void setCurrency ( Currency currency )
```

Parameters

- **currency** – the currency value to set

setCustomerAddsInvoice(boolean) method

Syntax

```
void setCustomerAddsInvoice ( boolean isCustomerAddsInvoice )
```

Parameters

- **dbCustomerAddsInvoice** – the dbCustomerAddsInvoice to set

setCustomerAddsInvoiceConfiguration(boolean) method

Syntax

```
void setCustomerAddsInvoiceConfiguration ( boolean  
isCustomerAddsInvoiceConfiguration )
```

Parameters

- **dbCustomerAddsInvoiceConfiguration** – the dbCustomerAddsInvoiceConfiguration to set

setDefaultPi(PaymentInstrument) method

Syntax

```
void setDefaultPi ( PaymentInstrument defaultPi )
```

Parameters

- **defaultPi** – the defaultPi to set

setFetchCycle(Duration) method

Syntax

```
void setFetchCycle ( Duration fetchCycle )
```

Parameters

- **setFetchCycle** – the setFetchCycle to set

setGroupId(int) method

Syntax

void setGroupId (int *groupId*)

Parameters

- **groupId** – the groupId to set

setHandlerType(Integer) method

Syntax

void setHandlerType (Integer *handlerType*)

Parameters

- **handlerType** – the handlerType to set

setInitialDelay(Duration) method

Syntax

void setInitialDelay (Duration *initialDelay*)

Parameters

- **initialDelay** – the initialDelay to set

setInvoiceConfigurationReferencePattern(String) method

Syntax

void setInvoiceConfigurationReferencePattern (String *invoiceConfigurationReferencePattern*)

Parameters

- **invoiceConfigurationReferencePattern** – the invoiceConfigurationReferencePattern to set

setInvoiceReferencePattern(String) method

Syntax

void setInvoiceReferencePattern (String *invoiceReferencePattern*)

Parameters

- **invoiceReferencePattern** – the invoiceReferencePattern to set

setIssuer(Customer) method

Syntax

```
void setIssuer ( Customer issuer )
```

Parameters

- **issuer** – the issuer to set

setIssuerAddsInvoice(boolean) method

Syntax

```
void setIssuerAddsInvoice ( boolean isIssuerAddsInvoice )
```

Parameters

- **dbIssuerAddsInvoice** – the dbIssuerAddsInvoice to set

setIssuerAddsInvoiceConfiguration(boolean) method

Syntax

```
void setIssuerAddsInvoiceConfiguration ( boolean  
isIssuerAddsInvoiceConfiguration )
```

Parameters

- **dbIssuerAddsInvoiceConfiguration** – the dbIssuerAddsInvoiceConfiguration to set

setLastFetch(Date) method

Syntax

```
void setLastFetch ( Date lastFetch )
```

setName(String) method

Syntax

```
void setName ( String name )
```

setReference(String) method

Syntax

void setReference (String *reference*)

Parameters

- **reference** – the reference to set

setSystemAddsInvoice(boolean) method

Syntax

void setSystemAddsInvoice (boolean *isSystemAddsInvoice*)

Parameters

- **dbSystemAddsInvoice** – the dbSystemAddsInvoice to set

setUseCase(UseCase) method

Syntax

void setUseCase (UseCase *useCase*)

Parameters

- **useCase** – the useCase to set

InvoiceTypeAttribute class

Syntax

```
public class InvoiceTypeAttribute
```

Usage

The `Attribute` is used for the invoice type attributes.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

InvoiceTypeAttributesPK class

Syntax

```
public class InvoiceTypeAttributesPK
```

Usage

This is the embedded Primary Key

© 2000-2009 by paybox solutions AG

Revision:1.14Name:not supported by cvs2svn

InvoiceTypeAttributesPK(InvoiceType, int) constructor

Constructs a new mapping PK.

Syntax

InvoiceTypeAttributesPK (InvoiceType invoiceType , int attributeTypeId)

Parameters

- **invoiceType** –
- **attribute** –

InvoiceTypeAttributesPK() constructor

The empty constructor.

Syntax

InvoiceTypeAttributesPK ()

equals(Object) method

Syntax

boolean equals (Object o)

Parameters

- **o** – the Object to compare with

Returns

true if this class is assignable from o's class and has the same PK values as this instance

Usage

Indicates if another object is

equal to

this one.

true if this class is assignable from o's class and has the same PK values as this instance

getAttributeTypeId() method

Syntax

```
int getAttributeTypeId ()
```

Returns

the attributeTypeId

Usage

the attributeTypeId

getInvoiceType() method

Syntax

```
InvoiceType getInvoiceType ()
```

Returns

the invoice type

Usage

the invoice type

hashCode() method

Syntax

```
int hashCode ()
```

Returns

a hash value for this instance

Usage

Returns a hash value.

a hash value for this instance

isSetAttributeTypeId() method

Syntax

```
boolean isSetAttributeTypeId ()
```


Returns

false if the `attributeTypeId` is null. Otherwise, it returns true.

Usage

Checks whether the

attributeTypeId

of this instance is set.

false if the `attributeTypeId` is null. Otherwise, it returns true.

isSetInvoiceType() method

Syntax

```
boolean isSetInvoiceType ()
```

Returns

false if the `invoiceType` is null. Otherwise, it returns true.

Usage

Checks whether the

invoiceType

of this instance is set.

false if the `invoiceType` is null. Otherwise, it returns true.

toString() method

Syntax

```
String toString ()
```

Returns

a `String` representing this instance

Usage

Returns a string representation of this object.

a `String` representing this instance

InvoiceTypeAttribute() constructor

Syntax

InvoiceTypeAttribute ()

Usage

Creates a new instance of

InvoiceTypeAttribute

.

InvoiceTypeAttribute(InvoiceTypeAttributesPK) constructor

Syntax

InvoiceTypeAttribute (InvoiceTypeAttributesPK *id*)

Parameters

- **id** – the PK value

Usage

Creates a new instance of

InvoiceTypeAttribute

with a preset primary key.

getAttributeTypeId() method

Syntax

int getAttributeTypeId ()

Returns

the attributeTypeId

Usage

the attributeTypeId

getInvoiceType() method

Syntax

InvoiceType getInvoiceType ()

Returns

the invoice type

Usage

the invoice type

getStrDefaultValue() method

Syntax

```
String getStrDefaultValue ()
```

Returns

the strValue

Usage

the strValue

isSetAttributeTypeId() method

Syntax

```
boolean isSetAttributeTypeId ()
```

Returns

false if the id or the id's attributeTypeId is null. Otherwise, it returns true.

Usage

Checks whether the

attributeTypeId

of this instance's PK is set.

false if the id or the id's attributeTypeId is null. Otherwise, it returns true.

isSetInvoiceType() method

Syntax

```
boolean isSetInvoiceType ()
```

Returns

false if the id or the id's invoiceType is null. Otherwise, it returns true.

Usage

Checks whether the

invoiceType

of this instance's PK is set.

false if the id or the id's invoiceType is null. Otherwise, it returns true.

isSetStrDefaultValue() method

Syntax

```
boolean isSetStrDefaultValue ()
```

Returns

false if the strDefaultValue is null. Otherwise, it returns true.

Usage

Checks whether the

strDefaultValue

of this instance is set.

false if the strDefaultValue is null. Otherwise, it returns true.

setStrDefaultValue(String) method

Syntax

```
void setStrDefaultValue ( String strDefaultValue )
```

Parameters

- **strDefaultValue** – the strValue to set

InvoiceTypeBatchConfig class

Syntax

```
public class InvoiceTypeBatchConfig
```

Usage

The InvoiceTypeOnlineCmdQueue model stores the requests which could not be sent to the Issuer of the InvoiceTypes

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

InvoiceTypeBatchConfig() constructor

Syntax

```
InvoiceTypeBatchConfig ()
```

Usage

Creates a new instance of

```
InvoiceTypeBatchConfig
```

.

InvoiceTypeBatchConfig(long) constructor

Syntax

```
InvoiceTypeBatchConfig ( long invTypeId )
```

Parameters

- **invTypeId** – the PK value

Usage

Creates a new instance of

```
InvoiceTypeBatchConfig
```

with a preset primary key.

getCutoffTime() method

Syntax

```
String getCutoffTime ()
```

Returns

the cutoffTime.

Usage

the cutoffTime.

getExportDate() method

Syntax

Date getExportDate ()

Returns

the Date representing the export time.

Usage

Returns the exportDate.

the Date representing the export time.

getExportSchedule() method

Syntax

String getExportSchedule ()

Returns

the exportSchedule.

Usage

the exportSchedule.

getExportUrl() method

Syntax

String getExportUrl ()

Returns

the exportUrl.

Usage

Returns the exportUrl.

the exportUrl.

getImportUrl() method

Syntax

String getImportUrl ()

Returns

the importUrl.

Usage

the importUrl.

isSetCutoffTime() method

Syntax

```
boolean isSetCutoffTime ()
```

Returns

false if the cutoffTime is null. Otherwise, it returns true.

Usage

Checks whether the

cutoffTime

of this instance is set.

false if the cutoffTime is null. Otherwise, it returns true.

isSetExportDate() method

Syntax

```
boolean isSetExportDate ()
```

Returns

false if the exportDate is null. Otherwise, it returns true.

Usage

Checks whether the

exportDate

of this instance is set.

false if the exportDate is null. Otherwise, it returns true.

isSetExportSchedule() method

Syntax

```
boolean isSetExportSchedule ()
```

Returns

false if the `exportSchedule` is null. Otherwise, it returns true.

Usage

Checks whether the

exportSchedule

of this instance is set.

false if the `exportSchedule` is null. Otherwise, it returns true.

isSetExportUrl() method

Syntax

```
boolean isSetExportUrl ()
```

Returns

false if the `exportUrl` is null . Otherwise, it returns true.

Usage

Checks whether the

exportUrl

of this instance is set.

false if the `exportUrl` is null . Otherwise, it returns true.

isSetImportUrl() method

Syntax

```
boolean isSetImportUrl ()
```

Returns

false if the `importUrl` is null . Otherwise, it returns true.

Usage

Checks whether the

importUrl

of this instance is set.

false if the `importUrl` is null . Otherwise, it returns true.

setCutoffTime(String) method

Syntax

```
void setCutoffTime ( String cutoffTime )
```

Parameters

- **cutoffTime** – the cutoffTime to set

Usage

Sets the cutoffTime of this invoice.

setExportDate(Date) method

Syntax

```
void setExportDate ( Date exportDate )
```

Parameters

- **exportDate** – the exportDate to set

Usage

Sets the exportDate.

setExportSchedule(String) method

Syntax

```
void setExportSchedule ( String exportSchedule )
```

Parameters

- **exportSchedule** – the exportSchedule to set

Usage

Sets the exportSchedule of this invoice.

setExportUrl(String) method

Syntax

```
void setExportUrl ( String exportUrl )
```

Parameters

- **exportUrl** – the exportUrl to set

Usage

Sets the exportUrl

setImportUrl(String) method

Syntax

```
void setImportUrl ( String importUrl )
```

Parameters

- **importUrl** – the importUrl to set

Usage

Sets the importUrl of this invoice.

InvoiceTypeBatchInvoices class

Syntax

```
public class InvoiceTypeBatchInvoices
```

Usage

This entity keeps track of the paid invoices that are managed by the OfflineBillPaymentHandler.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

InvoiceTypeBatchInvoices() constructor

Syntax

```
InvoiceTypeBatchInvoices ()
```

Usage

Creates a new instance of

```
InvoiceTypeBatchInvoices
```

.

getCancelTransaction() method

Syntax

Transaction getCancelTransaction ()

Returns

the associated Transaction

Usage

Returns the associated cancelTransaction of this instance.

the associated Transaction

getInvoice() method

Syntax

Invoice getInvoice ()

Returns

the associated Invoice

Usage

Returns the associated invoice of this instance.

the associated Invoice

getInvoiceType() method

Syntax

InvoiceType getInvoiceType ()

Returns

the associated InvoiceType

Usage

Returns the associated invoiceType of this instance.

the associated InvoiceType

getStatus() method

Syntax

int getStatus ()

Returns

the associated Integer

Usage

Returns the associated status of this instance.

the associated Integer

getTransaction() method

Syntax

Transaction getTransaction ()

Returns

the associated Transaction

Usage

Returns the associated transaction of this instance.

the associated Transaction

isSetCancelTransaction() method

Syntax

boolean isSetCancelTransaction ()

Returns

false if the cancelTransaction is null. Otherwise, it returns true.

Usage

Checks whether the

cancelTransaction

of this instance is set.

false if the cancelTransaction is null. Otherwise, it returns true.

isSetInvoice() method

Syntax

`boolean isSetInvoice ()`

Returns

false if the invoice is null. Otherwise, it returns true.

Usage

Checks whether the

invoice

of this instance is set.

false if the invoice is null. Otherwise, it returns true.

isSetInvoiceType() method

Syntax

`boolean isSetInvoiceType ()`

Returns

false if the invoiceType is null. Otherwise, it returns true.

Usage

Checks whether the

invoiceType

of this instance is set.

false if the invoiceType is null. Otherwise, it returns true.

isSetStatus() method

Syntax

`boolean isSetStatus ()`

Returns

false if the status is null. Otherwise, it returns true.

Usage

Checks whether the

status

of this instance is set.

false if the status is null. Otherwise, it returns true.

isSetTransaction() method

Syntax

```
boolean isSetTransaction ()
```

Returns

false if the invoice is null. Otherwise, it returns true.

Usage

Checks whether the

invoice

of this instance is set.

false if the invoice is null. Otherwise, it returns true.

setCancelTransaction(Transaction) method

Syntax

```
void setCancelTransaction ( Transaction cancelTransaction )
```

Parameters

- **cancelTransaction** – the cancelTransaction to set

Usage

Sets the

cancelTransaction

of this instance.

setInvoice(Invoice) method

Syntax

```
void setInvoice ( Invoice invoice )
```

Parameters

- **invoice** – the `invoice` to set

Usage

Sets the

invoice

of this instance.

setInvoiceType(InvoiceType) method

Syntax

```
void setInvoiceType ( InvoiceType invoiceType )
```

Parameters

- **invoiceType** – the `invoiceType` to set

Usage

Sets the

invoiceType

of this instance.

setStatus(int) method

Syntax

```
void setStatus ( int status )
```

Parameters

- **status** – the `status` to set

Usage

Sets the

status

of this instance.

setTransaction(Transaction) method

Syntax

```
void setTransaction ( Transaction transaction )
```

Parameters

- **transaction** – the transaction to set

Usage

Sets the

transaction

of this instance.

InvoiceTypeOnlineCmdQueue class

Syntax

```
public class InvoiceTypeOnlineCmdQueue
```

Usage

The `InvoiceTypeOnlineCmdQueue` model stores the requests which could not be sent to the Issuer of the `InvoiceTypes`

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

InvoiceTypeOnlineCmdQueue() constructor

Syntax

```
InvoiceTypeOnlineCmdQueue ()
```

Usage

Creates a new instance of

`InvoiceTypeOnlineCmdQueue`

.

getInvoice() method

Syntax

```
Invoice getInvoice ()
```

Returns

the associated Invoice

Usage

Returns the associated invoice of this instance.

the associated Invoice

getInvoiceType() method

Syntax

```
InvoiceType getInvoiceType ()
```

Returns

the invoiceType

Usage

the invoiceType

getRetryCount() method

Syntax

```
int getRetryCount ()
```

Returns

the count

Usage

the count

getStatus() method

Syntax

```
int getStatus ()
```

Returns

the status

Usage

the status

getTransferStatus() method

Syntax

```
int getTransferStatus ()
```

Returns

the transferStatus

Usage

the transferStatus

isSetInvoice() method

Syntax

```
boolean isSetInvoice ()
```

Returns

false if the invoice is null. Otherwise, it returns true.

Usage

Checks whether the

invoice

of this instance is set.

false if the invoice is null. Otherwise, it returns true.

isSetInvoiceType() method

Syntax

```
boolean isSetInvoiceType ()
```

Returns

false if the invoiceType is null. Otherwise, it returns true.

Usage

Checks whether the

invoiceType

of this instance is set.

false if the *invoiceType* is null. Otherwise, it returns true.

isSetRetryCount() method

Syntax

```
boolean isSetRetryCount ()
```

Returns

false if the count is null. Otherwise, it returns true.

Usage

Checks whether the

count

of this instance is set.

false if the count is null. Otherwise, it returns true.

isSetStatus() method

Syntax

```
boolean isSetStatus ()
```

Returns

false if the status is null. Otherwise, it returns true.

Usage

Checks whether the

status

of this instance is set.

false if the status is null. Otherwise, it returns true.

isSetTransferStatus() method

Syntax

```
boolean isSetTransferStatus ()
```

Returns

false if the `transferStatus` is null. Otherwise, it returns true.

Usage

Checks whether the

transferStatus

of this instance is set.

false if the `transferStatus` is null. Otherwise, it returns true.

setInvoice(Invoice) method

Syntax

```
void setInvoice ( Invoice invoice )
```

Parameters

- **invoice** – the `invoice` to set

Usage

Sets the

invoice

of this instance.

setInvoiceType(InvoiceType) method

Syntax

```
void setInvoiceType ( InvoiceType invoiceType )
```

Parameters

- **invoiceType** – the `invoiceType` to set

setRetryCount(int) method

Syntax

```
void setRetryCount ( int count )
```

Parameters

- **count** – the count to set

setStatus(int) method

Syntax

```
void setStatus ( int status )
```

Parameters

- **status** – the status to set

setTransferStatus(int) method

Syntax

```
void setTransferStatus ( int transferStatus )
```

Parameters

- **transferStatus** – the transferStatus to set

InvoiceTypeOnlineConfig class

Syntax

```
public class InvoiceTypeOnlineConfig
```

Usage

This entity stores the configuration data needed to send request via the SOAP interface to the biller.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

InvoiceTypeOnlineConfig() constructor

Syntax

`InvoiceTypeOnlineConfig ()`

Usage

Creates a new instance of

`InvoiceTypeOnlineConfig`

.

InvoiceTypeOnlineConfig(long) constructor

Syntax

`InvoiceTypeOnlineConfig ( long invTypeId )`

Parameters

- **invTypeId** – the PK value

Usage

Creates a new instance of

`InvoiceTypeOnlineConfig`

with a preset primary key.

getProtocolVersion() method

Syntax

`int getProtocolVersion ()`

Returns

the associated `Integer`

Usage

Returns the associated `protocolVersion` of this instance.

the associated `Integer`

getRetries() method

Syntax

```
int getRetries ()
```

Returns

the associated Integer

Usage

Returns the associated retries of this instance.

the associated Integer

getTimeoutSeconds() method

Syntax

```
int getTimeoutSeconds ()
```

Returns

the associated Integer

Usage

Returns the associated timeoutSeconds of this instance.

the associated Integer

getUrl() method

Syntax

```
String getUrl ()
```

Returns

the associated String

Usage

Returns the associated url of this instance.

the associated String

isSetProtocolVersion() method

Syntax

boolean isSetProtocolVersion ()

Returns

false if the protocolVersion is null. Otherwise, it returns true.

Usage

Checks whether the

protocolVersion

of this instance is set.

false if the protocolVersion is null. Otherwise, it returns true.

isSetRetries() method

Syntax

boolean isSetRetries ()

Returns

false if the retries is null. Otherwise, it returns true.

Usage

Checks whether the

retries

of this instance is set.

false if the retries is null. Otherwise, it returns true.

isSetTimeoutSeconds() method

Syntax

boolean isSetTimeoutSeconds ()

Returns

false if the protocolVersion is null. Otherwise, it returns true.

Usage

Checks whether the

timeoutSeconds

of this instance is set.

false if the `protocolVersion` is null. Otherwise, it returns true.

isSetUrl() method

Syntax

```
boolean isSetUrl ()
```

Returns

false if the `url` is null. Otherwise, it returns true.

Usage

Checks whether the

url

of this instance is set.

false if the `url` is null. Otherwise, it returns true.

setProtocolVersion(int) method

Syntax

```
void setProtocolVersion ( int protocolVersion )
```

Parameters

- **protocolVersion** – the `protocolVersion` to set

Usage

Sets the

protocolVersion

of this instance.

setRetries(int) method

Syntax

```
void setRetries ( int retries )
```

Parameters

- **retries** – the retries to set

Usage

Sets the

retries

of this instance.

setTimeoutSeconds(int) method

Syntax

```
void setTimeoutSeconds ( int timeoutSeconds )
```

Parameters

- **timeoutSeconds** – the timeoutSeconds to set

Usage

Sets the

timeoutSeconds

of this instance.

setUrl(String) method

Syntax

```
void setUrl ( String url )
```

Parameters

- **url** – the url to set

Usage

Sets the

url

of this instance.

Job class**Syntax**

```
public class Job
```

Usage

The `Job` configures a cron job to be executed by the `net.paybox.mobiliser.platform.CronJobScheduler`

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Job() constructor**Syntax**

```
Job ()
```

Usage

Creates a new instance of

```
Job
```

```
.
```

getCronSchedule() method**Syntax**

```
String getCronSchedule ()
```

Returns

the associated `String`

Usage

Returns the associated `cronSchedule` of this instance.

the associated `String`

getHandlerName() method**Syntax**

```
String getHandlerName ()
```

Returns

the associated `String`

Usage

Returns the associated `handlerName` of this instance.

the associated `String`

getImplementationType() method

Syntax

```
int getImplementationType ()
```

Returns

the associated `Integer`

Usage

Returns the associated `implementationType` of this instance.

the associated `Integer`

getImplementationURL() method

Syntax

```
String getImplementationURL ()
```

Returns

the associated `String`

Usage

Returns the associated `implementationURL` of this instance.

the associated `String`

getJobName() method

Syntax

```
String getJobName ()
```

Returns

the associated `String`

Usage

Returns the associated jobName of this instance.

the associated `String`

getLastExecution() method

Syntax

```
Date getLastExecution ()
```

Returns

the associated `Date`

Usage

Returns the associated lastExecution of this instance.

the associated `Date`

getMaxDelay() method

Syntax

```
Long getMaxDelay ()
```

Returns

the associated `Long`

Usage

Returns the associated maxDelay of this instance.

the associated `Long`

getMaxDuration() method

Syntax

```
Long getMaxDuration ()
```

Returns

the associated `Long`

Usage

Returns the associated maxDuration of this instance.

the associated Long

getParameter() method

Syntax

String getParameter ()

Returns

the associated String

Usage

Returns the associated parameter of this instance.

the associated String

isActive() method

Syntax

boolean isActive ()

Returns

true if the job is active; false otherwise

Usage

Indicates if this job is active

true if the job is active; false otherwise

isSetActive() method

Syntax

boolean isSetActive ()

Returns

false if the dbActive is null. Otherwise, it returns true.

Usage

Checks whether the

dbActive

of this instance is set.

false if the dbActive is null. Otherwise, it returns true.

isSetCronSchedule() method

Syntax

boolean isSetCronSchedule ()

Returns

false if the cronSchedule is null. Otherwise, it returns true.

Usage

Checks whether the

cronSchedule

of this instance is set.

false if the cronSchedule is null. Otherwise, it returns true.

isSetHandlerName() method

Syntax

boolean isSetHandlerName ()

Returns

false if the handlerName is null. Otherwise, it returns true.

Usage

Checks whether the

handlerName

of this instance is set.

false if the handlerName is null. Otherwise, it returns true.

isSetImplementationType() method

Syntax

boolean isSetImplementationType ()

Returns

false if the implementationType is null. Otherwise, it returns true.

Usage

Checks whether the

implementationType

of this instance is set.

false if the *implementationType* is null. Otherwise, it returns true.

isSetImplementationURL() method

Syntax

```
boolean isSetImplementationURL ()
```

Returns

false if the *implementationURL* is null. Otherwise, it returns true.

Usage

Checks whether the

implementationURL

of this instance is set.

false if the *implementationURL* is null. Otherwise, it returns true.

isSetJobName() method

Syntax

```
boolean isSetJobName ()
```

Returns

false if the *jobName* is null. Otherwise, it returns true.

Usage

Checks whether the

jobName

of this instance is set.

false if the *jobName* is null. Otherwise, it returns true.

isSetLastExecution() method

Syntax

`boolean isSetLastExecution ()`

Returns

false if the `lastExecution` is null. Otherwise, it returns true.

Usage

Checks whether the

lastExecution

of this instance is set.

false if the `lastExecution` is null. Otherwise, it returns true.

isSetMaxDelay() method

Syntax

`boolean isSetMaxDelay ()`

Returns

false if the `maxDelay` is null. Otherwise, it returns true.

Usage

Checks whether the

maxDelay

of this instance is set.

false if the `maxDelay` is null. Otherwise, it returns true.

isSetMaxDuration() method

Syntax

`boolean isSetMaxDuration ()`

Returns

false if the `maxDuration` is null. Otherwise, it returns true.

Usage

Checks whether the

maxDuration

of this instance is set.

false if the *maxDuration* is null. Otherwise, it returns true.

isSetParameter() method

Syntax

```
boolean isSetParameter ()
```

Returns

false if the parameter is null . Otherwise, it returns true.

Usage

Checks whether the

parameter

of this instance is set.

false if the *parameter* is null . Otherwise, it returns true.

setActive(boolean) method

Syntax

```
void setActive ( boolean active )
```

Parameters

- **active** – the active value to set

Usage

Sets the

active

of this instance.

setCronSchedule(String) method

Syntax

```
void setCronSchedule ( String cronSchedule )
```

Parameters

- **cronSchedule** – the `cronSchedule` to set

Usage

Sets the

cronSchedule

of this instance.

setHandlerName(String) method

Syntax

```
void setHandlerName ( String handlerName )
```

Parameters

- **handlerName** – the `handlerName` to set

Usage

Sets the

handlerName

of this instance.

setImplementationType(int) method

Syntax

```
void setImplementationType ( int implementationType )
```

Parameters

- **implementationType** – the `implementationType` to set

Usage

Sets the

implementationType

of this instance.

setImplementationURL(String) method

Syntax

void setImplementationURL (String *implementationURL*)

Parameters

- **implementationURL** – the implementationURL to set

Usage

Sets the

implementationURL

of this instance.

setJobName(String) method

Syntax

void setJobName (String *jobName*)

Parameters

- **jobName** – the jobName to set

Usage

Sets the

jobName

of this instance.

setLastExecution(Date) method

Syntax

void setLastExecution (Date *lastExecution*)

Parameters

- **lastExecution** – the lastExecution to set

Usage

Sets the

lastExecution

of this instance.

setMaxDelay(Long) method

Syntax

```
void setMaxDelay ( Long maxDelay )
```

Parameters

- **maxDelay** – the `maxDelay` to set

Usage

Sets the

maxDelay

of this instance.

setMaxDuration(Long) method

Syntax

```
void setMaxDuration ( Long maxDuration )
```

Parameters

- **maxDuration** – the `maxDuration` to set

Usage

Sets the

maxDuration

of this instance.

setParameter(String) method

Syntax

```
void setParameter ( String parameter )
```

Parameters

- **parameter** – the `parameter` to set

Usage

Sets the

parameter

of this instance.

JobHistory class

Syntax

```
public class JobHistory
```

Usage

The JobHistory tracks previously executed job runs

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

JobHistory() constructor

Syntax

```
JobHistory ()
```

Usage

Creates a new instance of

```
JobHistory
```

.

getEndDate() method

Syntax

```
Date getEndDate ()
```

Returns

the associated Date

Usage

Returns the associated endDate of this instance.

the associated Date

getJob() method

Syntax

Job getJob ()

Returns

the associated Job

Usage

Returns the associated job of this instance.

the associated Job

getLogEntry() method

Syntax

String getLogEntry ()

Returns

the associated String

Usage

Returns the associated logEntry of this instance.

the associated String

getStartDate() method

Syntax

Date getStartDate ()

Returns

the associated Date

Usage

Returns the associated startDate of this instance.

the associated Date

getStatus() method

Syntax

char getStatus ()

Returns

the associated char

Usage

Returns the associated status of this instance.

the associated char

isSetEndDate() method

Syntax

boolean isSetEndDate ()

Returns

false if the endDate is null . Otherwise, it returns true.

Usage

Checks whether the

endDate

of this instance is set.

false if the endDate is null . Otherwise, it returns true.

isSetJob() method

Syntax

boolean isSetJob ()

Returns

false if the job is null. Otherwise, it returns true.

Usage

Checks whether the

job

of this instance is set.

false if the job is null. Otherwise, it returns true.

isSetLogEntry() method

Syntax

```
boolean isSetLogEntry ()
```

Returns

false if the logEntry is null . Otherwise, it returns true.

Usage

Checks whether the

logEntry

of this instance is set.

false if the logEntry is null . Otherwise, it returns true.

isSetStartDate() method

Syntax

```
boolean isSetStartDate ()
```

Returns

false if the startDate is null . Otherwise, it returns true.

Usage

Checks whether the

startDate

of this instance is set.

false if the startDate is null . Otherwise, it returns true.

isSetStatus() method

Syntax

```
boolean isSetStatus ()
```

Returns

false if the status is null . Otherwise, it returns true.

Usage

Checks whether the

status

of this instance is set.

false if the status is null . Otherwise, it returns true.

setEndDate(Date) method

Syntax

```
void setEndDate ( Date endDate )
```

Parameters

- **endDate** – the endDate to set

Usage

Sets the

endDate

of this instance.

setJob(Job) method

Syntax

