

**SE のための
企業側 ISO20022 メッセージ
導入ガイド**

第 0.2 版

目次

第 1 章 はじめに.....	5
(1) 目的	5
(2) 対象読者	5
(3) 本書の内容	5
(4) 免責事項	6
第 2 章 金流商流連携	7
(1) 金流商流連携の処理の流れ.....	7
(2) 振込処理(振込明細挿入)	7
(3) 入金確認処理(振込明細抽出).....	8
① ISO20022 Pain、Camt とは.....	9
② 金融 EDI 情報格納領域とは.....	9
③ MIME ヘッダーとは.....	11
④ Base64 エンコードとは.....	11
第 3 章 金流情報作成方法	14
(1) 総合振込(ISO20022 Pain.001)の作り方	14
(2) 総合振込結果明細(ISO20022 Pain.002)の読み方	41
(3) 入出金明細(ISO20022 Camt.052)の読み方	65
(4) 振込入金通知(ISO20022 Camt.054)の読み方	98
(5) 総合振込依頼制御情報(ISO20022 Bah.001)の作り方	119
(6) 入出金取引明細制御情報(ISO20022 Bah.001)の作り方	130
(7) 振込入金通知依頼制御情報(ISO20022 Bah.001)の作り方	130
(8) 総合振込結果制御情報(ISO20022 Bah.001)の読み方	131
(9) 入出金取引明細結果制御情報(ISO20022 Bah.001)の読み方	135
(10) 振込入金通知結果制御情報(ISO20022 Bah.001)の読み方	135
第 4 章 金融EDI情報作成方法	136
(1) 商流情報項目	136
(2) 国連 CEFACT Remittance Advice	138
(3) 国連 CEFACT 支払案内メッセージの作り方	139
(4) 国連 CEFACT 支払案内メッセージの読み方	168
(5) 国連 CEFACT 基礎データ記述方法	192
① 金額データ(Amount)型記述方法	192
② バイナリーオブジェクト型記述方法	192
③ コード(Code)型記述方法	193
④ 日付・時刻データ型記述方法	194
⑤ ID データ(Identifier)型記述方法	197
⑥ 真偽値(Indicator)型記述方法	198
⑦ 単位付き数値(Measure)型記述方法	198
⑧ 数値(Numeric)型記述方法	199

⑨ 数量(Quantity)型記述方法	200
⑩ テキスト(Text)型記述方法	200
第 5 章 参考文献	202
第 6 章 付録	203
(1) XML スキーマ	203
① Pain.001 の XML スキーマ(チェック用)	203
② Pain.001 の XML スキーマ(JAXB 用)	203
③ Pain.002 の XML スキーマ(チェック用)	203
④ Pain.002 の XML スキーマ(JAXB 用)	203
⑤ Bah.001 の XML スキーマ(チェック用)	204
⑥ Bah.001 の XML スキーマ(JAXB 用)	204
⑦ Camt.052 の XML スキーマ(チェック用)	204
⑧ Camt.052 の XML スキーマ(JAXB 用)	204
⑨ Camt.054 の XML スキーマ(チェック用)	204
⑩ Camt.054 の XML スキーマ(出力チェック用)	205
⑪ Remittance Advice の XML スキーマ(チェック用)	205
⑫ Remittance Advice の XML スキーマ(JAXB 用)	205
(2) コード一覧	206
① 業界区分	206
② データ区分	207
③ 金額相殺項目理由コード	207
④ 税タイプコード(税区分)	208
⑤ 書類種別コード	209
(3) 取引ごとに識別に使われるコード	210
① コード表が登録されている場合	210
② コード表が登録されていない場合	210
(4) メッセージ例	212
① Pain.001 の例	212
② Pain.002 の例	212
③ Camt.052 の例	212
④ Camt.054 の例	212
⑤ Bah.001 の例	212
⑥ Remittance Advice の例	212
(5) ソフトウェア例	213
① Pain.001 の例	213
② Pain.002 の例	213
③ Camt.052 の例	213
④ Camt.054 の例	213
⑤ Bah.001 の例	213
⑥ Remittance Advice の例	214

(6)	JAXB の使用例.....	215
-----	----------------	-----

第1章 はじめに

(1) 目的

ISO 20022 は、XML を主要なデータ記述言語とした金融通信メッセージの国際規格である。ISO 20022 は、標準化された金融メッセージの作成を一義的な目的とするが、金融業務分野で利用されている様々な通信メッセージに対しインターオペラビリティ(相互運用性)を実現することが容易になる可能性がある。

全銀EDIシステムはISO20022のXMLメッセージを採用することで、大きな商流情報などの金融EDI情報を添付可能とする拡張を行っている。

しかし、これまで日本国内の金融通信メッセージとしては固定長電文が使われていたため、ISO20022のXMLメッセージを扱えるシステムはほとんど無い。このため、全銀EDIシステムを利用するためには、ISO20022のXMLメッセージを扱う機能を追加することが急務であるが、金融系のシステム・アプリケーションの開発者の多くはISO20022の様な大きなXMLメッセージの処理は不慣れでもある。

本ガイドは、ISO20022のXMLメッセージの作成や読み取りの具体的な方法を解説し、システム開発者の一助とすることを目的とする。

本ガイドは、ISO20022のXMLメッセージの作成や読み取りの具体的な方法を解説し、システム開発者の一助となる情報提供を目的とする。

(2) 対象読者

全銀EDIシステムなどの金融サービスに接続するシステム・ソフトウェアにおけるISO20022のXMLメッセージを作成する部分を開発するが、以下の様な技術者に本ガイドを読むことを推奨する。

- ISO20022 Pain、Camt メッセージを熟知していない技術者
- 商流情報向けの国連 CEFACT のフォーマット(XML)を熟知していない技術者
- XML 技術を熟知していない技術者

本ガイドでは、読者諸氏が Java プログラミング言語を既に理解していることを想定している。

(3) 本書の内容

第2章では金流商流連携の概要について、第3章では金流情報作成方法としてISO20022のメッセージの作り方について、第4章では金融EDI情報作成方法としてISO20022のメッセージに格納される商流情報の作り方について記述する。なお、金融EDI情報作成方法については、原則、利用者や各業界等が任意で決定し利用可能なものとなっている。

6

そこで本ガイドでは、一つ方式案（国連 CEFACT Remittance Advice メッセージ）を参考とし、商流情報を金融 EDI 情報にマッピングする一方式を事例として解説する。

また関連する資料を参考文献一覧にまとめ、付録として XML スキーマ、コード一覧、メッセージ例を添付している。

(4) 免責事項

本ガイドは主として、全銀 EDI システムなどの金融サービスに接続するシステム・ソフトウェアの開発技術者向けに情報提供目的で作成したものである。発行者および発行関係者は、本書の使用によって、確実な成果が生み出されることを何ら主張するものではありません。本書の作成には万全を期しているが、万が一誤りや不正確な情報があっても、発行者および発行関係者が一切の責任を負わないことをご了承願います。

第2章 金流商流連携

(1) 金流商流連携の処理の流れ

支払情報および入金情報に付加情報（拡張金融 EDI）を付与し交換する金流商流連携処理により、効率的な消込処理などが可能になる。

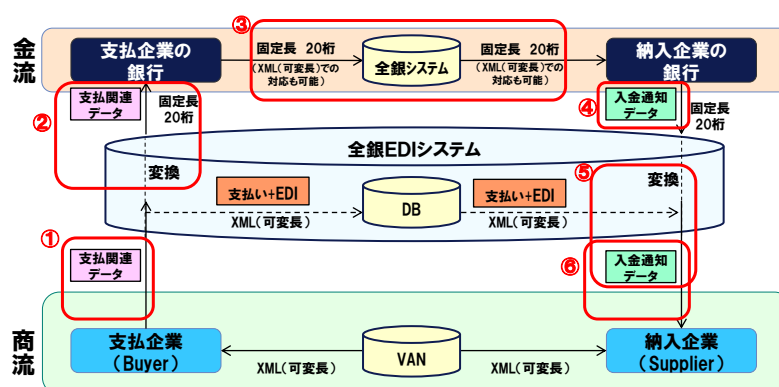


図 1 金流商流連携の流れ

具体的には、①支払企業が支払関連データを XML 形式で送る。②プラットフォーム内で支払関連データを変換して支払企業の銀行へ送る。③支払い企業の銀行は現状通り、固定長・20桁でデータのやり取りを行う。④納入企業の銀行が入金通知データを固定長・20桁で送る。⑤プラットフォーム内で入金通知データを変換して納入企業へ送る。⑥入金通知データを受信し、入金確認、消込などの処理を行う。

(2) 振込処理(振込明細挿入)

振込は送金方法の一種で、金融機関に開設された預貯金口座に宛てて、金銭を払い込むことである。多くの場合、企業自らが金融機関に解説した預貯金口座から、請求や納品した企業の預貯金口座に金銭を払い込む方法として用いられる。

振り込まれた金銭は入金として振り込まれた企業は知りうるが、どの請求や納品に対する振込かを特定する消込を行っている。この消込の作業効率向上に効果的な方法として、振込の基となる請求や納品かを特定できる情報を振込明細として付加することが考えられた。

金融 EDI システムを使った振込では、Pain.001.001.03 という XML メッセージを使って行えるため、大きな商流情報でも金融 EDI 情報として付加することが可能となり、数多くの請求を一度に払うまとめ払いでも振込明細を付加することが出来るように成る。

振込に必要である口座情報等共通な情報は、多くの企業は基本契約として情報交換し、マスターデータに格納している。このため必要に応じてマスターデータに格納されている口座情報等も利用し、振込明細とともに総合振込情報（Pain.001.001.03）を作成し、金融機関に振込を依頼する。

金融 EDI システムでは、振込の依頼に対して総合振込結果明細（Pain.002.001.03）が作成される。取引明細別処理結果により振込処理の正常またはエラーが判別でき、個別のエラー内容は識別表示および仕向け金融機関指示情報に記載される。

(3) 入金確認処理（振込明細抽出）

入金が行われると振込入金通知（Camt.054.001.02）を金融機関が作成する。また入金や出金の状況が入出金明細（Camt.052.001.02）として金融機関が作成する。

通知された振込入金通知や入出金明細の振込明細にある金融 EDI 情報から商流情報を抽出し、送付された請求書・請求明細や振込明細と突合し、消込を行う。

① ISO20022 Pain、Camt とは

Pain は国際規格 ISO20022 として標準化された送金指図（総合振込）を行うための XML 電文仕様であり、Camt も振込入金通知や入出金取引明細といった銀行から顧客企業に預金口座情報を通知するための国際標準 XML 電文仕様である。

Pain の項目と総合振込の項目との相互の対応付け（マッピング）を行い、Pain と総合振込の相互変換を可能とする。これによりこれまで使っていた総合振込の項目を使って Pain の電文の作成や、Pain の電文のどの項目を使って振込を行うかが明確になる。

pain.001.001.03、総合振込フォーマットマッピング									
ISO20022 (pain.001.001.03)				金融フォーマット振込項目 (総合振込フォーマット)				必須(M)	全銀協標準 (2014/7/4) (2014/7/15) (2014/7/16)
ISO Index No.	Gr	Message Item	Mult.	項目名	必須(M) 任意(O)	入力方法(金融フォーマット)	入力方法(Camt)	注: 金融フォーマットに該当項目がない場合も入力方法に記載	
		Document	[1..1]	(タグを生成)	-	-	メッセージのDocumentと指定	M	
		CustomerCreditTransferInitiation	[1..1]	(タグを生成)	-	-	※ <Document> xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:sd="http://www.w3.org/2001/XMLSchema-instance" xmlns="urn:iso:std:iso:20022:tech:xsd:pain.001.001.03"	M	
1.8		GroupHeader	[1..1]	(タグを生成)	-	-		M	
1.1		MessageIdentification	[1..1]	(タグを生成)	-	-	タグを生成し、顧客システムで管理されたメッセージを特定するためのIDを指定 ※テストデータの場合は先頭4文字に「TEST」となるIDが指定される。	M	テストデータの場合、先頭4文字が「TEST」となるIDを指定する。
1.2		CreationDateTime	[1..1]	(タグを生成)	-	-	タグを生成し、顧客システムでメッセージ作成日を指定	M	
1.3		Authorization	[0..2]						
1.4	(Or)	Code	[1..1]						
1.5	(Or)	Proprietary	[1..1]						
1.6		NumberOfTransactions	[1..1]	(タグを生成)	-	-	タグを生成し、合計件数を指定。	M	
1.7		ControlSum	[0..1]						
1.8		InitiatingParty	[1..1]	(タグを生成)	-	-	種を指定せず、空タグを設定する。 空タグ: <タグ名 />	M	
2.0		PaymentInformation	[1..n]	(タグを生成)	-	-		M	
2.1		PaymentInformationIdentification	[1..1]	(タグを生成)	-	-	タグを生成し、送金を特定するIDを指定	M	
2.2		PaymentMethod	[1..1]	(タグを生成)	-	-	タグを生成し、「TRF」を指定	M	
2.3		Batchbooking	[0..1]						
2.4		NumberOfTransactions	[0..1]						
2.5		ControlSum	[0..1]						
2.6		PaymentTypeInformation	[0..1]	(タグを生成)	-	-		M	
2.7		InstructionPriority	[0..1]						

表 1 Pain と総合振込の相互変換表例(一部)

誰もが同じ相互変換表を使うことで、Pain の使い方が同じになる。

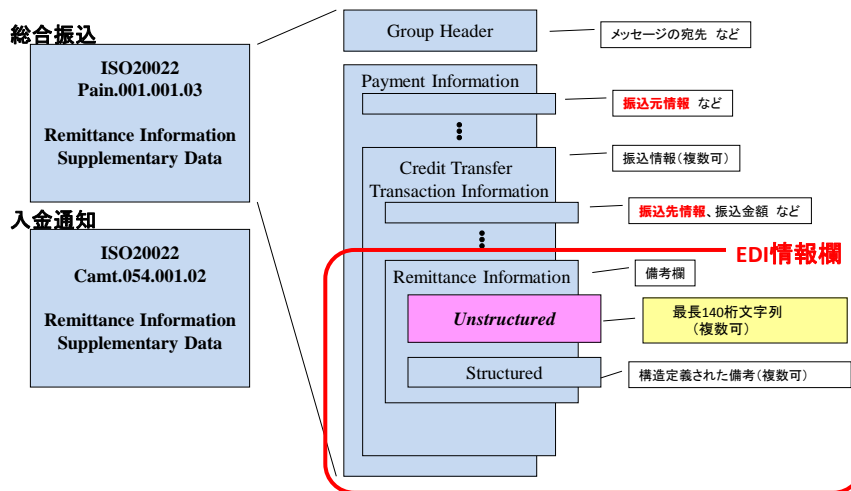
なお、上記は Pain の例であるが、Camt（振込入金通知、入出金明細に関する XML 電文）についても同様に項目の対応付け（マッピング）が必要となる。

