

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。

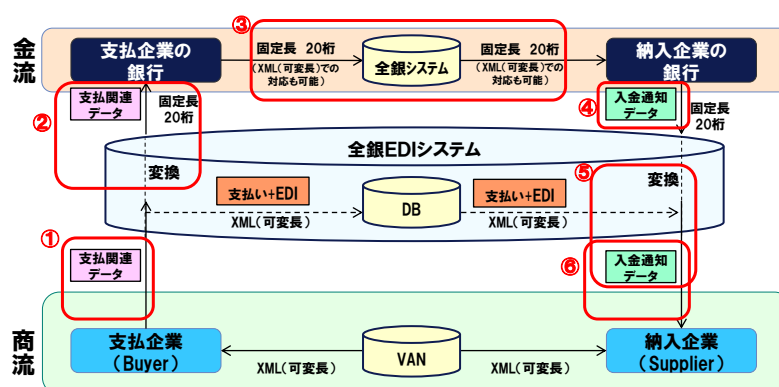


図 1 金流商流連携の流れ

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。

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を記載	
		Document	[1..1]	(タグを生成)	-	-	メッセージのDocumentと指定	※ <Document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:sd="http://www.w3.org/2001/XMLSchema-instance" uri="iso:std:iso:20022:tech:xsd:pain.001.001.03">	M
		CustomerCreditTransferInitiation	[1..1]	(タグを生成)	-	-			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		M
		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。

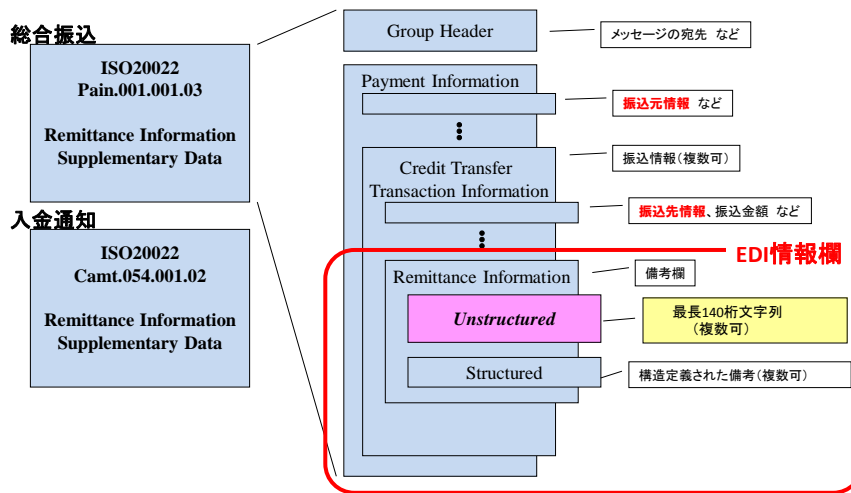
なお、上記は Pain の例であるが、Camt（振込入金通知、入出金明細に関する XML 電文）についても同様に項目の対応付け（マッピング）が必要となる。

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

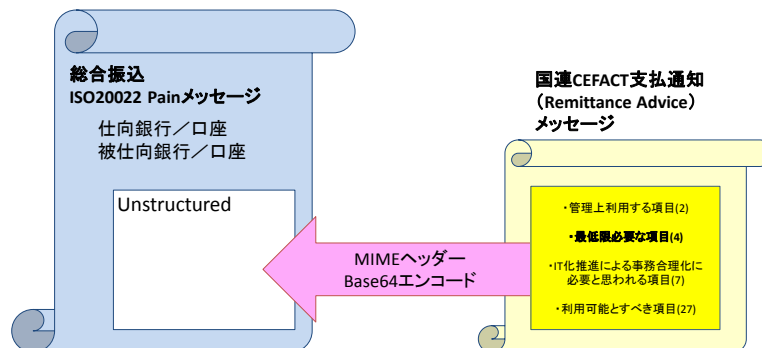
Unstructured 項目は 140 文字を記述でき、また繰り返し記述することが可能である。しかし、XML のタグを記述することは許されていない。このため XML 形式で記述されることが多い大きな商流情報は、そのまま Unstructured 項目に入れることが出来ない。