```
void setJob ( Job job )
```

Parameters

- **job** – the job to set

Usage

Sets the

job

of this instance.

setLogEntry(String) method

Syntax

```
void setLogEntry ( String logEntry )
```

Parameters

- **logEntry** – the logEntry to set

Usage

Sets the

logEntry

of this instance.

setStartDate(Date) method

Syntax

```
void setStartDate ( Date startDate )
```

Parameters

- **startDate** – the startDate to set

Usage

Sets the

startDate

of this instance.

setStatus(char) method

Syntax

```
void setStatus ( char status )
```

Parameters

- **status** – the status to set

Usage

Sets the

status

of this instance.

LimitClass class

Syntax

```
public class LimitClass
```

Usage

The `Limit` tracks the limits and turnover for different entities in the mobiliser platform like customers, payment instruments, wallet entries, etc.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

LimitClass() constructor

Syntax

```
LimitClass ()
```

Usage

Creates a new instance of

```
LimitClass
```

.

getAbsoluteCreditLimitAmount() method

Syntax

```
Long getAbsoluteCreditLimitAmount ()
```

Returns

the absoluteCreditLimitAmount

Usage

the absoluteCreditLimitAmount

getAbsoluteCreditLimitCount() method

Syntax

```
Long getAbsoluteCreditLimitCount ()
```

Returns

the absoluteCreditLimitCount

Usage

the absoluteCreditLimitCount

getAbsoluteDebitLimitAmount() method

Syntax

Long getAbsoluteDebitLimitAmount ()

Returns

the absoluteDebitLimitAmount

Usage

the absoluteDebitLimitAmount

getAbsoluteDebitLimitCount() method

Syntax

Long getAbsoluteDebitLimitCount ()

Returns

the absoluteDebitLimitCount

Usage

the absoluteDebitLimitCount

getCurrency() method

Syntax

Currency getCurrency ()

Returns

the dbCurrency

Usage

the dbCurrency

getDailyCreditLimitAmount() method

Syntax

Long getDailyCreditLimitAmount ()

Returns

the dailyCreditLimitAmount

Usage

the dailyCreditLimitAmount

getDailyCreditLimitCount() method

Syntax

Long getDailyCreditLimitCount ()

Returns

the dailyCreditLimitCount

Usage

the dailyCreditLimitCount

getDailyDebitLimitAmount() method

Syntax

Long getDailyDebitLimitAmount ()

Returns

the dailyDebitLimitAmount

Usage

the dailyDebitLimitAmount

getDailyDebitLimitCount() method

Syntax

Long getDailyDebitLimitCount ()

Returns

the dailyDebitLimitCount

Usage

the dailyDebitLimitCount

getMonthlyCreditLimitAmount() method

Syntax

Long getMonthlyCreditLimitAmount ()

Returns

the monthlyCreditLimitAmount

Usage

the monthlyCreditLimitAmount

getMonthlyCreditLimitCount() method

Syntax

Long getMonthlyCreditLimitCount ()

Returns

the monthlyCreditLimitCount

Usage

the monthlyCreditLimitCount

getMonthlyDebitLimitAmount() method

Syntax

Long getMonthlyDebitLimitAmount ()

Returns

the monthlyDebitLimitAmount

Usage

the monthlyDebitLimitAmount

getMonthlyDebitLimitCount() method

Syntax

Long getMonthlyDebitLimitCount ()

Returns

the monthlyDebitLimitCount

Usage

the monthlyDebitLimitCount

getName() method

Syntax

String getName ()

getSingleCreditLimitAmount() method

Syntax

Long getSingleCreditLimitAmount ()

getSingleCreditMinimum() method

Syntax

Long getSingleCreditMinimum ()

getSingleDebitLimitAmount() method

Syntax

Long getSingleDebitLimitAmount ()

getSingleDebitMinimum() method

Syntax

Long getSingleDebitMinimum ()

getWeeklyCreditLimitAmount() method

Syntax

Long getWeeklyCreditLimitAmount ()

Returns

the weeklyCreditLimitAmount

Usage

the weeklyCreditLimitAmount

getWeeklyCreditLimitCount() method

Syntax

Long getWeeklyCreditLimitCount ()

Returns

the weeklyCreditLimitCount

Usage

the weeklyCreditLimitCount

getWeeklyDebitLimitAmount() method

Syntax

Long getWeeklyDebitLimitAmount ()

Returns

the weeklyDebitLimitAmount

Usage

the weeklyDebitLimitAmount

getWeeklyDebitLimitCount() method

Syntax

Long getWeeklyDebitLimitCount ()

Returns

the weeklyDebitLimitCount

Usage

the weeklyDebitLimitCount

isSetAbsoluteCreditLimitAmount() method

Syntax

boolean isSetAbsoluteCreditLimitAmount ()

Returns

false if the absoluteCreditLimitAmount is null. Otherwise, it returns true.

Usage

Checks whether the

absoluteCreditLimitAmount

of this instance is set.

false if the absoluteCreditLimitAmount is null. Otherwise, it returns true.

isSetAbsoluteCreditLimitCount() method

Syntax

```
boolean isSetAbsoluteCreditLimitCount ()
```

Returns

false if the absoluteCreditLimitCount is null. Otherwise, it returns true.

Usage

Checks whether the

absoluteCreditLimitCount

of this instance is set.

false if the absoluteCreditLimitCount is null. Otherwise, it returns true.

isSetAbsoluteDebitLimitAmount() method

Syntax

```
boolean isSetAbsoluteDebitLimitAmount ()
```

Returns

false if the absoluteDebitLimitAmount is null. Otherwise, it returns true.

Usage

Checks whether the

absoluteDebitLimitAmount

of this instance is set.

false if the absoluteDebitLimitAmount is null. Otherwise, it returns true.

isSetAbsoluteDebitLimitCount() method

Syntax

```
boolean isSetAbsoluteDebitLimitCount ()
```

Returns

false if the absoluteDebitLimitCount is null. Otherwise, it returns true.

Usage

Checks whether the

absoluteDebitLimitCount

of this instance is set.

false if the absoluteDebitLimitCount is null. Otherwise, it returns true.

isSetCurrency() method

Syntax

```
boolean isSetCurrency ()
```

Returns

false if the dbCurrency is null. Otherwise, it returns true.

Usage

Checks whether the

dbCurrency

of this instance is set.

false if the dbCurrency is null. Otherwise, it returns true.

isSetDailyCreditLimitAmount() method

Syntax

```
boolean isSetDailyCreditLimitAmount ()
```

Returns

false if the dailyCreditLimitAmount is null. Otherwise, it returns true.

Usage

Checks whether the

dailyCreditLimitAmount

of this instance is set.

false if the `dailyCreditLimitAmount` is null. Otherwise, it returns true.

isSetDailyCreditLimitCount() method

Syntax

```
boolean isSetDailyCreditLimitCount ()
```

Returns

false if the `dailyCreditLimitCount` is null. Otherwise, it returns true.

Usage

Checks whether the

dailyCreditLimitCount

of this instance is set.

false if the `dailyCreditLimitCount` is null. Otherwise, it returns true.

isSetDailyDebitLimitAmount() method

Syntax

```
boolean isSetDailyDebitLimitAmount ()
```

Returns

false if the `dailyDebitLimitAmount` is null. Otherwise, it returns true.

Usage

Checks whether the

dailyDebitLimitAmount

of this instance is set.

false if the `dailyDebitLimitAmount` is null. Otherwise, it returns true.

*isSetDailyDebitLimitCount() method***Syntax**

```
boolean isSetDailyDebitLimitCount ()
```

Returns

false if the `dailyDebitLimitCount` is null. Otherwise, it returns true.

Usage

Checks whether the

dailyDebitLimitCount

of this instance is set.

false if the `dailyDebitLimitCount` is null. Otherwise, it returns true.

*isSetMonthlyCreditLimitAmount() method***Syntax**

```
boolean isSetMonthlyCreditLimitAmount ()
```

Returns

false if the `monthlyCreditLimitAmount` is null. Otherwise, it returns true.

Usage

Checks whether the

monthlyCreditLimitAmount

of this instance is set.

false if the `monthlyCreditLimitAmount` is null. Otherwise, it returns true.

*isSetMonthlyCreditLimitCount() method***Syntax**

```
boolean isSetMonthlyCreditLimitCount ()
```

Returns

false if the `monthlyCreditLimitCount` is null. Otherwise, it returns true.

Usage

Checks whether the

monthlyCreditLimitCount

of this instance is set.

false if the *monthlyCreditLimitCount* is null. Otherwise, it returns true.

isSetMonthlyDebitLimitAmount() method

Syntax

```
boolean isSetMonthlyDebitLimitAmount ()
```

Returns

false if the *monthlyDebitLimitAmount* is null. Otherwise, it returns true.

Usage

Checks whether the

monthlyDebitLimitAmount

of this instance is set.

false if the *monthlyDebitLimitAmount* is null. Otherwise, it returns true.

isSetMonthlyDebitLimitCount() method

Syntax

```
boolean isSetMonthlyDebitLimitCount ()
```

Returns

false if the *monthlyDebitLimitCount* is null. Otherwise, it returns true.

Usage

Checks whether the

monthlyDebitLimitCount

of this instance is set.

false if the *monthlyDebitLimitCount* is null. Otherwise, it returns true.

isSetName() method

Syntax

`boolean isSetName ()`

isSetSingleCreditLimitAmount() method

Syntax

`boolean isSetSingleCreditLimitAmount ()`

Returns

false if the `singleCreditLimitAmount` is null. Otherwise, it returns true.

Usage

Checks whether the

singleCreditLimitAmount

of this instance is set.

false if the `singleCreditLimitAmount` is null. Otherwise, it returns true.

isSetSingleCreditMinimum() method

Syntax

`boolean isSetSingleCreditMinimum ()`

Returns

false if the `singleCreditMinimum` is null. Otherwise, it returns true.

Usage

Checks whether the

singleCreditMinimum

of this instance is set.

false if the `singleCreditMinimum` is null. Otherwise, it returns true.

isSetSingleDebitLimitAmount() method

Syntax

`boolean isSetSingleDebitLimitAmount ()`

Returns

false if the `singleDebitLimitAmount` is null. Otherwise, it returns true.

Usage

Checks whether the

singleDebitLimitAmount

of this instance is set.

false if the `singleDebitLimitAmount` is null. Otherwise, it returns true.

isSetSingleDebitMinimum() method

Syntax

```
boolean isSetSingleDebitMinimum ()
```

Returns

false if the `singleDebitMinimum` is null. Otherwise, it returns true.

Usage

Checks whether the

singleDebitMinimum

of this instance is set.

false if the `singleDebitMinimum` is null. Otherwise, it returns true.

isSetWeeklyCreditLimitAmount() method

Syntax

```
boolean isSetWeeklyCreditLimitAmount ()
```

Returns

false if the `weeklyCreditLimitAmount` is null. Otherwise, it returns true.

Usage

Checks whether the

weeklyCreditLimitAmount

of this instance is set.

false if the `weeklyCreditLimitAmount` is null. Otherwise, it returns true.

*isSetWeeklyCreditLimitCount() method***Syntax**

```
boolean isSetWeeklyCreditLimitCount ()
```

Returns

false if the weeklyCreditLimitCount is null. Otherwise, it returns true.

Usage

Checks whether the

weeklyCreditLimitCount

of this instance is set.

false if the weeklyCreditLimitCount is null. Otherwise, it returns true.

*isSetWeeklyDebitLimitAmount() method***Syntax**

```
boolean isSetWeeklyDebitLimitAmount ()
```

Returns

false if the weeklyDebitLimitAmount is null. Otherwise, it returns true.

Usage

Checks whether the

weeklyDebitLimitAmount

of this instance is set.

false if the weeklyDebitLimitAmount is null. Otherwise, it returns true.

*isSetWeeklyDebitLimitCount() method***Syntax**

```
boolean isSetWeeklyDebitLimitCount ()
```

Returns

false if the weeklyDebitLimitCount is null. Otherwise, it returns true.

Usage

Checks whether the

weeklyDebitLimitCount

of this instance is set.

false if the *weeklyDebitLimitCount* is null. Otherwise, it returns true.

setAbsoluteCreditLimitAmount(Long) method

Syntax

void *setAbsoluteCreditLimitAmount* (Long *absoluteCreditLimitAmount*)

Parameters

- **absoluteCreditLimitAmount** – the *absoluteCreditLimitAmount* to set

setAbsoluteCreditLimitCount(Long) method

Syntax

void *setAbsoluteCreditLimitCount* (Long *absoluteCreditLimitCount*)

Parameters

- **absoluteCreditLimitCount** – the *absoluteCreditLimitCount* to set

setAbsoluteDebitLimitAmount(Long) method

Syntax

void *setAbsoluteDebitLimitAmount* (Long *absoluteDebitLimitAmount*)

Parameters

- **absoluteDebitLimitAmount** – the *absoluteDebitLimitAmount* to set

setAbsoluteDebitLimitCount(Long) method

Syntax

void *setAbsoluteDebitLimitCount* (Long *absoluteDebitLimitCount*)

Parameters

- **absoluteDebitLimitCount** – the *absoluteDebitLimitCount* to set

setCurrency(Currency) method

Syntax

void setCurrency (Currency *currency*)

Parameters

- **dbCurrency** – the dbCurrency to set

setDailyCreditLimitAmount(Long) method

Syntax

void setDailyCreditLimitAmount (Long *dailyCreditLimitAmount*)

Parameters

- **dailyCreditLimitAmount** – the dailyCreditLimitAmount to set

setDailyCreditLimitCount(Long) method

Syntax

void setDailyCreditLimitCount (Long *dailyCreditLimitCount*)

Parameters

- **dailyCreditLimitCount** – the dailyCreditLimitCount to set

setDailyDebitLimitAmount(Long) method

Syntax

void setDailyDebitLimitAmount (Long *dailyDebitLimitAmount*)

Parameters

- **dailyDebitLimitAmount** – the dailyDebitLimitAmount to set

setDailyDebitLimitCount(Long) method

Syntax

void setDailyDebitLimitCount (Long *dailyDebitLimitCount*)

Parameters

- **dailyDebitLimitCount** – the dailyDebitLimitCount to set

setMonthlyCreditLimitAmount(Long) method

Syntax

void setMonthlyCreditLimitAmount (Long *monthlyCreditLimitAmount*)

Parameters

- **monthlyCreditLimitAmount** – the monthlyCreditLimitAmount to set

setMonthlyCreditLimitCount(Long) method

Syntax

void setMonthlyCreditLimitCount (Long *monthlyCreditLimitCount*)

Parameters

- **monthlyCreditLimitCount** – the monthlyCreditLimitCount to set

setMonthlyDebitLimitAmount(Long) method

Syntax

void setMonthlyDebitLimitAmount (Long *monthlyDebitLimitAmount*)

Parameters

- **monthlyDebitLimitAmount** – the monthlyDebitLimitAmount to set

setMonthlyDebitLimitCount(Long) method

Syntax

void setMonthlyDebitLimitCount (Long *monthlyDebitLimitCount*)

Parameters

- **monthlyDebitLimitCount** – the monthlyDebitLimitCount to set

setName(String) method

Syntax

void setName (String *name*)

setSingleCreditLimitAmount(Long) method

Syntax

void setSingleCreditLimitAmount (Long *singleCreditLimitAmount*)

setSingleCreditMinimum(Long) method

Syntax

void setSingleCreditMinimum (Long *singleCreditMinimum*)

setSingleDebitLimitAmount(Long) method

Syntax

void setSingleDebitLimitAmount (Long *singleDebitLimitAmount*)

setSingleDebitMinimum(Long) method

Syntax

void setSingleDebitMinimum (Long *singleDebitMinimum*)

setWeeklyCreditLimitAmount(Long) method

Syntax

void setWeeklyCreditLimitAmount (Long *weeklyCreditLimitAmount*)

Parameters

- **weeklyCreditLimitAmount** – the weeklyCreditLimitAmount to set

setWeeklyCreditLimitCount(Long) method

Syntax

void setWeeklyCreditLimitCount (Long *weeklyCreditLimitCount*)

Parameters

- **weeklyCreditLimitCount** – the weeklyCreditLimitCount to set

setWeeklyDebitLimitAmount(Long) method

Syntax

void setWeeklyDebitLimitAmount (Long *weeklyDebitLimitAmount*)

Parameters

- **weeklyDebitLimitAmount** – the weeklyDebitLimitAmount to set

setWeeklyDebitLimitCount(Long) method

Syntax

```
void setWeeklyDebitLimitCount ( Long weeklyDebitLimitCount )
```

Parameters

- **weeklyDebitLimitCount** – the weeklyDebitLimitCount to set

LimitSet class

Syntax

```
public class LimitSet
```

Usage

The `LimitSet` defines the limit sets.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

LimitSet() constructor

Syntax

```
LimitSet ()
```

Usage

Creates a new instance of

```
LimitSet
```

.

getName() method

Syntax

```
String getName ()
```

isIndividual() method

Syntax

boolean isIndividual ()

isSetIndividual() method

Syntax

boolean isSetIndividual ()

Returns

false if the dbIndividual is null. Otherwise, it returns true.

Usage

Checks whether the

dbIndividual

of this instance is set.

false if the dbIndividual is null. Otherwise, it returns true.

isSetName() method

Syntax

boolean isSetName ()

Returns

false if the name is null. Otherwise, it returns true.

Usage

Checks whether the

name

of this instance is set.

false if the name is null. Otherwise, it returns true.

setIndividual(boolean) method

Syntax

void setIndividual (boolean *individual*)

setName(String) method

Syntax

```
void setName ( String name )
```

LimitSetClass class

Syntax

```
public class LimitSetClass
```

Usage

The `LimitSetClass` maps the limit sets and use cases to limit classes

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

LimitSetClass() constructor

Syntax

```
LimitSetClass ()
```

Usage

Creates a new instance of

```
LimitSetClass
```

.

getLimitClassId() method

Syntax

```
long getLimitClassId ()
```

Returns

the `limitClassId`

Usage

the `limitClassId`

getLimitSetId() method

Syntax

long getLimitSetId ()

Returns

the limitSetId

Usage

the limitSetId

getUseCaseId() method

Syntax

Integer getUseCaseId ()

Returns

the useCaseId

Usage

the useCaseId

isSetLimitClassId() method

Syntax

boolean isSetLimitClassId ()

Returns

false if the limitClassId is null. Otherwise, it returns true.

Usage

Checks whether the

limitClassId

of this instance is set.

false if the limitClassId is null. Otherwise, it returns true.

isSetLimitSetId() method

Syntax

boolean isSetLimitSetId ()

Returns

false if the limitSetId is null. Otherwise, it returns true.

Usage

Checks whether the

limitSetId

of this instance is set.

false if the limitSetId is null. Otherwise, it returns true.

isSetUseCaseId() method

Syntax

boolean isSetUseCaseId ()

Returns

false if the useCaseId is null . Otherwise, it returns true.

Usage

Checks whether the

useCaseId

of this instance is set.

false if the useCaseId is null . Otherwise, it returns true.

setLimitClassId(long) method

Syntax

void setLimitClassId (long *limitClassId*)

Parameters

- **limitClassId** – the limitClassId to set

setLimitSetId(long) method

Syntax

```
void setLimitSetId ( long limitSetId )
```

Parameters

- **limitSetId** – the limitSetId to set

setUseCaseId(Integer) method

Syntax

```
void setUseCaseId ( Integer useCaseId )
```

Parameters

- **useCaseId** – the useCaseId to set

LookupQuery class

Syntax

```
public class LookupQuery
```

Usage

A `LookupQuery` provides a default SQL query to retrieve ID and name of lookup entities such as countries, currencies, etc.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

LookupQuery() constructor

Syntax

```
LookupQuery ()
```

LookupQuery(String) constructor

Syntax

```
LookupQuery ( String id )
```

Parameters

- **id** –

Usage

Please use the

`newInstance`

method of the corresponding DAO.

getPrivilege() method

Syntax

`String getPrivilege ()`

getQuery() method

Syntax

`String getQuery ()`

setPrivilege(String) method

Syntax

`void setPrivilege ( String privilege )`

setQuery(String) method

Syntax

`void setQuery ( String query )`

OrgUnit class

Syntax

`public class OrgUnit`

Usage

The `OrgUnit` enables multi-client capability for the paybox system.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

*OrgUnit() constructor***Syntax**`OrgUnit ()`**Usage**

Creates a new instance of

`OrgUnit`

.

*OrgUnit(String) constructor***Syntax**`OrgUnit ( String id )`**Parameters**

- **id** – the PK value

Usage

Creates a new instance of

`OrgUnit`

with a preset primary key.

*getAutoCancelAfterMinutes() method***Syntax**`Integer getAutoCancelAfterMinutes ()`*getCurrency() method***Syntax**`Currency getCurrency ()`**Returns**

the OrgUnit's Currency

Usage

Returns the default currency of this

OrgUnit

.

the OrgUnit's Currency

getFeeSetId() method

Syntax

Long getFeeSetId ()

getLegalAddress() method

Syntax

Address getLegalAddress ()

getLimitSetId() method

Syntax

Long getLimitSetId ()

getLocale() method

Syntax

Locale getLocale ()

Returns

the OrgUnit's Locale

Usage

Returns the locale of this

OrgUnit

.

the OrgUnit's Locale

getName() method

Syntax

String getName ()

Returns

the name

Usage

Returns the long name of this

OrgUnit

.

the name

getTimeZone() method

Syntax

TimeZone getTimeZone ()

Returns

the OrgUnit's TimeZone

Usage

Returns the default time zone of the

OrgUnit

.

the OrgUnit's TimeZone

getVat() method

Syntax

BigDecimal getVat ()

Returns

a BigDecimal representing the VAT percentage

Usage

Returns the VAT value.

a BigDecimal representing the VAT percentage

isSetAutoCancelAfterMinutes() method

Syntax

boolean isSetAutoCancelAfterMinutes ()

Returns

false if the autoCancelAfterMinutes is null. Otherwise, it returns true.

Usage

Checks whether the

autoCancelAfterMinutes

of this instance is set.

false if the autoCancelAfterMinutes is null. Otherwise, it returns true.

isSetCurrency() method

Syntax

boolean isSetCurrency ()

Returns

false if the dbCurrency is null. Otherwise, it returns true.

Usage

Checks whether the

dbCurrency

of this instance is set.

false if the dbCurrency is null. Otherwise, it returns true.

isSetFeeSetId() method

Syntax

boolean isSetFeeSetId ()

Returns

false if the feeSetId is null. Otherwise, it returns true.

Usage

Checks whether the

feeSetId

of this instance is set.

false if the feeSetId is null. Otherwise, it returns true.

isSetLegalAddress() method

Syntax

```
boolean isSetLegalAddress ()
```

Returns

false if the legalAddress is null. Otherwise, it returns true.

Usage

Checks whether the

legalAddress

of this instance is set.

false if the legalAddress is null. Otherwise, it returns true.

isSetLimitSetId() method

Syntax

```
boolean isSetLimitSetId ()
```

Returns

false if the limitSetId is null. Otherwise, it returns true.

Usage

Checks whether the

limitSetId

of this instance is set.

false if the limitSetId is null. Otherwise, it returns true.

isSetLocale() method

Syntax

boolean isSetLocale ()

Returns

false if the dbLanguage and dbCountry is null. Otherwise, it returns true.

Usage

Checks whether the

dbLanguage

or

dbCountry

of this instance is set.

false if the dbLanguage and dbCountry is null. Otherwise, it returns true.

isSetName() method

Syntax

boolean isSetName ()

Returns

false if the name is null. Otherwise, it returns true.

Usage

Checks whether the

name

of this instance is set.

false if the name is null. Otherwise, it returns true.

isSetTimeZone() method

Syntax

boolean isSetTimeZone ()

Returns

false if the dbTimeZone is null. Otherwise, it returns true.

Usage

Checks whether the

dbTimeZone

of this instance is set.

false if the *dbTimeZone* is null. Otherwise, it returns true.

isSetVat() method

Syntax

boolean isSetVat ()

Returns

false if the *vat* is null. Otherwise, it returns true.

Usage

Checks whether the

vat

of this instance is set.

false if the *vat* is null. Otherwise, it returns true.

setAutoCancelAfterMinutes(Integer) method

Syntax

void setAutoCancelAfterMinutes (Integer *autoCancelAfterMinutes*)

setCurrency(Currency) method

Syntax

void setCurrency (Currency *currency*)

Parameters

- **currency** – the *currency* value to set

Usage

Sets the

currency

of this instance.

setFeeSetId(Long) method

Syntax

void setFeeSetId (Long *feeSetId*)

setLegalAddress(Address) method

Syntax

void setLegalAddress (Address *legalAddress*)

setLimitSetId(Long) method

Syntax

void setLimitSetId (Long *limitSetId*)

setLocale(Locale) method

Syntax

void setLocale (Locale *locale*)

Parameters

- **locale** – the locale value to set

Usage

Sets the

locale

of this instance.

setName(String) method

Syntax

void setName (String *name*)

Parameters

- **name** – the name value to set

Usage

Sets the

name

of this instance.

setTimeZone(TimeZone) method

Syntax

```
void setTimeZone ( TimeZone timeZone )
```

Parameters

- **timeZone** – the `timeZone` value to set

Usage

Sets the

timeZone

of this instance.

setVat(BigDecimal) method

Syntax

```
void setVat ( BigDecimal vat )
```

Parameters

- **vat** – the `vat` value to set

Usage

Sets the

vat

of this instance.

OtpType class

Syntax

```
public class OtpType
```

Usage

This entity defines a type of a one-time password.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

OtpType() constructor

Syntax

OtpType ()

Usage

Creates a new instance of

OtpType

.

OtpType(int) constructor

Syntax

OtpType (int *id*)

Parameters

- **id** – the PK value

Usage

Creates a new instance of

OtpType

with a preset primary key.

getLength() method

Syntax

int getLength ()

Returns

the OTP's length

Usage

Returns the length (character amount) of an OTP.

the OTP's length

getMaxWrongEntries() method

Syntax

```
int getMaxWrongEntries ()
```

Returns

the maxWrongEntries

Usage

the maxWrongEntries

getMessageTemplate() method

Syntax

```
String getMessageTemplate ()
```

getMessageTemplateType() method

Syntax

```
String getMessageTemplateType ()
```

getSentLimit() method

Syntax

```
Integer getSentLimit ()
```

getValidityPeriod() method

Syntax

```
Integer getValidityPeriod ()
```

Returns

the validity period (in [s]) (may be null)

Usage

Returns the validity period. If no period is given, the according OTP is valid forever.

the validity period (in [s]) (may be null)

isNumeric() method

Syntax

boolean isNumeric ()

Returns

true if a numeric OTP is defined; false if this defines an alphanumeric OTP

Usage

Indicates if this type defines a numeric or an alphanumeric OTP.

true if a numeric OTP is defined; false if this defines an alphanumeric OTP

isSetLength() method

Syntax

boolean isSetLength ()

Returns

false if the length is null. Otherwise, it returns true.

Usage

Checks whether the

length

of this instance is set.

false if the length is null. Otherwise, it returns true.

isSetMaxWrongEntries() method

Syntax

boolean isSetMaxWrongEntries ()

Returns

false if the maxWrongEntries is null. Otherwise, it returns true.

Usage

Checks whether the

maxWrongEntries

of this instance is set.

false if the `maxWrongEntries` is null. Otherwise, it returns true.

isSetMessageTemplate() method

Syntax

```
boolean isSetMessageTemplate ()
```

Returns

false if the `messageTemplate` is null. Otherwise, it returns true.

Usage

Checks whether the

messageTemplate

of this instance is set.

false if the `messageTemplate` is null. Otherwise, it returns true.

isSetMessageTemplateType() method

Syntax

```
boolean isSetMessageTemplateType ()
```

isSetNumeric() method

Syntax

```
boolean isSetNumeric ()
```

Returns

false if the `dbNumeric` is null . Otherwise, it returns true.

Usage

Checks whether the

dbNumeric

of this instance is set.

false if the `dbNumeric` is null . Otherwise, it returns true.

isSetSentLimit() method

Syntax

boolean isSetSentLimit ()

isSetValidityPeriod() method

Syntax

boolean isSetValidityPeriod ()

Returns

false if the validityPeriod is null. Otherwise, it returns true.

Usage

Checks whether the

validityPeriod

of this instance is set.

false if the validityPeriod is null. Otherwise, it returns true.

setLength(int) method

Syntax

void setLength (int *length*)

Parameters

- **length** – the length value to set

Usage

Sets the

length

of this instance.

setMaxWrongEntries(int) method

Syntax

void setMaxWrongEntries (int *maxWrongEntries*)

Parameters

- **maxWrongEntries** – the maxWrongEntries to set

setMessageTemplate(String) method

Syntax

`void setMessageTemplate ( String messageTemplate )`

setMessageTemplateType(String) method

Syntax

`void setMessageTemplateType ( String messageTemplateType )`

setNumeric(boolean) method

Syntax

`void setNumeric ( boolean numeric )`

Parameters

- **numeric** – the numeric value to set

Usage

Sets the

numeric

of this instance.

setSentLimit(Integer) method

Syntax

`void setSentLimit ( Integer sentLimit )`

setValidityPeriod(Integer) method

Syntax

`void setValidityPeriod ( Integer validityPeriod )`

Parameters

- **validityPeriod** – the validityPeriod value to set

Usage

Sets the

validityPeriod

of this instance.

Restriction class

Syntax