全銀 EDI システムにおける変換表は全銀協のホームページ (<https://www.zenginkyo.or.jp/abstract/efforts/smooth/xml/>) に掲載されているので参照されたい。

② 金融 EDI 情報格納領域とは

全銀 EDI システムでは、総合振込は Pain.001.001.03、振込入金通知は Camt.054.001.02、入出金明細は Camt.052.001.02 のフォーマットを採用している。それぞれ商流情報を入れる領域として Remittance Information の Unstructured 項目が指定されている。

全銀EDIシステムにおけるEDI情報格納可能領域



1

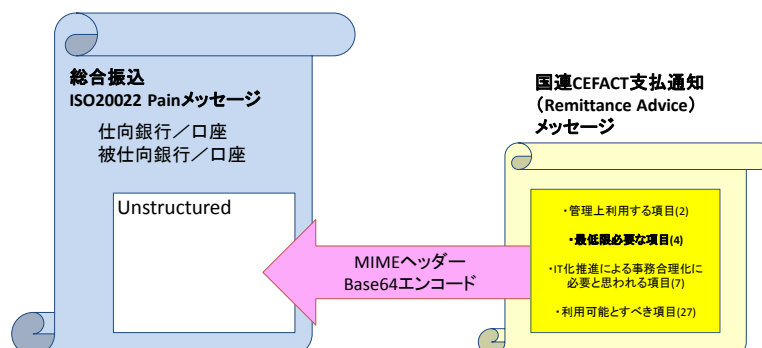
Unstructured 項目は 140 文字を記述でき、また繰り返し記述することが可能である。しかし、XML のタグを記述することは許されていない。このため XML 形式で記述されることが多い大きな商流情報は、そのまま Unstructured 項目に入れることが出来ない。

そこでバイナリデータなどを英数字と 3 種の記号 (+, /, =) だけの文字列情報としてコード化する Base64 エンコードを用いることが推奨されている。更に 140 文字以内の制限を守るため Base64 エンコードで通常用いられている 76 文字毎の改行を使い、複数の Unstructured 項目を繰り返し使う方法と、Base64 エンコードされていることが明らかとなるよう MIME ヘッダーの付加も推奨されている。

EDI情報のマッピングについて

EDI情報

国連CEFACT支払通知メッセージ(国際標準)中の40項目(必須4項目)が選定



- ・ 受取企業がデコードの要否を判断できるようにヘッダ情報を付加する。
- ・ 改行ごとに「Unstructurd」タグを設定する。

金融 EDI 情報格納領域には消込時の突合に使う商流情報を入れることになるが、格納する商流情報項目およびその情報の記述形式が明確化されないと取り出すことが出来ない。情報項目については、経済産業省が中心となり消込の突合に有益と思われる 40 項目がまず選定されている。情報の記述形式については様々な方法が検討されており、商取引の多様性から種々の記述形式が出てくると思われるが、多様過ぎるのも不便である。このため、金融 EDI 情報項目のデファクトスタンダード化も進むと推測される。

③ MIME ヘッダーとは

Multipurpose Internet Mail Extension (MIME) は、US-ASCII のテキストしか使用できないインターネットの電子メールでさまざまなフォーマット（書式）を扱えるようにする規格である。通常は MIME（マイム）と略される。

MIME は内容の記載フォーマットと共にコンテンツ内容を明らかにする補助情報を記載する MIME ヘッダーについても規格化している。様々な内容の記載フォーマットを規定しているが、ここでは XML データを記載する場合の MIME ヘッダーの具体例を以下に示す。

MIME ヘッダー

MIME-Version: 1.0

Content-Type: text/xml

Content-Transfer-Encoding: base64

MIME-Version は、MIME ヘッダーの最初に記載することが決められており、MIME の規定の版情報を表す。しかし、現在のところ版情報としては 1.0 しかない。しかし、「MIME-Version: 1.0」という行が先頭にあることで、それ以降は MIME の規定に従えば解析を行えることが期待できる。「MIME-Version: 1.0」で始められていない Unstructured 項目は平文として解析することとなる。

Content-Type は内容の記述方法を表し、本例は「XML データ」であることを示している。Content-Transfer-Encoding は内容のエンコード方法を表し、本例は「base64 エンコード」が行われていることを示している。

④ Base64 エンコードとは

Base64 は、データを 64 種類の印字可能な英数字のみを用いて、印字可能な文字しか扱うことの出来ない通信環境にてマルチバイト文字やバイナリデータを扱うためのエンコード方式である。MIME によって規定されていて、7 ビットのデータしか扱うことの出来ない電子メールにて広く利用されている。具体的には、A-Z, a-z, 0-9 までの 62 文字と、記号 2 つ (+, /)、さらにパディング（余った部分を詰める）のための記号として = が用いられる。

Base64 エンコードおよび MIME ヘッダーの例を以下に示す。

エンコード例: (1) 商流情報例

```
<CrossIndustryRemittanceAdvice
xmlns="urn:un:unece:uncefact:data:standard:CrossIndustryRemittanceAdvice:11">
  <CIReExchangedDocumentContext>
    <ID>R-H28-13</ID>
    <IssueDateTime>2017-01-23T00:00:00</IssueDateTime>
  </CIReExchangedDocumentContext>
  <CIReTradeSettlementPayment>
    <SpecifiedCIReSupplyChainTradeSettlement>
      <PayerCIReTradeParty>
        <ID>9010601021385</ID>
      </PayerCIReTradeParty>
    </SpecifiedCIReSupplyChainTradeSettlement>
  </CIReTradeSettlementPayment>
  <CIReSupplyChainTradeTransaction>
    <AssociatedCIReferencedDocument>
      <IssuerAssignedID>I-H27-12-1</IssuerAssignedID>
    </AssociatedCIReferencedDocument>
  </CIReSupplyChainTradeTransaction>
  <CIReSupplyChainTradeTransaction>
    <AssociatedCIReferencedDocument>
      <IssuerAssignedID>I-H27-12-2</IssuerAssignedID>
    </AssociatedCIReferencedDocument>
  </CIReSupplyChainTradeTransaction>
</CrossIndustryRemittanceAdvice>
```

支払通知番号
支払通知発行日
支払人企業法人コード
請求書番号 2
請求書番号 2

エンコード例: (2) base64エンコードとMIMEヘッダー付加

```
MIME-Version: 1.0
Content-Type: text/xml
Content-Transfer-Encoding: base64
```

MIMEヘッダ
(MIMEのバージョン、データのタイプ (この場合はXMLテキスト)、
エンコード方式 (この場合はbase64) が指定される)

```
77u/PENyb3NzSW5kdXN0cn1SZW1pdHRhbmN1QWR2aWN1IHhtbG5zPSJ1cm46dW46dW51Y2U6dW5j
ZWZhY3Q6ZGF0YUtpZdGFuZGFyZDpDcm9zc01uZHVzdHJ5UmVtaXROYW5jZUFkdmljZT0xMSI+PENJ
UkV4Y2hhbmd1ZERvY3VtZW50Q29udGV4dD48SUQ+Ui1IMjgtMTM8L01EPjxJc3N1ZURhdGVUaW11
PjIwMTctMDEtMjNUMDA6MDA6MDA8L01zc3V1RGFOZVRpbWU+PC9DSVJFeGNoYW5nZWREb2N1bWVu
dENvbnR1eHQ+PENJUKhUcmFkZVN1dHRsZW11bnRQYX1tZW50PjxTcGVjaWZpZWRSZSVJ1U3VwcGx5
Q2hhaW5UcmFkZVN1dHRsZW11bnQ+PFBheWVYQ01UcmFkZVBhcnR5PjxJRD45MDEwNjAxMDIxMzg1
PC9JRD48L1BheWVYQ01UcmFkZVBhcnR5PjxvU3B1Y21maWVkbG91SSFN1cHBseUNoYW1uVHJhZGVt
ZXR0bGVtZW50PjxvU3B1Y21maWVkbG91SSFN1cHBseUNoYW1uVHJhZGVtZXR0bGVtZW50PjxJ
c3N1ZXRJc3NpZ251ZE1EPkktSDI3LEyLTI8L01zc3V1ckFzc21nbmVksUQ+PC9Bc3NvY21hdGVk
Q01SZWZ1cmVudVtVckRG9jdW11bnQ+PC9DSVJ1U3VwcGx5Q2hhaW5UcmFkZVRyYW5zYWN0aW9uPjxv
Q3Jvc3NjbmR1c3RyeVJlbW10dGFuY2VBZHZpY2U+
```

base64により
エンコードされた文字列

エンコードされた文字列は76文字毎に改行される

エンコード例: (3)タグ(Unstrd)の付加

<Ustrd>MIME-Version: 1.0</Ustrd>
<Ustrd>Content-Type: text/xml</Ustrd>
<Ustrd>Content-Transfer-Encoding: base64</Ustrd>

MIMEヘッダをEDI情報欄に設定することにより、
受取企業はbase64でエンコードされたことを把握可能

<Ustrd>77u/PENyb3NzSW5kdXN0cn1SZW1pdHRhbmN1QWR2aWN1IHhtbG5zPSJ1cm46dW46dW51Y2U6dW5j</Ustrd>
<Ustrd>ZWZhY3Q6ZGF0YTpzdGFuZGFyZDpDcm9zc01uZHVzdHJ5UmVtaXR0YW5jZUFkdmljZToxMSI+PENJ</Ustrd>
<Ustrd>UkV4Y2hhbmd1ZERvY3VtZW50Q29udGV4dD48SUQ+Ui1IMjgtMTM8L01EPjxJc3N1ZURhdGVUaW11</Ustrd>
<Ustrd>PjIwMmEtMjE0MDA6MDA6MDA4L01zc3V1RGF0ZVRpbWU+PC9DSVJFeGNoYW5nZWREb2N1bWVu</Ustrd>
<Ustrd>dENvbnR1eHQ+PENJUkhUcmFkZVN1dHRsZW11bnRQYX1tZW50PjxTcGVjaWZpZWRSVJlU3VwcGx5</Ustrd>
<Ustrd>Q2hhaW5UcmFkZVN1dHRsZW11bnQ+PFBheWVyQ01UcmFkZVBhcnR5PjxJRd45MDEwNjAxMDIxMzg1</Ustrd>
<Ustrd>PC9JRd48L1BheWVyQ01UcmFkZVBhcnR5PjwvU3B1Y2lmaWVhQ01SSFN1cHBseUNoYW1uVHJhZGVt</Ustrd>
<Ustrd>ZXRoOGVtZW50PjwvQ01SSFRyYWR1U2V0dGx1bWVudFBheW11bnQ+PENJU1RTdXBwbH1DaGFpb1Ry</Ustrd>
<Ustrd>YWR1VHJhbnNhY3Rpb24+PEFzc29jaWFOZWRSVJlZmVyZW5jZWREb2N1bWVuZD48SXNzdWVyQXNz</Ustrd>
<Ustrd>aWduZWRJRd5JlUgyNy0xMi0xPC9Jc3N1ZXJBe3NpZ251ZE1EPjwvQXNzb2NpYXR1ZENJUmlmZXJ1</Ustrd>
<Ustrd>bmN1ZERvY3VtZW50PjwvQ01SVFN1cHBseUNoYW1uVHJhZGVUcmFuc2FjdG1vb2NpYXR1ZENJUmlmZXJ1bmN1ZERvY3VtZW50PjxJ</Ustrd>
<Ustrd>eUNoYW1uVHJhZGVUcmFuc2FjdG1vb2NpYXR1ZENJUmlmZXJ1bmN1ZERvY3VtZW50PjxJ</Ustrd>
<Ustrd>c3N1ZXJBe3NpZ251ZE1EPkktSDI3LTlEYlTI8L01zc3V1ckFzc2lmbWVhZSUQ+PC9Bc3NvY2lhdGVk</Ustrd>
<Ustrd>Q01SZWZ1cmVhZGVkRG9jdW11bnQ+PC9DSVJU3VwcGx5Q2hhaW5UcmFkZVRyYW5zYWN0aW9uPjwv</Ustrd>
<Ustrd>Q3Jvc3Njbmc1c3RyeVJlbnW10dGFuY2VBZHZpY2U+</Ustrd>

エンコードされた文字毎の各行を、Unstructuredの定義に用いるタグ「Ustrd」で定義する。

第3章 金流情報作成方法

XML 電文の作成や読み込には、DOM、SAX、StAX、JAXB、Caster など様々な方法がある。本ガイドでは JAVA プログラミング言語環境で共通に利用可能な JAXB(スキーマからデータにバインディングする技術)を用いて JAVA ライブラリを生成し、生成された JAVA ライブラリを活用し効率的にソフトウェアを作る方法を説明する。JAXB による JAVA ライブラリの作成方法については、「JAXB の使用例」を参照されたい。

JAXB を使い生成した JAVA ライブラリを使うとソフトウェアが冗長と見える場合もあるが、データバインディング技術を使ったソフトウェアでは、タグ名のタイプミスや終了タグの作成漏れなどの不注意なミスが生じない。作成出来るタグはデータ型的にコンパイル時に検出されるためすり抜け防止が図れ、品質の高いソフトを効率的に作成する利点がある。

ここでは、JAVA の開発環境 JDK が Windows7 上にインストールされていることを想定している。また例に記載している変数名は特に意味は無い。また日本語等の文字コードは JAVA の開発環境および OS の標準文字コードに依存するため、使用する動作環境に合わせることに注意されたい。

なお本ガイドに記載しているソフトウェアは開発環境 JDK1.8.0_171-b11 で検証しているが、作り方等の例示である。実際のシステムで使うメッセージを作る際の参考でしかないことに注意されたい。

(1) 総合振込(ISO20022 Pain.001)の作り方

ソフトウェアの作り方は、まず XML スキーマから JAVA の関数を生成する xjc コマンドで Pain.001.001.03 の XML スキーマから関数を生成する。xjc の使い方の詳細は、付録 (7) JAXB の使用例を参照されたい。

金融 EDI 情報が XML 形式の場合の基本の処理の流れは以下となる。

- ① 生成された関数を使い、総合振込情報を JAVA オブジェクトに変換する。
- ② JDK の Base64 関数を使い、金融 EDI 情報を Base64 エンコードする。
- ③ Unstructured 項目に該当する JAVA オブジェクトに格納する。
- ④ JAXB の marshaller を使い、XML に変換する。

金融 EDI 情報が平文形式の場合の基本の処理の流れは以下となる。

- ① 生成された関数を使い、総合振込情報を JAVA オブジェクトに変換する。
- ② 金融 EDI 情報を Unstructured 項目に該当する JAVA オブジェクトに格納する。
- ③ JAXB の marshaller を使い、XML に変換する。