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> 支払通知番号
 <IssueDateTime>2017-01-23T00:00:00</IssueDateTime>
 </CIReExchangedDocumentContext>
 <CIIRHTradeSettlementPayment>
 <SpecifiedCIIRHSupplyChainTradeSettlement>
 <PayerCITradeParty>
 <ID>9010601021385</ID> 支払人企業法人コード
 </PayerCITradeParty>
 </SpecifiedCIIRHSupplyChainTradeSettlement>
 </CIIRHTradeSettlementPayment>
 <CIIRTSupplyChainTradeTransaction>
 <AssociatedCIReferencedDocument>
 <IssuerAssignedID>I-H27-12-1</IssuerAssignedID>
 </AssociatedCIReferencedDocument>
 </CIIRTSupplyChainTradeTransaction>
 <CIIRTSupplyChainTradeTransaction>
 <AssociatedCIReferencedDocument>
 <IssuerAssignedID>I-H27-12-2</IssuerAssignedID>
 </AssociatedCIReferencedDocument>
 </CIIRTSupplyChainTradeTransaction>
</CrossIndustryRemittanceAdvice>

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/PENyb3NzSW5kdXN0cn1SZW1pdHRhbmN1QWR2aWN1IHhtbG5zPSJjcm46dW46dW51Y2U6dW5j
 ZWZhY3Q6ZGF0YTpdzGfuZGFyZDpDcm9zc0luZHVhZdHJ5UmVtaXR0YW5jZUFkdmljZT0xMSI+PENJ
 UkY4Y2hhbmdlZERyY3VtZW50Q29udGV4dD48SUQ+U11MjgtMTM8L01EPjxJc3N1ZURhdGVuYW11
 PjIwMTctMDEtMjNUMDA6MDA6MDA4SL01zc3V1RGFOZVRpbWU+PC9DSVJfEgNoY5WnZWREb2N1bWVu
 dEbnbnR1eHQ+PENJUKhlcFkFVN1dHRsZW1lbnRQYX1tZW50PjxtcGVjaWZpZWRSVJ1U3VwcGx5
 Q2hhaw5UcmFkFzV1dHRsZW1lbnQ+PFbHeWVyQ01UcmFkFzBhcnR5PjxJRd45MDEwNjAxdMlXmZGT
 PC9JRd48L1BheWVyQ01UcmFkFzBhcnR5PjwvU3B1Y21maWVvQ01SSFN1cHBseUNoYW1uVHJhZGVt
 ZXRobGVtZW50PjwvQ01SSFRyYWR1U2V0dGx1bWVudFBheW1lbnQ+PENJ1URtdXBwbH1daGFPb1Ry
 YWR1VHJhbnR5J3Rpb24+PEFzc29jaWFOZWRDSVJ1ZmVyZW5jZWREb2N1bWVudD48SXNzZWVvQXNz
 aWduZW9JRd45JLUgYX0xMi0xPC9Jc3N1ZXJCb2N251Z1EPjwvQXNzZW50PjxJR1ZENjUmVmZXJ1
 bmn1ZERyY3VtZW50PjwvQ01SVFN1cHBseUNoYW1uVHJhZGVuUcmFuc2FjdG1vb3J48Q01SVFN1cHBs
 eUNoYW1uVHJhZGVuUcmFuc2FjdG1vb3J48QXNzZW50PjxJR1ZENjUmVmZXJ1bmn1ZERyY3VtZW50PjxJ
 c3N1ZXJ1Z1EPjxJ1Z1EPkktSDI3LTYeY1tL8L01zc3VlcFkFzN1bmtK5SUQPC9Cbc3N1Y21hdGVk
 Q01SZWZlcmVtY2VkrG9jdW1lbnQ+PC9DSVJU3VwcGx5Q2hhaw5UcmFkFzVryY5W5YWN0aW9uPjwv
 Q3Jvc3NjbmR1c3RyeVJlbnw1dGfuY2VbZHZpY2U+