```
public class Restriction
```

Usage

The `Restriction` entity represents a transaction restriction. A transaction restriction restricts certain transaction on amount and frequencies.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Restriction() constructor

Syntax

```
Restriction ()
```

Usage

Creates a new instance of

```
Restriction
```

.

compareTo(Restriction) method

Syntax

```
int compareTo ( Restriction o )
```

getName() method

Syntax

```
String getName ()
```

getPriority() method

Syntax

Long getPriority ()

getRestrictionsGroup() method

Syntax

RestrictionsGroup getRestrictionsGroup ()

getRules() method

Syntax

Set< RestrictionRule > getRules ()

Returns

a Set containing the assigned Rules

Usage

Returns the restriction rules assigned to this instance.

a Set containing the assigned Rules

isPayer() method

Syntax

boolean isPayer ()

isSetName() method

Syntax

boolean isSetName ()

isSetPayer() method

Syntax

boolean isSetPayer ()

Returns

false if the dbIsPayer is null . Otherwise, it returns true.

Usage

Checks whether the

dbIsPayer

of this instance is set.

false if the dbIsPayer is null . Otherwise, it returns true.

isSetPriority() method

Syntax

boolean isSetPriority ()

isSetRestrictionsGroup() method

Syntax

boolean isSetRestrictionsGroup ()

isSetRules() method

Syntax

boolean isSetRules ()

Returns

false if the rules is null. Otherwise, it returns true.

Usage

Checks whether the

rules

of this instance is set.

false if the rules is null. Otherwise, it returns true.

setName(String) method

Syntax

void setName (String *name*)

setPayer(boolean) method

Syntax

void setPayer (boolean *isPayer*)

Parameters

- **isPayer** – the `isPayer` value to set

Usage

Sets the

isPayer

of this instance.

setPriority(Long) method

Syntax

`void setPriority ( Long priority )`

setRestrictionsGroup(RestrictionsGroup) method

Syntax

`void setRestrictionsGroup ( RestrictionsGroup restrictionsGroup )`

setRules(Set< RestrictionRule >) method

Syntax

`void setRules ( Set< RestrictionRule > rules )`

Parameters

- **rules** – the `rules` value to set

Usage

Sets the

rules

of this instance.

RestrictionForOrgUnit class

Syntax

`public class RestrictionForOrgUnit`

Usage

The `RestrictionForOrgUnit` entity represents a transaction restriction for an org unit.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

RestrictionForOrgUnitPK class

Syntax

```
public class RestrictionForOrgUnitPK
```

RestrictionForOrgUnitPK() constructor

Syntax

```
RestrictionForOrgUnitPK ()
```

Usage

Creates a new instance of

```
RestrictionForOrgUnitPK
```

.

RestrictionForOrgUnitPK(Restriction, OrgUnit) constructor

Syntax

```
RestrictionForOrgUnitPK ( Restriction restriction ,   OrgUnit orgUnit )
```

Parameters

- **restriction** – the restriction
- **orgUnit** – the org unit

Usage

Creates a new instance of

```
RestrictionForOrgUnitPK
```

.

equals(Object) method

Syntax

```
boolean equals ( Object o )
```


Parameters

- **o** – the Object to compare with

Returns

true if this class is assignable from o's class and has the same PK values as this instance

Usage

Indicates if another object is

equal to

this one.

true if this class is assignable from o's class and has the same PK values as this instance

getOrgUnit() method

Syntax

```
OrgUnit getOrgUnit ()
```

getRestriction() method

Syntax

```
Restriction getRestriction ()
```

hashCode() method

Syntax

```
int hashCode ()
```

Returns

a hash value for this instance

Usage

Returns a hash value.

a hash value for this instance

isSetOrgUnit() method

Syntax

```
boolean isSetOrgUnit ()
```

isSetRestriction() method

Syntax

boolean isSetRestriction ()

setOrgUnit(OrgUnit) method

Syntax

void setOrgUnit (OrgUnit *orgUnit*)

setRestriction(Restriction) method

Syntax

void setRestriction (Restriction *restriction*)

toString() method

Syntax

String toString ()

RestrictionForOrgUnit() constructor

Syntax

RestrictionForOrgUnit ()

Usage

Creates a new instance of

RestrictionForOrgUnit

.

RestrictionForOrgUnit(RestrictionForOrgUnitPK) constructor

Syntax

RestrictionForOrgUnit (RestrictionForOrgUnitPK *pk*)

Parameters

- **pk** – the PK value

Usage

Creates a new instance of
 RestrictionForOrgUnit

.

RestrictionForRiskCategory class**Syntax**

```
public class RestrictionForRiskCategory
```

Usage

The `RestrictionForRiskCategory` entity represents a transaction restriction for risk category.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

RestrictionForRiskCategoryPK class**Syntax**

```
public class RestrictionForRiskCategoryPK
```

RestrictionForRiskCategoryPK() constructor**Syntax**

```
RestrictionForRiskCategoryPK ()
```

Usage

Creates a new instance of
 RestrictionForRiskCategoryPK

.

RestrictionForRiskCategoryPK(Restriction, RiskCategory) constructor**Syntax**

```
RestrictionForRiskCategoryPK ( Restriction restriction ,  

  RiskCategory riskCategory )
```

Parameters

- **restriction** – the restriction
- **riskCategory** – the risk category

Usage

Creates a new instance of

RestrictionForRiskCategoryPK

.

equals(Object) method

Syntax

boolean equals (Object *o*)

Parameters

- **o** – the Object to compare with

Returns

true if this class is assignable from o's class and has the same PK values as this instance

Usage

Indicates if another object is

equal to

this one.

true if this class is assignable from o's class and has the same PK values as this instance

getRestriction() method

Syntax

Restriction getRestriction ()

getRiskCategory() method

Syntax

RiskCategory getRiskCategory ()

hashCode() method

Syntax

`int hashCode ()`

Returns

a hash value for this instance

Usage

Returns a hash value.

a hash value for this instance

isSetRestriction() method

Syntax

`boolean isSetRestriction ()`

isSetRiskCategory() method

Syntax

`boolean isSetRiskCategory ()`

setRestriction(Restriction) method

Syntax

`void setRestriction ( Restriction restriction )`

setRiskCategory(RiskCategory) method

Syntax

`void setRiskCategory ( RiskCategory orgUnit )`

toString() method

Syntax

`String toString ()`

RestrictionForRiskCategory() constructor

Syntax

RestrictionForRiskCategory ()

Usage

Creates a new instance of

RestrictionForRiskCategory

.

RestrictionForRiskCategory(RestrictionForRiskCategoryPK) constructor

Syntax

RestrictionForRiskCategory (RestrictionForRiskCategoryPK *pk*)

Parameters

- **pk** – the PK value

Usage

Creates a new instance of

RestrictionForRiskCategory

.

RestrictionOnOrgUnit class

Syntax

```
public class RestrictionOnOrgUnit
```

Usage

The `RestrictionOnOrgUnit` entity represents a transaction restriction which is applied on an org unit.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

*RestrictionOnOrgUnitPK class***Syntax**

```
public class RestrictionOnOrgUnitPK
```

*RestrictionOnOrgUnitPK() constructor***Syntax**

```
RestrictionOnOrgUnitPK ()
```

Usage

Creates a new instance of

```
RestrictionOnOrgUnitPK
```

.

*RestrictionOnOrgUnitPK(Restriction, OrgUnit) constructor***Syntax**

```
RestrictionOnOrgUnitPK ( Restriction restriction ,   OrgUnit orgUnit )
```

Parameters

- **restriction** – the restriction
- **orgUnit** – the org unit

Usage

Creates a new instance of

```
RestrictionOnOrgUnitPK
```

.

*equals(Object) method***Syntax**

```
boolean equals ( Object o )
```

Parameters

- **o** – the Object to compare with

Returns

true if this class is assignable from o's class and has the same PK values as this instance

Usage

Indicates if another object is

equal to

this one.

true if this class is assignable from o's class and has the same PK values as this instance

getOrgUnit() method

Syntax

OrgUnit getOrgUnit ()

getRestriction() method

Syntax

Restriction getRestriction ()

hashCode() method

Syntax

int hashCode ()

Returns

a hash value for this instance

Usage

Returns a hash value.

a hash value for this instance

isSetOrgUnit() method

Syntax

boolean isSetOrgUnit ()

isSetRestriction() method

Syntax

boolean isSetRestriction ()

setOrgUnit(OrgUnit) method

Syntax

void setOrgUnit (OrgUnit *orgUnit*)

setRestriction(Restriction) method

Syntax

void setRestriction (Restriction *restriction*)

toString() method

Syntax

String toString ()

RestrictionOnOrgUnit() constructor

Syntax

RestrictionOnOrgUnit ()

Usage

Creates a new instance of

RestrictionOnOrgUnit

.

RestrictionOnOrgUnit(RestrictionOnOrgUnitPK) constructor

Syntax

RestrictionOnOrgUnit (RestrictionOnOrgUnitPK *pk*)

Parameters

- **pk** – the PK value

Usage

Creates a new instance of

RestrictionOnOrgUnit

.

RestrictionOnRiskCategory class

Syntax

```
public class RestrictionOnRiskCategory
```

Usage

The `RestrictionOnRiskCategory` entity represents a transaction restriction which is applied on a risk category.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

RestrictionOnRiskCategoryPK class

Syntax

```
public class RestrictionOnRiskCategoryPK
```

RestrictionOnRiskCategoryPK() constructor

Syntax

```
RestrictionOnRiskCategoryPK ()
```

Usage

Creates a new instance of

```
RestrictionOnRiskCategoryPK
```

.

RestrictionOnRiskCategoryPK(Restriction, RiskCategory) constructor

Syntax

```
RestrictionOnRiskCategoryPK ( Restriction restriction , RiskCategory  
riskCategory )
```

Parameters

- **restriction** – the restriction
- **riskCategory** – the risk category

Usage

Creates a new instance of
 RestrictionOnRiskCategoryPK

.

equals(Object) method

Syntax

boolean equals (Object *o*)

Parameters

- **o** – the Object to compare with

Returns

true if this class is assignable from o's class and has the same PK values as this instance

Usage

Indicates if another object is

equal to

this one.

true if this class is assignable from o's class and has the same PK values as this instance

getRestriction() method

Syntax

Restriction getRestriction ()

getRiskCategory() method

Syntax

RiskCategory getRiskCategory ()

hashCode() method

Syntax

int hashCode ()

Returns

a hash value for this instance

Usage

Returns a hash value.

a hash value for this instance

isSetRestriction() method

Syntax

boolean isSetRestriction ()

isSetRiskCategory() method

Syntax

boolean isSetRiskCategory ()

setRestriction(Restriction) method

Syntax

void setRestriction (Restriction *restriction*)

setRiskCategory(RiskCategory) method

Syntax

void setRiskCategory (RiskCategory *orgUnit*)

toString() method

Syntax

String toString ()

RestrictionOnRiskCategory() constructor

Syntax

RestrictionOnRiskCategory ()

Usage

Creates a new instance of

RestrictionOnRiskCategory

.

RestrictionOnRiskCategory(RestrictionOnRiskCategoryPK) constructor

Syntax

`RestrictionOnRiskCategory ( RestrictionOnRiskCategoryPK pk )`

Parameters

- **pk** – the PK value

Usage

Creates a new instance of

`RestrictionOnRiskCategory`

.

RestrictionOnUseCase class

Syntax

```
public class RestrictionOnUseCase
```

Usage

The `RestrictionOnUseCase` entity represents a transaction restriction which is applied on an use case.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

RestrictionOnUseCasePK class

Syntax

```
public class RestrictionOnUseCasePK
```

RestrictionOnUseCasePK() constructor

Syntax

`RestrictionOnUseCasePK ()`

Usage

Creates a new instance of

RestrictionOnUseCasePK

.

RestrictionOnUseCasePK(Restriction, UseCase) constructor

Syntax

RestrictionOnUseCasePK (Restriction *restriction* , UseCase *useCase*)

Parameters

- **restriction** – the restriction
- **useCase** – the use case

Usage

Creates a new instance of

RestrictionOnUseCasePK

.

equals(Object) method

Syntax

boolean equals (Object *o*)

Parameters

- **o** – the Object to compare with

Returns

true if this class is assignable from o's class and has the same PK values as this instance

Usage

Indicates if another object is

equal to

this one.

true if this class is assignable from o's class and has the same PK values as this instance

getRestriction() method

Syntax

`Restriction getRestriction ()`

getUseCase() method

Syntax

`UseCase getUseCase ()`

hashCode() method

Syntax

`int hashCode ()`

Returns

a hash value for this instance

Usage

Returns a hash value.

a hash value for this instance

isSetRestriction() method

Syntax

`boolean isSetRestriction ()`

isSetUseCase() method

Syntax

`boolean isSetUseCase ()`

setRestriction(Restriction) method

Syntax

`void setRestriction ( Restriction restriction )`

setUseCase(UseCase) method

Syntax

```
void setUseCase ( UseCase useCase )
```

toString() method

Syntax

```
String toString ()
```

RestrictionOnUseCase() constructor

Syntax

```
RestrictionOnUseCase ()
```

Usage

Creates a new instance of

```
RestrictionOnUseCase
```

.

RestrictionOnUseCase(RestrictionOnUseCasePK) constructor

Syntax

```
RestrictionOnUseCase ( RestrictionOnUseCasePK pk )
```

Parameters

- **pk** – the PK value

Usage

Creates a new instance of

```
RestrictionOnUseCase
```

.

RestrictionRule class

Syntax

```
public class RestrictionRule
```


Usage

The `RestrictionRule` represents a transaction restriction rule, defining the specific limitations the transaction restriction should enforce.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

RestrictionRule() constructor

Syntax

```
RestrictionRule ()
```

Usage

Creates a new instance of

`RestrictionRule`

.

getCurrency() method

Syntax

```
String getCurrency ()
```

Returns

the currency of the configured amounts.

Usage

Returns the currency of the configured amounts.

the currency of the configured amounts.

getMaxTransactionAmount() method

Syntax

```
Long getMaxTransactionAmount ()
```

Returns

the maximum transaction amount; `null` if there is no such restriction

Usage

Returns the allowed maximum amount (inclusive) for a single transaction.
the maximum transaction amount; `null` if there is no such restriction

getMaxTransactions() method

Syntax

`Long getMaxTransactions ()`

Returns

the maximum number of transactions; `null` if there is no such restriction

Usage

Returns the maximum number of transaction allowed within the
configured timeframe

.

the maximum number of transactions; `null` if there is no such restriction

getMaxTransactionSum() method

Syntax

`Long getMaxTransactionSum ()`

Returns

the maximum transaction amount sum; `null` if there is no such restriction

Usage

Returns the maximum sum of all transaction amounts (of the affected type) allowed within
the

configured timeframe

.

the maximum transaction amount sum; `null` if there is no such restriction

getMinTransactionAmount() method

Syntax

`Long getMinTransactionAmount ()`

Returns

the minimum transaction amount; `null` if there is no such restriction

Usage

Returns the allowed minimum amount (inclusive) for a single transaction.

the minimum transaction amount; `null` if there is no such restriction

getRestriction() method

Syntax

```
Restriction getRestriction ()
```

Returns

the associated `RestrictionSet`

Usage

Returns the restriction set associated to this rule.

the associated `RestrictionSet`

getTimeframe() method

Syntax

```
Duration getTimeframe ()
```

Returns

the timeframe to consider for the given restrictions; `null` if the complete transaction history should be considered

Usage

Returns the timeframe to be considered for the

maximum number of transactions

and

maximum transaction amount sum

restrictions.

the timeframe to consider for the given restrictions; `null` if the complete transaction history should be considered

isSetCurrency() method

Syntax

```
boolean isSetCurrency ()
```

Returns

false if the dbCurrency is null. Otherwise, it returns true.

Usage

Checks whether the

dbCurrency

of this instance is set.

false if the dbCurrency is null. Otherwise, it returns true.

isSetMaxTransactionAmount() method

Syntax

```
boolean isSetMaxTransactionAmount ()
```

Returns

false if the maxTransactionAmount is null. Otherwise, it returns true.

Usage

Checks whether the

maxTransactionAmount

of this instance is set.

false if the maxTransactionAmount is null. Otherwise, it returns true.

isSetMaxTransactions() method

Syntax

```
boolean isSetMaxTransactions ()
```

Returns

false if the maxTransactions is null. Otherwise, it returns true.

Usage

Checks whether the

maxTransactions

of this instance is set.

false if the `maxTransactions` is null. Otherwise, it returns true.

isSetMaxTransactionSum() method

Syntax

```
boolean isSetMaxTransactionSum ()
```

Returns

false if the `maxTransactionSum` is null. Otherwise, it returns true.

Usage

Checks whether the

maxTransactionSum

of this instance is set.

false if the `maxTransactionSum` is null. Otherwise, it returns true.

isSetMinTransactionAmount() method

Syntax

```
boolean isSetMinTransactionAmount ()
```

Returns

false if the `minTransactionAmount` is null. Otherwise, it returns true.

Usage

Checks whether the

minTransactionAmount

of this instance is set.

false if the `minTransactionAmount` is null. Otherwise, it returns true.

isSetRestriction() method

Syntax

```
boolean isSetRestriction ()
```

Returns

false if the `restrictionSet` is null. Otherwise, it returns true.

Usage

Checks whether the

restrictionSet

of this instance is set.

false if the `restrictionSet` is null. Otherwise, it returns true.

isSetTimeframe() method

Syntax

```
boolean isSetTimeframe ()
```

Returns

false if the `dbTimeframe` is null. Otherwise, it returns true.

Usage

Checks whether the

dbTimeframe

of this instance is set.

false if the `dbTimeframe` is null. Otherwise, it returns true.

setCurrency(String) method

Syntax

```
void setCurrency ( String currency )
```

Parameters

- **currency** – the `currency` value to set

Usage

Sets the
currency
of this instance.

setMaxTransactionAmount(Long) method

Syntax

`void setMaxTransactionAmount ( Long maxTransactionAmount )`

Parameters

- **maxTransactionAmount** – the `maxTransactionAmount` value to set

Usage

Sets the
maxTransactionAmount
of this instance.

setMaxTransactions(Long) method

Syntax

`void setMaxTransactions ( Long maxTransactions )`

Parameters

- **maxTransactions** – the `maxTransactions` value to set

Usage

Sets the
maxTransactions
of this instance.

setMaxTransactionSum(Long) method

Syntax

`void setMaxTransactionSum ( Long maxTransactionSum )`

Parameters

- **maxTransactionSum** – the `maxTransactionSum` value to set

Usage

Sets the

maxTransactionSum

of this instance.

setMinTransactionAmount(Long) method

Syntax

```
void setMinTransactionAmount ( Long minTransactionAmount )
```

Parameters

- **minTransactionAmount** – the `minTransactionAmount` value to set

Usage

Sets the

minTransactionAmount

of this instance.

setRestriction(Restriction) method

Syntax

```
void setRestriction ( Restriction restriction )
```

Parameters

- **restrictionSet** – the `restrictionSet` value to set

Usage

Sets the

restrictionSet

of this instance.

setTimeframe(Duration) method

Syntax

```
void setTimeframe ( Duration timeframe )
```

Parameters

- **timeframe** – the timeframe value to set

Usage

Sets the

timeframe

of this instance.

RestrictionsGroup class

Syntax

```
public class RestrictionsGroup
```

Usage

The RestrictionsGroup entity represents a lookup for transaction restriction types.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

RestrictionsGroup() constructor

Syntax

```
RestrictionsGroup ()
```

Usage

Creates a new instance of

RestrictionsGroup

.

RestrictionsGroup(int) constructor

Syntax

RestrictionsGroup (int *id*)

Parameters

- **id** – the PK value

Usage

Creates a new instance of

RestrictionsGroup

.

RiskCategory class

Syntax

```
public class RiskCategory
```

Usage

Maps the table MOB_RISK_CATEGORY. Only valuable information in here is the default limit set id.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

RiskCategory() constructor

Syntax

RiskCategory ()

Usage

Creates a new instance of

RiskCategory

.

RiskCategory(int) constructor

Syntax

`RiskCategory ( int id )`

Parameters

- **id** – the PK value

Usage

Creates a new instance of

`RiskCategory`

with a preset primary key.

getLimitSetId() method

Syntax

`Long getLimitSetId ()`

getRole() method

Syntax

`String getRole ()`

isSetLimitSetId() method

Syntax

`boolean isSetLimitSetId ()`

Returns

false if the `limitSetId` is null. Otherwise, it returns true.

Usage

Checks whether the

limitSetId

of this instance is set.

false if the `limitSetId` is null. Otherwise, it returns true.

isSetRole() method

Syntax

```
boolean isSetRole ()
```

Returns

false if the role is null. Otherwise, it returns true.

Usage

Checks whether the

role

of this instance is set.

false if the role is null. Otherwise, it returns true.

setLimitSetId(Long) method

Syntax

```
void setLimitSetId ( Long limitSetId )
```

setRole(String) method

Syntax

```
void setRole ( String role )
```

Scale class

Syntax

```
public class Scale
```

Usage

The `Scale` models a generic scale that can be used for fees, commissions and the like.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

*Period class**Syntax*

```
public class Period
```

Usage

The `Period` models a specific validity period of a scale. There must be at least one period for each scale. Multiple periods can be defined for a single scale, but there is only one that is active.

© 2000-2008 by paybox solutions AG

Revision:1.12Name: not supported by cvs2svn

*Period() constructor***Syntax**

```
Period ()
```

Usage

Creates a new instance of

```
Period
```

```
.
```

*getCurrency() method***Syntax**

```
Currency getCurrency ()
```

Returns

the `Currency`

Usage

Returns the currency of this scale period.

the `Currency`

*getScale() method***Syntax**

```
Scale getScale ()
```

Returns

the associated Scale

Usage

Returns the scale this period is associated with.

the associated Scale

getSteps() method

Syntax

```
List< Step > getSteps ()
```

Returns

a List of the Steps assigned to this period

Usage

Returns the steps of this period.

a List of the Steps assigned to this period

getValidFrom() method

Syntax

```
Date getValidFrom ()
```

Returns

the valid-from Date

Usage

Returns the point in time since when this period is valid.

the valid-from Date

isSetCurrency() method

Syntax

```
boolean isSetCurrency ()
```

Returns

false if the dbCurrency is null. Otherwise, it returns true.

Usage

Checks whether the

dbCurrency

of this instance is set.

false if the dbCurrency is null. Otherwise, it returns true.

isSetScale() method

Syntax

```
boolean isSetScale ()
```

Returns

false if the scale is null . Otherwise, it returns true.

Usage

Checks whether the

scale

of this instance is set.

false if the scale is null . Otherwise, it returns true.

isSetSteps() method

Syntax

```
boolean isSetSteps ()
```

Returns

false if the steps is null . Otherwise, it returns true.

Usage

Checks whether the

steps

of this instance is set.

false if the steps is null . Otherwise, it returns true.

isValidFrom() method

Syntax

boolean isValidFrom ()

Returns

false if the validFrom is null. Otherwise, it returns true.

Usage

Checks whether the

validFrom

of this instance is set.

false if the validFrom is null. Otherwise, it returns true.

setCurrency(Currency) method

Syntax

void setCurrency (Currency *currency*)

Parameters

- **currency** – the currency value to set

Usage

Sets the

currency

of this instance.

setScale(Scale) method

Syntax

void setScale (Scale *scale*)

Parameters

- **scale** – the scale value to set

Usage

Sets the

scale

of this instance.

setSteps(List< Step >) method

Syntax

```
void setSteps ( List< Step > steps )
```

Parameters

- **steps** – the steps value to set

Usage

Sets the

steps

of this instance.

setValidFrom(Date) method

Syntax

```
void setValidFrom ( Date validFrom )
```

Parameters

- **validFrom** – the validFrom value to set

Usage

Sets the

validFrom

of this instance.

Step class

Syntax

```
public class Step
```

Usage

The `Steps` are used to define the various elements of a scale. The actual step to apply is determined by a threshold value.

© 2000-2007 by paybox solutions AG

Revision:1.12Name: not supported by cvs2svn

Step() constructor

Syntax

`Step ()`

Usage

Creates a new instance of

`Step`

.

Step(StepPk) constructor

Syntax

`Step ( StepPk id )`

Parameters

- **id** – the PK value

Usage

Creates a new instance of

`Step`

with a preset primary key.

calculate(BigDecimal) method

Syntax

`final BigDecimal calculate ( BigDecimal amount )`

Parameters

- **amount** – the amount to base the calculation on

Returns

the calculated amount as `BigDecimal`

Usage

Calculates the sum for the specified amount.

This method returns the result of

```
max(minAmount, min(maxAmount, onTopAmount + amount *
percentage / 100))
```

; given that a

`maxAmount`

is set. If there is no such max value, the

`min()`

operation is omitted.

the calculated amount as `BigDecimal`

calculate(long) method

Syntax

```
final BigDecimal calculate ( long amount )
```

Parameters

- **amount** – the amount to base the calculation on (as `long`)

Returns

the calculated amount as `BigDecimal`

Usage

Calculates the sum for the specified amount.

This method returns the result of

```
max(minAmount, min(maxAmount, onTopAmount + amount *
percentage / 100))
```

; given that a

`maxAmount`

is set. If there is no such max value, the

`min()`

operation is omitted.

the calculated amount as `BigDecimal`

calculateNetAmount(BigDecimal) method

Syntax

```
final BigDecimal calculateNetAmount ( BigDecimal grossAmount )
```

Parameters

- **grossAmount** – the gross amount to base the calculation on

Returns

the supposed net amount as `BigDecimal`

Usage

Calculates the net amount for the specified amount (that is supposed to include a sum calculated by the associated scale/step).

This method returns the result of

```
grossAmount - max(minAmount, min(maxAmount, diff))  
with  
diff = (grossAmount - onTopAmount) / (1 + percentage / 100)
```

given that a

`maxAmount`

is set. If there is no such max value, the

`min()`

operation is omitted.

the supposed net amount as `BigDecimal`

calculateNetAmount(long) method

Syntax

```
final BigDecimal calculateNetAmount ( long grossAmount )
```

Parameters

- **grossAmount** – the gross amount to base the calculation on (as `long`)

Returns

the supposed net amount as `BigDecimal`

Usage

Calculates the net amount for the specified amount (that is supposed to include a sum calculated by the associated scale/step).

the supposed net amount as `BigDecimal`

getMaximumAmount() method

Syntax

```
Long getMaximumAmount ()
```

Returns

the maximum amount; `null` if there is no limit

Usage

Returns the maximum amount to apply for this step.

the maximum amount; `null` if there is no limit

getMinimumAmount() method

Syntax

```
long getMinimumAmount ()
```

Returns

the minimum amount

Usage

Returns the minimum amount to apply for this step.

the minimum amount

getOnTopAmount() method

Syntax

```
long getOnTopAmount ()
```

Returns

the on-top amount

Usage

Returns the on-top amount to apply for this step.
the on-top amount

getPercentage() method

Syntax

BigDecimal getPercentage ()

Returns

the percentage

Usage

Returns the percentage to apply for this step.
the percentage

getPeriod() method

Syntax

Scale.Period getPeriod ()

Returns

the associated Scale.Period

Usage

Returns the scale period this instance is associated with.
the associated Scale.Period

getThreshold() method

Syntax

long getThreshold ()

Returns

the threshold amount

Usage

Returns the threshold amount determining if this step is applicable.

the threshold amount

isSetMaximumAmount() method

Syntax

```
boolean isSetMaximumAmount ()
```

Returns

false if the maximumAmount is null. Otherwise, it returns true.

Usage

Checks whether the

maximumAmount

of this instance is set.

false if the maximumAmount is null. Otherwise, it returns true.

isSetMinimumAmount() method

Syntax

```
boolean isSetMinimumAmount ()
```

Returns

false if the minimumAmount is null. Otherwise, it returns true.

Usage

Checks whether the

minimumAmount

of this instance is set.

false if the minimumAmount is null. Otherwise, it returns true.

isSetOnTopAmount() method

Syntax

```
boolean isSetOnTopAmount ()
```

Returns

false if the onTopAmount is null. Otherwise, it returns true.

Usage

Checks whether the

onTopAmount

of this instance is set.

false if the onTopAmount is null. Otherwise, it returns true.

isSetPercentage() method

Syntax

```
boolean isSetPercentage ()
```

Returns

false if the percentage is null. Otherwise, it returns true.

Usage

Checks whether the

percentage

of this instance is set.

false if the percentage is null. Otherwise, it returns true.

isSetPeriod() method

Syntax

```
boolean isSetPeriod ()
```

Returns

false if the id or the id's period is null. Otherwise, it returns true.

Usage

Checks whether the

period

of this instance's PK is set.

false if the id or the id's period is null. Otherwise, it returns true.

isSetThreshold() method

Syntax

`boolean isSetThreshold ()`

Returns

false if the id or the id's threshold is null. Otherwise, it returns true.

Usage

Checks whether the

threshold

of this instance's PK is set.

false if the id or the id's threshold is null. Otherwise, it returns true.

setMaximumAmount(Long) method

Syntax

`void setMaximumAmount ( Long maximumAmount )`

Parameters

- **maximumAmount** – the `maximumAmount` value to set

Usage

Sets the

maximumAmount

of this instance.

setMinimumAmount(long) method

Syntax

`void setMinimumAmount ( long minimumAmount )`

Parameters

- **minimumAmount** – the `minimumAmount` value to set

Usage

Sets the

minimumAmount

of this instance.

setOnTopAmount(long) method

Syntax