なお、金融 EDI 情報の作り方は、第 4 章を参照されたい。

次に JAVA のプログラムを記述するが、まず準備として総合振込情報を JAVA オブジェクトに変換するための xjc コマンドで生成した関数、更に生成した XML の検証、金融 EDI 情報の Base64 エンコード、日時データや金額データを作成するためのランタイムルーチン等をインポートする。

```
import javax.xml.bind.JAXBContext;
import javax.xml.bind.Marshaller;

import iso.std.iso._20022.tech.xsd.pain_001_001.*;

import java.io.File;
import java.io.StringWriter;
import java.io.PrintStream;

import javax.xml.bind.Unmarshaller;
import java.io.StringReader;

import javax.xml.XMLConstants;
import javax.xml.validation.Schema;
import javax.xml.validation.SchemaFactory;
import javax.xml.validation.Validator;

import javax.xml.transform.stream.StreamSource;

import java.nio.charset.Charset;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.stream.Collectors;

import java.io.BufferedReader;
import java.util.Base64;

import javax.xml.datatype.XMLGregorianCalendar;
import javax.xml.datatype.DatatypeFactory;
import javax.xml.datatype.DatatypeConstants;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.GregorianCalendar;

import java.math.BigDecimal;

public class Pain001w {
    public static void main(String[] args) {
```

最初に以下の Document で始まる以下の XML に当たる JAVA オブジェクト群を生成する。

```
<Document xmlns="urn:iso:std:iso:20022:tech:xsd:pain.001.001.03">
  <CstmrCdtTrfInitn>
    <GrpHdr>
      <MsgId>msgid</MsgId>
      ...
    </GrpHdr>
    ...
  </CstmrCdtTrfInitn>
</Document>
```

まず、<Document></Document> と <CstmrCdtTrfInitn></CstmrCdtTrfInitn> に当たる JAVA オブジェクト(Document オブジェクト、CstmrCdtTrfInitn オブジェクト)を生成し、Document オブジェクトの中に CstmrCdtTrfInitn オブジェクトを挿入する。

GrpHdr オブジェクトを生成し、CstmrCdtTrfInitn オブジェクトの中に挿入する。更に GrpHdr オブジェクトの MsgId フィールドに、企業が採番する XML メッセージ単位の識別番号、例えば文字列”msgid”を挿入する。

```
// XMLドキュメントルート
Document doc = new Document();

// 総合振込依頼ルート
CustomerCreditTransferInitiationV03 ccti
    = new CustomerCreditTransferInitiationV03();
doc.setCstmrCdtTrfInitn(ccti);

// グループヘッダー情報
GroupHeader32 gh1 = new GroupHeader32();
ccti.setGrpHdr(gh1);

// グループメッセージ ID
gh1.setMsgId("msgid");
```


MsgId の次に、CreDtTm で始まる以下の XML に当たる JAVA オブジェクト群を生成する。

```
<Document>
  <CstmrCdtTrfInitn>
    <GrpHdr>
      ...
      <CreDtTm>2018-03-02T20:11:10.000+09:00</CreDtTm>
      <NbOfTxS>1</NbOfTxS>
      <InitgPty/>
    </GrpHdr>
    ...
  </CstmrCdtTrfInitn>
</Document>
```

CreDtTm オブジェクトは、XML の DateTime 型を表現する xmlGregCal 型のオブジェクトとして以下の様に生成し、GrpHdr オブジェクトの CreDtTm フィールドに挿入する。

次に NbOfTxS フィールドに支払情報数の 1 を挿入する。更に空の InitgPty オブジェクトを生成し、GrpHdr オブジェクトの InitgPty フィールドに挿入する。

```
// XML ファイル作成日付
try {
    XMLGregorianCalendar xmlGregCal1
        = DatatypeFactory.newInstance()
            .newXMLGregorianCalendar(new GregorianCalendar());
    gh1.setCreDtTm(xmlGregCal1);

    xmlGregCal1.setYear(2018);
    xmlGregCal1.setMonth(3);
    xmlGregCal1.setDay(2);
    xmlGregCal1.setTimezone(9);
    xmlGregCal1.setTime(20, 11, 10);

    // System.out.println(xmlGregCal);
} catch (Exception e) {
    e.printStackTrace();
}

// 支払情報数
gh1.setNbOfTxS("1");

// 開始集団
PartyIdentification32 pi1 = new PartyIdentification32();
gh1.setInitgPty(pi1);
```

GrpHdr の次に、PmtInf で始まる以下の XML に当たる JAVA オブジェクト群を生成する。

```
<Document xmlns="urn:iso:std:iso:20022:tech:xsd:pain.001.001.03">
  <CstmrCdtTrfInitn>
    ...
    <PmtInf>
      <PmtInfId>pmtinfid1</PmtInfId>
      <PmtMtd>TRF</PmtMtd>
      <NbOfTxS>1</NbOfTxS>
      <CtrlSum>12345</CtrlSum>
      <PmtTpInf>
        <CtgyPurp>
          <Cd>OTHR</Cd>
        </CtgyPurp>
      </PmtTpInf>
      <ReqdExctnDt>2018-03-02+09:00</ReqdExctnDt>
      ...
    </PmtInf>
  </CstmrCdtTrfInitn>
</Document>
```

次の PmtInf オブジェクトは複数作成が可能な為、List オブジェクトを getPmtInf メソッドで取り出し、そこに生成した PmtInf オブジェクトを追加する。

次に PmtInf オブジェクトの PmtInfId フィールドに文字列を挿入する。”pmtinfid1”は例で特に意味は無く、仕分明細を特定するために企業が採番するユニークな識別番号を設定されたい。更に PmtInf オブジェクトの PmtMtd フィールドには、列挙型 PaymentMethod3Code の定数 TRF を挿入する。また PmtInf オブジェクトの NbOfTxS フィールドに合計件数の 1 を挿入する。

```
// 支払情報
PaymentInstructionInformation3 pii = new PaymentInstructionInformation3();
ccti.getPmtInf().add(pii);

// 支払情報 ID
pii.setPmtInfId("pmtinfid1");
pii.setPmtMtd(PaymentMethod3Code.TRF);

// 合計件数
pii.setNbOfTxS("1");
```

次に CtrlSum フィールドに BigDecimal 形式で合計金額 12345 を挿入する。更に支払情報種別の種別コード OTHR を挿入する。

```
// 合計金額
pii.setCtrlSum(new BigDecimal("12345"));

// 支払種別情報
PaymentTypeInformation19 pti = new PaymentTypeInformation19();
pii.setPmtTpInf(pti);
CategoryPurpose1Choice cp1 = new CategoryPurpose1Choice();
pti.setCtgyPurp(cp1);
cp1.setCd("OTHR");
```

取組日を表す ReqdExctnDt オブジェクトは、XML の DateTime 型を表現する xmlGregCal 型のオブジェクトを挿入する。xmlGregCal 型オブジェクトの生成方法には種々あるが、ここでは文字列で記述した日時を変換し生成している。生成した xmlGregCal 型オブジェクトは GrpHdr オブジェクトの ReqdExctnDt フィールドに挿入する。

```
// 取組日
try {
    DateFormat format1 = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
    Date date1 = format1.parse("2018-03-02 20:11:10");
    GregorianCalendar cal1 = new GregorianCalendar();
    cal1.setTime(date1);
    XMLGregorianCalendar xmlGregCal2
        = DatatypeFactory.newInstance()
            .newXMLGregorianCalendar(cal1);
    pii.setReqdExctnDt(xmlGregCal2);
    // System.out.println(xmlGregCal);
} catch (Exception e) {
    e.printStackTrace();
}
```

ReqdExctnDt の次に、Dbtr で始まる以下の XML に当たる JAVA オブジェクト群を生成する。

```
<Document>
  <CstmrCdtTrfInitn>
    ...
    <PmtInf>
      ...
      <Dbtr>
        <Id>
          <OrgId>
            <Othr>
              <Id>1234567890</Id>
              <SchmeNm>
                <Cd>BANK</Cd>
              </SchmeNm>
            </Othr>
            <Othr>
              <Id>1234567890123</Id>
              <SchmeNm>
                <Cd>TXID</Cd>
              </SchmeNm>
            </Othr>
          </OrgId>
        </Id>
      </Dbtr>
      ...
    </PmtInf>
  </CstmrCdtTrfInitn>
</Document>
```

PmtInf オブジェクトの Dbtr オブジェクトに依頼人の振込依頼人コード、例えば 1234567890 を一つの Othr オブジェクトの Id フィールドに挿入する。また、銀行がそのコードを払い出していることを示す値 BANK を振込依頼人概要コードに挿入する。二つ目の Othr オブジェクトの Id フィールドに振込依頼人法人番号(法人マイナンバー)、例えば 1234567890123 を挿入する。法人番号であることを示す TXID を振込依頼人法人番号概要コードに挿入する。

```
// 振込依頼人情報
PartyIdentification32 pi2 = new PartyIdentification32();
pii.setDbtr(pi2);
Party6Choice p1 = new Party6Choice();
pi2.setId(p1);
OrganisationIdentification4 oid1 = new OrganisationIdentification4();
p1.setOrgId(oid1);

// 振込依頼人コード
GenericOrganisationIdentification1 goid1
    = new GenericOrganisationIdentification1();
oid1.getOthr().add(goid1);
goid1.setId("1234567890");
OrganisationIdentificationSchemeName1Choice oidshnm1
    = new OrganisationIdentificationSchemeName1Choice();
goid1.setSchmeNm(oidshnm1);
oidshnm1.setCd("BANK");

// 振込依頼人法人番号(法人マイナンバー)情報
GenericOrganisationIdentification1 goid2
    = new GenericOrganisationIdentification1();
oid1.getOthr().add(goid2);
goid2.setId("1234567890123");
OrganisationIdentificationSchemeName1Choice oidshnm2
    = new OrganisationIdentificationSchemeName1Choice();
goid2.setSchmeNm(oidshnm2);
oidshnm2.setCd("TXID");
```

Dbtr の次に、DbtrAcct で始まる以下の XML に当たる JAVA オブジェクト群を生成する。

```
<Document>
  <CstmrCdtTrfInitn>
    ...
    <PmtInf>
      ...
      <DbtrAcct>
        <Id>
          <Othr>
            <Id>0010000001</Id>
          </Othr>
        </Id>
        <Tp>
          <Prtry>1</Prtry>
        </Tp>
      </DbtrAcct>
      ...
    </PmtInf>
  </CstmrCdtTrfInitn>
</Document>
```

PmtInf オブジェクトの DbtrAcct オブジェクトの Id フィールドに振込依頼人口座番号、例えば 0010000001 を挿入する。Tp フィールドに振込預金人預金種別、例えば普通口座の 1 を挿入する。

```
// 振込依頼人口座情報
CashAccount16 ca1 = new CashAccount16();
pii.setDbtrAcct(ca1);
AccountIdentification4Choice ai1 = new AccountIdentification4Choice();
ca1.setId(ai1);

// 振込依頼人口座番号
GenericAccountIdentification1 gai1 = new GenericAccountIdentification1();
gai1.setId("0010000001");
ai1.setOthr(gai1);

// 振込依頼人預金種目
CashAccountType2 cat1 = new CashAccountType2();
cat1.setPrtry("1");
ca1.setTp(cat1);
```

DbtrAcct の次に、振込側の銀行情報に当たる DbtrAgt で始まる以下の XML に対応する JAVA オブジェクト群を生成する。

```
<Document>
  <CstmrCdtTrfInitn>
    ...
    <PmtInf>
      ...
      <DbtrAgt>
        <FinInstnId>
          <ClrSysMmbId>
            <ClrSysId>
              <Cd>JPZGN</Cd>
            </ClrSysId>
          <MmbId>0000</MmbId>
        </ClrSysMmbId>
        <Nm>シムケギンコウメイ</Nm>
      </FinInstnId>
      <BrnchId>
        <Id>000</Id>
        <Nm>シムケシテンメイ</Nm>
      </BrnchId>
    </DbtrAgt>
    <UltmtDbtr>
      <Nm>フリコミイライニンメイ</Nm>
    </UltmtDbtr>
    ...
  </PmtInf>
</CstmrCdtTrfInitn>
</Document>
```

DbtrAgt オブジェクトの FinInstnId フィールドの ClrSysMmbId フィールドの ClrSysId フィールドの Cd フィールドに全銀システムを表す JPZGN を挿入し、ClrSysMmbId フィールドの MmbId フィールドに仕向銀行番号、例えば 0000 を挿入する。FinInstnId フィールドの Nm フィールドに仕向銀行名、例えばシムケギンコウメイを挿入する。

FinInstnId フィールドの BranchId フィールドの Id フィールドに仕向支店番号、例えば 000 を挿入し、Nm フィールドに仕向支店名、例えばシムケシテンメイを挿入する。

UltmtDbtr フィールドの Nm フィールドに振込依頼人名、例えばフリコミライニンメイを挿入する。

```
// 仕向金融機関情報
BranchAndFinancialInstitutionIdentification4 bfii1
    = new BranchAndFinancialInstitutionIdentification4();
pii.setDbtrAgt(bfii1);
FinancialInstitutionIdentification7 fii1
    = new FinancialInstitutionIdentification7();
bfii1.setFinInstnId(fii1);
ClearingSystemMemberIdentification2 csmi1
    = new ClearingSystemMemberIdentification2();
fii1.setClrSysMmbId(csmi1);
ClearingSystemIdentification2Choice csid1
    = new ClearingSystemIdentification2Choice();
csmi1.setClrSysId(csid1);
csid1.setCd("JPZGN");

// 仕向銀行番号
csmi1.setMmbId("0000");

// 仕向銀行名
fii1.setNm("シムケギンコウメイ");

BranchData2 bd1 = new BranchData2();
bfii1.setBrnchId(bd1);

// 仕向支店番号
bd1.setId("000");

// 仕向支店名
bd1.setNm("シムケシテンメイ");

// 振込依頼人名
PartyIdentification32 upi = new PartyIdentification32();
pii.setUltmtDbtr(upi);
upi.setNm("フリコミライニンメイ");
```


UltmtDbtr の次に、一つの振込に当たる CdtTrfTxInf で始まる以下の XML に対応する JAVA オブジェクト群を生成する。

```
<Document>
  <CstmrCdtTrfInitn>
    ...
    <PmtInf>
      ...
      <CdtTrfTxInf>
        <PmtId>
          <EndToEndId>E4AE332F626448268E3852CA76A81620
          </EndToEndId>
        </PmtId>
        <Amt>
          <InstdAmt Ccy="JPY">12345</InstdAmt>
        </Amt>
        ...
      </CdtTrfTxInf>
    </PmtInf>
  </CstmrCdtTrfInitn>
</Document>
```

振込側と入金側の双方で唯一となるような取引明細識別番号、例えば E4AE332F626448268E3852CA76A81620 を生成し、EndToEndId フィールドに設定する。