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 7 6 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>PjIwTctMDEtMjNUMDA6MDA6MDA8L01zc3V1RGFOZVRpbWU+PC9DSVJFeGN0YW5nZWREb2N1bWVu</Ustrd>
<Ustrd>dENvbnR1eHQ+PENJUkhUcmFkZVN1dHRsZW11bnRQYX1tZW50PjxTcGVjaWZpZWRDSVJlU3VwcGx5</Ustrd>
<Ustrd>Q2hhaW5UcmFkZVN1dHRsZW11bnQ+PFBheWVyQ01UcmFkZVBhcnR5PjxJRd45MDEwNjAxMDIxMzg1</Ustrd>
<Ustrd>PC9JRd48L1BheWVyQ01UcmFkZVBhcnR5PjwvU3B1Y2lmaWVhQ01SSFN1cHBseUNoYW1uVHJhZGVt</Ustrd>
<Ustrd>ZXR0bGVtZW50PjwvQ01SSFRyYWR1U2V0dGx1bWVudFBheW11bnQ+PENJU1RTdXBwbH1DaGFpb1Ry</Ustrd>
<Ustrd>YWR1VHJhbnNhY3Rpb24+PEFzc29jaWFOZWRDSVJlZmVyZW5jZWREb2N1bWVuZD48SXNzdWVyQXNz</Ustrd>
<Ustrd>aWduZWRJRd5JlUgyNy0xMi0xPC9Jc3N1ZXJBe3NpZ251ZE1EPjwvQXNzb2NpYXR1ZENJUmlmZXJl</Ustrd>
<Ustrd>bmN1ZERvY3VtZW50PjwvQ01SVFN1cHBseUNoYW1uVHJhZGVUcmFuc2FjdG1vb2NpYXR1ZENJUmlmZXJlbnN1ZERvY3VtZW50PjxJ</Ustrd>
<Ustrd>eUNoYW1uVHJhZGVUcmFuc2FjdG1vb2NpYXR1ZENJUmlmZXJlbnN1ZERvY3VtZW50PjxJ</Ustrd>
<Ustrd>c3N1ZXJBe3NpZ251ZE1EPkktSDI3LTlEYlTI8L01zc3V1ckFzc2lmbWVhZSUQ+PC9Bc3NvY2lhdGVk</Ustrd>
<Ustrd>Q01SZWZ1cmVudGVkRG9jdW11bnQ+PC9DSVJlU3VwcGx5Q2hhaW5UcmFkZVRyYW5zYWN0aW9uPjwv</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
aaamt.setValue(new BigDecimal(12345));
aaamt.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 BranchAndFinancialInstitutionIdentification40();
ctti.setCdtrAgt(bfii2);
FinancialInstitutionIdentification7 fii2
    = new FinancialInstitutionIdentification70();
bfii2.setFinInstnId(fii2);
ClearingSystemMemberIdentification2 csmi2
    = new ClearingSystemMemberIdentification20();
fii2.setClrSysMmbId(csmi2);
ClearingSystemIdentification2Choice csid2
    = new ClearingSystemIdentification2Choice0();
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 GenericFinancialIdentification10();
fii2.setOthr(gfi);
gfi.setId("1111");

// 被仕向支店情報
BranchData2 bd2 = new BranchData20();
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;
    }
}
```


(2) 総合振込結果明細 (ISO20022 Pain.002) の読み方

まず `xjc` コマンドで `Pain.002.001.03` 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.pain_002_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 Pain002r {
    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;
        }
    }
}
```

Pain.002.001.03 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 Pain.002 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("pain.002.001.03.xml");
try {
    SchemaFactory sFactory1
        = SchemaFactory.newInstance
            (XMLConstants.W3C_XML_SCHEMA_NS_URI);
    Schema schema1
        = sFactory1.newSchema(new File("pain.002.001.03.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/CstmrPmtStsRpt/GrpHdr/");
try {
    out.println("MsgId[" + doc.getCstmrPmtStsRpt().getGrpHdr().getMsgId()
        + "]");
} catch (NullPointerException e) {
    out.println("[No MsgId]");
}

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

```

次に JAVA オブジェクトから総合振込依頼に記述されたグループメッセージ ID とグループメッセージ名称 ID、総合振込依頼に記述された XML ファイル作成日時と支払情報数を参照する。

```
// グループメッセージ ID (総合振込依頼)
out.println("/Document/CstmrPmtStsRpt/OrgnlGrpInfAndSts/");
try {
    out.println("OrgnlMsgId["
        + doc.getCstmrPmtStsRpt().getOrgnlGrpInfAndSts()
        .getOrgnlMsgId() + "]);
} catch(NullPointerException e) {
    out.println("[No OrgnlMsgId]");
}

// グループメッセージ名称 ID - "pain.001"
out.println("/Document/CstmrPmtStsRpt/OrgnlGrpInfAndSts/");
try {
    out.println("OrgnlMsgNmId["
        + doc.getCstmrPmtStsRpt().getOrgnlGrpInfAndSts()
        .getOrgnlMsgNmId() + "]);
} catch(NullPointerException e) {
    out.println("[No OrgnlMsgNmId]");
}