```
void setOnTopAmount ( long onTopAmount )
```

Parameters

- **onTopAmount** – the onTopAmount value to set

Usage

Sets the

onTopAmount

of this instance.

setPercentage(BigDecimal) method

Syntax

```
void setPercentage ( BigDecimal percentage )
```

Parameters

- **percentage** – the percentage value to set

Usage

Sets the

percentage

of this instance.

StepPk class

Syntax

```
public class StepPk
```

Usage

The StepPk class is the embeddable primary key object for a scale's step.

© 2000-2007 by paybox solutions AG

Revision:1.12Name:not supported by cvs2svn

StepPk() constructor

Syntax

StepPk ()

Usage

Creates a new instance of

StepPk

.

StepPk(Period, long) constructor

Syntax

StepPk (Period *period* , long *threshold*)

Parameters

- **period** –
- **threshold** –

Usage

Creates a new instance of

StepPk

.

equals(Object) method

Syntax

boolean equals (Object *o*)

Parameters

- **o** – the Object to compare with

Returns

true if this class is assignable from o's class and has the same PK values as this instance

Usage

Indicates if another object is

equal to

this one.

true if this class is assignable from o's class and has the same PK values as this instance

getPeriod() method

Syntax

Period getPeriod ()

Returns

the period

Usage

the period

getThreshold() method

Syntax

long getThreshold ()

Returns

the threshold

Usage

the threshold

hashCode() method

Syntax

int hashCode ()

Returns

a hash value for this instance

Usage

Returns a hash value.

a hash value for this instance

isSetPeriod() method

Syntax

`boolean isSetPeriod ()`

Returns

false if the period is null. Otherwise, it returns true.

Usage

Checks whether the

period

of this instance is set.

false if the period is null. Otherwise, it returns true.

isSetThreshold() method

Syntax

`boolean isSetThreshold ()`

Returns

false if the threshold is null. Otherwise, it returns true.

Usage

Checks whether the

threshold

of this instance is set.

false if the threshold is null. Otherwise, it returns true.

toString() method

Syntax

`String toString ()`

Returns

a `String` representing this instance

Usage

Returns a string representation of this object.

a `String` representing this instance

Scale() constructor

Syntax

`Scale ()`

Usage

Creates a new instance of

`Scale`

.

calculate(Step, long) method

Syntax

`final BigDecimal calculate ( Step step , long amount )`

Parameters

- **step** – the `Scale.Step` to consider
- **amount** – the amount to base the calculation on

Returns

the calculated amount as `BigDecimal`

Usage

Calculates the sum for the specified (scale) step and amount.

This method returns the result of

```
max(minAmount, min(maxAmount, onTopAmount + amount *  
percentage / 100))
```

; given that a

`maxAmount`

is set. If there is no such max value, the

`min()`

operation is omitted.

the calculated amount as `BigDecimal`

getPeriods() method

Syntax

```
List< Period >getPeriods ()
```

Returns

the periods value

Usage

Returns the

periods

of this instance.

the periods value

isSetPeriods() method

Syntax

```
boolean isSetPeriods ()
```

Returns

false if the periods is null. Otherwise, it returns true.

Usage

Checks whether the

periods

of this instance is set.

false if the periods is null. Otherwise, it returns true.

setPeriods(List< Period >) method

Syntax

```
void setPeriods ( List< Period > periods )
```

Parameters

- **periods** – the periods value to set

Usage

Sets the

periods

of this instance.

LOG variable

The Log instance to use.

Syntax

```
final Logger LOG
```

SessionPolicy class

Syntax

```
public class SessionPolicy
```

Usage

The `SecurityPolicy` models a security policy for the authentication processes, including credential length, block threshold and time, credential expiration and retention.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

SessionPolicy() constructor

Syntax

```
SessionPolicy ()
```

Usage

Creates a new instance of

```
SessionPolicy
```

.

getMaxSessionsAllowed() method

Syntax

```
int getMaxSessionsAllowed ()
```


getName() method

Syntax

String getName ()

Returns

the name

Usage

the name

getSessionTimeoutSeconds() method

Syntax

int getSessionTimeoutSeconds ()

isExtendSessionOnActivity() method

Syntax

boolean isExtendSessionOnActivity ()

isKillOldestWhenLimitHit() method

Syntax

boolean isKillOldestWhenLimitHit ()

isRecycleWhenLimitHit() method

Syntax

boolean isRecycleWhenLimitHit ()

isSetExtendSessionOnActivity() method

Syntax

boolean isSetExtendSessionOnActivity ()

Returns

false if the dbExtendSessionOnActivity is null. Otherwise, it returns true.

Usage

Checks whether the

dbExtendSessionOnActivity

of this instance is set.

false if the dbExtendSessionOnActivity is null. Otherwise, it returns true.

isSetKillOldestWhenLimitHit() method

Syntax

```
boolean isSetKillOldestWhenLimitHit ()
```

Returns

false if the dbKillOldestWhenLimitHit is null. Otherwise, it returns true.

Usage

Checks whether the

dbKillOldestWhenLimitHit

of this instance is set.

false if the dbKillOldestWhenLimitHit is null. Otherwise, it returns true.

isSetMaxSessionsAllowed() method

Syntax

```
boolean isSetMaxSessionsAllowed ()
```

Returns

false if the maxSessionsAllowed is null. Otherwise, it returns true.

Usage

Checks whether the

maxSessionsAllowed

of this instance is set.

false if the maxSessionsAllowed is null. Otherwise, it returns true.

*isSetName() method***Syntax**

```
boolean isSetName ()
```

Returns

false if the name is null. Otherwise, it returns true.

Usage

Checks whether the

name

of this instance is set.

false if the name is null. Otherwise, it returns true.

*isSetRecycleWhenLimitHit() method***Syntax**

```
boolean isSetRecycleWhenLimitHit ()
```

Returns

false if the dbRecycleWhenLimitHit is null. Otherwise, it returns true.

Usage

Checks whether the

dbRecycleWhenLimitHit

of this instance is set.

false if the dbRecycleWhenLimitHit is null. Otherwise, it returns true.

*isSetSessionTimeoutSeconds() method***Syntax**

```
boolean isSetSessionTimeoutSeconds ()
```

Returns

false if the sessionTimeoutSeconds is null. Otherwise, it returns true.

Usage

Checks whether the

sessionTimeoutSeconds

of this instance is set.

false if the *sessionTimeoutSeconds* is null. Otherwise, it returns true.

setExtendSessionOnActivity(boolean) method

Syntax

void *setExtendSessionOnActivity* (boolean *extendSessionOnActivity*)

setKillOldestWhenLimitHit(boolean) method

Syntax

void *setKillOldestWhenLimitHit* (boolean *killOldestWhenLimitHit*)

setMaxSessionsAllowed(int) method

Syntax

void *setMaxSessionsAllowed* (int *maxSessionsAllowed*)

setName(String) method

Syntax

void *setName* (String *name*)

Parameters

- **name** – the name to set

setRecycleWhenLimitHit(boolean) method

Syntax

void *setRecycleWhenLimitHit* (boolean *recycleWhenLimitHit*)

setSessionTimeoutSeconds(int) method

Syntax

void *setSessionTimeoutSeconds* (int *sessionTimeoutSeconds*)

SubTransactionType class**Syntax**

```
public class SubTransactionType
```

Usage

The `SubTransactionType` extends the core `Transaction.Type` by providing additional, Money-specific information.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

SubTransactionType() constructor**Syntax**

```
SubTransactionType ()
```

Usage

Creates a new instance of

```
SubTransactionType
```

```
.
```

SubTransactionType(int) constructor**Syntax**

```
SubTransactionType ( int id )
```

Parameters

- **id** – the PK value

Usage

Creates a new instance of

```
SubTransactionType
```

with a preset primary key.

isBillable() method

Syntax

`boolean isBillable ()`

Returns

`true` if transactions can be billed; `false` otherwise

Usage

Indicates if transactions of this type can be billed.

`true` if transactions can be billed; `false` otherwise

isCancel() method

Syntax

`boolean isCancel ()`

Returns

`true` if transactions are cancel txns; `false` otherwise

Usage

Indicates if transactions of this type are

Cancel

transactions.

`true` if transactions are cancel txns; `false` otherwise

isSetBillable() method

Syntax

`boolean isSetBillable ()`

Returns

`false` if the `dbBillable` is null. Otherwise, it returns `true`.

Usage

Checks whether the

dbBillable

of this instance is set.

false if the dbBillable is null. Otherwise, it returns true.

isSetCancel() method

Syntax

```
boolean isSetCancel ()
```

Returns

false if the dbCancel is null. Otherwise, it returns true.

Usage

Checks whether the

dbCancel

of this instance is set.

false if the dbCancel is null. Otherwise, it returns true.

setBillable(boolean) method

Syntax

```
void setBillable ( boolean billable )
```

Parameters

- **billable** – the billable value to set

Usage

Sets the

billable

of this instance.

setCancel(boolean) method

Syntax

```
void setCancel ( boolean cancel )
```

Parameters

- **cancel** – the cancel value to set

Usage

Sets the

cancel

of this instance.

TransactionTemplateMapping class

Syntax

```
public class TransactionTemplateMapping
```

Usage

The TransactionTemplateMapping holds the configuration for sending transaction related notifications to customers *Money Mobiliser*.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

TransactionTemplateMapping() constructor

Syntax

```
TransactionTemplateMapping ()
```

Usage

Creates a new instance of

```
TransactionTemplateMapping
```

.

getErrorCode() method

Syntax

```
int getErrorCode ()
```

Returns

the errorCode

Usage

the errorCode

getOrgunit() method

Syntax

```
String getOrgunit ()
```

Returns

the orgunit

Usage

the orgunit

getTemplate() method

retrieves the name of the gw-msg template that is bound to the configuration

Syntax

```
String getTemplate ()
```

getTxnStatus() method

Syntax

```
int getTxnStatus ()
```

Returns

the txnStatus

Usage

the txnStatus

getUseCase() method

Syntax

```
int getUseCase ()
```

Returns

the useCase

Usage

the useCase

isActive() method

Syntax

```
boolean isActive ()
```

Returns

Boolean#TRUE if this fee applies to the payee (i.e. the payee must pay the fee);

Boolean#FALSE if it applies to the payer; null if it applies to neither (see getCustomer())

Usage

Indicates if the fee applies to the payee, payer, or neither of them.

Boolean#TRUE if this fee applies to the payee (i.e. the payee must pay the fee);

Boolean#FALSE if it applies to the payer; null if it applies to neither (see getCustomer())

isPayer() method

Syntax

```
boolean isPayer ()
```

Returns

the dbIsPayer

Usage

the dbIsPayer

isSetActive() method

Syntax

```
boolean isSetActive ()
```

Returns

false if the dbActive is null. Otherwise, it returns true.

Usage

Checks whether the

dbActive

of this instance is set.

false if the dbActive is null. Otherwise, it returns true.

isSetErrorCode() method

Syntax

```
boolean isSetErrorCode ()
```

Returns

false if the errorCode is null . Otherwise, it returns true.

Usage

Checks whether the

errorCode

of this instance is set.

false if the errorCode is null . Otherwise, it returns true.

isSetOrgunit() method

Syntax

```
boolean isSetOrgunit ()
```

Returns

false if the orgunit is null. Otherwise, it returns true.

Usage

Checks whether the

orgunit

of this instance is set.

false if the orgunit is null. Otherwise, it returns true.

isSetPayer() method

Syntax

```
boolean isSetPayer ()
```

Returns

false if dbIsPayer is null . Otherwise, it returns true.

Usage

Checks whether the

dbIsPayer

attribute of this instance is set.

false if dbIsPayer is null. Otherwise, it returns true.

isSetTemplate() method

Syntax

```
boolean isSetTemplate ()
```

Returns

false if the template is null. Otherwise, it returns true.

Usage

Checks whether the

template

of this instance is set.

false if the template is null. Otherwise, it returns true.

isSetTest() method

Syntax

```
boolean isSetTest ()
```

Returns

false if dbIsTest is null. Otherwise, it returns true.

Usage

Checks whether the

dbIsTest

attribute of this instance is set.

false if dbIsTest is null. Otherwise, it returns true.

isSetTxnStatus() method

Syntax

```
boolean isSetTxnStatus ()
```

Returns

false if the txnStatus is null . Otherwise, it returns true.

Usage

Checks whether the

txnStatus

of this instance is set.

false if the txnStatus is null . Otherwise, it returns true.

isSetUseCase() method

Syntax

```
boolean isSetUseCase ()
```

Returns

false if the useCase is null. Otherwise, it returns true.

Usage

Checks whether the

useCase

of this instance is set.

false if the useCase is null. Otherwise, it returns true.

isTest() method

Syntax

```
boolean isTest ()
```

Returns

the dbIsTest

Usage

the dbIsTest

setActive(boolean) method

Syntax

void setActive (boolean *active*)

Parameters

- **active** – the active value to set

Usage

Sets the

active

of this instance.

setErrorCode(int) method

Syntax

void setErrorCode (int *errorCode*)

setOrgunit(String) method

Syntax

void setOrgunit (String *orgunit*)

setPayer(boolean) method

Syntax

void setPayer (boolean *payer*)

setTemplate(String) method

Syntax

void setTemplate (String *template*)

setTest(boolean) method

Syntax

void setTest (boolean *test*)

setTxnStatus(int) method

Syntax

```
void setTxnStatus ( int txnStatus )
```

setUseCase(int) method

Syntax

```
void setUseCase ( int useCase )
```

DEFAULT_ERROR_CODE variable

This one is used for unknown errors, e.g.

Syntax

```
final int DEFAULT_ERROR_CODE
```

Usage

when there is no template found for a specific error code (!=0) the template for the default error code is returned

UmgrPrivilege class

Syntax

```
public class UmgrPrivilege
```

Usage

The UmgrPrivilege entity represents a (customer) privilege in the mobiliser system. Privileges can be either assigned directly to a customer to via a UmgrRole.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

UmgrPrivilege() constructor

Syntax

```
UmgrPrivilege ()
```

Usage

Creates a new instance of

```
UmgrPrivilege
```

.

UmgrPrivilege(String) constructor

Syntax

UmgrPrivilege (String *id*)

Parameters

- **id** – the PK value

Usage

Creates a new instance of

UmgrPrivilege

with a preset primary key.

getDescription() method

Syntax

String getDescription ()

setDescription(String) method

Syntax

void setDescription (String *description*)

UmgrRole class

Syntax

```
public class UmgrRole
```

Usage

The UmgrRole entity represents a (customer) role in the mobiliser system. Roles can be assigned to Customers, either directly or via the customer type or the risk category.;

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

*UmgrRole() constructor***Syntax**

```
UmgrRole ()
```

Usage

Creates a new instance of

```
UmgrRole
```

```
.
```

*UmgrRole(String) constructor***Syntax**

```
UmgrRole ( String id )
```

Parameters

- **id** – the PK value

Usage

Creates a new instance of

```
UmgrRole
```

with a preset primary key.

*getDescription() method***Syntax**

```
String getDescription ()
```

*isSetDescription() method***Syntax**

```
boolean isSetDescription ()
```

*setDescription(String) method***Syntax**

```
void setDescription ( String description )
```

UmgrRolePrivilege class

Syntax

```
public class UmgrRolePrivilege
```

Usage

This entity maps a UmgrPrivilege to a UmgrRole.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

UmgrRolePrivilegePk class

Syntax

```
public class UmgrRolePrivilegePk
```

Usage

The UmgrRolePrivilegePk class is the embeddable primary key object for a UMGR role privilege.

© 2000-2007 by paybox solutions AG

Revision:1.3Name:not supported by cvs2svn

UmgrRolePrivilegePk() constructor

Syntax

```
UmgrRolePrivilegePk ()
```

Usage

Creates a new instance of

```
UmgrRolePrivilegePk
```

.

UmgrRolePrivilegePk(String, String) constructor

Syntax

```
UmgrRolePrivilegePk ( String role ,   String privilege )
```

Parameters

- **customer** –
- **customerAttributeTypeId** –

Usage

Creates a new instance of
UmgrRolePrivilegePk

.

equals(Object) method

Syntax

boolean equals (Object *obj*)

getPrivilege() method

Syntax

String getPrivilege ()

getRole() method

Syntax

String getRole ()

hashCode() method

Syntax

int hashCode ()

isSetPrivilege() method

Syntax

boolean isSetPrivilege ()

isSetRole() method

Syntax

boolean isSetRole ()

toString() method

Syntax

`String toString ()`

Returns

a `String` representing this instance

Usage

Returns a string representation of this object.

a `String` representing this instance

UmgrRolePrivilege() constructor

Syntax

`UmgrRolePrivilege ()`

Usage

Creates a new instance of

`UmgrRolePrivilege`

.

UmgrRolePrivilege(UmgrRolePrivilegePk) constructor

Syntax

`UmgrRolePrivilege ( UmgrRolePrivilegePk id )`

Parameters

- **id** – the PK value

Usage

Creates a new instance of

`UmgrRolePrivilege`

with a preset primary key.

getPrivilege() method

Syntax

String getPrivilege ()

getRole() method

Syntax

String getRole ()

isSetPrivilege() method

Syntax

boolean isSetPrivilege ()

isSetRole() method

Syntax

boolean isSetRole ()

UseCase class

Syntax

public class UseCase

Usage

The UseCase specifies use cases for money transactions. This information is orthogonal to Transaction.Type and specifies, what *kind* of transaction was done - independently of the technical transaction type. Each use case is configured here by providing additional, Money-specific information.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

UseCase() constructor

Syntax

UseCase ()

Usage

Creates a new instance of

UseCase

.

UseCase(int) constructor

Syntax

UseCase (int *id*)

Parameters

- **id** – the PK value

Usage

Creates a new instance of

UseCase

with a preset primary key.

getAutoCancelAfterMinutes() method

Syntax

Integer getAutoCancelAfterMinutes ()

Returns

the autoCancelAfterMinutes

Usage

the autoCancelAfterMinutes

getPiTypes() method

Syntax

Set< UseCasePaymentInstrumentType > getPiTypes ()

Returns

the required privilege for this use case.

Usage

Returns the required privilege for this use case.

the required privilege for this use case.

isActorPayee() method

Syntax

```
boolean isActorPayee ()
```

Returns

true if the actor is payee; false otherwise

Usage

Indicates if the actor is payee.

true if the actor is payee; false otherwise

isAnonymousParticipantAllowed() method

Syntax

```
boolean isAnonymousParticipantAllowed ()
```

Returns

true if payer and payee may be identical; false otherwise

Usage

Indicates if an unknown participant should be created during transaction processing (with BLR set).

true if payer and payee may be identical; false otherwise

isAutoCaptureForbidden() method

Syntax

```
boolean isAutoCaptureForbidden ()
```

Returns

the dbAutoCaptureForbidden

Usage

the dbAutoCaptureForbidden

isAutoCaptureRequired() method

Syntax

boolean isAutoCaptureRequired ()

Returns

the dbAutoCaptureRequired

Usage

the dbAutoCaptureRequired

isCopyExchangeRateFromAuthorisation() method

Syntax

boolean isCopyExchangeRateFromAuthorisation ()

Returns

true if payer and payee may be identical; false otherwise

Usage

Indicates if the exchange rate from the authorisation should be used for following sub transactions.

true if payer and payee may be identical; false otherwise

isDifferingAmountAllowed() method

Syntax

boolean isDifferingAmountAllowed ()

Returns

true if transaction amounts can differ; false otherwise

Usage

Indicates if transactions of this type allow differing amounts for

sub transactions

, e.g. if the capture amount can be smaller than the authorisation amount.

true if transaction amounts can differ; false otherwise

isDisableLimits() method

Syntax

```
boolean isDisableLimits ()
```

Returns

true if limit checks are to be ignored for this use case; false otherwise

Usage

Indicates if limit checks are to be ignored for this use case.

true if limit checks are to be ignored for this use case; false otherwise

isIdenticalPayerPayeeAllowed() method

Syntax

```
boolean isIdenticalPayerPayeeAllowed ()
```

Returns

true if payer and payee may be identical; false otherwise

Usage

Indicates if payer and payee may be identical.

true if payer and payee may be identical; false otherwise

isPublic() method

Syntax

```
boolean isPublic ()
```

Returns

true if this use case is public; false otherwise

Usage

Indicates if this use case is public or for internal use.

true if this use case is public; false otherwise

isSetActorPayee() method

Syntax

boolean isSetActorPayee ()

Returns

false if the dbIsActorPayee is null. Otherwise, it returns true.

Usage

Checks whether the

dbIsActorPayee

of this instance is set.

false if the dbIsActorPayee is null. Otherwise, it returns true.

isSetAnonymousParticipantAllowed() method

Syntax

boolean isSetAnonymousParticipantAllowed ()

Returns

false if the dbAnonymousParticipantAllowed is null. Otherwise, it returns true.

Usage

Checks whether the

dbAnonymousParticipantAllowed

of this instance is set.

false if the dbAnonymousParticipantAllowed is null. Otherwise, it returns true.

isSetAutoCancelAfterMinutes() method

Syntax

boolean isSetAutoCancelAfterMinutes ()

Returns

false if the autoCancelAfterMinutes is null. Otherwise, it returns true.

Usage

Checks whether the

autoCancelAfterMinutes

of this instance is set.

false if the `autoCancelAfterMinutes` is null. Otherwise, it returns true.

isSetAutoCaptureForbidden() method

Syntax

```
boolean isSetAutoCaptureForbidden ()
```

Returns

false if the `dbAutoCaptureForbidden` is null. Otherwise, it returns true.

Usage

Checks whether the

dbAutoCaptureForbidden

of this instance is set.

false if the `dbAutoCaptureForbidden` is null. Otherwise, it returns true.

isSetAutoCaptureRequired() method

Syntax

```
boolean isSetAutoCaptureRequired ()
```

Returns

false if the `autoCancelAfterMinutes` is null. Otherwise, it returns true.

Usage

Checks whether the

autoCancelAfterMinutes

of this instance is set.

false if the `autoCancelAfterMinutes` is null. Otherwise, it returns true.

isSetCopyExchangeRateFromAuthorisation() method

Syntax

boolean isSetCopyExchangeRateFromAuthorisation ()

Returns

false if the dbCopyExchangeRateFromAuthorisation is null . Otherwise, it returns true.

Usage

Checks whether the

dbCopyExchangeRateFromAuthorisation

of this instance is set.

false if the dbCopyExchangeRateFromAuthorisation is null . Otherwise, it returns true.

isSetDifferingAmountAllowed() method

Syntax

boolean isSetDifferingAmountAllowed ()

Returns

false if the dbDifferingAmountAllowed is null. Otherwise, it returns true.

Usage

Checks whether the

dbDifferingAmountAllowed

of this instance is set.

false if the dbDifferingAmountAllowed is null. Otherwise, it returns true.

isSetDisableLimits() method

Syntax

boolean isSetDisableLimits ()

Returns

false if the dbDisableLimits is null. Otherwise, it returns true.

Usage

Checks whether the

dbDisableLimits

of this instance is set.

false if the dbDisableLimits is null. Otherwise, it returns true.

isSetIdenticalPayerPayeeAllowed() method

Syntax

```
boolean isSetIdenticalPayerPayeeAllowed ()
```

Returns

false if the dbIdenticalPayerPayeeAllowed is null. Otherwise, it returns true.

Usage

Checks whether the

dbIdenticalPayerPayeeAllowed

of this instance is set.

false if the dbIdenticalPayerPayeeAllowed is null. Otherwise, it returns true.

isSetPiTypes() method

Syntax

```
boolean isSetPiTypes ()
```

Returns

false if the piTypes is null. Otherwise, it returns true.

Usage

Checks whether the

piTypes

of this instance is set.

false if the piTypes is null. Otherwise, it returns true.

isSetPublic() method

Syntax

boolean isSetPublic ()

Returns

false if the dbIsPublic is null. Otherwise, it returns true.

Usage

Checks whether the

dbIsPublic

of this instance is set.

false if the dbIsPublic is null. Otherwise, it returns true.

setActorPayee(boolean) method

Syntax

void setActorPayee (boolean *actorPayee*)

Parameters

- **actorPayee** – the actorPayee value to set

Usage

Sets the

actorPayee

of this instance.

setAnonymousParticipantAllowed(boolean) method

Syntax

void setAnonymousParticipantAllowed (boolean
anonymousParticipantAllowed)

Parameters

- **anonymousParticipantAllowed** – the anonymousParticipantAllowed value to set

Usage

Sets the

anonymousParticipantAllowed

of this instance.

setAutoCancelAfterMinutes(Integer) method

Syntax

`void setAutoCancelAfterMinutes ( Integer autoCancelAfterMinutes )`

Parameters

- **autoCancelAfterMinutes** – the autoCancelAfterMinutes to set

setAutoCaptureForbidden(boolean) method

Syntax

`void setAutoCaptureForbidden ( boolean autoCaptureForbidden )`

Parameters

- **dbAutoCaptureForbidden** – the dbAutoCaptureForbidden to set

setAutoCaptureRequired(boolean) method

Syntax

`void setAutoCaptureRequired ( boolean autoCaptureRequired )`

Parameters

- **dbAutoCaptureRequired** – the dbAutoCaptureRequired to set

setCopyExchangeRateFromAuthorisation(boolean) method

Syntax

`void setCopyExchangeRateFromAuthorisation ( boolean copyExchangeRateFromAuthorisation )`

Parameters

- **copyExchangeRateFromAuthorisation** – the `copyExchangeRateFromAuthorisation` value to set

Usage

Sets the

copyExchangeRateFromAuthorisation

of this instance.

setDifferingAmountAllowed(boolean) method

Syntax

void setDifferingAmountAllowed (boolean *differingAmountAllowed*)

Parameters

- **differingAmountAllowed** – the differingAmountAllowed value to set

Usage

Sets the

differingAmountAllowed

of this instance.

setDisableLimits(boolean) method

Syntax

void setDisableLimits (boolean *disableLimits*)

Parameters

- **disableLimits** – the disableLimits value to set

Usage

Sets the

disableLimits

of this instance.

setIdenticalPayerPayeeAllowed(boolean) method

Syntax

void setIdenticalPayerPayeeAllowed (boolean *identicalPayerPayeeAllowed*)

Parameters

- **identicalPayerPayeeAllowed** – the `identicalPayerPayeeAllowed` value to set

Usage

Sets the

identicalPayerPayeeAllowed

of this instance.

setPiTypes(Set< UseCasePaymentInstrumentType >) method

Syntax

`void setPiTypes ( Set< UseCasePaymentInstrumentType > piTypes )`

Parameters

- **piTypes** – the `piTypes` value to set

Usage

Sets the

piTypes

of this instance.

setPublic(boolean) method

Syntax

`void setPublic ( boolean isPublic )`

Parameters

- **isPublic** – the `isPublic` value to set

Usage

Sets the

isPublic

of this instance.

UseCaseFee class

Syntax

```
public class UseCaseFee
```

Usage

The `UseCaseFee` models a configuration of a specific business scenario to certain fee types. Precisely, a money fee type is assigned to a combination of

- money *use case*
- *Sub transaction type*
- payee's *OrgUnit*
- payer's *OrgUnit*
- payee/payer flag

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

UseCaseFee() constructor

Syntax

```
UseCaseFee ()
```

Usage

Creates a new instance of

```
UseCaseFee
```

.

getFeeType() method

Syntax

```
FeeType getFeeType ()
```

Returns

the assigned `MoneyFee.Type`

Usage

Returns the fee type assigned to this configuration.

the assigned `MoneyFee.Type`

getOrgUnitPayee() method

Syntax

`OrgUnit getOrgUnitPayee ()`

Returns

the affected payee's `OrgUnit`

Usage

Returns the

`OrgUnit`

of the affected payee.

the affected payee's `OrgUnit`

getOrgUnitPayer() method

Syntax

`OrgUnit getOrgUnitPayer ()`

Returns

the affected payer's `OrgUnit`

Usage

Returns the

`OrgUnit`

of the affected payer.

the affected payer's `OrgUnit`

getPayeePiType() method

Syntax

`PaymentInstrument.Type getPayeePiType ()`

Returns

the assigned `PaymentInstrument.Type`

Usage

Returns the payee's PI type assigned to this configuration.
the assigned `PaymentInstrument.Type`

getPayerPiType() method

Syntax

`PaymentInstrument.Type getPayerPiType ()`

Returns

the assigned `PaymentInstrument.Type`

Usage

Returns the payer's PI type assigned to this configuration.
the assigned `PaymentInstrument.Type`

getSubTransactionType() method

Syntax

`SubTransactionType getSubTransactionType ()`

Returns

the affected *sub Transaction.Type*

Usage

Returns the type of the affected
sub transaction

.

the affected *sub Transaction.Type*

getUseCase() method

Syntax

`UseCase getUseCase ()`

Returns

the affected `UseCase`.

Usage

Returns the type of the affected

Use case

.

the affected UseCase.

isCopyFromAuthorisation() method

Syntax