次に、振込金 12345 円を Amt オブジェクトの InstdAmt フィールドの Value に設定する。合わせて通貨単位の円に対応する JPY を CCy属性に設定する。

```
// 取引明細
CreditTransferTransactionInformation10 ctti
    = new CreditTransferTransactionInformation10();
pii.getCdtTrfTxInf().add(ctti);
PaymentIdentification1 pid = new PaymentIdentification1();
ctti.setPmtId(pid);

// 取引明細識別番号（振込依頼人発行）
pid.setEndToEndId("E4AE332F626448268E3852CA76A81620");

// 振込金額情報
AmountType3Choice amt = new AmountType3Choice();
ctti.setAmt(amt);
ActiveOrHistoricCurrencyAndAmount aamt
    = new ActiveOrHistoricCurrencyAndAmount();
amt.setInstdAmt(aamt);

// 振込金額
aamt.setValue(new BigDecimal(12345));
aamt.setCcy("JPY");
```

Amt の次に、振込側の銀行情報に当たる CdtrAgt で始まる以下の XML に対応する JAVA オブジェクト群を生成する。

```

<Document>
  <CstmrCdtTrfInitn>
    ...
    <PmtInf>
      ...
      <CdtTrfTxInf>
        ...
        <CdtrAgt>
          <FinInstnId>
            <ClrSysMmbId>
              <ClrSysId>
                <Cd>JPZGN</Cd>
              </ClrSysId>
            <MmbId>9999</MmbId>
          </ClrSysMmbId>
          <Nm>ヒシムケギンコウメイ</Nm>
          <Othr>
            <Id>1111</Id>
          </Othr>
        </FinInstnId>
        <BrnchId>
          <Id>999</Id>
          <Nm>ヒシムケシテンメイ</Nm>
        </BrnchId>
      </CdtrAgt>
      ...
    </CdtTrfTxInf>
  </PmtInf>
</CstmrCdtTrfInitn>
</Document>

```

CdtrAgt オブジェクトの FinInstnId フィールドの ClrSysMmbId フィールドの ClrSysId フィールドの Cd フィールドに全銀システムを表す JPZGN を挿入し、ClrSysMmbId フィールドの MmbId フィールドに被仕向銀行番号、例えば 9999 を挿入する。FinInstnId フィールドの Nm フィールドに被仕向銀行名、例えばヒシムケギンコウメイを挿入する。

手形交換所番号がある場合は、FinInstnId フィールドの Othr フィールドの Id フィールドに挿入する。

FinInstnId フィールドの BranchId フィールドの Id フィールドに被仕向支店番号、例えば 999 を挿入し、Nm フィールドに被仕向支店名、例えばヒシムケシテンメイを挿入する。

```
// 被仕向銀行情報
BranchAndFinancialInstitutionIdentification4 bfii2
    = new BranchAndFinancialInstitutionIdentification4();
ctti.setCdtrAgt(bfii2);
FinancialInstitutionIdentification7 fii2
    = new FinancialInstitutionIdentification7();
bfii2.setFinInstnId(fii2);
ClearingSystemMemberIdentification2 csmi2
    = new ClearingSystemMemberIdentification2();
fii2.setClrSysMmbId(csmi2);
ClearingSystemIdentification2Choice csid2
    = new ClearingSystemIdentification2Choice();
csmi2.setClrSysId(csid2);
csid2.setCd("JPZGN");

// 被仕向銀行番号
csmi2.setMmbId("9999");

// 被仕向銀行名
fii2.setNm("ヒシムケギンコウメイ");

// 手形交換所番号
GenericFinancialIdentification1 gfi = new GenericFinancialIdentification1();
fii2.setOthr(gfi);
gfi.setId("1111");

// 被仕向支店情報
BranchData2 bd2 = new BranchData2();
bfii2.setBrnchId(bd2);

// 被仕向支店番号
bd2.setId("999");

// 被仕向支店名
bd2.setNm("ヒシムケシテンメイ");
```

CdtrAgt の次に、Cdtr で始まる以下の XML に当たる JAVA オブジェクト群を生成する。

```

<Document>
  <CstmrCdtTrfInitn>
    ...
    <PmtInf>
      ...
      <CdtTrfTxInf>
        ...
        <Cdtr>
          <Nm>ウケトリニンメイ</Nm>
          <Id>
            <OrgId>
              <Othr>
                <Id>1234567890123</Id>
                <SchmeNm>
                  <Cd>TXID</Cd>
                </SchmeNm>
              </Othr>
            <Othr>
              <Id>1234567890</Id>
              <SchmeNm>
                <Prtry>Customer Code1</Prtry>
              </SchmeNm>
            </Othr>
            <Othr>
              <Id>1234567890</Id>
              <SchmeNm>
                <Prtry>Customer Code2</Prtry>
              </SchmeNm>
            </Othr>
          </OrgId>
        </Id>
      </Cdtr>
      ...
    </CdtTrfTxInf>
  </PmtInf>
</CstmrCdtTrfInitn>
</Document>

```

Cdtr オブジェクトの Nm フィールドに受取人名、例えばウケトリニンメイを挿入する。一つ目の Othr オブジェクトの Id フィールドに受取人法人番号、例えば 1234567890123 を挿入し、法人番号であることを示す TXID を振込依頼人法人番号概要コードに挿入する。

```
// 受取人情報
PartyIdentification32 pi3 = new PartyIdentification32();
ctti.setCdtr(pi3);
Party6Choice p2 = new Party6Choice();
pi3.setId(p2);
OrganisationIdentification4 oid2 = new OrganisationIdentification4();
p2.setOrgId(oid2);

// 受取人名
pi3.setNm("ウケトリニンメイ");

// 受取人法人番号(法人マイナンバー)情報
GenericOrganisationIdentification1 goid3
    = new GenericOrganisationIdentification1();
oid2.getOthr().add(goid3);
goid3.setId("1234567890123");
OrganisationIdentificationSchemeName1Choice oidshnm3
    = new OrganisationIdentificationSchemeName1Choice();
goid3.setSchmeNm(oidshnm3);
oidshnm3.setCd("TXID");
```

二つ目の **Othr** オブジェクトの **Id** フィールドに顧客コード1、例えば 1234567890 を挿入し、顧客コード 1 を示す **Customer Code1** を顧客コード概要情報に挿入する。三つ目の **Othr** オブジェクトの **Id** フィールドに顧客コード 2、例えば 1234567890 を挿入し、顧客コード 2 を示す **Customer Code2** を顧客コード概要情報に挿入する。

```
// 顧客コード1情報
GenericOrganisationIdentification1 goid4
    = new GenericOrganisationIdentification1();
oid2.getOthr().add(goid4);
goid4.setId("1234567890");
OrganisationIdentificationSchemeName1Choice oidshnm4
    = new OrganisationIdentificationSchemeName1Choice();
goid4.setSchmeNm(oidshnm4);
oidshnm4.setPrtry("Customer Code1");

// 顧客コード2情報
GenericOrganisationIdentification1 goid5
    = new GenericOrganisationIdentification1();
oid2.getOthr().add(goid5);
goid5.setId("1234567890");
OrganisationIdentificationSchemeName1Choice oidshnm5
    = new OrganisationIdentificationSchemeName1Choice();
goid5.setSchmeNm(oidshnm5);
oidshnm5.setPrtry("Customer Code2");
```

Cdtr の次に、CdtrAcct で始まる以下の XML に当たる JAVA オブジェクト群を生成する。

```
<Document>
  <CstmrCdtTrfInitn>
    ...
    <PmtInf>
      ...
      <CdtTrfTxInf>
        ...
        <CdtrAcct>
          <Id>
            <Othr>
              <Id>0010000001</Id>
            </Othr>
          </Id>
          <Tp>
            <Prtry>1</Prtry>
          </Tp>
        </CdtrAcct>
        ...
      </CdtTrfTxInf>
    </PmtInf>
  </CstmrCdtTrfInitn>
</Document>
```

CdtrAcct オブジェクトの Id フィールドに受取人口座番号、例えば 0010000001 を挿入する。Tp フィールドに受取人預金種別、例えば普通口座の 1 を挿入する。

```
// 受取人口座情報
CashAccount16 ca2 = new CashAccount16();
ctti.setCdtrAcct(ca2);
AccountIdentification4Choice ai2 = new AccountIdentification4Choice();
ca2.setId(ai2);

// 受取人口座番号
GenericAccountIdentification1 gai2 = new GenericAccountIdentification1();
gai2.setId("0010000001");
ai2.setOthr(gai2);

// 受取人預金種目
CashAccountType2 cat2 = new CashAccountType2();
cat2.setPrtry("1");
ca2.setTp(cat2);
```


CdtrAcct の次に、InstrForCdtrAgt で始まる以下の XML に当たる JAVA オブジェクト群を生成する。

```
<Document>
  <CstmrCdtTrfInitn>
    ...
    <PmtInf>
      ...
      <CdtTrfTxInf>
        ...
        <InstrForCdtrAgt>
          <InstrInf>7</InstrInf>
        </InstrForCdtrAgt>
        <InstrForDbtrAgt>Y:dummy01:dummy0123456789du
        </InstrForDbtrAgt>
        <Purp>
          <Prtry>0</Prtry>
        </Purp>
        ...
      </CdtTrfTxInf>
    </PmtInf>
  </CstmrCdtTrfInitn>
</Document>
```

InstrForCdtrAgt オブジェクトの InstrInf フィールドに振込指定区分、例えばテレ振込の 7 を挿入する。InstrForDbtrAgt フィールドに識別表示と仕向金融機関指示情報、例えば Y、dummy01、dummy0123456789du を挿入する。Purp フィールドの Prtry フィールドに新規コード、例えば 0 を挿入する。

```
// 振込指定区分情報
InstructionForCreditorAgent1 ifca = new InstructionForCreditorAgent1();
ctti.getInstrForCdtrAgt().add(ifca);

// 振込指定区分
ifca.setInstrInf("7");

// 識別表示および仕向金融機関指示情報
ctti.setInstrForDbtrAgt(
    new InstrForDbtrAgt("Y", "dummy01", "dummy0123456789du")
        .Unparse());

// 新規コード
Purpose2Choice purp = new Purpose2Choice();
ctti.setPurp(purp);
purp.setPrtry("0");
```

次に、金融 EDI 情報が XML 形式であると想定し、以下の様な金融 EDI 情報を表す XML に対応する JAVA オブジェクト群を生成する。

```
<Document>
  <CstmrCdtTrfInitn>
    ...
    <PmtInf>
      ...
      <CdtTrfTxInf>
        ...
        <RmtInf>
          <Ustrd>MIME-Version: 1.0</Ustrd>
          <Ustrd>Content-Type: text/xml</Ustrd>
          <Ustrd>Content-Transfer-Encoding: base64</Ustrd>
          <Ustrd>PENyb3NzSW5kdXN0cnlSZW1pdHRhbmNIQWR2a ...
          </Ustrd>
        </RmtInf>
      </CdtTrfTxInf>
    </PmtInf>
  </CstmrCdtTrfInitn>
</Document>
```

まず、MIME ヘッダーの 3 行をを挿入する。次にファイル、例えば CrossIndustryRemittanceAdvice.xml に格納された XML データの金融 EDI 情報を Base64 エンコードし、Unstructured タグを付加し、挿入する。

```
// 商流情報
RemittanceInformation5 ri = new RemittanceInformation5();
ctti.setRmtInf(ri);

// XML 形式の場合
// MIME ヘッダー
ri.getUstrd().add("MIME-Version: 1.0");
ri.getUstrd().add("Content-Type: text/xml");
ri.getUstrd().add("Content-Transfer-Encoding: base64");

// 商流情報ファイル(平文形式)
String path = "CrossIndustryRemittanceAdvice.xml";

String crlf = System.getProperty("line.separator");
try {
    String edi = Files.lines(Paths.get(path), Charset.forName("UTF-8"))
        .collect(Collectors.joining(crlf));
    String encoded = Base64.getMimeEncoder()
        .encodeToString(edi.getBytes());
    //System.out.println(encoded);

    BufferedReader br = new BufferedReader(new StringReader(encoded));
    String s;
    while ((s = br.readLine()) != null) {
        ri.getUstrd().add(s);
        //System.out.println("<Ustrd>" + s + "</Ustrd>");
    }
} catch (Exception e) {
    e.printStackTrace();
}
```

金融 EDI 情報が XML 形式でない場合は、MIME ヘッダーの付加、Base64 エンコードは不要のため、Unstructured タグを付加し、挿入する。

```
// 商流情報
RemittanceInformation5 ri = new RemittanceInformation5();
ctti.setRmtInf(ri);

// 平文形式の場合
// 商流情報ファイル(平文形式)
String path = "RemittanceAdvice.txt";
try {
    BufferedReader br = Files.newBufferedReader
        (Paths.get(path), Charset.forName("UTF-8"));
    String s;
    while ((s = br.readLine()) != null) {
        ri.getUstrd().add(s);
        //System.out.println("<Ustrd>" + s + "</Ustrd>");
    }
} catch (Exception e) {
    e.printStackTrace();
}
```

ここまでで生成した JAVA オブジェクト群を、marshaller 機能を使った XML 化と検証用 XML スキーマ、例えば pain.001.001.03.xsd を使った検証を行い、文字列領域に格納する。更にテスト的に標準出力に出力している。

```
StringWriter stringWriter = new StringWriter();
try {
    SchemaFactory sf
        = SchemaFactory.newInstance
            (XMLConstants.W3C_XML_SCHEMA_NS_URI);
    Schema schema = sf.newSchema(new File("pain.001.001.03.xsd"));

    JAXBContext context = JAXBContext.newInstance(Document.class);
    Marshaller marshaller = context.createMarshaller();
    marshaller.setSchema(schema);
    marshaller.setEventHandler(new MyValidationEventHandler());

    // インデントした形式とする場合
    marshaller.setProperty
        (Marshaller.JAXB_FORMATTED_OUTPUT, true);
    // XML ヘッダーを生成しない場合
    // marshaller.setProperty(Marshaller.JAXB_FRAGMENT, true);
    // 出力対象により XML ヘッダーの扱いが変わることに注意
    // XML ヘッダーを直接生成する場合
    // stringWriter.write
    // ("<?xml version='1.0' encoding='UTF-8' standalone='no'?">");

    marshaller.marshal(doc, stringWriter);

    // 標準出力に出す場合
    // PrintStream out = new PrintStream(System.out, true, "UTF-8");
    // BOM を付ける場合、以下の 3 行を実行する。
    // out.write(0xef);
    // out.write(0xbb);
    // out.write(0xbf);
    // out.print(stringWriter.toString());
} catch (Exception e) {
    e.printStackTrace();
}
}
```