// XML ファイル作成日時 (総合振込依頼)
out.println("/Document/CstmrPmtStsRpt/OrgnlGrpInfAndSts/");
try {
    out.println("OrgnlCreDtTm["
        + doc.getCstmrPmtStsRpt().getOrgnlGrpInfAndSts()
        .getOrgnlCreDtTm() + "]);
} catch(NullPointerException e) {
    out.println("[No OrgnlCreDtTm]");
}

// 支払情報数 (総合振込依頼)
out.println("/Document/CstmrPmtStsRpt/OrgnlGrpInfAndSts/");
try {
    out.println("OrgnlNbOfTxs["
        + doc.getCstmrPmtStsRpt().getOrgnlGrpInfAndSts()
        .getOrgnlNbOfTxs() + "]);
} catch(NullPointerException e) {
    out.println("[No OrgnlNbOfTxs]");
}
```

次に JAVA オブジェクトから全体処理結果、総合振込依頼に記述された支払情報 ID と合計件数を参照する。StsRsnInfタグ、OrgnlPmtInfAndStsタグは複数記述可能で List 型で取得される。このためインデックス参照の get メソッドを使い、当該要素が無い場合は IndexOutOfBoundsException の例外が発生することに注意されたい。

```
// 全体処理結果
out.println
    ("/Document/CstmrPmtStsRpt/OrgnlGrpInfAndSts/StsRsnInf[0]/Rsn/");
try {
    out.println("OrgnlNbOfTxt["
        + doc.getCstmrPmtStsRpt().getOrgnlGrpInfAndSts()
        .getStsRsnInf().get(0).getRsn().getPrtry() + "]");
} catch (NullPointerException e) {
    out.println("[No OrgnlNbOfTxt]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No StsRsnInf]");
}

// 支払情報 ID (総合振込依頼)
out.println("/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/");
try {
    out.println("OrgnlPmtInfID["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
        .getOrgnlPmtInfId() + "]");
} catch (NullPointerException e) {
    out.println("[No OrgnlPmtInfID]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts]");
}

// 合計件数 (総合振込依頼)
out.println("/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/");
try {
    out.println("OrgnlNbOfTxS["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
        .getOrgnlNbOfTxS() + "]");
} catch (NullPointerException e) {
    out.println("[No OrgnlNbOfTxS]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts]");
}
```

次に JAVA オブジェクトから複数の TxInfAnsSts タグに含まれる EDI キー情報、総合振込依頼に記述された振込依頼人発行する取引明細識別番号を参照する。そのための for ループを用いている。

```

if(doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().size() <= 0) {
    out.println("-- No OrgnlPmtInfAndSts --");
    return;
}

for(int i = 0;
    i < doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
        .getTxInfAndSts().size(); i++) {

    out.println("¥n--" + i + "--¥n");

    // EDI キー情報
    out.println(
        "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
            + i + "]/");
    try {
        out.println("OrgnlInstrId["
            + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
                .getTxInfAndSts().get(0).getOrgnlInstrId() + "]);
    } catch(NullPointerException e) {
        out.println("[No OrgnlInstrId]");
    } catch(IndexOutOfBoundsException e) {
        out.println("[No OrgnlPmtInfAndSts/TxInfAndSts]");
    }

    // 取引明細識別番号(振込依頼人発行)(総合振込依頼)
    out.println(
        "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
            + i + "]/");
    try {
        out.println("OrgnlEndToEndId["
            + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
                .getTxInfAndSts().get(0).getOrgnlEndToEndId() + "]);
    } catch(NullPointerException e) {
        out.println("[No OrgnlEndToEndId]");
    } catch(IndexOutOfBoundsException e) {
        out.println("[No OrgnlPmtInfAndSts/TxInfAndSts]");
    }
}

```

次に JAVA オブジェクトから複数の TxInfAnsSts タグに含まれる取引明細別処理結果、識別情報および仕向記入期間指示情報を参照する。

```
// 取引明細別処理結果
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]/StsRsnInf/Rsn/");
try {
    out.println("Prtry["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
            .getTxInfAndSts().get(0).getStsRsnInf().get(0)
            .getRsn().getPrtry() + "]");
} catch (NullPointerException e) {
    out.println("[No Prtry]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts/StsRsnInf]");
}