```
boolean isCopyFromAuthorisation ()
```

Returns

true if fees of this type should be copied from the authorisation; false if they should be calculated normally

Usage

Indicates if the fee should be copied from the authorisation. Note that for copied fields, the value of

isPayeeFee()

will be ignored.

true if fees of this type should be copied from the authorisation; false if they should be calculated normally

isIncludeInLimit() method

Syntax

```
boolean isIncludeInLimit ()
```

Returns

true if it should be included in limit; false if it should not be included in limit

Usage

Indicates if this use case fee should be included in the limit.

true if it should be included in limit; false if it should not be included in limit

isPayeeFee() method

Syntax

boolean isPayeeFee ()

Returns

true if fees of this type apply to the payee (i.e. the payee must pay the fee); false if they apply to the payer

Usage

Indicates if the fee applies to the payee or payer.

true if fees of this type apply to the payee (i.e. the payee must pay the fee); false if they apply to the payer

isSetCopyFromAuthorisation() method

Syntax

boolean isSetCopyFromAuthorisation ()

Returns

false if the dbCopyFromAuthorisation is null. Otherwise, it returns true.

Usage

Checks whether the

dbCopyFromAuthorisation

of this instance is set.

false if the dbCopyFromAuthorisation is null. Otherwise, it returns true.

isSetFeeType() method

Syntax

boolean isSetFeeType ()

Returns

false if the feeType is null. Otherwise, it returns true.

Usage

Checks whether the

feeType

of this instance is set.

false if the *feeType* is null. Otherwise, it returns true.

isSetIncludeInLimit() method

Syntax

```
boolean isSetIncludeInLimit ()
```

Returns

false if the *dbincludeInLimit* is null. Otherwise, it returns true.

Usage

Checks whether the

dbincludeInLimit

of this instance is set.

false if the *dbincludeInLimit* is null. Otherwise, it returns true.

isSetOrgUnitPayee() method

Syntax

```
boolean isSetOrgUnitPayee ()
```

Returns

false if the *orgUnitPayee* is null. Otherwise, it returns true.

Usage

Checks whether the

orgUnitPayee

of this instance is set.

false if the *orgUnitPayee* is null. Otherwise, it returns true.

isSetOrgUnitPayer() method

Syntax

```
boolean isSetOrgUnitPayer ()
```

Returns

false if the `orgUnitPayer` is null. Otherwise, it returns true.

Usage

Checks whether the

orgUnitPayer

of this instance is set.

false if the `orgUnitPayer` is null. Otherwise, it returns true.

isSetPayeeFee() method

Syntax

```
boolean isSetPayeeFee ()
```

Returns

false if the `dbpayeeFee` is null. Otherwise, it returns true.

Usage

Checks whether the

dbpayeeFee

of this instance is set.

false if the `dbpayeeFee` is null. Otherwise, it returns true.

isSetPayeePiType() method

Syntax

```
boolean isSetPayeePiType ()
```

Returns

false if the `payeePiType` is null. Otherwise, it returns true.

Usage

Checks whether the

payeePiType

of this instance is set.

false if the `payeePiType` is null. Otherwise, it returns true.

isSetPayerPiType() method

Syntax

```
boolean isSetPayerPiType ()
```

Returns

false if the payerPiType is null. Otherwise, it returns true.

Usage

Checks whether the

payerPiType

of this instance is set.

false if the payerPiType is null. Otherwise, it returns true.

isSetSubTransactionType() method

Syntax

```
boolean isSetSubTransactionType ()
```

Returns

false if the subTransactionType is null. Otherwise, it returns true.

Usage

Checks whether the

subTransactionType

of this instance is set.

false if the subTransactionType is null. Otherwise, it returns true.

isSetUseCase() method

Syntax

```
boolean isSetUseCase ()
```

Returns

false if the useCase is null. Otherwise, it returns true.

Usage

Checks whether the

useCase

of this instance is set.

false if the useCase is null. Otherwise, it returns true.

setCopyFromAuthorisation(boolean) method

Syntax

void setCopyFromAuthorisation (boolean *copyFromAuthorisation*)

Parameters

- **copyFromAuthorisation** – the copyFromAuthorisation value to set

Usage

Sets the

copyFromAuthorisation

of this instance.

setFeeType(FeeType) method

Syntax

void setFeeType (FeeType *type*)

Parameters

- **type** – the type value to set

Usage

Sets the

type

of this instance.

setIncludeInLimit(boolean) method

Syntax

void setIncludeInLimit (boolean *includeInLimit*)

Parameters

- **includeInLimit** – the `includeInLimit` value to set

Usage

Sets the

includeInLimit

of this instance.

setOrgUnitPayee(OrgUnit) method

Syntax

```
void setOrgUnitPayee ( OrgUnit orgUnitPayee )
```

Parameters

- **orgUnitPayee** – the `orgUnitPayee` value to set

Usage

Sets the

orgUnitPayee

of this instance.

setOrgUnitPayer(OrgUnit) method

Syntax

```
void setOrgUnitPayer ( OrgUnit orgUnitPayer )
```

Parameters

- **orgUnitPayer** – the `orgUnitPayer` value to set

Usage

Sets the

orgUnitPayer

of this instance.

setPayeeFee(boolean) method

Syntax

void setPayeeFee (boolean *payeeFee*)

Parameters

- **payeeFee** – the payeeFee value to set

Usage

Sets the

payeeFee

of this instance.

setPayeePiType(PaymentInstrument.Type) method

Syntax

void setPayeePiType (PaymentInstrument.Type *payeePiType*)

Parameters

- **payeePiType** – the payeePiType value to set

Usage

Sets the

payeePiType

of this instance.

setPayerPiType(PaymentInstrument.Type) method

Syntax

void setPayerPiType (PaymentInstrument.Type *payerPiType*)

Parameters

- **payerPiType** – the payerPiType value to set

Usage

Sets the

payerPiType

of this instance.

setSubTransactionType(SubTransactionType) method

Syntax

```
void setSubTransactionType ( SubTransactionType subTransactionType )
```

Parameters

- **subTransactionType** – the subTransactionType value to set

Usage

Sets the

subTransactionType

of this instance.

setUseCase(UseCase) method

Syntax

```
void setUseCase ( UseCase useCase )
```

Parameters

- **useCase** – the useCase value to set

Usage

Sets the

useCase

of this instance.

UseCasePaymentInstrumentType class

Syntax

```
public class UseCasePaymentInstrumentType
```

Usage

The UseCasePaymentInstrumentType restricts use cases to specific payment instrument types to be used by the payer and payee.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express

permission of SAP AG. The information contained herein may be changed without prior notice.

UseCasePaymentInstrumentType() constructor

Syntax

```
UseCasePaymentInstrumentType ()
```

Usage

Creates a new instance of

```
UseCasePaymentInstrumentType
```

.

getPaymentInstrumentType() method

Syntax

```
PaymentInstrument.Type getPaymentInstrumentType ()
```

Returns

the allowed PI type.

Usage

Returns the allowed PI type.

the allowed PI type.

getUseCase() method

Syntax

```
UseCase getUseCase ()
```

Returns

the use case to be restricted.

Usage

Returns the use case to be restricted.

the use case to be restricted.

isPayee() method

Syntax

boolean isPayee ()

Returns

true if this use case is public; false otherwise

Usage

Indicates if this use case is public or for internal use.

true if this use case is public; false otherwise

isSetPayee() method

Syntax

boolean isSetPayee ()

Returns

false if the dbPayee is null. Otherwise, it returns true.

Usage

Checks whether the

dbPayee

of this instance is set.

false if the dbPayee is null. Otherwise, it returns true.

isSetPaymentInstrumentType() method

Syntax

boolean isSetPaymentInstrumentType ()

Returns

false if the piType is null. Otherwise, it returns true.

Usage

Checks whether the

piType

of this instance is set.

false if the *piType* is null. Otherwise, it returns true.

isSetUseCase() method

Syntax

boolean isSetUseCase ()

Returns

false if the *useCase* is null. Otherwise, it returns true.

Usage

Checks whether the

useCase

of this instance is set.

false if the *useCase* is null. Otherwise, it returns true.

setPayee(boolean) method

Syntax

void setPayee (boolean *dbPayee*)

Parameters

- **dbPayee** – the *dbPayee* value to set

Usage

Sets the

dbPayee

of this instance.

setPaymentInstrumentType(PaymentInstrument.Type) method

Syntax

void setPaymentInstrumentType (PaymentInstrument.Type *piType*)

Parameters

- **piType** – the piType value to set

Usage

Sets the

piType

of this instance.

setUseCase(UseCase) method

Syntax

```
void setUseCase ( UseCase useCase )
```

Parameters

- **useCase** – the useCase value to set

Usage

Sets the

useCase

of this instance.

UseCasePrivilege class**Syntax**

```
public class UseCasePrivilege
```

Usage

This class is to control access rights to use cases. The access type can either by payer, payer, caller, caller-self, or caller-parent. caller-self means that the caller is identical to the actor. caller-parent means that the caller is the child of the actor

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

UseCasePrivilege() constructor

Syntax

```
UseCasePrivilege ()
```

Usage

Creates a new instance of

UseCasePrivilege

.

UseCasePrivilege(Integer) constructor

Syntax

UseCasePrivilege (Integer *id*)

Parameters

- **id** – the PK value

Usage

Creates a new instance of

UseCasePrivilege

with a preset primary key.

getCallerParentPrivilege() method

Syntax

String getCallerParentPrivilege ()

getCallerPrivilege() method

Syntax

String getCallerPrivilege ()

getCallerSelfPrivilege() method

Syntax

String getCallerSelfPrivilege ()

getCheckerPrivilege() method

Syntax

String getCheckerPrivilege ()

getExecutePrivilege() method

Syntax

String getExecutePrivilege ()

getMakerPrivilege() method

Syntax

String getMakerPrivilege ()

getPayeePrivilege() method

Syntax

String getPayeePrivilege ()

getPayerPrivilege() method

Syntax

String getPayerPrivilege ()

isSetCallerParentPrivilege() method

Syntax

boolean isSetCallerParentPrivilege ()

isSetCallerPrivilege() method

Syntax

boolean isSetCallerPrivilege ()

isSetCallerSelfPrivilege() method

Syntax

boolean isSetCallerSelfPrivilege ()

isSetCheckerPrivilege() method

Syntax

boolean isSetCheckerPrivilege ()

isSetExecutePrivilege() method

Syntax

boolean isSetExecutePrivilege ()

isSetMaker() method

Syntax

boolean isSetMaker ()

isSetPayeePrivilege() method

Syntax

boolean isSetPayeePrivilege ()

isSetPayerPrivilege() method

Syntax

boolean isSetPayerPrivilege ()

setCallerParentPrivilege(String) method

Syntax

void setCallerParentPrivilege (String *callerParentPrivilege*)

setCallerPrivilege(String) method

Syntax

void setCallerPrivilege (String *callerPrivilege*)

setCallerSelfPrivilege(String) method

Syntax

void setCallerSelfPrivilege (String *callerSelfPrivilege*)

setCheckerPrivilege(String) method

Syntax

void setCheckerPrivilege (String *checkerPrivilege*)

setExecutePrivilege(String) method

Syntax

```
void setExecutePrivilege ( String executePrivilege )
```

setMakerPrivilege(String) method

Syntax

```
void setMakerPrivilege ( String makerPrivilege )
```

setPayeePrivilege(String) method

Syntax

```
void setPayeePrivilege ( String payeePrivilege )
```

setPayerPrivilege(String) method

Syntax

```
void setPayerPrivilege ( String payerPrivilege )
```

ValidationRule class

Syntax

```
public class ValidationRule
```

Usage

The `ValidationRule` table holds configuration for the customizable validation implemented by the `MobiliserServiceValidator`, which can register to the generic service filter chain through the `FilterAdvice`.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

ValidationRule() constructor

Syntax

```
ValidationRule ()
```

Usage

Creates a new instance of

ValidationRule

.

getEndpoint() method

Syntax

String getEndpoint ()

Returns

the endpoint name.

Usage

Returns the endpoint name this rule applies to.

the endpoint name.

getExpression() method

Syntax

String getExpression ()

Returns

the expression that must be applied.

Usage

Returns the expression that must be applied.

the expression that must be applied.

getMethod() method

Syntax

String getMethod ()

Returns

the method name this rule applies to.

Usage

Returns the method name this rule applies to.

the method name this rule applies to.

*getPriority() method***Syntax**`int getPriority ()`**Returns**

the priority of this rule.

Usage

Returns the priority of this rule.

the priority of this rule.

*isRequest() method***Syntax**`boolean isRequest ()`**Returns**

true if this rule applies to a request; false if it applies to a response

Usage

Indicates if this rule applies to a request.

true if this rule applies to a request; false if it applies to a response

*isSetEndpoint() method***Syntax**`boolean isSetEndpoint ()`**Returns**

false if the endpoint is null. Otherwise, it returns true.

Usage

Checks whether the

endpoint

of this instance is set.

false if the endpoint is null. Otherwise, it returns true.

isSetExpression() method

Syntax

```
boolean isSetExpression ()
```

Returns

false if the expression is null. Otherwise, it returns true.

Usage

Checks whether the

expression

of this instance is set.

false if the expression is null. Otherwise, it returns true.

isSetMethod() method

Syntax

```
boolean isSetMethod ()
```

Returns

false if the method is null. Otherwise, it returns true.

Usage

Checks whether the

method

of this instance is set.

false if the method is null. Otherwise, it returns true.

isSetPriority() method

Syntax

```
boolean isSetPriority ()
```

Returns

false if the priority is null. Otherwise, it returns true.

Usage

Checks whether the

priority

of this instance is set.

false if the `priority` is null. Otherwise, it returns true.

isSetRequest() method

Syntax

```
boolean isSetRequest ()
```

Returns

false if the `dbRequest` is null . Otherwise, it returns true.

Usage

Checks whether the

dbRequest

of this instance is set.

false if the `dbRequest` is null . Otherwise, it returns true.

setEndpoint(final String) method

Syntax

```
void setEndpoint ( final String endpoint )
```

Parameters

- **endpoint** – the endpoint value to set

Usage

Sets the

endpoint

of this instance.

setExpression(final String) method

Syntax

```
void setExpression ( final String expression )
```

Parameters

- **expression** – the expression value to set

Usage

Sets the

expression

of this instance.

setMethod(final String) method

Syntax

```
void setMethod ( final String method )
```

Parameters

- **method** – the method value to set

Usage

Sets the

method

of this instance.

setPriority(final int) method

Syntax

```
void setPriority ( final int priority )
```

Parameters

- **priority** – the priority value to set

Usage

Sets the

priority

of this instance.

setRequest(final boolean) method

Syntax

```
void setRequest ( final boolean request )
```

Parameters

- **active** – the request value to set

Usage

Sets the

request

of this instance.

Version class

Syntax

```
public class Version
```

Usage

The `Version` tracks the database version for all applications accessing the database directly.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

DbVersion class

Syntax

```
public class DbVersion
```

Usage

Simple bean to hold an application's version for compatibility checks with the Version bean.

© 2000-2008 by paybox solutions AG

Revision:1.1Name:not supported by cvs2svn

DbVersion(String, int, int) constructor

Syntax

```
DbVersion ( String application , int major , int minor )
```

Parameters

- **application** –
- **major** –
- **minor** –

Usage

Creates a new instance of

DbVersion

.

getApplication() method

Syntax

String getApplication ()

Returns

the application

Usage

the application

getMajor() method

Syntax

int getMajor ()

Returns

the major

Usage

the major

getMinor() method

Syntax

int getMinor ()

Returns

the minor

Usage

the minor

setApplication(String) method

Syntax

```
void setApplication ( String application )
```

Parameters

- **application** – the application to set

setMajor(int) method

Syntax

```
void setMajor ( int major )
```

Parameters

- **major** – the major to set

setMinor(int) method

Syntax

```
void setMinor ( int minor )
```

Parameters

- **minor** – the minor to set

toString() method

Syntax

```
String toString ()
```

Returns

a string representation of this bean.

Usage

Returns a string representation of this bean.

a string representation of this bean.

Version() constructor

Syntax

`Version ()`

Usage

Creates a new instance of

`Version`

.

Version(String) constructor

Syntax

`Version ( String id )`

Parameters

- **id** – the PK value

Usage

Creates a new instance of

`Version`

with a preset primary key.

getApplicationDescription() method

Syntax

`String getApplicationDescription ()`

Returns

a short description of the application.

Usage

Returns a short description of the application.

a short description of the application.

getCvsTag() method

Syntax

String getCvsTag ()

Returns

the application's CVS tag with which this version was introduced.

Usage

Returns the application's CVS tag with which this version was introduced.

the application's CVS tag with which this version was introduced.

getMajorVersion() method

Syntax

int getMajorVersion ()

Returns

the major version number.

Usage

Returns the major version number.

the major version number.

getMinorVersion() method

Syntax

int getMinorVersion ()

Returns

the minor version number.

Usage

Returns the minor version number.

the minor version number.

isCompatibleWith(DbVersion) method

Syntax

boolean isCompatibleWith (DbVersion *version*)

Parameters

- **version** – the version to check.

Returns

true if this version is equal or newer, but major revision number is equal.

Usage

Checks for version compatibility. Returns true if this version is equal or newer, but major revision number is equal.

true if this version is equal or newer, but major revision number is equal.

isSetApplicationDescription() method

Syntax

boolean isSetApplicationDescription ()

Returns

false if the applicationDescription is null. Otherwise, it returns true.

Usage

Checks whether the

applicationDescription

of this instance is set.

false if the applicationDescription is null. Otherwise, it returns true.

isSetCvsTag() method

Syntax

boolean isSetCvsTag ()

Returns

false if the cvsTag is null. Otherwise, it returns true.

Usage

Checks whether the

cvstag

of this instance is set.

false if the *cvstag* is null. Otherwise, it returns true.

isSetMajorVersion() method

Syntax

```
boolean isSetMajorVersion ()
```

Returns

false if the *majorVersion* is null. Otherwise, it returns true.

Usage

Checks whether the

majorVersion

of this instance is set.

false if the *majorVersion* is null. Otherwise, it returns true.

isSetMinorVersion() method

Syntax

```
boolean isSetMinorVersion ()
```

Returns

false if the *minorVersion* is null. Otherwise, it returns true.

Usage

Checks whether the

minorVersion

of this instance is set.

false if the *minorVersion* is null. Otherwise, it returns true.

setApplicationDescription(String) method

Syntax

void setApplicationDescription (String *applicationDescription*)

Parameters

- **applicationDescription** – the applicationDescription value to set

Usage

Sets the

application description

of this instance.

setCvsTag(String) method

Syntax

void setCvsTag (String *cvsTag*)

Parameters

- **cvsTag** – the cvsTag value to set

Usage

Sets the

cvs tag

of this instance.

setMajorVersion(int) method

Syntax

void setMajorVersion (int *majorVersion*)

Parameters

- **majorVersion** – the majorVersion value to set

Usage

Sets the

major version

of this instance.

setMinorVersion(int) method

Syntax

```
void setMinorVersion ( int minorVersion )
```

Parameters

- **minorVersion** – the `minorVersion` value to set

Usage

Sets the

minor version

of this instance.

WrkBulkProcessing class

Syntax

```
public class WrkBulkProcessing
```

Usage

The `WrkBulkProcessing` represents a maker/checker configuration for bulk processing file types. Allowing to define `UmgrPrivilege s` for each action all tied to a specific bulk processing file type.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

WrkBulkProcessing() constructor

Syntax

```
WrkBulkProcessing ()
```

WrkBulkProcessing(Integer) constructor

Syntax

```
WrkBulkProcessing ( Integer id )
```

Parameters

- **id** –

Usage

Please use the

`newInstance`

method of the corresponding DAO.

getCheckerPrivilege() method

Syntax

`String getCheckerPrivilege ()`

getExecutePrivilege() method

Syntax

`String getExecutePrivilege ()`

getMakerPrivilege() method

Syntax

`String getMakerPrivilege ()`

isSetCheckerPrivilege() method

Syntax

`boolean isSetCheckerPrivilege ()`

isSetExecutePrivilege() method

Syntax

`boolean isSetExecutePrivilege ()`

isSetMakerPrivilege() method

Syntax

`boolean isSetMakerPrivilege ()`

setCheckerPrivilege(String) method

Syntax

```
void setCheckerPrivilege ( String checkerPrivilege )
```

setExecutePrivilege(String) method

Syntax

```
void setExecutePrivilege ( String executePrivilege )
```

setMakerPrivilege(String) method

Syntax

```
void setMakerPrivilege ( String makerPrivilege )
```

transaction package

Members

All public members of the transaction package.

- **Authorisation class** –
- **AuthorisationCancel class** –
- **Capture class** –
- **CaptureCancel class** –
- **Commission class** –
- **Fee class** –
- **InvoiceTransaction class** –
- **SubTransaction class** –
- **SvaBooking class** –
- **TraceableRequest class** –
- **Transaction class** –
- **TransactionAttribute class** –
- **XmlMessage class** –

Authorisation class

Syntax

```
public class Authorisation
```

Usage

The *Authorisation* models an *authorisation* transaction.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Authorisation() constructor

Syntax

```
Authorisation ()
```

Usage

Creates a new instance of

```
Authorisation
```

.

AuthorisationCancel class

Syntax

```
public class AuthorisationCancel
```

Usage

The `AuthorisationCancel` models an *authorisation cancel* transaction.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

AuthorisationCancel() constructor

Syntax

```
AuthorisationCancel ()
```

Usage

Creates a new instance of

```
AuthorisationCancel
```

.

Capture class

Syntax

```
public class Capture
```

Usage

The `Capture` models a *capture* transaction.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Capture() constructor

Syntax

```
Capture ()
```

Usage

Creates a new instance of

```
Capture
```

.

CaptureCancel class

Syntax

```
public class CaptureCancel
```

Usage

The `CaptureCancel` models a *capture cancel* transaction.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

CaptureCancel() constructor

Syntax

```
CaptureCancel ()
```

Usage

Creates a new instance of

CaptureCancel

.

Commission class

Syntax

```
public class Commission
```

Usage

The `Commission` models a commission charged by the *Money Mobiliser*.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Commission() constructor

Syntax

```
Commission ()
```

Usage

Creates a new instance of

Commission

.

getAmount() method

Syntax

```
long getAmount ()
```

Returns

the fee's amount

Usage

Returns the amount of this fee.

the fee's amount

getAmountNanos() method

Syntax

Integer getAmountNanos ()

Returns

the fee amount's fraction; may be null

Usage

Returns the fraction part of the fee amount.

For the two values

amount

and

fraction

, the following equation is valid:

$$\text{fee} = \text{amount} + \text{fraction} * 10^{\sup>-9\sup>}$$

the fee amount's fraction; may be null

getCommission() method

Syntax

BigDecimal getCommission ()

Returns

a BigDecimal representing the summed up fee amount

Usage

Returns the sum of the four fields

amount

,

amountNanos

a BigDecimal representing the summed up fee amount

getCustomer() method

Syntax

Customer getCustomer ()

Returns

the affected Customer

Usage

Returns the customer affected by this commission.

the affected Customer

getStatus() method

Syntax

int getStatus ()

Returns

the status

Usage

the status

getStranaction() method

Syntax

Transaction getStranaction ()

Returns

the involved SubTransaction

Usage

Returns the settlement transaction this fee was cleared with

the involved SubTransaction

getTransaction() method

Syntax

SubTransaction getTransaction ()

Returns

the involved SubTransaction

Usage

Returns the transaction this fee is involved with.

the involved SubTransaction

getType() method

Syntax

```
FeeType getType ()
```

Returns

the fee's Type

Usage

Returns the type of this fee.

the fee's Type

isSetAmount() method

Syntax

```
boolean isSetAmount ()
```

Returns

false if the amount is null. Otherwise, it returns true.

Usage

Checks whether the

amount

of this instance is set.

false if the amount is null. Otherwise, it returns true.

isSetAmountNanos() method

Syntax

```
boolean isSetAmountNanos ()
```

Returns

false if the amountNanos is null. Otherwise, it returns true.

Usage

Checks whether the

amountNanos

of this instance is set.

false if the amountNanos is null. Otherwise, it returns true.

isSetCustomer() method

Syntax

```
boolean isSetCustomer ()
```

Returns

false if the customer is null. Otherwise, it returns true.

Usage

Checks whether the

customer

of this instance is set.

false if the customer is null. Otherwise, it returns true.

isSetStatus() method

Syntax

```
boolean isSetStatus ()
```

Returns

false if the status is null. Otherwise, it returns true.

Usage

Checks whether the

status

of this instance is set.

false if the status is null. Otherwise, it returns true.

isSetStransaction() method

Syntax

```
boolean isSetStransaction ()
```

Returns

false if the stransaction is null. Otherwise, it returns true.

Usage

Checks whether the

stransaction

of this instance is set.

false if the stransaction is null. Otherwise, it returns true.

isSetTransaction() method

Syntax

```
boolean isSetTransaction ()
```

Returns

false if the transaction is null. Otherwise, it returns true.

Usage

Checks whether the

transaction

of this instance is set.

false if the transaction is null. Otherwise, it returns true.

isSetType() method

Syntax

```
boolean isSetType ()
```

Returns

false if the type is null. Otherwise, it returns true.

Usage

Checks whether the

type

of this instance is set.

false if the `type` is null. Otherwise, it returns true.

setAmount(long) method

Syntax

```
void setAmount ( long amount )
```

Parameters

- **amount** – the amount value to set

Usage

Sets the

amount

of this instance.

setAmountNanos(Integer) method

Syntax

```
void setAmountNanos ( Integer amountNanos )
```

Parameters

- **amountNanos** – the amountNanos value to set

Usage

Sets the

amountNanos

of this instance.

setCommission(final BigDecimal) method

Syntax

```
void setCommission ( final BigDecimal amount )
```

Parameters

- **amount** – the amount to set

Exceptions

- **NullPointerException** – if amount is null

Usage

Sets the appropriate fee amount and nanos for the specified decimal. Note that the nanos could be positive or negative, depending on the rounding that applied.

setCustomer(Customer) method

Syntax

```
void setCustomer ( Customer customer )
```

Parameters

- **customer** – the customer value to set

Usage

Sets the

customer

of this instance.

setStatus(int) method

Syntax

```
void setStatus ( int status )
```

Parameters

- **status** – the status to set

setStranction(Transaction) method

Syntax

```
void setStranction ( Transaction stranction )
```

Parameters

- **transaction** – the transaction value to set

Usage

Sets the

transaction

by which this commission was cleared

setTransaction(SubTransaction) method

Syntax

```
void setTransaction ( SubTransaction transaction )
```

Parameters

- **transaction** – the transaction value to set

Usage

Sets the

transaction

of this instance.

setType(FeeType) method

Syntax

```
void setType ( FeeType type )
```

Parameters

- **type** – the type value to set

Usage

Sets the

type

of this instance.

Fee class**Syntax**

```
public class Fee
```

Usage

The `Fee` models a fee charged by the *Money Mobiliser*.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Fee() constructor**Syntax**

```
Fee ()
```

Usage

Creates a new instance of

```
Fee
```

```
.
```

getAmount() method**Syntax**

```
long getAmount ()
```

Returns

the fee's amount

Usage

Returns the amount of this fee.

the fee's amount

getAmountNanos() method**Syntax**

```
Integer getAmountNanos ()
```

Returns

the fee amount's fraction; may be null

Usage

Returns the fraction part of the fee amount.

For the two values

amount

and

fraction

, the following equation is valid:

```
fee = amount + fraction * 10-9
```

the fee amount's fraction; may be null

getCurrency() method

Syntax

Currency getCurrency ()

Returns

the transaction's Currency

Usage

Returns the currency this transaction is executed with.

the transaction's Currency

getCustomer() method

Syntax

Customer getCustomer ()

Returns

the affected Customer

Usage

Returns the customer affected by this fee. This value should only be set if

isPayeeFee()

is

null

.

the affected Customer

getExchangeRatePayee() method

Syntax

BigDecimal getExchangeRatePayee ()

Returns

the exchange rate used for the payee.

Usage

The exchange rate used for the payee.

the exchange rate used for the payee.

getExchangeRatePayer() method

Syntax

BigDecimal getExchangeRatePayer ()

Returns

the exchange rate used for the payer.

Usage

The exchange rate used for the payer.

the exchange rate used for the payer.

getFeeAmount() method

Syntax

BigDecimal getFeeAmount ()

Returns

a BigDecimal representing the summed up fee amount

Usage

Returns the sum of the four fields

amount

,

amountNanos

,

vat

and

vatNanos

. VAT is only added if not included in the amount yet.

a `BigDecimal` representing the summed up fee amount

getFeeAmountAsLong() method

Syntax

`long getFeeAmountAsLong ()`

Returns

a `long` representing the summed up fee amount

Usage

Returns the sum of the four fields

amount

,

amountNanos

,

vat

and

vatNanos

as rounded

long

.

a `long` representing the summed up fee amount

getFeeType() method

Syntax

FeeType getFeeType ()

Returns

the feeType

Usage

the feeType

getNetFeeAmount() method

Syntax

BigDecimal getNetFeeAmount ()

Returns

a BigDecimal representing the exact net fee amount

Usage

Returns the sum of the two fields

amount

and

amountNanos

.

a BigDecimal representing the exact net fee amount

getText() method

Syntax

String getText ()

Returns

the fee's text; may be null

Usage

Returns the text of this fee.

the fee's text; may be null

getTransaction() method

Syntax

SubTransaction getTransaction ()

Returns

the involved SubTransaction

Usage

Returns the transaction this fee is involved with.

the involved SubTransaction

getVat() method

Syntax

Long getVat ()

Returns

the fee's VAT amount; may be null

Usage

Returns the VAT amount of this fee.

the fee's VAT amount; may be null

getVatFeeAmount() method

Syntax

BigDecimal getVatFeeAmount ()

Returns

a BigDecimal representing the exact VAT fee amount

Usage

Returns the sum of the two fields

vat

and

vatNanos

.

a `BigDecimal` representing the exact VAT fee amount

getVatNanos() method

Syntax

Integer getVatNanos ()

Returns

the vat amount's fraction; may be `null`

Usage

Returns the fraction part of the vat amount.

For the two values

vatAmount

and

vatFraction

, the following equation is valid:

$$\text{vat} = \text{vatAmount} + \text{vatFraction} * 10^{\sup{-9}}$$

the vat amount's fraction; may be `null`

isIncludeInLimit() method

Syntax

boolean isIncludeInLimit ()

Returns

`true` if it should be included in limit; `false` if it should not be included in limit

Usage

Indicates if this fee should be included in the limit.

`true` if it should be included in limit; `false` if it should not be included in limit

isPayeeFee() method

Syntax

Boolean isPayeeFee ()

Returns

Boolean#TRUE if this fee applies to the payee (i.e. the payee must pay the fee);
Boolean#FALSE if it applies to the payer; null if it applies to neither (see getCustomer())

Usage

Indicates if the fee applies to the payee, payer, or neither of them.

Boolean#TRUE if this fee applies to the payee (i.e. the payee must pay the fee);
Boolean#FALSE if it applies to the payer; null if it applies to neither (see getCustomer())

isSetAmount() method

Syntax

boolean isSetAmount ()

Returns

false if the amount is null. Otherwise, it returns true.

Usage

Checks whether the

amount

of this instance is set.

false if the amount is null. Otherwise, it returns true.

isSetAmountNanos() method

Syntax

boolean isSetAmountNanos ()

Returns

false if the amountNanos is null. Otherwise, it returns true.

Usage

Checks whether the

amountNanos

of this instance is set.

false if the amountNanos is null. Otherwise, it returns true.

isSetCurrency() method

Syntax

```
boolean isSetCurrency ()
```

Returns

false if the dbCurrency is null. Otherwise, it returns true.

Usage

Checks whether the

dbCurrency

of this instance is set.

false if the dbCurrency is null. Otherwise, it returns true.

isSetCustomer() method

Syntax

```
boolean isSetCustomer ()
```

Returns

false if the customer is null. Otherwise, it returns true.

Usage

Checks whether the

customer

of this instance is set.

false if the customer is null. Otherwise, it returns true.

isSetExchangeRatePayee() method

Syntax

```
boolean isSetExchangeRatePayee ()
```

Returns

false if the `exchangeRatePayee` is null. Otherwise, it returns true.

Usage

Checks whether the

exchangeRatePayee

of this instance is set.

false if the `exchangeRatePayee` is null. Otherwise, it returns true.

isSetExchangeRatePayer() method

Syntax

```
boolean isSetExchangeRatePayer ()
```

Returns

false if the `exchangeRatePayer` is null. Otherwise, it returns true.

Usage

Checks whether the

exchangeRatePayer

of this instance is set.

false if the `exchangeRatePayer` is null. Otherwise, it returns true.

isSetFeeType() method

Syntax

```
boolean isSetFeeType ()
```

Returns

false if the `feeType` is null. Otherwise, it returns true.

Usage

Checks whether the

feeType

of this instance is set.

false if the `feeType` is null. Otherwise, it returns true.

*isSetIncludeInLimit() method***Syntax**

```
boolean isSetIncludeInLimit ()
```

Returns

false if the `dbincludeInLimit` is null. Otherwise, it returns true.

Usage

Checks whether the

dbincludeInLimit

of this instance is set.

false if the `dbincludeInLimit` is null. Otherwise, it returns true.

*isSetPayeeFee() method***Syntax**

```
boolean isSetPayeeFee ()
```

Returns

false if the `dbpayeeFee` is null. Otherwise, it returns true.

Usage

Checks whether the

dbpayeeFee

of this instance is set.

false if the `dbpayeeFee` is null. Otherwise, it returns true.

*isSetText() method***Syntax**

```
boolean isSetText ()
```

Returns

false if the `text` is null. Otherwise, it returns true.

Usage

Checks whether the

text

of this instance is set.

false if the text is null. Otherwise, it returns true.

isSetTransaction() method

Syntax

```
boolean isSetTransaction ()
```

Returns

false if the transaction is null. Otherwise, it returns true.

Usage

Checks whether the

transaction

of this instance is set.

false if the transaction is null. Otherwise, it returns true.

isSetVat() method

Syntax

```
boolean isSetVat ()
```

Returns

false if the vat is null. Otherwise, it returns true.

Usage

Checks whether the

vat

of this instance is set.

false if the vat is null. Otherwise, it returns true.

*isSetVatNanos() method***Syntax**

```
boolean isSetVatNanos ()
```

Returns

false if the vatNanos is null. Otherwise, it returns true.

Usage

Checks whether the

vatNanos

of this instance is set.

false if the vatNanos is null. Otherwise, it returns true.

*setAmount(final BigDecimal) method***Syntax**

```
void setAmount ( final BigDecimal amount )
```

Parameters

- **amount** – the amount to set

Exceptions

- **NullPointerException** – if amount is null

Usage

Sets the appropriate fee amount and nanos for the specified decimal. Note that the nanos could be positive or negative, depending on the rounding that applied.

*setAmount(long) method***Syntax**

```
void setAmount ( long amount )
```

Parameters

- **amount** – the amount value to set

Usage

Sets the

amount

of this instance.

setAmountNanos(Integer) method

Syntax

```
void setAmountNanos ( Integer amountNanos )
```

Parameters

- **amountNanos** – the amountNanos value to set

Usage

Sets the

amountNanos

of this instance.

setCurrency(Currency) method

Syntax

```
void setCurrency ( Currency currency )
```

Parameters

- **currency** – the currency value to set

Usage

Sets the

currency

of this instance.

setCurrency(String) method

Syntax

```
void setCurrency ( String currency )
```

Parameters

- **currency** – the `currency` value to set

Usage

Sets the

currency

of this instance.

setCustomer(Customer) method

Syntax

```
void setCustomer ( Customer customer )
```

Parameters

- **customer** – the `customer` value to set

Usage

Sets the

customer

of this instance.

setExchangeRatePayee(BigDecimal) method

Syntax

```
void setExchangeRatePayee ( BigDecimal exchangeRatePayee )
```

Parameters

- **exchangeRatePayee** – the `exchangeRatePayee` value to set

Usage

Sets the

exchangeRatePayee

of this instance.

setExchangeRatePayer(BigDecimal) method

Syntax

`void setExchangeRatePayer ( BigDecimal exchangeRatePayer )`

Parameters

- **exchangeRatePayer** – the `exchangeRatePayer` value to set

Usage

Sets the

exchangeRatePayer

of this instance.

setFeeType(FeeType) method

Syntax

`void setFeeType ( FeeType feeType )`

Parameters

- **feeType** – the `feeType` to set

setIncludeInLimit(boolean) method

Syntax

`void setIncludeInLimit ( boolean includeInLimit )`

Parameters

- **includeInLimit** – the `includeInLimit` value to set

Usage

Sets the

includeInLimit

of this instance.

setPayeeFee(Boolean) method

Syntax

void setPayeeFee (Boolean *payeeFee*)

Parameters

- **payeeFee** – the `payeeFee` value to set

Usage

Sets the

payeeFee

of this instance.

setText(String) method

Syntax

void setText (String *text*)

Parameters

- **text** – the `text` value to set

Usage

Sets the

text

of this instance.

setTransaction(SubTransaction) method

Syntax

void setTransaction (SubTransaction *transaction*)

Parameters

- **transaction** – the `transaction` value to set

Usage

Sets the

transaction

of this instance.

setVat(final BigDecimal) method

Syntax

```
void setVat ( final BigDecimal vat )
```

Parameters

- **vat** – the VAT amount to set; `null` is allowed

Usage

Sets the appropriate VAT amount and nanos for the specified decimal. Note that the nanos could be positive or negative, depending on the rounding that applied.

setVat(Long) method

Syntax

```
void setVat ( Long vat )
```

Parameters

- **vat** – the `vat` value to set

Usage

Sets the

vat

of this instance.

setVatNanos(Integer) method

Syntax

```
void setVatNanos ( Integer vatNanos )
```

Parameters

- **vatNanos** – the `vatNanos` value to set

Usage

Sets the

vatNanos

of this instance.

InvoiceTransaction class

Syntax

```
public class InvoiceTransaction
```

Usage

The `InvoiceTransaction` represents a transaction towards an invoice reducing the remaining amount.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

InvoiceTransaction() constructor

Syntax

```
InvoiceTransaction ()
```

Usage

Creates a new instance of

```
InvoiceTransaction
```

.

getInvoice() method

Syntax

```
Invoice getInvoice ()
```

Returns

the invoice

Usage

the invoice

getTransaction() method

Syntax

Transaction getTransaction ()

Returns

the transaction

Usage

the transaction

isSetInvoice() method

Syntax

boolean isSetInvoice ()

Returns

false if the invoice is null. Otherwise, it returns true.

Usage

Checks whether the

invoice

of this instance is set.

false if the invoice is null. Otherwise, it returns true.

isSetTransaction() method

Syntax

boolean isSetTransaction ()

Returns

false if the transaction is null. Otherwise, it returns true.

Usage

Checks whether the

transaction

of this instance is set.

false if the transaction is null. Otherwise, it returns true.

setInvoice(Invoice) method

Syntax

```
void setInvoice ( Invoice invoice )
```

Parameters

- **invoice** – the invoice to set

setTransaction(Transaction) method

Syntax

```
void setTransaction ( Transaction transaction )
```

Parameters

- **transaction** – the transaction to set

SubTransaction class

Syntax

```
public class SubTransaction
```

Derived classes

- *Authorisation* on page 1497
- *AuthorisationCancel* on page 1498
- *Capture* on page 1499
- *CaptureCancel* on page 1499

Usage

The SubTransaction is the base class for all Money transactions.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

SubTransaction() constructor

Syntax

```
SubTransaction ()
```

Usage

Creates a new instance of

`SubTransaction`

.

addFee(Fee) method

Syntax

`void addFee ( Fee fee )`

Parameters

- **fee** – the `fee` value to add

Usage

Adds the

fee

to this instance.

getBalancePayee() method

Syntax

`Long getBalancePayee ()`

Returns

the balance on the payee's PI after subtransaction processing.

Usage

Returns the balance on the payee's PI after subtransaction processing.

the balance on the payee's PI after subtransaction processing.

getBalancePayer() method

Syntax

`Long getBalancePayer ()`

Returns

the balance on the payer's PI after subtransaction processing.

Usage

Returns the balance on the payer's PI after subtransaction processing.
the balance on the payer's PI after subtransaction processing.

getCallbackStatus() method

Syntax

```
int getCallbackStatus ()
```

Returns

the callbackStatus

Usage

the callbackStatus

getCallbackUrl() method

Syntax

```
String getCallbackUrl ()
```

Returns

the callbackUrl

Usage

the callbackUrl

getCaller() method

Syntax

```
Customer getCaller ()
```

Returns

the caller

Usage

the caller

getCallerIdentification() method

Syntax

String getCallerIdentification ()

Returns

the callerIdentification

Usage

the callerIdentification

getDiscrimaterValue() method

Syntax

int getDiscrimaterValue ()

getErrorCode() method

Syntax

int getErrorCode ()

Returns

the errorCode

Usage

the errorCode

getExchangeRatePayee() method

Syntax

BigDecimal getExchangeRatePayee ()

Returns

the exchange rate used for the payee.

Usage

The exchange rate used for the payee.

the exchange rate used for the payee.

getExchangeRatePayer() method

Syntax

BigDecimal getExchangeRatePayer ()

Returns

the exchange rate used for the payer.

Usage

The exchange rate used for the payer.

the exchange rate used for the payer.

getFees() method

Syntax

List< Fee > getFees ()

Returns

a List of involved MoneyFees

Usage

Returns the fees involved in this transaction.

a List of involved MoneyFees

getOrigin() method

Syntax

String getOrigin ()

Returns

the origin

Usage

the origin

getPayeePayment() method

Syntax

Long getPayeePayment ()

Returns

the payeePayment

Usage

the payeePayment

getPayerPayment() method

Syntax

Long getPayerPayment ()

Returns

the payerPayment

Usage

the payerPayment

getReturnUrl() method

Syntax

String getReturnUrl ()

Returns

the returnUrl

Usage

the returnUrl

getSubAmount() method

Syntax

long getSubAmount ()

Returns

the transaction's amount

Usage

Returns the amount of this transction.

the transaction's amount

getSubAmountPayee() method

Syntax

long getSubAmountPayee ()

Returns

the transaction amount in payee's currency.

Usage

Returns the amount of this transaction in payee's currency.

the transaction amount in payee's currency.

getSubAmountPayer() method

Syntax

long getSubAmountPayer ()

Returns

the transaction amount in payer's currency.

Usage

Returns the amount of this transaction in payer's currency.

the transaction amount in payer's currency.

getSubTransactionType() method

Syntax

SubTransactionType getSubTransactionType ()

Returns

the subTransactionType

Usage

the subTransactionType

getTraceNumber() method

Syntax

String getTraceNumber ()

Returns

the traceNumber

Usage

the traceNumber

getTransaction() method

Syntax

Transaction getTransaction ()

Returns

the transaction

Usage

the transaction

getTransactionDate() method

Returns the actual Transaction Date.

Syntax

Date getTransactionDate ()

Returns

TransactionDate

Usage

TransactionDate

getTransactionReferencePayee() method

Syntax

String getTransactionReferencePayee ()

Returns

the payee's transaction reference

Usage

Returns the transaction reference of the payee's payment instrument.

the payee's transaction reference

getTransactionReferencePayer() method

Syntax

String getTransactionReferencePayer ()

Returns

the payer's transaction reference

Usage

Returns the transaction reference of the payer's payment instrument.

the payer's transaction reference

getTransactionTimestampPayee() method

Syntax

Date getTransactionTimestampPayee ()

Returns

the payee's transaction Date

Usage

Returns the transaction timestamp of the payee's payment instrument.

the payee's transaction Date

getTransactionTimestampPayer() method

Syntax

Date getTransactionTimestampPayer ()

Returns

the payer's transaction Date

Usage

Returns the transaction timestamp of the payer's payment instrument.

the payer's transaction Date

getVatAmount() method

Syntax

Long getVatAmount ()

Returns

the transaction's VAT amount

Usage

Returns the VAT share of the transaction amount.

the transaction's VAT amount

isSetBalancePayee() method

Syntax

boolean isSetBalancePayee ()

Returns

false if the balancePayee is null. Otherwise, it returns true.

Usage

Checks whether the

balancePayee

of this instance is set.

false if the balancePayee is null. Otherwise, it returns true.

isSetBalancePayer() method

Syntax

boolean isSetBalancePayer ()

Returns

false if the balancePayer is null. Otherwise, it returns true.

Usage

Checks whether the

balancePayer

of this instance is set.

false if the balancePayer is null. Otherwise, it returns true.

isSetCallbackStatus() method

Syntax

```
boolean isSetCallbackStatus ()
```

Returns

false if the callbackStatus is null. Otherwise, it returns true.

Usage

Checks whether the

callbackStatus

of this instance is set.

false if the callbackStatus is null. Otherwise, it returns true.

isSetCallbackUrl() method

Syntax

```
boolean isSetCallbackUrl ()
```

Returns

false if the callbackUrl is null. Otherwise, it returns true.

Usage

Checks whether the

callbackUrl

of this instance is set.

false if the callbackUrl is null. Otherwise, it returns true.

isSetCaller() method

Syntax

```
boolean isSetCaller ()
```

Returns

false if the caller is null. Otherwise, it returns true.

Usage

Checks whether the

caller

of this instance is set.

false if the caller is null. Otherwise, it returns true.

isSetCallerIdentification() method

Syntax

```
boolean isSetCallerIdentification ()
```

Returns

false if the callerIdentification is null. Otherwise, it returns true.

Usage

Checks whether the

callerIdentification

of this instance is set.

false if the callerIdentification is null. Otherwise, it returns true.

isSetErrorCode() method

Syntax

```
boolean isSetErrorCode ()
```

Returns

false if the errorCode is null . Otherwise, it returns true.

Usage

Checks whether the

errorCode

of this instance is set.

false if the errorCode is null . Otherwise, it returns true.

isSetExchangeRatePayee() method

Syntax

```
boolean isSetExchangeRatePayee ()
```

Returns

false if the exchangeRatePayee is null. Otherwise, it returns true.

Usage

Checks whether the

exchangeRatePayee

of this instance is set.

false if the exchangeRatePayee is null. Otherwise, it returns true.

isSetExchangeRatePayer() method

Syntax

```
boolean isSetExchangeRatePayer ()
```

Returns

false if the exchangeRatePayer is null. Otherwise, it returns true.

Usage

Checks whether the

exchangeRatePayer

of this instance is set.

false if the exchangeRatePayer is null. Otherwise, it returns true.

isSetFees() method

Syntax

```
boolean isSetFees ()
```

Returns

false if the fees is null. Otherwise, it returns true.

Usage

Checks whether the

fees

of this instance is set.

false if the fees is null. Otherwise, it returns true.

isSetOrigin() method

Syntax

```
boolean isSetOrigin ()
```

Returns

false if the origin is null. Otherwise, it returns true.

Usage

Checks whether the

origin

of this instance is set.

false if the origin is null. Otherwise, it returns true.

isSetPayeePayment() method

Syntax

```
boolean isSetPayeePayment ()
```

Returns

false if the payeePayment is null. Otherwise, it returns true.

Usage

Checks whether the

payeePayment

of this instance is set.

false if the payeePayment is null. Otherwise, it returns true.

isSetPayerPayment() method

Syntax

```
boolean isSetPayerPayment ()
```

Returns

false if the payerPayment is null. Otherwise, it returns true.

Usage

Checks whether the

payerPayment

of this instance is set.

false if the payerPayment is null. Otherwise, it returns true.

isSetReturnUrl() method

Syntax

```
boolean isSetReturnUrl ()
```

Returns

false if the returnUrl is null . Otherwise, it returns true.

Usage

Checks whether the

returnUrl

of this instance is set.

false if the returnUrl is null . Otherwise, it returns true.

isSetSubAmount() method

Syntax

```
boolean isSetSubAmount ()
```

Returns

false if the subAmount is null . Otherwise, it returns true.

Usage

Checks whether the

subAmount

of this instance is set.

false if the subAmount is null . Otherwise, it returns true.

isSetSubTransactionType() method

Syntax

```
boolean isSetSubTransactionType ()
```

Returns

false if the subTransactionType is null. Otherwise, it returns true.

Usage

Checks whether the

subTransactionType

of this instance is set.

false if the subTransactionType is null. Otherwise, it returns true.

isSetTraceNumber() method

Syntax

```
boolean isSetTraceNumber ()
```

Returns

false if the traceNumber is null. Otherwise, it returns true.

Usage

Checks whether the

traceNumber

of this instance is set.

false if the traceNumber is null. Otherwise, it returns true.

isSetTransaction() method

Syntax

```
boolean isSetTransaction ()
```

Returns

false if the transaction is null. Otherwise, it returns true.

Usage

Checks whether the

transaction

of this instance is set.

false if the transaction is null. Otherwise, it returns true.

isSetTransactionReferencePayee() method

Syntax

```
boolean isSetTransactionReferencePayee ()
```

Returns

false if the transactionReferencePayee is null. Otherwise, it returns true.

Usage

Checks whether the

transactionReferencePayee

of this instance is set.

false if the transactionReferencePayee is null. Otherwise, it returns true.

isSetTransactionReferencePayer() method

Syntax

```
boolean isSetTransactionReferencePayer ()
```

Returns

false if the transactionReferencePayer is null. Otherwise, it returns true.

Usage

Checks whether the

transactionReferencePayer

of this instance is set.

false if the transactionReferencePayer is null. Otherwise, it returns true.

isSetTransactionTimestampPayee() method

Syntax

```
boolean isSetTransactionTimestampPayee ()
```

Returns

false if the transactionTimestampPayee is null. Otherwise, it returns true.

Usage

Checks whether the

transactionTimestampPayee

of this instance is set.

false if the transactionTimestampPayee is null. Otherwise, it returns true.

isSetTransactionTimestampPayer() method

Syntax

```
boolean isSetTransactionTimestampPayer ()
```

Returns

false if the transactionTimestampPayer is null. Otherwise, it returns true.

Usage

Checks whether the

transactionTimestampPayer

of this instance is set.

false if the transactionTimestampPayer is null. Otherwise, it returns true.

*isSetVatAmount() method***Syntax**

```
boolean isSetVatAmount ()
```

Returns

false if the vatAmount is null . Otherwise, it returns true.

Usage

Checks whether the

vatAmount

of this instance is set.

false if the vatAmount is null . Otherwise, it returns true.

*setBalancePayee(Long) method***Syntax**

```
void setBalancePayee ( Long balancePayee )
```

Parameters

- **balancePayee** – the balancePayee value to set

Usage

Sets the

balancePayee

of this instance.

*setBalancePayer(Long) method***Syntax**

```
void setBalancePayer ( Long balancePayer )
```

Parameters

- **balancePayer** – the balancePayer value to set

Usage

Sets the

balancePayer

of this instance.

setCallbackStatus(int) method

Syntax

```
void setCallbackStatus ( int callbackStatus )
```

Parameters

- **callbackStatus** – the callbackStatus to set

setCallbackUrl(String) method

Syntax

```
void setCallbackUrl ( String callbackUrl )
```

Parameters

- **callbackUrl** – the callbackUrl to set

setCaller(Customer) method

Syntax

```
void setCaller ( Customer caller )
```

Parameters

- **caller** – the caller to set

setCallerIdentification(String) method

Syntax

```
void setCallerIdentification ( String callerIdentification )
```

Parameters

- **callerIdentification** – the callerIdentification to set

setErrorCode(int) method

Syntax

```
void setErrorCode ( int errorCode )
```

Parameters

- **errorCode** – the errorCode to set

setExchangeRatePayee(BigDecimal) method

Syntax

```
void setExchangeRatePayee ( BigDecimal exchangeRatePayee )
```

Parameters

- **exchangeRatePayee** – the exchangeRatePayee value to set

Usage

Sets the

exchangeRatePayee

of this instance.

setExchangeRatePayer(BigDecimal) method

Syntax

```
void setExchangeRatePayer ( BigDecimal exchangeRatePayer )
```

Parameters

- **exchangeRatePayer** – the exchangeRatePayer value to set

Usage

Sets the

exchangeRatePayer

of this instance.

setFees(List< Fee >) method

Syntax

void setFees (List< Fee > *fees*)

Parameters

- **fees** – the fees value to set

Usage

Sets the

fees

of this instance.

setOrigin(String) method

Syntax

void setOrigin (String *origin*)

Parameters

- **origin** – the origin to set

setPayeePayment(Long) method

Syntax

void setPayeePayment (Long *payeePayment*)

Parameters

- **payeePayment** – the payeePayment to set

setPayerPayment(Long) method

Syntax

void setPayerPayment (Long *payerPayment*)

Parameters

- **payerPayment** – the payerPayment to set

setReturnUrl(String) method

Syntax

`void setReturnUrl ( String returnUrl )`

Parameters

- **returnUrl** – the returnUrl to set

setSubAmount(long) method

Syntax

`void setSubAmount ( long amount )`

Parameters

- **amount** – the amount value to set

Usage

Sets the

amount

of this instance.

setSubTransactionType(SubTransactionType) method

Syntax

`void setSubTransactionType ( SubTransactionType subTransactionType )`

Parameters

- **subTransactionType** – the subTransactionType to set

setTraceNumber(String) method

Syntax

`void setTraceNumber ( String traceNumber )`

Parameters

- **traceNumber** – the traceNumber to set

setTransaction(Transaction) method

Syntax

void setTransaction (Transaction *transaction*)

Parameters

- **transaction** – the transaction to set

setTransactionReferencePayee(String) method

Syntax

void setTransactionReferencePayee (String *transactionReferencePayee*)

Parameters

- **transactionReferencePayee** – the transactionReferencePayee value to set

Usage

Sets the

transactionReferencePayee

of this instance.

setTransactionReferencePayer(String) method

Syntax

void setTransactionReferencePayer (String *transactionReferencePayer*)

Parameters

- **transactionReferencePayer** – the transactionReferencePayer value to set

Usage

Sets the

transactionReferencePayer

of this instance.

setTransactionTimestampPayee(Date) method

Syntax

void setTransactionTimestampPayee (Date *transactionTimestampPayee*)

Parameters

- **transactionTimestampPayee** – the transactionTimestampPayee value to set

Usage

Sets the

transactionTimestampPayee

of this instance.

setTransactionTimestampPayer(Date) method

Syntax

void setTransactionTimestampPayer (Date *transactionTimestampPayer*)

Parameters

- **transactionTimestampPayer** – the transactionTimestampPayer value to set

Usage

Sets the

transactionTimestampPayer

of this instance.

setVatAmount(Long) method

Syntax

void setVatAmount (Long *vatAmount*)

Parameters

- **vatAmount** – the vatAmount value to set

Usage

Sets the

vatAmount

of this instance.

SvaBooking class

Syntax

```
public class SvaBooking
```

Usage

The SvaBooking manages the Stored Value Account Bookings of a customer.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

SvaBooking() constructor

Syntax

```
SvaBooking ()
```

Usage

Creates a new instance of

```
SvaBooking
```

.

getAmount() method

Syntax

```
long getAmount ()
```

Returns

the amount

Usage

the amount

getBalance() method

Syntax

```
long getBalance ()
```

Returns

the balance

Usage

the balance

getBookingStatus() method

Syntax

String getBookingStatus ()

Returns

the dbBookingStatus

Usage

the dbBookingStatus

getBookingType() method

Syntax

String getBookingType ()

Returns

the dbBookingType

Usage

the dbBookingType

getRefSvaBooking() method

Syntax

SvaBooking getRefSvaBooking ()

Returns

the refSvaBooking

Usage

the refSvaBooking

getSvaAccount() method

Syntax

StoredValueAccount getSvaAccount ()

Returns

the svaAccount

Usage

the svaAccount

getTxnId() method

Syntax

long getTxnId ()

Returns

the txnId

Usage

the txnId

isSetAmount() method

Syntax

boolean isSetAmount ()

Returns

false if the amount is null. Otherwise, it returns true.

Usage

Checks whether the

amount

of this instance is set.

false if the amount is null. Otherwise, it returns true.

isSetBalance() method

Syntax

boolean isSetBalance ()

Returns

false if the balance is null. Otherwise, it returns true.

Usage

Checks whether the

balance

of this instance is set.

false if the balance is null. Otherwise, it returns true.

isSetBookingStatus() method

Syntax

boolean isSetBookingStatus ()

Returns

false if the dbBookingStatus is null. Otherwise, it returns true.

Usage

Checks whether the

dbBookingStatus

of this instance is set.

false if the dbBookingStatus is null. Otherwise, it returns true.

isSetBookingType() method

Syntax

boolean isSetBookingType ()

Returns

false if the dbBookingType is null. Otherwise, it returns true.

Usage

Checks whether the

dbBookingType

of this instance is set.

false if the dbBookingType is null. Otherwise, it returns true.

isSetRefSvaBooking() method

Syntax

```
boolean isSetRefSvaBooking ()
```

Returns

false if the refSvaBooking is null. Otherwise, it returns true.

Usage

Checks whether the

refSvaBooking

of this instance is set.

false if the refSvaBooking is null. Otherwise, it returns true.

isSetSvaAccount() method

Syntax

```
boolean isSetSvaAccount ()
```

Returns

false if the svaAccount is null. Otherwise, it returns true.

Usage

Checks whether the

svaAccount

of this instance is set.

false if the svaAccount is null. Otherwise, it returns true.

*isSetTxnId() method***Syntax**

```
boolean isSetTxnId ()
```

Returns

false if the `txnId` is null. Otherwise, it returns true.

Usage

Checks whether the

txnId

of this instance is set.

false if the `txnId` is null. Otherwise, it returns true.

*setAmount(long) method***Syntax**

```
void setAmount ( long amount )
```

Parameters

- **amount** – the amount to set

*setBalance(long) method***Syntax**

```
void setBalance ( long balance )
```

Parameters

- **balance** – the balance to set

*setBookingStatus(String) method***Syntax**

```
void setBookingStatus ( String bookingStatus )
```

Parameters

- **dbBookingStatus** – the `dbBookingStatus` to set

setBookingType(String) method

Syntax

```
void setBookingType ( String bookingType )
```

Parameters

- **bookingType** – the dbBookingType to set

setRefSvaBooking(SvaBooking) method

Syntax

```
void setRefSvaBooking ( SvaBooking refSvaBooking )
```

Parameters

- **refSvaBooking** – the refSvaBooking to set

setSvaAccount(StoredValueAccount) method

Syntax

```
void setSvaAccount ( StoredValueAccount svaAccount )
```

Parameters

- **svaAccount** – the svaAccount to set

setTxnId(long) method

Syntax