InstForDbtrAgt の文字列を生成する InstrForDbtrAgt クラスを以下に示す。

```
import java.util.regex.Pattern;
import java.util.regex.Matcher;

public class InstrForDbtrAgt {

    protected String id;
    protected String dummy1;
    protected String dummy2;
    protected Pattern p;

    InstrForDbtrAgt() {
        this.p = Pattern.compile("([^\:]*)([^\:]*)([^\:]*)?");
    }

    InstrForDbtrAgt(String id, String dummy1, String dummy2) {
        this.p = Pattern.compile("([^\:]*)([^\:]*)([^\:]*)?");
        this.id = id;
        this.dummy1 = dummy1;
        this.dummy2 = dummy2;
    }

    public String Unparse() {
        return this.id + ":" + this.dummy1 + ":" + this.dummy2;
    }

    public InstrForDbtrAgt Parse(String v) {

        Matcher m = p.matcher(v);

        if (m.find()){
            this.id = m.group(1);
            this.dummy1 = m.group(2);
            if(m.groupCount() > 2) {
                this.dummy1 = m.group(3);
            }
        }
        return this;
    }
}
```

InstrForDbtrAgt クラスの続きを以下に示す。

```
public InstrForDbtrAgt setId(String v) {
    this.id = v;
    return this;
}

public String getId() {
    return this.id;
}

public void setDummy1(String v) {
    this.dummy1 = v;
}

public InstrForDbtrAgt putDummy1(String v) {
    this.dummy1 = v;
    return this;
}

public String getDummy1() {
    return this.dummy1;
}

public void setDummy2(String v) {
    this.dummy2 = v;
}

public InstrForDbtrAgt putDummy2(String v) {
    this.dummy2 = v;
    return this;
}

public String getDummy2() {
    return this.dummy2;
}
}
```

最後に **Marshaller** で使った検証用の例外処理ルーチンを以下に示す。

```
import javax.xml.bind.ValidationEvent;
import javax.xml.bind.ValidationEventHandler;

public class MyValidationEventHandler implements ValidationEventHandler {

    public boolean handleEvent(ValidationEvent event) {
        System.out.println("¥nEVENT");
        System.out.println("SEVERITY:  " + event.getSeverity());
        System.out.println("MESSAGE:  " + event.getMessage());
        System.out.println("LINKED EXCEPTION:  " + event.getLinkedException());
        System.out.println("LOCATOR");
        System.out.println("    LINE NUMBER:  "
            + event.getLocator().getLineNumber());
        System.out.println("    COLUMN NUMBER:  "
            + event.getLocator().getColumnNumber());
        System.out.println("    OFFSET:  " + event.getLocator().getOffset());
        System.out.println("    OBJECT:  " + event.getLocator().getObject());
        System.out.println("    NODE:  " + event.getLocator().getNode());
        System.out.println("    URL:  " + event.getLocator().getURL());
        return true;
    }
}
```


(3) 入出金明細 (ISO20022 Camt.052) の読み込み方

xjc コマンドで Camt.052.001.02 の XML スキーマから生成された関数、読み込んだ XML の検証、金融 EDI 情報の Base64 エンコード、日時データや金額データを作成するためのランタイムルーチン等をインポートする。

最初に出力用ストリームの文字コードを UTF-8 に指定する。

```
import java.io.File;
import java.io.StringReader;
import java.io.PrintStream;

import java.util.List;
import java.util.Base64;
import java.util.regex.Pattern;
import java.util.regex.Matcher;

import iso.std.iso._20022.tech.xsd.camt_052_001.*;

import javax.xml.bind.JAXBContext;
import javax.xml.bind.Unmarshaller;
import javax.xml.XMLConstants;
import javax.xml.validation.Schema;
import javax.xml.validation.SchemaFactory;

public class Camt052r {
    public static void main(String[] args) {

        PrintStream out = null;
        try {
            out = new PrintStream(System.out, true, "UTF-8");
            // out = System.out;
        } catch (Exception e) {
            e.printStackTrace();
            return;
        }
    }
}
```

Camt.052.001.02 の XML スキーマから生成された関数と共に、JAXB の unmarshaller を使って Camt.052 のメッセージを JAVA オブジェクトに変換する。併せてスキーマを使った検証も行う。

次に JAVA オブジェクトからグループメッセージ ID と XML ファイル作成日時を参照する。当該要素が無い場合、NullPointerException の例外が発生することに注意されたい。

```

Document doc = new Document();

File file = new File("camt.052.001.02.xml");
try {
    SchemaFactory sFactory1
        = SchemaFactory.newInstance(
            XMLConstants.W3C_XML_SCHEMA_NS_URI);
    Schema schema1
        = sFactory1.newSchema(new File("camt.052.001.02.xsd"));

    JAXBContext context1 = JAXBContext.newInstance(Document.class);
    Unmarshaller unmarshaller1 = context1.createUnmarshaller();
    unmarshaller1.setSchema(schema1);
    doc = (Document) unmarshaller1.unmarshal(file);

} catch (Exception e) {
    e.printStackTrace();
    return;
}

// グループメッセージ ID
out.println("/Document/BkToCstmrAcctRpt/GrpHdr/");
try {
    out.println("MsgId["
        + doc.getBkToCstmrAcctRpt().getGrpHdr().getMsgId() + "]);
} catch (NullPointerException e) {
    out.println("[No MsgId]");
}

// XML ファイル作成日時
out.println("/Document/BkToCstmrAcctRpt/GrpHdr/");
try {
    out.println("CreDtTm["
        + doc.getBkToCstmrAcctRpt().getGrpHdr().getCreDtTm() + "]);
} catch (NullPointerException e) {
    out.println("[No CreDtTm]");
}

```

次に JAVA オブジェクトから入出金取引明細情報 ID 、入出金取引明細情報作成日、勘定日 (自) を参照する。get メソッドを使う場合、当該オブジェクトが無い場合は `IndexOutOfBoundsException` の例外が発生することに注意されたい。

```
// 入出金取引明細情報 ID
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/");
try {
    out.println("ID[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getId() + "]");
} catch(NullPointerException e) {
    out.println("[No ID]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt]");
}

// 入出金取引明細情報作成日
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/");
try {
    out.println("CreDtTm["
        + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getCreDtTm() + "]");
} catch(NullPointerException e) {
    out.println("[No CreDtTm]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt]");
}

// 勘定日 (自)
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/FrToDt");
try {
    out.println("FrDtTm["
        + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getFrToDt().getFrDtTm() + "]");
} catch(NullPointerException e) {
    out.println("[No FrDtTm]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt]");
}
```

次に JAVA オブジェクトから勘定日(至)、預金種目および通帳・証書区分を参照する。

```
// 勘定日(至)
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/FrToDt");
try {
    out.println("ToDtTm["
        + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getFrToDt().getToDtTm() + "]");
} catch(NullPointerException e) {
    out.println("[No ToDtTm]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt]");
}

// 口座番号
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Acct/Id/Othr/");
try {
    out.println("Id["
        + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getAcct().getId().getOthr().getId() + "]");
} catch(NullPointerException e) {
    out.println("[No Id]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt]");
}

// 預金種目および通帳・証書区分
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Acct/Tp/");
try {
    out.println("Prtry["
        + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getAcct().getTp().getPrtry() + "]");
} catch(NullPointerException e) {
    out.println("[No Prtry]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt]");
}
```

次に JAVA オブジェクトから口座名、受取人法人番号(法人マイナンバー)、受取人法人番号(法人マイナンバー)概要コードを参照する。

```
// 口座名
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Acct/");
try {
    out.println("Nm["
        + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getAcct().getNm() + "]);
} catch(NullPointerException e) {
    out.println("[No Nm]");
}

// 受取人法人番号(法人マイナンバー)
out.println(
    "/Document/BkToCstmrAcctRpt/Rpt[0]/Acct/Ownr/Id/OrgId/Othr[0]/");
try {
    out.println("Id[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getAcct().getOwnr().getId().getOrgId().getOthr().get(0).getId() + "]);
} catch(NullPointerException e) {
    out.println("[No Id]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Othr]");
}

// 受取人法人番号(法人マイナンバー)概要コード
out.println(
    "/Document/BkToCstmrAcctRpt/Rpt[0]/Acct/Ownr/Id/OrgId/Othr[0]/SchmeNm/");
try {
    out.println("Cd[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getAcct().getOwnr().getId().getOrgId()
        .getOthr().get(0).getSchmeNm().getCd() + "]);
} catch(NullPointerException e) {
    out.println("[No Cd]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Othr]");
}
```

次に JAVA オブジェクトから銀行コード、銀行名、支店コードを参照する。

```
// 銀行コード
out.println(
"/Document/BkToCstmrAcctRpt/Rpt[0]/Acct/Svcr/FinInstnId/ClrSysMmbId/");
try {
    out.println("MmbId[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getAcct().getSvcr().getFinInstnId().getClrSysMmbId().getMmbId()
        + "]");
} catch(NullPointerException e) {
    out.println("[No MmbId]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt]");
}

// 銀行名
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Acct/Svcr/FinInstnId/");
try {
    out.println("Nm[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getAcct().getSvcr().getFinInstnId().getNm() + "]");
} catch(NullPointerException e) {
    out.println("[No Nm]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt]");
}

// 支店コード
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Acct/Svcr/BrnchId/");
try {
    out.println("Id[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getAcct().getSvcr().getBrnchId().getId() + "]");
} catch(NullPointerException e) {
    out.println("[No Id]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt]");
}
```

次に JAVA オブジェクトから支店名、取引前残高種別コード、取引前残高を参照する。

```
// 支店名
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Acct/Svcr/BrnchId/");
try {
    out.println("Nm[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getAcct().getSvcr().getBrnchId().getNm() + "]");
} catch(NullPointerException e) {
    out.println("[No Nm]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt]");
}

// 取引前残高種別コード
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Bal[0]/Tp/CdOrPrtry/");
try {
    out.println("Cd[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getBal().get(0).getTp().getCdOrPrtry().getCd() + "]");
} catch(NullPointerException e) {
    out.println("[No Cd]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt]");
}

// 取引前残高
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Bal[0]/");
try {
    out.println("Amt["
        + doc.getBkToCstmrAcctRpt().getRpt().get(0).getBal().get(0)
        .getAmt().getValue() + "]");
    out.println("Ccy["
        + doc.getBkToCstmrAcctRpt().getRpt().get(0).getBal().get(0)
        .getAmt().getCcy() + "]");
} catch(NullPointerException e) {
    out.println("[No Amt]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Bal]");
}
```

次に JAVA オブジェクトから取引前残高貸越区分、取引前残高勘定日(自)、取引後残高種別コードを参照する。なお、取引前残高関連と取引後残高関連は同じ要素を 2 つ使い記述されるため、`get` の引数を 0 と 1 として使い分けることに注意されたい。

```
// 取引前残高貸越区分
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Bal[0]/");
try {
    out.println("CdtDbtInd["
        + doc.getBkToCstmrAcctRpt().getRpt().get(0).getBal().get(0)
        .getCdtDbtInd() + "];");
} catch(NullPointerException e) {
    out.println("[No CdtDbtInd]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Bal]");
}

// 取引前残高勘定日(自)
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Bal[0]/Dt/");
try {
    out.println("Dt["
        + doc.getBkToCstmrAcctRpt().getRpt().get(0).getBal().get(0)
        .getDt().getDt() + "];");
} catch(NullPointerException e) {
    out.println("[No Dt]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Bal]");
}

// 取引後残高種別コード
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Bal[1]/Tp/CdOrPrtry/");
try {
    out.println("Cd["
        + doc.getBkToCstmrAcctRpt().getRpt().get(0).getBal().get(1)
        .getTp().getCdOrPrtry().getCd() + "];");
} catch(NullPointerException e) {
    out.println("[No Cd]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Bal]");
}
```


次に JAVA オブジェクトから取引後残高、取引後残高貸越区分、取引後残高勘定日(至)を参照する。
取引後残高の金額には通貨単位も記述できる。

```
// 取引後残高
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Bal[1]/");
try {
    out.println("Amt["
        + doc.getBkToCstmrAcctRpt().getRpt().get(0).getBal().get(1)
        .getAmt().getValue() + "]);
    out.println("Ccy["
        + doc.getBkToCstmrAcctRpt().getRpt().get(0).getBal().get(1)
        .getAmt().getCcy() + "]);
} catch(NullPointerException e) {
    out.println("[No Amt]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Bal]");
}

// 取引後残高貸越区分
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Bal[1]/");
try {
    out.println("CdtDbtInd["
        + doc.getBkToCstmrAcctRpt().getRpt().get(0).getBal().get(1)
        .getCdtDbtInd() + "]);
} catch(NullPointerException e) {
    out.println("[No CdtDbtInd]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Bal]");
}

// 取引後残高勘定日(至)
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Bal[1]/Dt/");
try {
    out.println("Dt["
        + doc.getBkToCstmrAcctRpt().getRpt().get(0).getBal().get(1)
        .getDt().getDt() + "]);
} catch(NullPointerException e) {
    out.println("[No Dt]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Bal]");
}
```

次に JAVA オブジェクトからデータレコード件数、入金件数、入金額合計を参照する。

```
// データレコード件数
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/TxSummry/TtlNtries/");
try {
    out.println("NbOfNtrie[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getTxSummry().getTtlNtries().getNbOfNtries() + "]");
} catch(NullPointerException e) {
    out.println("[No NbOfNtrie]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt]");
}

// 入金件数
out.println(
    "/Document/BkToCstmrAcctRpt/Rpt[0]/TxSummry/TtlCdtNtries/");
try {
    out.println("NbOfNtrie[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getTxSummry().getTtlCdtNtries().getNbOfNtries() + "]");
} catch(NullPointerException e) {
    out.println("[No NbOfNtrie]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt]");
}

// 入金額合計
out.println(
    "/Document/BkToCstmrAcctRpt/Rpt[0]/TxSummry/TtlCdtNtries/");
try {
    out.println("Sum[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getTxSummry().getTtlCdtNtries().getSum() + "]");
} catch(NullPointerException e) {
    out.println("[No Sum]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt]");
}
```

次に JAVA オブジェクトから出金件数、出金額合計、入金額合計を参照する。出金額合計には通貨単位が付加されないことに注意されたい。

出金額合計までがヘッダーに記述され、以降は入出金毎に繰り返し記述される。そのため、for 文で参照を繰り返している。

```
// 出金件数
out.println(
    "/Document/BkToCstmrAcctRpt/Rpt[0]/TxSummry/TtlDbtNtries/");
try {
    out.println("NbOfNtrie[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getTxSummry().getTtlDbtNtries().getNbOfNtries() + "]");
} catch(NullPointerException e) {
    out.println("[No NbOfNtrie]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt]");
}

// 出金額合計
out.println(
    "/Document/BkToCstmrAcctRpt/Rpt[0]/TxSummry/TtlDbtNtries/");
try {
    out.println("Sum[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getTxSummry().getTtlDbtNtries().getSum() + "]");
} catch(NullPointerException e) {
    out.println("[No Sum]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt]");
}

if(doc.getBkToCstmrAcctRpt().getRpt().size() <= 0) {
    out.println("-- No Rpt --");
    return;
}

for(int i = 0; i < doc.getBkToCstmrAcctRpt().getRpt().get(0).getNtry().size();
    i++) {

    out.println("¥n--" + i + "--¥n");
}
```