// 識別情報および仕向記入期間指示情報
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]/StsRsnInf/");
try {
    out.println("AddtlInf["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
            .getTxInfAndSts().get(0).getStsRsnInf().get(0)
            .getAddtlInf() + "]");
} catch (NullPointerException e) {
    out.println("[No AddtlInf]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts/StsRsnInf]");
}
```

次に JAVA オブジェクトから複数の TxInfAnsSts タグに含まれる総合振込依頼に記述された取引明細情報と取組日を参照する。

```
// 取引明細情報(総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]/OrgnlTxRef/Amt/");
try {
    out.println("InstdAmt["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
        .getTxInfAndSts().get(0).getOrgnlTxRef().getAmt()
        .getInstdAmt().getValue() + "]);
    out.println("Ccy["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
        .getTxInfAndSts().get(0).getOrgnlTxRef().getAmt()
        .getInstdAmt().getCcy() + "]);
} catch(NullPointerException e) {
    out.println("[No InstdAmt]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts]");
}

// 取組日(総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]/OrgnlTxRef/");
try {
    out.println("ReqdExctnDt["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
        .getTxInfAndSts().get(0).getOrgnlTxRef().getReqdExctnDt()
        + "]);
} catch(NullPointerException e) {
    out.println("[No ReqdExctnDt]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts]");
}
```

次に JAVA オブジェクトから複数の TxInfAnsSts タグに含まれる総合振込依頼に記述された新規コードと振込指定区分を参照する。

```
// 新規コード(総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]/OrgnlTxRef/PmtTpInf/SvcLvl/");
try {
    out.println("Prtry["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
            .getTxInfAndSts().get(0).getOrgnlTxRef().getPmtTpInf()
            .getSvcLvl().getPrtry() + "]);
} catch(NullPointerException e) {
    out.println("[No Prtry]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts]");
}

// 振込指定区分(総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]/OrgnlTxRef/PmtTpInf/LclInstrm/");
try {
    out.println("Prtry["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
            .getTxInfAndSts().get(0).getOrgnlTxRef().getPmtTpInf()
            .getLclInstrm().getPrtry() + "]);
} catch(NullPointerException e) {
    out.println("[No Prtry]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts]");
}
```


次に JAVA オブジェクトから複数の TxInfAnsSts タグに含まれる総合振込依頼に記述された識別コードと支払方法を参照する。

```
// 識別コード(総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]/OrgnlTxRef/PmtTpInf/CtgyPurp/");
try {
    out.println("Cd["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
            .getTxInfAndSts().get(0).getOrgnlTxRef().getPmtTpInf()
            .getCtgyPurp().getCd() + "]);
} catch(NullPointerException e) {
    out.println("[No Cd]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts]");
}

// 支払方法(総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]/OrgnlTxRef/");
try {
    out.println("PmtMtd["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
            .getTxInfAndSts().get(0).getOrgnlTxRef().getPmtMtd() + "]);
} catch(NullPointerException e) {
    out.println("[No PmtMtd]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts]");
}
```

次に JAVA オブジェクトから複数の TxInfAnsSts タグに含まれる金融 EDI 情報を参照し、更に Ustrds タグへの格納が XML か平文か判定する。

```
// 金融 EDI 情報 (総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]/OrgnlTxRef/RmtInf/");
try {
    out.println("Ustrd["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
        .getTxInfAndSts().get(0).getOrgnlTxRef().getRmtInf()
        .getUstrd().get(0) + "]");

    List<String> ustrds
        = doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
        .getTxInfAndSts().get(0).getOrgnlTxRef()
        .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) + "]");
```

XML の場合は MIME ヘッダーの確認と Base64 のデコードを行う。なお、if-else の単純なネストとならなかったためインデントが異なることに注意されたい。

```

        } 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) + "]");
                } 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]");
        }
    }

```

次に JAVA オブジェクトから複数の TxInfAnsSts タグに含まれる総合振込依頼に記述された振込依頼人名と取引企業コードである振込依頼人コードを参照する。

```
// 振込依頼人名(総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]/OrgnlTxRef/UltmtDbtr/");
try {
    out.println("Nm["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
            .getTxInfAndSts().get(0).getOrgnlTxRef()
            .getUltmtDbtr().getNm() + "]);
} catch(NullPointerException e) {
    out.println("[No Nm]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts]");
}