```
void setTxnId ( long txnId )
```

Parameters

- **txnId** – the txnId to set

TraceableRequest class

Syntax

```
public class TraceableRequest
```

Usage

This is used to store XML response messages into the DB as CLOBS. .

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

TraceableRequest() constructor

Syntax

```
TraceableRequest ()
```

Usage

Creates a new instance of

```
TraceableRequest
```

.

getJaxbClassName() method

Syntax

```
String getJaxbClassName ()
```

getOrigin() method

Syntax

```
String getOrigin ()
```

getResponse() method

Syntax

```
String getResponse ()
```

getTimeSegment() method

Syntax

```
long getTimeSegment ()
```

getTraceNo() method

Syntax

```
String getTraceNo ()
```

isProcessingComplete() method

Syntax

boolean isProcessingComplete ()

isSetJaxbClassName() method

Syntax

boolean isSetJaxbClassName ()

Returns

false if the jaxbClassName is null. Otherwise, it returns true.

Usage

Checks whether the

jaxbClassName

of this instance is set.

false if the jaxbClassName is null. Otherwise, it returns true.

isSetOrigin() method

Syntax

boolean isSetOrigin ()

Returns

false if the origin is null. Otherwise, it returns true.

Usage

Checks whether the

origin

of this instance is set.

false if the origin is null. Otherwise, it returns true.

isSetProcessingComplete() method

Syntax

boolean isSetProcessingComplete ()

Returns

false if the `dbprocessingComplete` is null. Otherwise, it returns true.

Usage

Checks whether the

dbprocessingComplete

of this instance is set.

false if the `dbprocessingComplete` is null. Otherwise, it returns true.

isSetResponse() method

Syntax

```
boolean isSetResponse ()
```

Returns

false if the response is null. Otherwise, it returns true.

Usage

Checks whether the

response

of this instance is set.

false if the response is null. Otherwise, it returns true.

isSetTimeSegment() method

Syntax

```
boolean isSetTimeSegment ()
```

Returns

false if the `timeSegment` is null. Otherwise, it returns true.

Usage

Checks whether the

timeSegment

of this instance is set.

false if the `timeSegment` is null. Otherwise, it returns true.

isSetTraceNo() method

Syntax

boolean isSetTraceNo ()

Returns

false if the traceNo is null. Otherwise, it returns true.

Usage

Checks whether the

traceNo

of this instance is set.

false if the traceNo is null. Otherwise, it returns true.

setJaxbClassName(String) method

Syntax

void setJaxbClassName (String *jaxbClassName*)

setOrigin(String) method

Syntax

void setOrigin (String *origin*)

setProcessingComplete(boolean) method

Syntax

void setProcessingComplete (boolean *processingComplete*)

setResponse(String) method

Syntax

void setResponse (String *response*)

setTimeSegment(long) method

Syntax

void setTimeSegment (long *timeSegment*)

*setTraceNo(String) method***Syntax**

```
void setTraceNo ( String traceNo )
```

Transaction class**Syntax**

```
public class Transaction
```

Usage

The `Transaction` class is the base class for all Mobiliser transactions.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

*Transaction() constructor***Syntax**

```
Transaction ()
```

Usage

Creates a new instance of

```
Transaction
```

```
.
```

*getAmount() method***Syntax**

```
Long getAmount ()
```

Returns

the amount

Usage

the amount

getAttribute(int) method

Returns the first attribute filtered by the key.

Syntax

```
String getAttribute ( int key )
```

Parameters

- **key** –

Returns

The value of the key

Usage

The value of the key

getAttributes(int) method

Returns a list of attributes filtered by the key.

Syntax

```
Set< String > getAttributes ( int key )
```

Parameters

- **key** –

Returns

Returns the list of all attributes matching the key

Usage

Returns the list of all attributes matching the key

getAttributes() method

Syntax

```
Set< TransactionAttribute > getAttributes ()
```

Returns

a Set containing the TransactionAttributes

Usage

Returns the transaction's attributes

a Set containing the TransactionAttributes

getAuthCode() method

Syntax

String getAuthCode ()

Returns

the authCode

Usage

the authCode

getAuthMethodPayee() method

Syntax

Integer getAuthMethodPayee ()

Returns

the dbAuthMethodPayee

Usage

the dbAuthMethodPayee

getAuthMethodPayer() method

Syntax

Integer getAuthMethodPayer ()

Returns

the dbAuthMethodPayer

Usage

the dbAuthMethodPayer

getCurrency() method

Syntax

String getCurrency ()

Returns

the dbCurrency

Usage

the dbCurrency

getErrorCode() method

Syntax

int getErrorCode ()

Returns

the errorCode

Usage

the errorCode

getExpiration() method

Syntax

Date getExpiration ()

Returns

the expiration

Usage

the expiration

getOrderChannel() method

Syntax

Integer getOrderChannel ()

Returns

the orderChannel

Usage

the orderChannel

getOrderId() method

Syntax

String getOrderId ()

Returns

the orderId

Usage

the orderId

getPayee() method

Syntax

Customer getPayee ()

Returns

the payee

Usage

the payee

getPayeeIdentification() method

Syntax

String getPayeeIdentification ()

Returns

the payeeIdentification

Usage

the payeeIdentification

getPayer() method

Syntax

Customer getPayer ()

Returns

the payer

Usage

the payer

getPayerIdentification() method

Syntax

String getPayerIdentification ()

Returns

the payerIdentification

Usage

the payerIdentification

getPaymentInstrumentPayee() method

Syntax

PaymentInstrument getPaymentInstrumentPayee ()

Returns

the paymentInstrumentPayee

Usage

the paymentInstrumentPayee

getPaymentInstrumentPayer() method

Syntax

PaymentInstrument getPaymentInstrumentPayer ()

Returns

the paymentInstrumentPayer

Usage

the paymentInstrumentPayer

getRefTransaction() method

Syntax

Transaction getRefTransaction ()

Returns

the refTransaction

Usage

the refTransaction

getText() method

Syntax

String getText ()

Returns

the text

Usage

the text

getTimestamp() method

Syntax

Date getTimestamp ()

Returns

the timestamp

Usage

the timestamp

getTransactionStatus() method

Syntax

int getTransactionStatus ()

Returns

the transactionatus

Usage

the transactionatus

getUseCase() method

Syntax

UseCase getUseCase ()

Returns

the useCase

Usage

the useCase

isAutoCapture() method

Syntax

boolean isAutoCapture ()

Returns

the dbAutoCapture

Usage

the dbAutoCapture

isCancelled() method

Syntax

boolean isCancelled ()

Returns

the dbCancelled

Usage

the dbCancelled

isDeliveryAddressRequested() method

Syntax

boolean isDeliveryAddressRequested ()

Returns

the dbDeliveryAddressRequested

Usage

the dbDeliveryAddressRequested

isLocked() method

Syntax

```
boolean isLocked ()
```

Returns

the dbLocked

Usage

the dbLocked

isSetAmount() method

Syntax

```
boolean isSetAmount ()
```

Returns

false if the amount is null. Otherwise, it returns true.

Usage

Checks whether the

amount

of this instance is set.

false if the amount is null. Otherwise, it returns true.

isSetAttributes() method

Syntax

```
boolean isSetAttributes ()
```

Returns

false if the attributes is null. Otherwise, it returns true.

Usage

Checks whether the

attributes

of this instance is set.

false if the attributes is null. Otherwise, it returns true.

isSetAuthCode() method

Syntax

```
boolean isSetAuthCode ()
```

Returns

false if the authCode is null. Otherwise, it returns true.

Usage

Checks whether the

authCode

of this instance is set.

false if the authCode is null. Otherwise, it returns true.

isSetAuthMethodPayee() method

Syntax

```
boolean isSetAuthMethodPayee ()
```

Returns

false if the dbAuthMethodPayee is null. Otherwise, it returns true.

Usage

Checks whether the

dbAuthMethodPayee

of this instance is set.

false if the dbAuthMethodPayee is null. Otherwise, it returns true.

isSetAuthMethodPayer() method

Syntax

boolean isSetAuthMethodPayer ()

Returns

false if the dbAuthMethodPayer is null. Otherwise, it returns true.

Usage

Checks whether the

dbAuthMethodPayer

of this instance is set.

false if the dbAuthMethodPayer is null. Otherwise, it returns true.

isSetAutoCapture() method

Syntax

boolean isSetAutoCapture ()

Returns

false if the dbAutoCapture is null. Otherwise, it returns true.

Usage

Checks whether the

dbAutoCapture

of this instance is set.

false if the dbAutoCapture is null. Otherwise, it returns true.

isSetCancelled() method

Syntax

boolean isSetCancelled ()

Returns

false if the dbCancelled is null . Otherwise, it returns true.

Usage

Checks whether the

dbCancelled

of this instance is set.

false if the dbCancelled is null . Otherwise, it returns true.

isSetCurrency() method

Syntax

boolean isSetCurrency ()

Returns

false if the dbCurrency is null. Otherwise, it returns true.

Usage

Checks whether the

dbCurrency

of this instance is set.

false if the dbCurrency is null. Otherwise, it returns true.

isSetDeliveryAddressRequested() method

Syntax

boolean isSetDeliveryAddressRequested ()

Returns

false if the dbDeliveryAddressRequested is null. Otherwise, it returns true.

Usage

Checks whether the

dbDeliveryAddressRequested

of this instance is set.

false if the dbDeliveryAddressRequested is null. Otherwise, it returns true.

isSetErrorCode() method

Syntax

`boolean isSetErrorCode ()`

Returns

false if the errorCode is null . Otherwise, it returns true.

Usage

Checks whether the

errorCode

of this instance is set.

false if the errorCode is null . Otherwise, it returns true.

isSetExpiration() method

Syntax

`boolean isSetExpiration ()`

Returns

false if the expiration is null. Otherwise, it returns true.

Usage

Checks whether the

expiration

of this instance is set.

false if the expiration is null. Otherwise, it returns true.

isSetLocked() method

Syntax

`boolean isSetLocked ()`

Returns

false if the dbLocked is null. Otherwise, it returns true.

Usage

Checks whether the

dbLocked

of this instance is set.

false if the dbLocked is null. Otherwise, it returns true.

isSetOrderChannel() method

Syntax

```
boolean isSetOrderChannel ()
```

Returns

false if the orderChannel is null. Otherwise, it returns true.

Usage

Checks whether the

orderChannel

of this instance is set.

false if the orderChannel is null. Otherwise, it returns true.

isSetOrderId() method

Syntax

```
boolean isSetOrderId ()
```

Returns

false if the orderId is null. Otherwise, it returns true.

Usage

Checks whether the

orderId

of this instance is set.

false if the orderId is null. Otherwise, it returns true.

isSetPayee() method

Syntax

```
boolean isSetPayee ()
```

Returns

false if the payee is null. Otherwise, it returns true.

Usage

Checks whether the

payee

of this instance is set.

false if the payee is null. Otherwise, it returns true.

isSetPayeeIdentification() method

Syntax

```
boolean isSetPayeeIdentification ()
```

Returns

false if the payeeIdentification is null. Otherwise, it returns true.

Usage

Checks whether the

payeeIdentification

of this instance is set.

false if the payeeIdentification is null. Otherwise, it returns true.

isSetPayer() method

Syntax

```
boolean isSetPayer ()
```

Returns

false if the payer is null. Otherwise, it returns true.

Usage

Checks whether the

payer

of this instance is set.

false if the payer is null. Otherwise, it returns true.

isSetPayerIdentification() method

Syntax

```
boolean isSetPayerIdentification ()
```

Returns

false if the payerIdentification is null. Otherwise, it returns true.

Usage

Checks whether the

payerIdentification

of this instance is set.

false if the payerIdentification is null. Otherwise, it returns true.

isSetPaymentInstrumentPayee() method

Syntax

```
boolean isSetPaymentInstrumentPayee ()
```

Returns

false if the paymentInstrumentPayee is null. Otherwise, it returns true.

Usage

Checks whether the

paymentInstrumentPayee

of this instance is set.

false if the paymentInstrumentPayee is null. Otherwise, it returns true.

isSetPaymentInstrumentPayer() method

Syntax

`boolean isSetPaymentInstrumentPayer ()`

Returns

false if the `paymentInstrumentPayer` is null. Otherwise, it returns true.

Usage

Checks whether the

paymentInstrumentPayer

of this instance is set.

false if the `paymentInstrumentPayer` is null. Otherwise, it returns true.

isSetRefTransaction() method

Syntax

`boolean isSetRefTransaction ()`

Returns

false if the `refTransaction` is null. Otherwise, it returns true.

Usage

Checks whether the

refTransaction

of this instance is set.

false if the `refTransaction` is null. Otherwise, it returns true.

isSetTest() method

Syntax

`boolean isSetTest ()`

Returns

false if the `dbTest` is null. Otherwise, it returns true.

Usage

Checks whether the

dbTest

of this instance is set.

false if the dbTest is null. Otherwise, it returns true.

isSetText() method

Syntax

```
boolean isSetText ()
```

Returns

false if the text is null. Otherwise, it returns true.

Usage

Checks whether the

text

of this instance is set.

false if the text is null. Otherwise, it returns true.

isSetTimestamp() method

Syntax

```
boolean isSetTimestamp ()
```

Returns

false if the timestamp is null . Otherwise, it returns true.

Usage

Checks whether the

timestamp

of this instance is set.

false if the timestamp is null . Otherwise, it returns true.

*isSetTransactionStatus() method***Syntax**

```
boolean isSetTransactionStatus ()
```

Returns

false if the transactionStatus is null. Otherwise, it returns true.

Usage

Checks whether the

transactionStatus

of this instance is set.

false if the transactionStatus is null. Otherwise, it returns true.

*isSetUseCase() method***Syntax**

```
boolean isSetUseCase ()
```

Returns

false if the useCase is null. Otherwise, it returns true.

Usage

Checks whether the

useCase

of this instance is set.

false if the useCase is null. Otherwise, it returns true.

*isTest() method***Syntax**

```
boolean isTest ()
```

Returns

the dbTest

Usage

the dbTest

setAmount(Long) method

Syntax

void setAmount (Long *amount*)

Parameters

- **amount** – the amount to set

setAttributes(Set< TransactionAttribute >) method

Syntax

void setAttributes (Set< TransactionAttribute > *attributes*)

Parameters

- **attributes** – the attributes value to set

Usage

Sets the

attributes

of this instance.

setAuthCode(String) method

Syntax

void setAuthCode (String *authCode*)

Parameters

- **authCode** – the authCode to set

setAuthMethodPayee(Integer) method

Syntax

void setAuthMethodPayee (Integer *dbAuthMethodPayee*)

Parameters

- **dbAuthMethodPayee** – the dbAuthMethodPayee to set

setAuthMethodPayer(Integer) method

Syntax

```
void setAuthMethodPayer ( Integer dbAuthMethodPayer )
```

Parameters

- **dbAuthMethodPayer** – the dbAuthMethodPayer to set

setAutoCapture(boolean) method

Syntax

```
void setAutoCapture ( boolean autoCapture )
```

Parameters

- **dbAutoCapture** – the dbAutoCapture to set

setCancelled(boolean) method

Syntax

```
void setCancelled ( boolean cancelled )
```

Parameters

- **dbCancelled** – the dbCancelled to set

setCurrency(String) method

Syntax

```
void setCurrency ( String dbCurrency )
```

Parameters

- **dbCurrency** – the dbCurrency to set

setDeliveryAddressRequested(boolean) method

Syntax

```
void setDeliveryAddressRequested ( boolean deliveryAddressRequested )
```

Parameters

- **dbDeliveryAddressRequested** – the dbDeliveryAddressRequested to set

setErrorCode(int) method

Syntax

```
void setErrorCode ( int errorCode )
```

Parameters

- **errorCode** – the errorCode to set

setExpiration(Date) method

Syntax

```
void setExpiration ( Date expiration )
```

Parameters

- **expiration** – the expiration to set

setLocked(boolean) method

Syntax

```
void setLocked ( boolean locked )
```

Parameters

- **dbLocked** – the dbLocked to set

setOrderChannel(Integer) method

Syntax

```
void setOrderChannel ( Integer orderChannel )
```

Parameters

- **orderChannel** – the orderChannel to set

setOrderId(String) method

Syntax

```
void setOrderId ( String orderId )
```

Parameters

- **orderId** – the orderId to set

setPayee(Customer) method

Syntax

```
void setPayee ( Customer payee )
```

Parameters

- **payee** – the payee to set

setPayeeIdentification(String) method

Syntax

```
void setPayeeIdentification ( String payeeIdentification )
```

Parameters

- **payeeIdentification** – the payeeIdentification to set

setPayer(Customer) method

Syntax

```
void setPayer ( Customer payer )
```

Parameters

- **payer** – the payer to set

setPayerIdentification(String) method

Syntax

```
void setPayerIdentification ( String payerIdentification )
```

Parameters

- **payerIdentification** – the payerIdentification to set

setPaymentInstrumentPayee(PaymentInstrument) method

Syntax

```
void setPaymentInstrumentPayee ( PaymentInstrument  
paymentInstrumentPayee )
```

Parameters

- **paymentInstrumentPayee** – the paymentInstrumentPayee to set

setPaymentInstrumentPayer(PaymentInstrument) method

Syntax

```
void setPaymentInstrumentPayer ( PaymentInstrument  
paymentInstrumentPayer )
```

Parameters

- **paymentInstrumentPayer** – the paymentInstrumentPayer to set

setRefTransaction(Transaction) method

Syntax

```
void setRefTransaction ( Transaction refTransaction )
```

Parameters

- **refTransaction** – the refTransaction to set

setTest(boolean) method

Syntax

```
void setTest ( boolean test )
```

Parameters

- **dbTest** – the dbTest to set

setText(String) method

Syntax

```
void setText ( String text )
```


Parameters

- **text** – the text to set

setTimestamp(Date) method

Syntax

```
void setTimestamp ( Date timestamp )
```

Parameters

- **timestamp** – the timestamp to set

setTransactionStatus(int) method

Syntax

```
void setTransactionStatus ( int transactionStatus )
```

Parameters

- **transactionatus** – the transactionatus to set

setUseCase(UseCase) method

Syntax

```
void setUseCase ( UseCase useCase )
```

Parameters

- **useCase** – the useCase to set

TransactionAttribute class

Syntax

```
public class TransactionAttribute
```

Usage

The `TransactionNetwork` class represents the relations between transactions.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

TransactionAttributePk class

Syntax

```
public class TransactionAttributePk
```

Usage

The TransactionAttributePk class is the embeddable primary key object for a transaction Attribute.

© 2000-2007 by paybox solutions AG

Revision:1.3Name:not supported by cvs2svn

TransactionAttributePk() constructor

Syntax

```
TransactionAttributePk ()
```

Usage

Creates a new instance of

```
TransactionAttributePk
```

.

TransactionAttributePk(Transaction, int) constructor

Syntax

```
TransactionAttributePk ( Transaction transaction , int  
transactionAttributeTypeId )
```

Parameters

- **transaction** –
- **transactionAttributeTypeId** –

Usage

Creates a new instance of

```
TransactionAttributePk
```

.

equals(Object) method

Syntax

`boolean equals ( Object o )`

Parameters

- **o** – the Object to compare with

Returns

true if this class is assignable from o's class and has the same PK values as this instance

Usage

Indicates if another object is

equal to

this one.

true if this class is assignable from o's class and has the same PK values as this instance

getTransaction() method

Syntax

`Transaction getTransaction ()`

Returns

the transaction

Usage

the transaction

getTransactionAttributeTypeId() method

Syntax

`int getTransactionAttributeTypeId ()`

Returns

the attribute

Usage

the attribute

hashCode() method

Syntax

`int hashCode ()`

Returns

a hash value for this instance

Usage

Returns a hash value.

a hash value for this instance

isSetTransaction() method

Syntax

`boolean isSetTransaction ()`

Returns

false if the transaction is null. Otherwise, it returns true.

Usage

Checks whether the

transaction

of this instance is set.

false if the transaction is null. Otherwise, it returns true.

isSetTransactionAttributeTypeId() method

Syntax

`boolean isSetTransactionAttributeTypeId ()`

Returns

false if the transactionAttributeTypeId is null. Otherwise, it returns true

Usage

Checks whether the

transactionAttributeTypeId

of this instance is set.

false if the transactionAttributeTypeId is null. Otherwise, it returns true

toString() method

Syntax

String toString ()

Returns

a String representing this instance

Usage

Returns a string representation of this object.

a String representing this instance

TransactionAttribute() constructor

Syntax

TransactionAttribute ()

Usage

Creates a new instance of

TransactionAttribute

.

TransactionAttribute(TransactionAttributePk) constructor

Syntax

TransactionAttribute (TransactionAttributePk *id*)

Parameters

- **id** – the PK value

Usage

Creates a new instance of

TransactionAttribute

with a preset primary key.

getTransaction() method

Syntax

Transaction getTransaction ()

Returns

the transaction

Usage

the transaction

getTransactionAttributeTypeId() method

Syntax

int getTransactionAttributeTypeId ()

Returns

the attribute

Usage

the attribute

getValue() method

Syntax

String getValue ()

Returns

the value

Usage

the value

isSetTransaction() method

Syntax

boolean isSetTransaction ()

Returns

false if the id or the id's transaction is null. Otherwise, it returns true.

Usage

Checks whether the

transaction

of this instance's PK is set.

false if the id or the id's transaction is null. Otherwise, it returns true.

isSetTransactionAttributeTypeId() method

Syntax

```
boolean isSetTransactionAttributeTypeId ()
```

Returns

false if the id or the id's transactionAttributeTypeId is null. Otherwise, it returns true.

Usage

Checks whether the

transactionAttributeTypeId

of this instance's PK is set.

false if the id or the id's transactionAttributeTypeId is null. Otherwise, it returns true.

isSetValue() method

Syntax

```
boolean isSetValue ()
```

Returns

false if the value is null. Otherwise, it returns true.

Usage

Checks whether the

value

of this instance is set.

false if the value is null. Otherwise, it returns true.

setValue(String) method

Syntax

```
void setValue ( String value )
```

Parameters

- **value** – the value to set

XmlMessage class

Syntax

```
public class XmlMessage
```

Usage

This is used to store XML messages into the DB as BLOBS. There are some specific methods for marshalling and unmarshalling of JAXB beans.

2013 SAP AG or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

XmlMessage() constructor

Syntax

```
XmlMessage ( )
```

Usage

Creates a new instance of

```
XmlMessage
```

.

getCustomer() method

Syntax

```
Customer getCustomer ( )
```

Returns

the customer

Usage

the customer

getJaxbClassName() method

Syntax

```
String getJaxbClassName ()
```

Returns

the jaxbClassName

Usage

the jaxbClassName

getMessage() method

Syntax

```
String getMessage ()
```

Returns

the message

Usage

the message

getSubTxn() method

Syntax

```
SubTransaction getSubTxn ()
```

Returns

the txn

Usage

the txn

getSystem() method

Syntax

```
String getSystem ()
```

Returns

the system from where the message was received or sent to

Usage

the system from where the message was received or sent to

isIncoming() method

Syntax

```
Boolean isIncoming ()
```

Returns

Indicates if this was an incoming message

Usage

Indicates if this was an incoming message

isSetCustomer() method

Syntax

```
boolean isSetCustomer ()
```

Returns

false if the customer is null. Otherwise, it returns true.

Usage

Checks whether the

customer

of this instance is set.

false if the customer is null. Otherwise, it returns true.

isSetIncoming() method

Syntax

```
boolean isSetIncoming ()
```

Returns

false if the dbIncoming is null. Otherwise, it returns true.

Usage

Checks whether the

dbIncoming

of this instance is set.

false if the dbIncoming is null. Otherwise, it returns true.

isSetJaxbClassName() method

Syntax

```
boolean isSetJaxbClassName ()
```

Returns

false if the jaxbClassName is null. Otherwise, it returns true.

Usage

Checks whether the

jaxbClassName

of this instance is set.

false if the jaxbClassName is null. Otherwise, it returns true.

isSetMessage() method

Syntax

```
boolean isSetMessage ()
```

Returns

false if the message is null. Otherwise, it returns true.

Usage

Checks whether the

message

of this instance is set.

false if the message is null. Otherwise, it returns true.

isSetSubTxn() method

Syntax

boolean isSetSubTxn ()

Returns

false if the subTxn is null. Otherwise, it returns true.

Usage

Checks whether the

subTxn

of this instance is set.

false if the subTxn is null. Otherwise, it returns true.

isSetSystem() method

Syntax

boolean isSetSystem ()

Returns

false if the system is null. Otherwise, it returns true.

Usage

Checks whether the

system

of this instance is set.

false if the system is null. Otherwise, it returns true.

restoreJaxbBean(XmlMessage) method

Convenience method to marshall a JAXB bean into the XML_MESSAGES table.

Syntax

Object restoreJaxbBean (XmlMessage *xmlMessage*) throws
JAXBException, ClassNotFoundException

Parameters

- **session** –
- **jaxbMessage** –
- **txn** –
- **customer** –
- **incoming** –
- **system** –
- **xmlMessage** –

Returns

the ID of the XmlMessage that was created
the unmarshalled message

Exceptions

- **JAXBException** – Convenience method to unmarshall a JAXB bean from the XML_MESSAGES table.
- **JAXBException** –
- **ClassNotFoundException** –

Usage

the ID of the XmlMessage that was created

the unmarshalled message

setCustomer(Customer) method

Syntax

`void setCustomer ( Customer customer )`

Parameters

- **customer** – the customer to set

setIncoming(Boolean) method

Syntax

`void setIncoming ( Boolean incoming )`

Parameters

- **incoming** – the incoming value to set

Usage

Sets the

active

of this instance.

setJaxbClassName(String) method

Syntax

void setJaxbClassName (String *jaxbClassName*)

Parameters

- **jaxbClassName** – the jaxbClassName to set

setMessage(String) method

Syntax

void setMessage (String *message*)

Parameters

- **message** – the message to set

setSubTxn(SubTransaction) method

Syntax

void setSubTxn (SubTransaction *subTxn*)

Parameters

- **txn** – the txn to set

setSystem(String) method

Syntax

void setSystem (String *system*)

Parameters

- **system** – the system from where the message was received or sent to

svcpkg package

Members

All public members of the svcpkg package.

- **businesslogic package** –
- **dao package** –

businesslogic package

Members

All public members of the businesslogic package.

- **ISvcPkgCrudLogic interface** – 2013 SAP AG or an SAP affiliate company.
- **ISvcPkgLogicServices interface** – 2013 SAP AG or an SAP affiliate company.
- **MandatoryAndOptionalTncVersions class** – 2013 SAP AG or an SAP affiliate company.
- **TncVersionUpdate class** – 2013 SAP AG or an SAP affiliate company.

ISvcPkgCrudLogic interface

2013 SAP AG or an SAP affiliate company.

Syntax

```
public interface ISvcPkgCrudLogic
```

Usage

All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

createMobiliserServicePackage(com.sybase365.mobiliser.money.svcpkg.dao.model.MobiliserServicePackage, long) method

Syntax

```
long createMobiliserServicePackage
( com.sybase365.mobiliser.money.svcpkg.dao.model.MobiliserServicePackage bean , long callerId )
```

Parameters

- **bean** –

- **callerId** –

createSvcPkgCustomer(com.sybase365.mobiliser.money.svcpkg.dao.model.SvcPkgCustomer, long) method

Syntax

```
long createSvcPkgCustomer  
( com.sybase365.mobiliser.money.svcpkg.dao.model.SvcPkgCustomer bean , long callerId )
```

Parameters

- **bean** –
- **callerId** –

createSvcPkgTnc(com.sybase365.mobiliser.money.svcpkg.dao.model.SvcPkgTnc, long) method

Syntax

```
long createSvcPkgTnc  
( com.sybase365.mobiliser.money.svcpkg.dao.model.SvcPkgTnc bean , long callerId )
```

Parameters

- **bean** –
- **callerId** –

createTermAndCondition(com.sybase365.mobiliser.money.svcpkg.dao.model.TermAndCondition, long) method

Syntax

```
long createTermAndCondition  
( com.sybase365.mobiliser.money.svcpkg.dao.model.TermAndCondition bean , long callerId )
```

Parameters

- **bean** –
- **callerId** –

createTncVersion(com.sybase365.mobiliser.money.svcpkg.dao.model.TncVersion, long) method

Syntax

```
long createTncVersion  
( com.sybase365.mobiliser.money.svcpkg.dao.model.TncVersion  
bean , long callerId )
```

Parameters

- **bean** –
- **callerId** –

createTncVersionCustomer(com.sybase365.mobiliser.money.svcpkg.dao.model.TncVersionCustomer, long) method

Syntax

```
long createTncVersionCustomer  
( com.sybase365.mobiliser.money.svcpkg.dao.model.TncVersionCu  
stomer bean , long callerId )
```

Parameters

- **bean** –
- **callerId** –

deleteMobiliserServicePackage(long, long) method

Syntax

```
void deleteMobiliserServicePackage ( long id , long callerId )
```

Parameters

- **id** –
- **callerId** –

deleteSvcPkgCustomer(long, long) method

Syntax

```
void deleteSvcPkgCustomer ( long id , long callerId )
```

Parameters

- **id** –
- **callerId** –

deleteSvcPkgTnc(long, long) method

Syntax