次に JAVA オブジェクトから取引金額、入払区分、取引訂正通知区分を参照する。取引金額には通貨単位も記述できる。

```
// 取引金額
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry[" + i + "]/");
try {
    out.println("Amt[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getAmt().getValue() + "]/");
    out.println("Ccy[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getAmt().getCcy() + "]/");
} catch(NullPointerException e) {
    out.println("[No Amt]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry]");
}

// 入払区分
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry[" + i + "]/");
try {
    out.println("CdtDbtInd[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getCdtDbtInd() + "]/");
} catch(NullPointerException e) {
    out.println("[No CdtDbtInd]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry]");
}

// 取引訂正通知区分
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry[" + i + "]/");
try {
    out.println("RvsInd[" + (doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).isRvsInd() ? "true" : "false") + "]/");
} catch(NullPointerException e) {
    out.println("[No RvsInd]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry]");
}
```

次に JAVA オブジェクトから取引ステータス、勘定日、預入・払出日を参照する。

```
// 取引ステータス - "BOOK"
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry[" + i + "]/");
try {
    out.println("Sts[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getSts() + "]);
} catch(NullPointerException e) {
    out.println("[No Sts]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry]");
}

// 勘定日
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
    + i + "]/BookgDt/");
try {
    out.println("Dt[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getBookgDt().getDt() + "]);
} catch(NullPointerException e) {
    out.println("[No Dt]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry]");
}

// 預入・払出日
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry[" + i + "]/ValDt/");
try {
    out.println("Dt[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getValDt().getDt() + "]);
} catch(NullPointerException e) {
    out.println("[No Dt]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry]");
}
```

次に JAVA オブジェクトから取引区分、定期性口座課税情報合計、税区分を参照する。

```
// 取引区分
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
           + i + "]/BkTxCd/Prtry/");
try {
    out.println("Cd[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
               .getNtry().get(i).getBkTxCd().getPrtry().getCd() + "]);
} catch(NullPointerException e) {
    out.println("[No Cd]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry]");
}

// 定期性口座課税情報合計
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
           + i + "]/Chrgs[0]/");
try {
    out.println("Amt[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
               .getNtry().get(i).getChrgs().get(0).getAmt().getValue() + "]);
    out.println("Ccy[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
               .getNtry().get(i).getChrgs().get(0).getAmt().getCcy() + "]);
} catch(NullPointerException e) {
    out.println("[No Amt]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/Chrgs]");
}

// 税区分
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
           + i + "]/Chrgs[0]/Tax/");
try {
    out.println("Id[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
               .getNtry().get(i).getChrgs().get(0).getTax().getId() + "]);
} catch(NullPointerException e) {
    out.println("[No Id]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/Chrgs]");
}
```

次に JAVA オブジェクトから税率、税額、合計利息を参照する。税額、合計利息には通貨単位も記述できる。

```
// 税率
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
           + i + "]/Chrgs[0]/Tax/");
try {
    out.println("Rate[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
               .getNtry().get(i).getChrgs().get(0).getTax().getRate() + "]);
} catch(NullPointerException e) {
    out.println("[No Rate]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/Chrgs]");
}

// 税額
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
           + i + "]/Chrgs[0]/Tax/");
try {
    out.println("Amt[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
               .getNtry().get(i).getChrgs().get(0).getTax().getAmt().getValue()
               + "]);
    out.println("Ccy[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
               .getNtry().get(i).getChrgs().get(0).getTax().getAmt().getCcy()
               + "]);
} catch(NullPointerException e) {
    out.println("[No Amt]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/Chrgs]");
}

// 合計利息
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
           + i + "]/Intrst[0]/");
try {
    out.println("Amt[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
               .getNtry().get(i).getIntrst().get(0).getAmt().getValue() + "]);
    out.println("Ccy[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
               .getNtry().get(i).getIntrst().get(0).getAmt().getCcy() + "]);
} catch(NullPointerException e) {
    out.println("[No Amt]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/Intrst]");
}
```

次に JAVA オブジェクトから合計利息貸越区分、利率、当初預入日を参照する。

```
// 合計利息貸越区分
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
           + i + "]/Intrt[0]/");
try {
    out.println("CdtDbtInd[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
               .getNtry().get(i).getIntrst().get(0).getCdtDbtInd() + "]");
} catch(NullPointerException e) {
    out.println("[No CdtDbtInd]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/Intrt]");
}

// 利率
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
           + i + "]/Intrt[0]/Rate[0]/Tp");
try {
    out.println("Pctg[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
               .getNtry().get(i).getIntrst().get(0).getRate().get(0)
               .getTp().getPctg() + "]");
} catch(NullPointerException e) {
    out.println("[No Pctg]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/Intrt/Rate]");
}

// 当初預入日
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
           + i + "]/Intrt[0]/FrToDt/");
try {
    out.println("FrDtTm[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
               .getNtry().get(i).getIntrst().get(0).getFrToDt().getFrDtTm()
               + "]");
} catch(NullPointerException e) {
    out.println("[No FrDtTm]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/Intrt]");
}
```


次に JAVA オブジェクトから満期日、当初預入日または満期日、取引明細識別番号(振込依頼人発行)を参照する。

```
// 満期日
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
           + i + "]/Intrt[0]/FrToDt/");
try {
    out.println("ToDtTm[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
               .getNtry().get(i).getIntrst().get(0).getFrToDt().getToDtTm() + "]);
} catch(NullPointerException e) {
    out.println("[No ToDtTm]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/Intrt]");
}

// 当初預入日または満期日
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
           + i + "]/Intrt[0]/");
try {
    out.println("Rsn[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
               .getNtry().get(i).getIntrst().get(0).getRsn() + "]);
} catch(NullPointerException e) {
    out.println("[No Rsn]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/Intrt]");
}

// 取引明細識別番号(振込依頼人発行)
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
           + i + "]/NtryDtls[0]/TxDtls[0]/Refs/");
try {
    out.println("EndToEndId[" + doc.getBkToCstmrAcctRpt()
               .getRpt().get(0).getNtry().get(i).getNtryDtls().get(0)
               .getTxDtls().get(0).getRefs().getEndToEndId() + "]);
} catch(NullPointerException e) {
    out.println("[No EndToEndId]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls]");
}
```

次に JAVA オブジェクトから照会識別情報種別、照会番号または識別番号、取引明細種別ドメインコードを参照する。

```
// 照会識別情報種別 - "Refenece/Identification Number"
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
           + i + "]/NtryDtls[0]/TxDtls[0]/Refs/Prtry/");
try {
    out.println("Tp[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
               .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
               .getRefs().getPrtry().getTp() + "]");
} catch(NullPointerException e) {
    out.println("[No Tp]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls]");
}

// 照会番号または識別番号
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
           + i + "]/NtryDtls[0]/TxDtls[0]/Refs/Prtry/");
try {
    out.println("Ref[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
               .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
               .getRefs().getPrtry().getRef() + "]");
} catch(NullPointerException e) {
    out.println("[No Ref]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls]");
}

// 取引明細種別ドメインコード - "PMNT"
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
           + i + "]/NtryDtls[0]/TxDtls[0]/BkTxCd/Domn/");
try {
    out.println("Cd[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
               .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
               .getBkTxCd().getDomn().getCd() + "]");
} catch(NullPointerException e) {
    out.println("[No Cd]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls]");
}
```

次に JAVA オブジェクトから取引明細種別系列コード、取引明細種別サブ系列コード、期間利息を参照する。

```
// 取引明細種別系列コード - "RCDT"
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
    + i + "]/NtryDtls[0]/TxDtls[0]/BkTxCd/Domn/Fmly/");
try {
    out.println("Cd[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
        .getBkTxCd().getDomn().getFmly().getCd() + "]);
} catch(NullPointerException e) {
    out.println("[No Cd]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls]");
}

// 取引明細種別サブ系列コード - "DMCT"
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
    + i + "]/NtryDtls[0]/TxDtls[0]/BkTxCd/Domn/Fmly/");
try {
    out.println("SubFmlyCd[" + doc.getBkToCstmrAcctRpt()
        .getRpt().get(0).getNtry().get(i).getNtryDtls().get(0)
        .getTxDtls().get(0).getBkTxCd().getDomn().getFmly()
        .getSubFmlyCd() + "]);
} catch(NullPointerException e) {
    out.println("[No SubFmlyCd]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls]");
}

// 期間利息
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
    + i + "]/NtryDtls[0]/TxDtls[0]/Intrst[0]/");
try {
    out.println("Amt[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
        .getIntrst().get(0).getAmt().getValue() + "]);
    out.println("Ccy[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
        .getIntrst().get(0).getAmt().getCcy() + "]);
} catch(NullPointerException e) {
    out.println("[No Amt]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls/Intrst]");
}
```

次に JAVA オブジェクトから期間利息貸越区分、中間払区分、中間払利率を参照する。

```
// 期間利息貸越区分
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
    + i + "]/NtryDtls[0]/TxDtls[0]/Intrst[0]/");
try {
    out.println("CdtDbtInd[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
        .getIntrst().get(0).getCdtDbtInd() + "]/");
} catch (NullPointerException e) {
    out.println("[No CdtDbtInd]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No Rpt/NtryDtls/Ntry/TxDtls/Intrst]");
}

// 中間払区分
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
    + i + "]/NtryDtls[0]/TxDtls[0]/Intrst[0]/Tp/");
try {
    out.println("Prtry[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
        .getIntrst().get(0).getTp().getPrtry() + "]/");
} catch (NullPointerException e) {
    out.println("[No Prtry]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls/Intrst]");
}

// 中間払利率
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
    + i + "]/NtryDtls[0]/TxDtls[0]/Intrst[0]/Rate[0]/Tp/");
try {
    out.println("Pctg[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
        .getIntrst().get(0).getRate().get(0).getTp().getPctg() + "]/");
} catch (NullPointerException e) {
    out.println("[No Pctg]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls/Intrst]");
}
```

次に JAVA オブジェクトから期間(1)または期間(2)、期後利息、期後利息貸越区分を参照する。

```
// 期間(1)または期間(2)
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
    + i + "]/NtryDtls[0]/TxDtls[0]/Intrst[0]/Rate[0]/Tp/");
try {
    out.println("Rsn[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
        .getIntrst().get(0).getRsn() + "]);
} catch(NullPointerException e) {
    out.println("[No Rsn]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls/Intrst]");
}

// 期後利息
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
    + i + "]/NtryDtls[0]/TxDtls[0]/Intrst[1]/");
try {
    out.println("Amt[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
        .getIntrst().get(1).getAmt().getValue() + "]);
    out.println("Ccy[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
        .getIntrst().get(1).getAmt().getCcy() + "]);
} catch(NullPointerException e) {
    out.println("[No Amt]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls/Intrst]");
}

// 期後利息貸越区分
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
    + i + "]/NtryDtls[0]/TxDtls[0]/Intrst[1]/");
try {
    out.println("CdtDbtInd[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
        .getIntrst().get(1).getCdtDbtInd() + "]);
} catch(NullPointerException e) {
    out.println("[No CdtDbtInd]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls/Intrst]");
}
```

次に JAVA オブジェクトから期後払区分、期後払利率、期後期間を参照する。

```
// 期後払区分
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
    + i + "]/NtryDtls[0]/TxDtls[0]/Intrst[1]/");
try {
    out.println("CdtDbtInd[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
        .getIntrst().get(1).getTp().getPrtry() + "]);
} catch(NullPointerException e) {
    out.println("[No CdtDbtInd]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls/Intrst]");
}

// 期後払利率
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
    + i + "]/NtryDtls[0]/TxDtls[0]/Intrst[1]/Rate[0]/Tp/");
try {
    out.println("Pctg[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
        .getIntrst().get(1).getRate().get(0).getTp().getPctg() + "]);
} catch(NullPointerException e) {
    out.println("[No Pctg]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls/Intrst]");
}

// 期後期間
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
    + i + "]/NtryDtls[0]/TxDtls[0]/Intrst[1]/");
try {
    out.println("Rsn[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
        .getIntrst().get(1).getRsn() + "]);
} catch(NullPointerException e) {
    out.println("[No Rsn]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls/Intrst]");
}
```

次に JAVA オブジェクトから税引後利息、税引後利息貸越区分、税引後払区分を参照する。

```
// 税引後利息
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
    + i + "]/NtryDtls[0]/TxDtls[0]/Intrst[2]/");
try {
    out.println("Amt[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
        .getIntrst().get(2).getAmt().getValue() + "]);
    out.println("Ccy[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
        .getIntrst().get(2).getAmt().getCcy() + "]);
} catch (NullPointerException e) {
    out.println("[No Amt]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls/Intrst]");
}

// 税引後利息貸越区分
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
    + i + "]/NtryDtls[0]/TxDtls[0]/Intrst[2]/");
try {
    out.println("CdtDbtInd[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
        .getIntrst().get(2).getCdtDbtInd() + "]);
} catch (NullPointerException e) {
    out.println("[No CdtDbtInd]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls/Intrst]");
}

// 税引後払区分
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
    + i + "]/NtryDtls[0]/TxDtls[0]/Intrst[2]/Tp/");
try {
    out.println("Prtry[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
        .getIntrst().get(2).getTp().getPrtry() + "]);
} catch (NullPointerException e) {
    out.println("[No Prtry]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls/Intrst]");
}
```

次に JAVA オブジェクトから振込依頼人名または契約者番号、振込依頼人コード、振込依頼人概要コードを参照する。

```
// 振込依頼人名または契約者番号
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
    + i + "]/NtryDtls[0]/TxDtls[0]/RltdPties/Dbtr/");
try {
    out.println("Nm[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
        .getRltdPties().getDbtr().getNm() + "]);
} catch(NullPointerException e) {
    out.println("[No Nm]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls]");
}

// 振込依頼人コード
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
    + i + "]/NtryDtls[0]/TxDtls[0]/RltdPties/Dbtr/Id/OrgId/Othr[0]/");
try {
    out.println("Id[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
        .getRltdPties().getDbtr()
        .getId().getOrgId().getOthr().get(0).getId() + "]);
} catch(NullPointerException e) {
    out.println("[No Id]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls/Othr]");
}

// 振込依頼人概要コード - "BANK"
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry[" + i
    + "]/NtryDtls[0]/TxDtls[0]/RltdPties/Dbtr/Id/OrgId/Othr[0]/SchmeNm/");
try {
    out.println("Cd[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
        .getRltdPties().getDbtr().getId().getOrgId().getOthr().get(0)
        .getSchmeNm().getCd() + "]);
} catch(NullPointerException e) {
    out.println("[No Cd]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls/Othr]");
}
```