```
void deleteSvcPkgTnc ( long id ,   long callerId )
```

Parameters

- **id** –
- **callerId** –

deleteTermAndCondition(long, long) method

Syntax

```
void deleteTermAndCondition ( long id ,   long callerId )
```

Parameters

- **id** –
- **callerId** –

deleteTncVersion(long, long) method

Syntax

```
void deleteTncVersion ( long id ,   long callerId )
```

Parameters

- **id** –
- **callerId** –

deleteTncVersionCustomer(long, long) method

Syntax

```
void deleteTncVersionCustomer ( long id ,   long callerId )
```

Parameters

- **id** –

- **callerId** –

getAllServicePackages(boolean, boolean, boolean, long) method

Syntax

```
java.util.List<  
com.sybase365.mobiliser.money.svcpkg.dao.model.MobiliserServicePackage > getAllServicePackages ( boolean activeOnly , boolean selfregistrationOnly , boolean autoAssignOnly , long callerId )
```

Parameters

- **activeOnly** –
- **selfregistrationOnly** –
- **autoAssignOnly** –
- **callerId** –

getCurrentTncVersionByTermAndConditionId(long, long) method

Syntax

```
com.sybase365.mobiliser.money.svcpkg.dao.model.TncVersion  
getCurrentTncVersionByTermAndConditionId ( long tncId , long callerId )
```

Parameters

- **tncId** –
- **callerId** –

getMobiliserServicePackage(long, long) method

Syntax

```
com.sybase365.mobiliser.money.svcpkg.dao.model.MobiliserServicePackage  
getMobiliserServicePackage ( long id , long callerId )
```

Parameters

- **id** –
- **callerId** –

getServicePackagesByCustomerId(long, boolean, boolean, long) method

Syntax

```
java.util.List<  
com.sybase365.mobiliser.money.svcpkg.dao.model.MobiliserServicePackage >
```

```
icePackage >getServicePackagesByCustomerId ( long customerId ,  
boolean activeOnly , boolean tncAcceptedOnly , long callerId )
```

Parameters

- **customerId** –
- **activeOnly** –
- **tncAcceptedOnly** –
- **callerId** –

getSvcPkgCustomer(long, long) method

Syntax

```
com.sybase365.mobiliser.money.svcpkg.dao.model.SvcPkgCustome  
r getSvcPkgCustomer ( long id , long callerId )
```

Parameters

- **id** –
- **callerId** –

getSvcPkgTnc(long, long) method

Syntax

```
com.sybase365.mobiliser.money.svcpkg.dao.model.SvcPkgTnc  
getSvcPkgTnc ( long id , long callerId )
```

Parameters

- **id** –
- **callerId** –

getSvcPkgTnCsBySvcPkgId(long, long) method

Syntax

```
java.util.List<  
com.sybase365.mobiliser.money.svcpkg.dao.model.SvcPkgTnc >  
getSvcPkgTnCsBySvcPkgId ( long svcPkgId , long callerId )
```

Parameters

- **svcPkgId** –
- **callerId** –

getTermAndCondition(long, long) method

Syntax

```
com.sybase365.mobiliser.money.svcpkg.dao.model.TermAndCondition  
getTermAndCondition ( long id , long callerId )
```

Parameters

- **id** –
- **callerId** –

getTermAndConditions(boolean, long) method

Syntax

```
java.util.List<  
com.sybase365.mobiliser.money.svcpkg.dao.model.TermAndCondition  
> getTermAndConditions ( boolean activeOnly , long callerId )
```

Parameters

- **activeOnly** –
- **callerId** –

getTermAndConditionsByServicePackageId(long, java.lang.Boolean, long) method

Syntax

```
java.util.List<  
com.sybase365.mobiliser.money.svcpkg.dao.model.TermAndCondition  
> getTermAndConditionsByServicePackageId ( long svcPkgId ,  
java.lang.Boolean mandatory , long callerId )
```

Parameters

- **svcPkgId** –
- **mandatory** –
- **callerId** –

getTncVersion(long, long) method

Syntax

```
com.sybase365.mobiliser.money.svcpkg.dao.model.TncVersion  
getTncVersion ( long id , long callerId )
```

Parameters

- **id** –
- **callerId** –

getTncVersionCustomer(long, long) method

Syntax

```
com.sybase365.mobiliser.money.svcpkg.dao.model.TncVersionCustomer  
getTncVersionCustomer ( long id , long callerId )
```

Parameters

- **id** –
- **callerId** –

getTncVersionsByTncId(long, long) method

Syntax

```
java.util.List<  
com.sybase365.mobiliser.money.svcpkg.dao.model.TncVersion >  
getTncVersionsByTncId ( long tncId , long callerId )
```

Parameters

- **tncId** –
- **callerId** –

updateMobiliserServicePackage(com.sybase365.mobiliser.money.svcpkg.dao.model.MobiliserServicePackage, long) method

Syntax

```
void updateMobiliserServicePackage  
( com.sybase365.mobiliser.money.svcpkg.dao.model.MobiliserServicePackage bean , long callerId )
```

Parameters

- **bean** –
- **callerId** –

updateSvcPkgCustomer(com.sybase365.mobiliser.money.svcpkg.dao.model.SvcPkgCustomer, long) method

Syntax

```
void updateSvcPkgCustomer  
( com.sybase365.mobiliser.money.svcpkg.dao.model.SvcPkgCustom  
er bean , long callerId )
```

Parameters

- **bean** –
- **callerId** –

updateSvcPkgTnc(com.sybase365.mobiliser.money.svcpkg.dao.model.SvcPkgTnc, long) method

Syntax

```
void updateSvcPkgTnc  
( com.sybase365.mobiliser.money.svcpkg.dao.model.SvcPkgTnc  
bean , long callerId )
```

Parameters

- **bean** –
- **callerId** –

updateTermAndCondition(com.sybase365.mobiliser.money.svcpkg.dao.model.TermAndCondition, long) method

Syntax

```
void updateTermAndCondition  
( com.sybase365.mobiliser.money.svcpkg.dao.model.TermAndCondi  
tion bean , long callerId )
```

Parameters

- **bean** –
- **callerId** –

updateTncVersion(com.sybase365.mobiliser.money.svcpkg.dao.model.TncVersion, long) method

Syntax

```
void updateTncVersion  
( com.sybase365.mobiliser.money.svcpkg.dao.model.TncVersion  
bean , long callerId )
```

Parameters

- **bean** –
- **callerId** –

updateTncVersionCustomer(com.sybase365.mobiliser.money.svcpkg.dao.model.TncVersionCustomer, long) method

Syntax

```
void updateTncVersionCustomer  
( com.sybase365.mobiliser.money.svcpkg.dao.model.TncVersionCu  
stomer bean , long callerId )
```

Parameters

- **bean** –
- **callerId** –

ISvcPkgLogicServices interface

2013 SAP AG or an SAP affiliate company.

Syntax

```
public interface ISvcPkgLogicServices
```

Usage

All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

acceptTermAndConditionVersions(long, List< Long >, long) method

Syntax

```
void acceptTermAndConditionVersions ( long customerId , List< Long  
> tncVerIds , long callerId )
```


*assignDefaultServicePackages(long, long) method***Syntax**

```
List< MobiliserServicePackage > assignDefaultServicePackages  
( long customerId , long callerID )
```

*assignServicePackageByCustomer(long, long, long) method***Syntax**

```
List< TncVersion > assignServicePackageByCustomer ( long  
customerId , long svcpkgId , long callerID )
```

*disableServicePackagesForCustomer(long, long, long) method***Syntax**

```
void disableServicePackagesForCustomer ( long customerId , long  
servicePackageId , long callerId )
```

*getCustomerPrivilegesFromServicePackages(long, long) method***Syntax**

```
List< String > getCustomerPrivilegesFromServicePackages ( long  
customerId , long callerId )
```

getDefaultTncVersions(int, String) method

Before a customer gets registered, this service is invoked to retrieve the set of standard TNCs that is to be displayed to the user.

Syntax

```
MandatoryAndOptionalTncVersions getDefaultTncVersions ( int  
customerTypeId , String orgUnitId )
```

Parameters

- **customerTypeId** –
- **orgUnitId** –

*getOpenTermAndConditions(long, long) method***Syntax**

```
List< TncVersion > getOpenTermAndConditions ( long customerID ,  
long callerID )
```

getServicePackagesForSelfRegistration(long, long) method

Syntax

```
List< MobiliserServicePackage >  
getServicePackagesForSelfRegistration ( long customerId , long  
callerId )
```

getTncsForServicePackage(long, long) method

Syntax

```
MandatoryAndOptionalTncVersions getTncsForServicePackage ( long  
svcpkgId , long calledId )
```

getUpdatedTncVersions(long, long) method

Syntax

```
List< TncVersionUpdate > getUpdatedTncVersions ( long customerId ,  
long callerID )
```

recalculateServicePackageEnablement(long, long) method

Syntax

```
void recalculateServicePackageEnablement ( long customerId , long  
callerID )
```

rejectTermsAndConditionVersion(long, List< Long >, long) method

Syntax

```
void rejectTermsAndConditionVersion ( long customerId , List< Long  
> tncVersionIds , long callerID )
```

MandatoryAndOptionalTncVersions class

2013 SAP AG or an SAP affiliate company.

Syntax

```
public class MandatoryAndOptionalTncVersions
```

Usage

All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getMandatoryTncVersions() method

Syntax

```
List< TncVersion > getMandatoryTncVersions ()
```

getOptionalTncVersions() method

Syntax

```
List< TncVersion > getOptionalTncVersions ()
```

mandatoryTncVersions variable

Syntax

```
List< TncVersion > mandatoryTncVersions
```

optionalTncVersions variable

Syntax

```
List< TncVersion > optionalTncVersions
```

TncVersionUpdate class

2013 SAP AG or an SAP affiliate company.

Syntax

```
public class TncVersionUpdate
```

Usage

All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

getNewVersion() method

Syntax

```
TncVersion getNewVersion ()
```

getOldVersion() method

Syntax

```
TncVersion getOldVersion ()
```

setNewVersion(TncVersion) method

Syntax

```
void setNewVersion ( TncVersion newVersion )
```

Parameters

- **newVersion** –

setOldVersion(TncVersion) method

Syntax

```
void setOldVersion ( TncVersion oldVersion )
```

Parameters

- **oldVersion** –

newVersion variable

Syntax

```
TncVersion newVersion
```

oldVersion variable

Syntax

```
TncVersion oldVersion
```

dao package

Members

All public members of the dao package.

- **api package** – 2013 SAP AG or an SAP affiliate company.
- **model package** – 2013 SAP AG or an SAP affiliate company.

api package

2013 SAP AG or an SAP affiliate company.

Members

All public members of the api package.

- **MobiliserServicePackageDao interface** – Interface ServicePackageDao-Contains the methods exposed related to the Service Package calls - Including CRUD operations and calls for business logic.
- **SvcPkgCustomerDao interface** –
- **SvcPkgTncDao interface** –
- **TermAndConditionDao interface** –
- **TncVersionCustomerDao interface** –
- **TncVersionDao interface** –

Usage

All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

MobiliserServicePackageDao interface

Interface ServicePackageDao-Contains the methods exposed related to the Service Package calls - Including CRUD operations and calls for business logic.

Syntax

```
public interface MobiliserServicePackageDao
```

```
getAllServicePackages(@MobiliserParam(name="activeOnly", required=true,
description="When set to false, no filter is applied") boolean,
@MobiliserParam(name="selfregistrationOnly", required=true, description="When
set to false, no filter is applied") boolean,
@MobiliserParam(name="autoAssignOnly", required=true, description="When set
to false, no filter is applied") boolean) method
```

The implementation of this method should return All Service Packages (Active) which are available for Self registration and can be auto assigned.

Syntax

```
List< MobiliserServicePackage > getAllServicePackages
( @MobiliserParam(name="activeOnly", required=true,
description="When set to false, no filter is applied") boolean
activeOnly , @MobiliserParam(name="selfregistrationOnly",
required=true, description="When set to false, no filter is
applied") boolean selfregistrationOnly ,
@MobiliserParam(name="autoAssignOnly", required=true,
description="When set to false, no filter is applied") boolean
autoAssignOnly )
```

getAvailableServicePackages(int, String, boolean, boolean, boolean) method

The implementation of this method should return Available Service Packages (Active) for Customer Type, OrgUnit.

Syntax

```
List< MobiliserServicePackage > getAvailableServicePackages  
( int customerTypeId , String orgUnitId , boolean isActive , boolean  
autoAssignOnly , boolean selfRegistrationOnly )
```

Usage

This method filters the Auto Assigned and Self Registered Packages

getServicePackagesByCustomerId(@MobiliserParam(name="customerId", required=true, description="The id of the customer") long, @MobiliserParam(name="activeOnly", required=true, description="When set to false, no filter is applied") boolean, @MobiliserParam(name="tncAcceptedOnly", required=true, description="Only return those for which all mandatory TNCs are accepted. When set to false, no filter is applied") boolean) method

This method returns all the MobiliserServicePackages For the given customer based on the customer id.

Syntax

```
List< MobiliserServicePackage > getServicePackagesByCustomerId  
( @MobiliserParam(name="customerId", required=true,  
description="The id of the customer") long customerId ,  
@MobiliserParam(name="activeOnly", required=true,  
description="When set to false, no filter is applied") boolean  
activeOnly , @MobiliserParam(name="tncAcceptedOnly",  
required=true, description="Only return those for which all  
mandatory TNCs are accepted. When set to false, no filter is  
applied") boolean tncAcceptedOnly )
```

Parameters

- **customerId** –
- **activeOnly** –
- **tncAcceptedOnly** –

SvcPkgCustomerDao interface

Syntax

```
public interface SvcPkgCustomerDao
```

getCustomerAssignedSvcpkgs(Customer, MobiliserServicePackage) method

The method implementation will give all the active service packages that are assigned to the customer.

Syntax

```
List< SvcPkgCustomer > getCustomerAssignedSvcpkgs ( Customer
customer , MobiliserServicePackage svcpkg )
```

getServicePackagesByCustomerId(long, boolean, boolean) method

This method should return the active Service Packages for the Customer for which the Tncs have been accepted and the date is valid.

Syntax

```
List< SvcPkgCustomer > getServicePackagesByCustomerId ( long
customerId , boolean activeOnly , boolean tncAcceptedOnly )
```

getSvcPkgPrivileges(long) method

This method returns the privileges assigned to customer (Based on the assigned MobiliserServicePackages)

Syntax

```
List< String > getSvcPkgPrivileges ( long customerId )
```

Parameters

- **customerId** –

SvcPkgTncDao interface**Syntax**

```
public interface SvcPkgTncDao
```

deleteByTermAndCondition(TermAndCondition) method

This method deletes by the TermAndCondition.

Syntax

```
void deleteByTermAndCondition ( TermAndCondition tnc )
```

Parameters

- **tnc** –

getSvcPkgTncBySvcPkgAndTnc(long, long) method

Gets the SvcpkgTnc based on the service package id and tnc id.

Syntax

```
SvcPkgTnc getSvcPkgTncBySvcPkgAndTnc ( long svcpkgId , long tncId )
```

getSvcPkgTnCsBySvcPkgId(@MobiliserParam(name="svcPkgId", required=true, description="The id of the Service Package") long) method

This method returns the Terms and Conditions associated with the Service Package Id.

Syntax

```
List< SvcPkgTnc > getSvcPkgTnCsBySvcPkgId  
( @MobiliserParam(name="svcPkgId", required=true,  
description="The id of the Service Package") long svcpkgId )
```

getSvcPkgTncsByTncId(long) method

This method will return the service packages associated with the given terms and conditions.

Syntax

```
List< SvcPkgTnc > getSvcPkgTncsByTncId ( long tncId )
```

TermAndConditionDao interface

Syntax

```
public interface TermAndConditionDao
```

getAllValidTermAndConditionsByCustomerId(long) method

This method returns the list of all the valid TermAndConditions for a given customer.

Syntax

```
List< TermAndCondition >  
getAllValidTermAndConditionsByCustomerId ( long customerId )
```

Parameters

- **customerId** –

getTermAndConditions(@MobiliserParam(name="activeOnly", required=true, description="When set to false, no filter is applied") boolean) boolean method

This method returns the Tncs based on active flag's values.

Syntax

```
List< TermAndCondition > getTermAndConditions
( @MobiliserParam(name="activeOnly", required=true,
description="When set to false, no filter is applied") boolean
activeOnly )
```

Usage

isActive= true -- Only the Active Terms and conditions and isActive=false --All the terms and conditions

getTermAndConditionsByServicePackageId(@MobiliserParam(name="svcPkgId", required=true, description="The ID of the Service Package for which to retrieve the TermAndConditions") long, @MobiliserParam(name="mandatory", required=false, description="Returns the mandatory or optional TNCs or all if not set (null)") Boolean) method

This method returns the list of the applicable TermAndCondition for a given ServicePackage.

Syntax

```
List< TermAndCondition >
getTermAndConditionsByServicePackageId
( @MobiliserParam(name="svcPkgId", required=true,
description="The ID of the Service Package for which to
retrieve the TermAndConditions") long svcPkgId ,
@MobiliserParam(name="mandatory", required=false,
description="Returns the mandatory or optional TNCs or all if
not set (null)") Boolean mandatory )
```

Parameters

- **svcPkgId** –
- **mandatory** –

TncVersionCustomerDao interface

Syntax

```
public interface TncVersionCustomerDao
```

getTncVersionCustomersByCustomer(Customer) method

This method returns the list of TncVersionCustomer for the given Customer.

Syntax

```
List< TncVersionCustomer > getTncVersionCustomersByCustomer  
( Customer customer )
```

Parameters

- **customer** –

*getTncVersionCustomersByCustomerAndTncVersion(Customer, TncVersion)
method*

This method returns the list of TncVersionCustomers based on the Customer and TncVersion.

Syntax

```
List< TncVersionCustomer >  
getTncVersionCustomersByCustomerAndTncVersion ( Customer  
customer , TncVersion tncVersion )
```

Parameters

- **customer** –
- **tncVersion** –

TncVersionDao interface

Syntax

```
public interface TncVersionDao
```

*getCurrentTncVersionByTermAndConditionId(@MobiliserParam(name="tncId",
required=true, length=18, description="The primary key of the TermAndCondition")
long) method*

This method returns the Current TncVersion for the given TermAndCondition.

Syntax

```
TncVersion getCurrentTncVersionByTermAndConditionId  
( @MobiliserParam(name="tncId", required=true, length=18,  
description="The primary key of the TermAndCondition") long  
tncId )
```

Parameters

- **tncId** –

getCurrentTncVersionsByServicePackageId(long) method

This method returns the current TncVersions for given ServicePackage.

Syntax

```
List< TncVersion > getCurrentTncVersionsByServicePackageId  
( long mobiliserServicePackageId )
```

Parameters

- **mobiliserServicePackageId** –

getMaxVersionNumber(TermAndCondition) method

This method returns the maximum version number for the given tnc.

Syntax

```
int getMaxVersionNumber ( TermAndCondition tncID )
```

getTncVersionByTnCAndVersion(TermAndCondition, int) method

This method returns the TncVersion by TermAndCondition and Version number.

Syntax

```
TncVersion getTncVersionByTnCAndVersion ( TermAndCondition tnc ,  
int version )
```

Parameters

- **tnc** –
- **version** –

getTncVersionsByCustomerId(long) method

This method returns the accepted TncVersions for the given customer.

Syntax

```
List< TncVersion > getTncVersionsByCustomerId ( long customerId )
```

Parameters

- **customerId** –

getTncVersionsByTncId(@MobiliserParam(name="tncId", required=true, length=18, description="The primary key of the TermAndCondition") long) method
This method returns the TncVersions for the given TermAndCondition id.

Syntax

```
List< TncVersion > getTncVersionsByTncId  
( @MobiliserParam(name="tncId", required=true, length=18,  
description="The primary key of the TermAndCondition") long  
tncId )
```

Parameters

- **tncId** –

getValidTncVersions(TermAndCondition) method

This method returns all the valid TncVersions for the given TermAndCondition.

Syntax

```
List< TncVersion > getValidTncVersions ( TermAndCondition tnc )
```

Parameters

- **tnc** –

model package

2013 SAP AG or an SAP affiliate company.

Members

All public members of the model package.

- **MobiliserServicePackage class** –
- **SvcPkgCustomer class** –
- **SvcPkgTnc class** –
- **TermAndCondition class** –
- **TncVersion class** –
- **TncVersionCustomer class** –

Usage

All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

MobiliserServicePackage class

Syntax

```
public class MobiliserServicePackage
```

getCustomerType() method

Syntax

```
CustomerType getCustomerType ()
```

getMaxAssignmentPerCust() method

Syntax

```
int getMaxAssignmentPerCust ()
```

getOrgUnit() method

Syntax

```
OrgUnit getOrgUnit ()
```

getServicePackageDescription() method

Syntax

```
String getServicePackageDescription ()
```

getServiceProvider() method

Syntax

```
Customer getServiceProvider ()
```

getUmgrRole() method

Syntax

```
UmgrRole getUmgrRole ()
```

isActive() method

Syntax

```
boolean isActive ()
```

isAutoAssign() method

Syntax

boolean isAutoAssign ()

isSelfRegistration() method

Syntax

boolean isSelfRegistration ()

isSetActive() method

Syntax

boolean isSetActive ()

isSetAutoAssign() method

Syntax

boolean isSetAutoAssign ()

isSetCustomerType() method

Syntax

boolean isSetCustomerType ()

isSetMaxAssignmentPerCust() method

Syntax

boolean isSetMaxAssignmentPerCust ()

isSetOrgUnit() method

Syntax

boolean isSetOrgUnit ()

isSetSelfRegistration() method

Syntax

boolean isSetSelfRegistration ()

isSetServicePackageDescription() method

Syntax

`boolean isSetServicePackageDescription ()`

isSetServiceProvider() method

Syntax

`boolean isSetServiceProvider ()`

isSetUmgrRole() method

Syntax

`boolean isSetUmgrRole ()`

setActive(boolean) method

Syntax

`void setActive ( boolean active )`

Parameters

- **active** –

setAutoAssign(boolean) method

Syntax

`void setAutoAssign ( boolean autoAssign )`

Parameters

- **autoAssign** –

setCustomerType(CustomerType) method

Syntax

`void setCustomerType ( CustomerType customerType )`

Parameters

- **customerType** –

setMaxAssignmentPerCust(int) method

Syntax

void setMaxAssignmentPerCust (int *maxAssignmentPerCust*)

Parameters

- **maxAssignmentPerCust** –

setOrgUnit(OrgUnit) method

Syntax

void setOrgUnit (OrgUnit *orgUnit*)

Parameters

- **orgUnit** –

setSelfRegistration(boolean) method

Syntax

void setSelfRegistration (boolean *selfRegistration*)

Parameters

- **selfRegistration** –

setServicePackageDescription(String) method

Syntax

void setServicePackageDescription (String *servicePackageDescription*)

Parameters

- **servicePackageDescription** –

setServiceProvider(Customer) method

Syntax

void setServiceProvider (Customer *serviceProvider*)

Parameters

- **serviceProvider** –

setUmgrRole(UmgrRole) method

Syntax

```
void setUmgrRole ( UmgrRole umgrRole )
```

Parameters

- **umgrRole** –

SvcPkgCustomer class

Syntax

```
public class SvcPkgCustomer
```

SvcPkgCustomer() constructor

Constructor.

Syntax

```
SvcPkgCustomer ( )
```

getCustomer() method

Syntax

```
Customer getCustomer ( )
```

getDateEnabledUntil() method

Syntax

```
Date getDateEnabledUntil ( )
```

getDateEnableFrom() method

Syntax

```
Date getDateEnableFrom ( )
```

getDateLastUsed() method

Syntax

```
Date getDateLastUsed ( )
```

getMobiliserServicePackage() method

Syntax

MobiliserServicePackage getMobiliserServicePackage ()

isActive() method

Syntax

boolean isActive ()

isSetActive() method

Syntax

boolean isSetActive ()

isSetDateEnabledUntil() method

Syntax

boolean isSetDateEnabledUntil ()

isSetDateEnableFrom() method

Syntax

boolean isSetDateEnableFrom ()

isSetDateLastUsed() method

Syntax

boolean isSetDateLastUsed ()

isSetTncAccepted() method

Syntax

boolean isSetTncAccepted ()

isTncAccepted() method

Syntax

boolean isTncAccepted ()

setActive(boolean) method

Syntax

void setActive (boolean *active*)

Parameters

- **active** –

setCustomer(Customer) method

Syntax

void setCustomer (Customer *customer*)

Parameters

- **customer** –

setEnabledUntil(Date) method

Syntax

void setEnabledUntil (Date *dateEnabledUntil*)

Parameters

- **dateEnabledUntil** –

setEnabledFrom(Date) method

Syntax

void setEnabledFrom (Date *dateEnableFrom*)

Parameters

- **dateEnableFrom** –

setLastUsed(Date) method

Syntax

void setLastUsed (Date *dateLastUsed*)

Parameters

- **dateLastUsed** –

setMobiliserServicePackage(MobiliserServicePackage) method

Syntax

```
void setMobiliserServicePackage ( MobiliserServicePackage  
mobiliserServicePackage )
```

Parameters

- **mobiliserServicePackage** –

setTncAccepted(boolean) method

Syntax

```
void setTncAccepted ( boolean tncAccepted )
```

Parameters

- **tncAccepted** –

SvcPkgTnc class

Syntax

```
public class SvcPkgTnc
```

getMobiliserServicePackage() method

Syntax

```
MobiliserServicePackage getMobiliserServicePackage ()
```

getTermAndCondition() method

Syntax

```
TermAndCondition getTermAndCondition ()
```

isMandatory() method

Syntax

```
boolean isMandatory ()
```

isSetMandatory() method

Syntax

```
boolean isSetMandatory ()
```

setMandatory(boolean) method

Syntax

```
void setMandatory ( boolean mandatory )
```

Parameters

- **mandatory** –

setMobiliserServicePackage(MobiliserServicePackage) method

Syntax

```
void setMobiliserServicePackage ( MobiliserServicePackage  
mobiliserServicePackage )
```

Parameters

- **mobiliserServicePackage** –

setTermAndCondition(TermAndCondition) method

Syntax

```
void setTermAndCondition ( TermAndCondition termAndCondition )
```

Parameters

- **termAndCondition** –

TermAndCondition class

Syntax

```
public class TermAndCondition
```

getDescription() method

Syntax

```
String getDescription ()
```

isActive() method

Syntax

boolean isActive ()

isSetActive() method

Syntax

boolean isSetActive ()

isSetDescription() method

Syntax

boolean isSetDescription ()

setActive(boolean) method

Syntax

void setActive (boolean *active*)

Parameters

- **active** –

setDescription(String) method

Syntax

void setDescription (String *description*)

Parameters

- **description** –

TncVersion class

Syntax

public class TncVersion

getCaptionText() method

Syntax

String getCaptionText ()

getFromDate() method

Syntax

Date getFromDate ()

getTermAndCondition() method

Syntax

TermAndCondition getTermAndCondition ()

getTncText() method

Syntax

String getTncText ()

getTncUrl() method

Syntax

String getTncUrl ()

getToDate() method

Syntax

Date getToDate ()

getVersion() method

Syntax

int getVersion ()

isSetCaptionText() method

Syntax

boolean isSetCaptionText ()

isSetFromDate() method

Syntax

boolean isSetFromDate ()

isSetTermAndCondition() method

Syntax

boolean isSetTermAndCondition ()

isSetTncText() method

Syntax

boolean isSetTncText ()

isSetTncUrl() method

Syntax

boolean isSetTncUrl ()

isSetToDate() method

Syntax

boolean isSetToDate ()

isSetVersion() method

Syntax

boolean isSetVersion ()

setCaptionText(String) method

Syntax

void setCaptionText (String *captionText*)

Parameters

- **captionText** –

setFromDate(Date) method

Syntax

void setFromDate (Date *fromDate*)

Parameters

- **fromDate** –

setTermAndCondition(TermAndCondition) method

Syntax

`void setTermAndCondition ( TermAndCondition termAndCondition )`

Parameters

- **termAndCondition** –

setTncText(String) method

Syntax

`void setTncText ( String tncText )`

Parameters

- **tncText** –

setTncUrl(String) method

Syntax

`void setTncUrl ( String tncUrl )`

Parameters

- **tncUrl** –

setToDate(Date) method

Syntax

`void setToDate ( Date toDate )`

Parameters

- **toDate** –

setVersion(Integer) method

Syntax

`void setVersion ( Integer version )`

Parameters

- **version** –

setVersion(int) method

Syntax

```
void setVersion ( int version )
```

Parameters

- **version** –

TncVersionCustomer class

Syntax

```
public class TncVersionCustomer
```

getApprovalDate() method

Syntax

```
Date getApprovalDate ()
```

getCustomer() method

Syntax

```
Customer getCustomer ()
```

getTncVersion() method

Syntax

```
TncVersion getTncVersion ()
```

setApprovalDate(Date) method

Syntax

```
void setApprovalDate ( Date approvalDate )
```

Parameters

- **approvalDate** –

setCustomer(Customer) method

Syntax

`void setCustomer ( Customer customer )`

Parameters

- **customer** –

setTncVersion(TncVersion) method

Syntax

`void setTncVersion ( TncVersion tncVersion )`

Parameters

- **tncVersion** –

com.sybase365.mobiliser.money package