次に JAVA オブジェクトから振込依頼人法人番号(法人マイナンバー)、振込依頼人法人番号(法人マイナンバー)概要コード、仕向銀行名を参照する。

```
// 振込依頼人法人番号(法人マイナンバー)
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry[" + i
    + "]/NtryDtls[0]/TxDtls[0]/RltdPties/Dbtr/Id/OrgId/Othr[1]/");
try {
    out.println("Id[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
        .getRltdPties().getDbtr().getId().getOrgId().getOthr().get(1)
        .getId() + "]);
} catch(NullPointerException e) {
    out.println("[No Id]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls/Othr]");
}

// 振込依頼人法人番号(法人マイナンバー)概要コード - "TXID"
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry[" + i
    + "]/NtryDtls[0]/TxDtls[0]/RltdPties/Dbtr/Id/OrgId/Othr[1]/SchmeNm/");
try {
    out.println("Cd[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
        .getRltdPties().getDbtr().getId().getOrgId().getOthr().get(1)
        .getSchmeNm().getCd() + "]);
} catch(NullPointerException e) {
    out.println("[No Cd]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls/Othr]");
}

// 仕向銀行名
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry[" + i
    + "]/NtryDtls[0]/TxDtls[0]/RltdAgts/DbtrAgt/FinInstnId/");
try {
    out.println("Nm[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
        .getRltdAgts().getDbtrAgt().getFinInstnId().getNm() + "]);
} catch(NullPointerException e) {
    out.println("[No Nm]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls]");
}
```

次に JAVA オブジェクトから仕向支店名、僚店支店名、EDI 情報を参照する。

```
// 仕向支店名
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry[" + i
           + "]/NtryDtls[0]/TxDtls[0]/RltdAgts/DbtrAgt/BrnchId/");
try {
    out.println("Nm[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
               .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
               .getRltdAgts().getDbtrAgt().getBrnchId().getNm() + "]);
} catch(NullPointerException e) {
    out.println("[No Nm]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls]");
}

// 僚店支店名
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry[" + i
           + "]/NtryDtls[0]/TxDtls[0]/RltdAgts/CdtrAgt/BrnchId/");
try {
    out.println("Id[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
               .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
               .getRltdAgts().getCdtrAgt().getBrnchId().getId() + "]);
} catch(NullPointerException e) {
    out.println("[No Id]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls]");
}

// EDI 情報
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry[" + i
           + "]/NtryDtls[0]/TxDtls[0]/RltdRmtInf[0]/");
try {
    out.println("RmtId[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
               .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
               .getRltdRmtInf().get(0).getRmtId() + "]);
} catch(NullPointerException e) {
    out.println("[No RmtId]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls/RltdRmtInf]");
}
```

次に JAVA オブジェクトから金融 EDI 情報を抽出する。Ustrds オブジェクトがあり、先頭に「MIME-Version 1.0」があれば XML 形式と判断している。

```
// 金融 EDI 情報
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry[" + i
    + "]/NtryDtls[0]/TxDtls[0]/RmtInf");
try {
    List<String> ustrds = null;
    ustrds = doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(0).getTxDtls().get(0)
        .getRmtInf().getUstrd();
    if(ustrds == null) {
        out.println("No Remittance Advices");
    } else {
        Pattern p = Pattern.compile(
            "^[Mm][Ii][Mm][Ee]-[Vv][Ee][Rr][Ss][Ii][Oo][Nn]:[ \t]*1.0");
        if (!p.matcher(ustrds.get(0)).find()){

            // 平文の処理
            out.println("No MIME-Version [" + ustrds.get(0) + "]");
        } else {
            p = Pattern.compile(
                "^[Cc][oO][nN][tT][eE][nN][tT]-[Tt][yY][pP][eE][ \t]*:[ \t]*[tT][eE][xX][tT]/[xX][mM][lL]");
            ustrds.remove(0);
            if (!p.matcher(ustrds.get(0)).find()){
                out.println("No Content-Type: text/xml ["
                    + ustrds.get(0) + "]");
            } else {
                p = Pattern.compile(
                    "^[Cc][oO][nN][tT][eE][nN][tT]-[Tt][rR][aA][nN][sS][fF][eE][rR]-" +
                    "[Ee][nN][cC][oO][dD][iI][nN][gG][ \t]*:[ \t]*[Bb][aA][sS][eE]64");
                ustrds.remove(0);
                if (!p.matcher(ustrds.get(0)).find()){
                    out.println("No Content-Transfer-Encoding: base64 ["
                        + ustrds.get(0) + "]");
                }
            }
        }
    }
}
```

以下は金融 EDI 情報を抽出の続きで、更に JAVA オブジェクトから手形・小切手番号の参照をする。

```

        } else {
            ustrds.remove(0);
            StringBuilder sb = new StringBuilder();
            ustrds.forEach(s -> sb.append(s));
            String decoded
                = new String(Base64.getDecoder().decode(sb.toString()));

            // XML の処理
            out.println(decoded);
        }}}
    }

} catch (NullPointerException e) {
    out.println("[No Ustrd]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls/Ustrd]");
}

// 手形・小切手番号
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
    + i + "]/NtryDtls[1]/TxDtls[0]/Refs/");
try {
    out.println("ChqNb[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(1).getTxDtls().get(0)
        .getRefs().getChqNb() + "]");
} catch (NullPointerException e) {
    out.println("[No ChqNb]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls]");
}

```

次に JAVA オブジェクトから手形・小切手区分、手形・小切手区分詳細を参照する。

```
// 手形・小切手区分
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
    + i + "]/NtryDtls[1]/TxDtls[0]/Refs/Prtry/");
try {
    out.println("Tp[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(1).getTxDtls().get(0)
        .getRefs().getPrtry().getTp() + "]);
} catch(NullPointerException e) {
    out.println("[No Tp]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls]");
}

// 手形・小切手区分詳細 - "0"
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
    + i + "]/NtryDtls[1]/TxDtls[0]/Refs/Prtry/");
try {
    out.println("Ref[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(1).getTxDtls().get(0)
        .getRefs().getPrtry().getRef() + "]);
} catch(NullPointerException e) {
    out.println("[No Ref]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls]");
}
```

次に JAVA オブジェクトからうち他店券金額、手形・小切手情報種別ドメインコードを参照する。

```
// うち他店券金額
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
+ i + "]/NtryDtls[1]/TxDtls[0]/AmtDtls/");
try {
    out.println("Amt[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(1).getTxDtls().get(0)
        .getAmtDtls().getTxAmt().getAmt().getValue() + "]);
    out.println("Ccy[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(1).getTxDtls().get(0)
        .getAmtDtls().getTxAmt().getAmt().getCcy() + "]);
} catch (NullPointerException e) {
    out.println("[No Amt]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls]");
}

// 手形・小切手情報種別ドメインコード - "PMNT"
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
+ i + "]/NtryDtls[1]/TxDtls[0]/BkTxCd/Domn/");
try {
    out.println("Cd[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(1).getTxDtls().get(0)
        .getBkTxCd().getDomn().getCd() + "]);
} catch (NullPointerException e) {
    out.println("[No Cd]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls]");
}
```

次に JAVA オブジェクトからうち他店券金額、手形・小切手情報種別ドメインコードを参照する。

```
// うち他店券金額
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
+ i + "]/NtryDtls[1]/TxDtls[0]/AmtDtls/");
try {
    out.println("Amt[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(1).getTxDtls().get(0)
        .getAmtDtls().getTxAmt().getAmt().getValue() + "]);
    out.println("Ccy[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(1).getTxDtls().get(0)
        .getAmtDtls().getTxAmt().getAmt().getCcy() + "]);
} catch (NullPointerException e) {
    out.println("[No Amt]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls]");
}

// 手形・小切手情報種別ドメインコード - "PMNT"
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry["
+ i + "]/NtryDtls[1]/TxDtls[0]/BkTxCd/Domn/");
try {
    out.println("Cd[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getNtry().get(i).getNtryDtls().get(1).getTxDtls().get(0)
        .getBkTxCd().getDomn().getCd() + "]);
} catch (NullPointerException e) {
    out.println("[No Cd]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls]");
}
```

次に JAVA オブジェクトからうち交換呈示日、日付種別、不渡返還日を参照する。

```
// 交換呈示日
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry[" + i
          + "]/NtryDtls[1]/TxDtls[0]/RltdDts/AcceptncDtTm/");
try {
    out.println("AcceptncDtTm[" + doc.getBkToCstmrAcctRpt()
          .getRpt().get(0).getNtry().get(i).getNtryDtls().get(1)
          .getTxDtls().get(0).getRltdDts().getAcceptncDtTm() + "]);
} catch(NullPointerException e) {
    out.println("[No ]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls]");
}

// 日付種別 - "Dishonored Return Date"
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry[" + i
          + "]/NtryDtls[1]/TxDtls[0]/RltdDts/Prtry[0]/");
try {
    out.println("Tp[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
          .getNtry().get(i).getNtryDtls().get(1).getTxDtls().get(0)
          .getRltdDts().getPrtry().get(0).getTp() + "]);
} catch(NullPointerException e) {
    out.println("[No Tp]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls]");
}

// 不渡返還日
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry[" + i
          + "]/NtryDtls[1]/TxDtls[0]/RltdDts/Prtry[0]/Dt/");
try {
    out.println("Dt[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
          .getNtry().get(i).getNtryDtls().get(1).getTxDtls().get(0)
          .getRltdDts().getPrtry().get(0).getDt().getDt() + "]);
} catch(NullPointerException e) {
    out.println("[No Dt]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry/NtryDtls/TxDtls]");
}
```


次に JAVA オブジェクトから業務内容および仕向金融機関指示情報を参照する。ここまでが個々の入出金に関連する情報項目として繰り返される。更に、金融機関指示情報を参照する。

```
// 業務内容および仕向金融機関指示情報
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/Ntry[" + i + "]/");
try {
    out.println("AddtlNtryInf[" + doc.getBkToCstmrAcctRpt()
        .getRpt().get(0).getNtry().get(i).getAddtlNtryInf() + "]);
} catch(NullPointerException e) {
    out.println("[No AddtlNtryInf]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt/Ntry]");
}

}

out.println("¥n--  --¥n");

// 金融機関指示情報
out.println("/Document/BkToCstmrAcctRpt/Rpt[0]/");
try {
    out.println("AddtlRptInf[" + doc.getBkToCstmrAcctRpt()
        .getRpt().get(0).getAddtlRptInf() + "]);
} catch(NullPointerException e) {
    out.println("[No AddtlRptInf]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Rpt]");
}

}

}
```

第4章 金融EDI情報作成方法

金融 EDI 情報の項目および記述形式については、原則、利用者や各業界等が任意で決定し利用可能なものとなっている。

ただし、経済産業省・中小企業庁が中心となり、「金融 EDI における商流情報等のあり方検討会議」にて消込の突合に有益と思われる 40 項目が選定・公表されている。さらに、本 40 項目をベースに、全銀 EDI システムより提供される「振込電文 (XML ファイル) 簡易作成機能 (S-ZEDI)」において登録可能な金融 EDI 情報項目 (18 項目) が公表されており、今後、金融 EDI 情報項目のデファクトスタンダード化も進むと推測される。

そのため、ここでは「金融 EDI における商流情報等のあり方検討会議」にて整理された 40 項目について、一つの記述形式案 (国連 CEFAC Remittance Advice メッセージ) を参考とし、商流情報を金融 EDI 情報にマッピングする一方式を解説する。本作成方法を強く推奨するものではないが、作成方法の一例として参考にさせていただきたいと考える。

(1) 商流情報項目

Unstructured 項目の中に入れる商流情報は、経産省が主導して産業界が標準化を図り、以下の 40 項目が選定されている。(付録 1 参照)

○管理上利用する項目

- ・業界区分 (情報項目番号 : UN01005486/UN01005472)
- ・データ区分 (情報項目番号 : UN01005481/UN01005472)

○最低限必要な項目

- ・支払通知番号 (情報項目番号 : UN01008372) (※1)
- ・支払通知発行日 (情報項目番号 : UN01008376) (※1)
- ・請求書番号 (情報項目番号 : UN01005580) (※2)
- ・支払人企業法人コード (情報項目番号 : UN01008795/UN01005756/UN01005757) (※3)

(※1) 支払対象債務・支払日・支払金額・支払方法 (振込か電債か) を通知する文書に付すもの。該当する文書が存在しない場合は記載せず、金融機関側で自動付番 (振込みの際に使われている既存の受付番号等) を利用)。

(※2) 請求書 (ないしそれに類する書類) を発行していない場合は記載不要

(※3) 法人マイナンバーを持たない事業者 (個人事業主等) については記載不要

○IT 化推進による事務合理化に必要と思われる項目

- ・受取人企業法人コード（情報項目番号：UN01008794/UN01005756/UN01005757）
- ・請求先企業名（情報項目番号：UN01008586/UN01005756/UN01005759）
- ・請求先企業法人コード（情報項目番号：UN01008586/UN01005756/UN01005757）
- ・支払金額（明細）（情報項目番号：UN01008478）
- ・税額（情報項目番号：UN01005833）
- ・税区分（情報項目番号：UN01005834）
- ・税率（情報項目番号：UN01005836）

○利用可能とすべき項目

- ・支払番号（情報項目番号：UN01008498）
- ・受取人企業連絡先電話番号（情報項目番号：UN01005860）
- ・支払人企業連絡先電話番号（情報項目番号：UN01005860）
- ・請求先連絡担当者（情報項目番号：UN01005720）
- ・請求先連絡先部門（情報項目番号：UN01005721）
- ・請求先電話番号（情報項目番号：UN01005860）
- ・行番号（情報項目番号：UN01008833/UN01008361/UN01008363）
- ・発注番号（情報項目番号：UN01005580）
- ・受注番号（情報項目番号：UN01005580）
- ・単価（情報項目番号：UN01005792）
- ・数量（情報項目番号：UN01011464）
- ・納入番号（情報項目番号：UN01005627）
- ・納入日時（情報項目番号：UN01005628）
- ・製品コード（情報項目番号：UN01005813）
- ・製品名（情報項目番号：UN01005815）
- ・支払内容（情報項目番号：UN01005560）
- ・契約名（情報項目番号：UN01005589）
- ・締日（情報項目番号：UN01012129）
- ・入金予定日（情報項目番号：UN01012130）
- ・納品伝票番号（情報項目番号：UN01008733/UN01008361/UN01008363）
- ・請求書発行日（情報項目番号：UN01005582）
- ・金額相殺理由コード（UN01011095/UN01011098）
- ・相殺金額（UN01011095/UN01011101）
- ・受取人企業名（情報項目番号：UN01008794/UN01005756/UN01005759）（※4）
- ・支払人企業名（情報項目番号：UN01008795/UN01005756/UN01005759）（※4）
- ・支払合計金額（情報項目番号：UN01008471）（※4）
- ・支払日時（情報項目番号：UN01008500）（※4）

（※4）XML 電文移行対象取引（予定）に、下表に示すように既に代替可能と思われる項目が存在するため、EDI 情報欄への記載不要との整理が可能と考えられる項目。

	今回整理案	XML 電文移行対象取引（代替候補案）		
		総合振込	振込入金通知	入出金取引明細
項目名	受取人企業名	受取人名	口座名	口座名
	支払人企業名	振込依頼人名	振込依頼人名	振込依頼人名
	支払合計金額	振込金額	金額	取引金額
	支払日時	取組日	勘定日、起算日	勘定日、預入・払出日

本プロジェクトにおいては、消込処理における突合項目をまず格納するとともに、請求書から選定された40項目が抽出できるのであれば、必須項目に拘らず当該項目を格納することがよいと考える。突合項目を増やす際に既に格納している項目であれば、変更の影響が低減する。

商流情報は、注文、請求、支払の段階で作成される情報項目を引き継ぎ収集する。

(2) 国連 CEFACT Remittance Advice

国連 CEFACT Remittance Advice は、UN/CEFACT Cross Industry Remittance Advice で支払案内という支払内容を支払先に通知する目的のメッセージで、XML データとして提示されている。

XML データは開始タグや終了タグといった共通の記述規則は決まっているが、どのようなタグを用いているか、またタグ付された項目の意味が何かなどは個別に規定される。特にタグ名などデータの構造に関する規定は、XML データを解析し業務に用いるデータとするために重要で、XML スキーマとして記述される。

この様に多様な XML スキーマが作られていくため、個々の XML データがどの XML スキーマに従って作られているかを機械的に判別できるよう名前（名前空間名）を付与している。XML スキーマに付与されている名前空間名と同じ名前を XML データにも記載されていると、持っている XML スキーマで解析できるかを判別することが可能となる。国連 CEFACT Remittance Advice に付与されている名前空間名を以下に示す。

```
urn:un:unece:uncefact:data:standard:CrossIndustryRemittanceAdvice:13
```

urn は名前の生成ルールの一つを表し、un は国連、unece は国連欧州経済委員会、uncefact は国連 CEFACT を表す。国連欧州経済委員会は国連 CEFACT の上位機関である。data はデータフォーマット、即ちメッセージを表し、他に code（コード）、identifier（識別子）が使われている。CrossIndustryRemittanceAdvice は支払案内、13 は 13 版を表している。

国連 CEFACT 支払案内メッセージの項目と商流情報の項目との相互の対応付け（マッピング）を行い相互変換を可能とする。詳細は、国連 CEFACT 支払案内メッセージ設計書を参照されたい。

(4) 国連 CEFAC 支払案内メッセージの読み込み方

xjc コマンドで Cross Industry Remittance Advice の XML スキーマから生成された関数、読み込んだ XML の検証、金融 EDI 情報の Base64 エンコード、日時データや金額データを作成するためのランタイムルーチン等をインポートする。

最初に出力用ストリームの文字コードを UTF-8 に指定する。更に商流情報をファイルから入力すると仮定し、バッファ読み込みを準備する。

```
import java.io.File;
import java.io.BufferedReader;
import java.io.PrintStream;

import java.nio.file.Paths;
import java.nio.file.Files;
import java.nio.charset.Charset;

import javax.xml.XMLConstants;
import javax.xml.validation.Schema;
import javax.xml.validation.SchemaFactory;

import javax.xml.stream.XMLInputFactory;
import javax.xml.stream.XMLEventReader;
import javax.xml.stream.events.XMLEvent;

import javax.xml.bind.JAXBContext;
import javax.xml.bind.Unmarshaller;
import un.unece.uncefact.data.standard.crossindustryremittanceadvice._11.*;

public class RAIN {
    public static void main(String[] args) {

        try {
            PrintStream out = new PrintStream(System.out, true, "UTF-8");

            // 商流情報ファイル
            String path = "CrossIndustryRemittanceAdvice.xml";
            BufferedReader br = Files.newBufferedReader(
                Paths.get(path), Charset.forName("UTF-8"));
            if(br.markSupported()) {
                // out.println("mark supported");
                br.mark(1024);
            }
        }
    }
}
```

次に JAXP ライブラリを使い、XML の最初の要素(ルート要素)を取得し、その名前空間名を取り出す。名前空間名が国連 CEFACT Remittance Advice (Cross Industry Remittance Advice) の名前空間名と一致したら、JAXB のライブラリを使った読み込みに変更する。併せて XML スキーマ (CrossIndustryRemittanceAdvice.xsd に格納されていることを仮定)を使った検証も行う。

```
String ns = "";
XMLInputFactory factory = XMLInputFactory.newInstance();
XMLStreamReader r = factory.createXMLStreamReader(br);

while(r.hasNext()) {
    XMLEvent e = r.nextEvent();
    if(e.isStartElement()) {
        ns = e.asStartElement().getName().getNamespaceURI();
        out.println("namespace["+ ns +"]");
        break;
    }
}

if(ns ==
"urn:un:unece:uncefact:data:standard:CrossIndustryRemittanceAdvice:11") {

    // Cross Industry Remittance Advice の処理
    if(br.markSupported()) {
        // out.println("mark reset");
        br.reset();
    } else {
        br = Files.newBufferedReader(Paths.get(path),
            Charset.forName("UTF-8"));
    }

    SchemaFactory sFactory = SchemaFactory
        .newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);
    Schema schema = sFactory.newSchema(
        new File("CrossIndustryRemittanceAdvice.xsd"));

    Unmarshaller unmarshaller = JAXBContext.newInstance(
        CrossIndustryRemittanceAdvice.class).createUnmarshaller();
    unmarshaller.setSchema(schema);

    CrossIndustryRemittanceAdvice cira =
        (CrossIndustryRemittanceAdvice) unmarshaller.unmarshal(br);
}
```

次にデータ区分、業界区分、支払通知番号、支払通知発行日を参照する。

```
// データ区分
out.println("CIExchangedDocumentContext/"
    + "BusinessProcessSpecifiedCIDocumentContextParameter/");
try {
    out.println("ID[" + cira.getCIExchangedDocumentContext()
        .getBusinessProcessSpecifiedCIDocumentContextParameter(
)
        .getID().getValue() + "]");
} catch (NullPointerException e) {
    out.println("[No ID]");
}

// 業界区分
out.println("CIExchangedDocumentContext/"
    + "SubsetSpecifiedCIDocumentContextParameter/");
try {
    out.println("ID[" + cira.getCIExchangedDocumentContext()
        .getSubsetSpecifiedCIDocumentContextParameter()
        .getID().getValue() + "]");
} catch (NullPointerException e) {
    out.println("[No ID]");
}

// 支払通知番号
out.println("CIExchangedDocument/");
try {
    out.println("ID[" + cira.getCIExchangedDocument()
        .getID().getValue() + "]");
} catch (NullPointerException e) {
    out.println("[No ID]");
}

// 支払通知発行日
out.println("CIExchangedDocument/");
try {
    out.println("IssueDateTime/DateTime[" + cira
        .getCIExchangedDocument().getIssueDateTime()
        .getDateTime() + "]");
} catch (NullPointerException e) {
    out.println("[No IssueDateTime/DateTime]");
}
```

(2) コード一覧

① 業界区分

識別子データ型	Identifier. Type		
コード表 ID	Identification Scheme. Identifier	JEC0001	国連 CEFACT 日本委員会で管理する ID
コード表名	Identification Scheme. Name. Text	Domain code	業界区分コード表
管理機関 ID	Identification Scheme Agency. Identifier	413	国連 CEFACT にて附番された ID
管理機関名	Identification Scheme. Agency Name. Text	JP, JEC	国連 CEFACT 日本委員会
コード表版 ID	Identification Scheme. Version. Identifier	JPS2018A	
コード値	名称	説明	
JPSBASE	業界横断 EDI 基本	業界横断 EDI の基本となるメッセージで、各業務領域メッセージ設計における参照モデルとして利用する。	
JPSLGPC	自治体消耗品購買	豊田市役所一般購買の EDI 実証実験で定義。	
JPSSMED	中小企業共通 EDI	中小企業共通 EDI 標準であり、基本、製造業、商社購買およびプロジェクト取引を含む。	
JPSFEDI	金融 EDI	金流商流情報連携で使われる金融関連メッセージ。	
JPSSCDL	スケジューリング SCM	需要予測／納入指示の組み合わせによるジャストインタイム方式 EDI。	
JPSTEDI	貿易手続円滑化	貿易手続関連 EDI。	

② データ区分

識別子 データ 型	Identifier. Type		
コード表 ID	Identification Scheme. Identifier	JEC0002	国連 CEFACT 日本委員会で管理する ID
管理機関 ID	Identification Scheme. Name. Text	Domain code	プロセスタイプ区分コード (金融 EDI データ区分)
管理機関名	Identification Scheme Agency. Identifier	413	国連 CEFACT にて附番された ID
コード表名	Identification Scheme. Agency Name. Text	JP, JEC	国連 CEFACT 日本委員会
コード表版 ID	Identification Scheme. Version. Identifier	JPS2018A	
コード値	名称	説明	
01	基本	業界横断 EDI において振込に添付する基本的商流情報として利用する。	
02	拡張	業界横断 EDI において振込に添付する商流情報に振込に関する金融情報を含める拡張して利用する。	

③ 金額相殺項目理由コード

識別子 データ型	Code. Type		
コード表 ID	Code List. Identifier	JEC0003	JEC 版コード
管理機関 ID	Code List. Agency. Identifier	413	国連 CEFACT 日本委員会
管理機関名	Code List. Agency. Name	JEC	国連 CEFACT 日本委員会
コード表名	Code List. Name. Text	Balance out	金額相殺理由コード

		reason code	
コード表版 ID	Code List. Version. Identifier	JPS2018A	
コード値	名称	説明	
001		振込手数料	
002		倉庫利用料	
003		報奨金	
004		直売部品	
005		棚卸資産譲渡	
006		固定資産賃貸料	
007		ネットワーク利用料	
008		データ提供料	
009		研修受講料	
010		前月違算相殺	

④ 税タイプコード(税区分)

識別子 データ 型	Code. Type		
コード表 ID	Code List. Identifier	5153	国連 CEFACT で管理
管理機関 ID	Code List. Agency. Identifier	6	UN/ECE
管理機関名	Code List. Agency. Name	UN/ECE	国連欧州経済委員会
コード表名	Code List. Name. Text	Duty or tax or fee type name code	税タイプコード
コード表版 ID	Code List. Version. Identifier	JPS2018 A	JEC 版サブセット
コード値	名称	説明	
AAA	Petroleum tax	石油税	
AAD	Tobacco tax	タバコ税	
AAE	Energy fee	エネルギー税	
AAF	Coffee tax	コーヒー税	
AAK	Mineral oil tax	鉱物石油税	
AAL	Special tax	特別税	
AAM	Insurance tax	保険税	
ADD	Anti-dumping duty	反ダンピング税	
CAP	Agricultural levy	農業徴収税	

CAR	Car tax	自動車税
CST	Commodity specific tax	消費特別税(軽減税率適用)
CUD	Customs duty	関税
CVD	Countervailing duty	相殺関税
ENV	Environmental tax	環境税
EXC	Excise duty	消費税
EXP	Agricultural export rebate	農業輸出リベート
FET	Federal excise tax	連邦消費税
FRE	Free	無税
GCN	General construction tax	建設税
GST	Goods and services tax	物品サービス税
ILL	Illuminants tax	光源税
IMP	Import tax	輸入税
IND	Individual tax	個別税
LAC	Business license fee	事業免許税
LCN	Local construction tax	建設地方税
LOC	Local sales tax	地方販売税
OTH	Other taxes	その他税

⑤ 書類種別コード

識別子データ型	Code. Type		
コード表 ID	Code List. Identifier	1001	国連 CEFACT で管理
管理機関 ID	Code List. Agency. Identifier	6	UN/ECE
管理機関名	Code List. Agency. Name	UN/ECE	国連欧州経済委員会
コード表名	Code List. Name. Text	Document name code	文書種別コード
コード表版 ID	Code List. Version. Identifier	JPS2018A	JEC 版サブセット
コード値	名称	説明	
35	Inventory report	在庫報告書	
105	Purchase Order	注文書	
231	Purchase Order Response	注文回答	
270	Delivery note	納品書	

312	Acknowledgement message	了解確認文書
315	Contract	契約書
351	Despatch advice	出荷通知書
380	Commercial invoice	請求書
481	Remittance advice	支払通知書

Remittance Advice には使われていないが、既に定義されているコードについては、「共通コード表 V4_20180702.xlsx」を参照のこと。

(3) 取引ごとに識別に使われるコード

① コード表が登録されている場合

企業コード、製品コードや法人番号など、コード表が存在し、かつ多数の組織間で共有することが望ましい場合は、当該コード表の管理機関を登録し、コード表を公開することが望ましい。

管理機関コードの登録やコード表の設定は、「5. 業界区分ごとに管理されるコード表」に準ずる。

法人番号については、管理機関(国税庁)が、コード表 ID=3055、管理機関 ID=402、管理機関名称=JP, National Tax Agency で国連 CEFACT に登録済である。

管理機関 ID	402	管理機関名称	国税庁(法人番号発番機関)
コード表 ID	名称	説明	
houjin-bango	houjin-bango	法人番号	

② コード表が登録されていない場合

発注番号、受注番号、納品伝票番号、請求書番号、支払通知番号や明細行番号は、取引自称や取引文書を一意的に識別するものであり、附番者により任意に設定される。通常、コード表は持たないため登録されない。

また相対の企業間程度でしか扱うことを想定していないコード表などは登録されていない場合もある。

「発注番号」など種別がその他 ID スキーマ識別値表にある場合、コード表 ID は以下の様に設定できる。但しコード表版 ID は設定されない。

コード表 ID: ZZZZaaaa: xxxxxxxxxxxxxx

ZZZZaaaa はその他 ID スキーマ識別値

xxxxxxxxxxxxxx は附番者の法人番号

例えば、サプライチェーン情報基盤研究会(法人番号:9010005023375)の発注番号(その他 ID スキーマ識別値:ZZZZ0105)の場合、コード表 ID は以下の様になる。

ZZZZ0105: 9010005023375

但し、附番者の法人番号が不明の場合は、コロン以降は付加しない。

コード表 ID		コード表名称	その他 ID スキーマ識別値
管理機関 ID	413	管理機関名称	国連 CEFAC 日本委員会
コード値	名称	説明	
ZZZZ0481	支払通知番号		
ZZZZ0450	支払番号		
ZZZZ001	行番号		
ZZZZ0380	請求書番号		
ZZZZ0270	納品伝票番号		
ZZZZ0105	発注番号		
ZZZZA105	受注番号		
ZZZZ0270	納入番号		
ZZZZ002	製品コード		

SE のための企業側 ISO20022 メッセージ導入ガイド 第 0.2 版

2018 年 3 月 23 日 第 0.1 版発行

2018 年 x 月 xx 日 第 0.2 版発行

発行者 株式会社 NTT データ