// 振込依頼人コード(取引企業コード)(総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]/OrgnlTxRef/Dbtr/Id/OrgId/Othr[0]/");
try {
    out.println("Id["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
            .getTxInfAndSts().get(0).getOrgnlTxRef().getDbtr().getId()
            .getOrgId().getOthr().get(0).getId() + "]);
} catch(NullPointerException e) {
    out.println("[No Id]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts/Othr]");
}
```

次に JAVA オブジェクトから複数の TxInfAnsSts タグに含まれる総合振込依頼に記述された振込依頼人概要コードと法人マイナンバーとも呼ばれる振込依頼人法人番号を参照する。

```
// 振込依頼人概要コード(総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]/OrgnlTxRef/Dbtr/Id/OrgId/Othr[0]/SchmeNm/");
try {
    out.println("Cd["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
            .getTxInfAndSts().get(0).getOrgnlTxRef().getDbtr().getId()
            .getOrgId().getOthr().get(0).getSchmeNm().getCd() + "]);
} catch(NullPointerException e) {
    out.println("[No Cd]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts/Othr]");
}

// 振込依頼人法人番号(法人マイナンバー)(総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]/OrgnlTxRef/Dbtr/Id/OrgId/Othr[1]/");
try {
    out.println("Id["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
            .getTxInfAndSts().get(0).getOrgnlTxRef().getDbtr().getId()
            .getOrgId().getOthr().get(1).getId() + "]);
} catch(NullPointerException e) {
    out.println("[No Id]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts/Othr]");
}
```

次に JAVA オブジェクトから複数の TxInfAnsSts タグに含まれる総合振込依頼に記述された法人マイナンバーとも呼ばれる振込依頼人法人番号の概要コードと振込依頼人口座番号を参照する。

```
// 振込依頼人法人番号(法人マイナンバー)概要コード(総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]/OrgnlTxRef/Dbtr/Id/OrgId/Othr[1]/SchmeNm/");
try {
    out.println("Cd["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
            .getTxInfAndSts().get(0).getOrgnlTxRef().getDbtr().getId()
            .getOrgId().getOthr().get(1).getSchmeNm().getCd() + "]);
} catch(NullPointerException e) {
    out.println("[No Cd]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts/Othr]");
}

// 振込依頼人口座番号(総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]/OrgnlTxRef/DbtrAcct/Id/Othr/");
try {
    out.println("Id["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
            .getTxInfAndSts().get(0).getOrgnlTxRef().getDbtrAcct().getId()
            .getOthr().getId() + "]);
} catch(NullPointerException e) {
    out.println("[No Id]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts]");
}
```

次に JAVA オブジェクトから複数の TxInfAnsSts タグに含まれる総合振込依頼に記述された振込依頼人預金種目と仕向決済システム識別コードを参照する。

```
// 振込依頼人預金種目(総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]/OrgnlTxRef/DbtrAcct/Tp/");
try {
    out.println("Prtry["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
        .getTxInfAndSts().get(0).getOrgnlTxRef().getDbtrAcct()
        .getTp().getPrtry() + "]);
} catch(NullPointerException e) {
    out.println("[No Prtry]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts]");
}

// 仕向決済システム識別コード(総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]"
        + "/OrgnlTxRef/DbtrAgt/FinInstnId/ClrSysMmbId/ClrSysId/");
try {
    out.println("Cd["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
        .getTxInfAndSts().get(0).getOrgnlTxRef().getDbtrAgt()
        .getFinInstnId().getClrSysMmbId().getClrSysId().getCd() + "]);
} catch(NullPointerException e) {
    out.println("[No Cd]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts]");
}
```

次に JAVA オブジェクトから複数の TxInfAnsSts タグに含まれる総合振込依頼に記述された仕向銀行番号と仕向銀行名を参照する。

```
// 仕向銀行番号(総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]" + "/OrgnlTxRef/DbtrAgt/FinInstnId/ClrSysMmbId/");
try {
    out.println("MmbId["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
            .getTxInfAndSts().get(0).getOrgnlTxRef().getDbtrAgt()
            .getFinInstnId().getClrSysMmbId().getMmbId() + "]");
} catch(NullPointerException e) {
    out.println("[No MmbId]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts]");
}

// 仕向銀行名(総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]" + "/OrgnlTxRef/DbtrAgt/FinInstnId/");
try {
    out.println("Nm["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
            .getTxInfAndSts().get(0).getOrgnlTxRef().getDbtrAgt()
            .getFinInstnId().getNm() + "]");
} catch(NullPointerException e) {
    out.println("[No Nm]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts]");
}
```


次に JAVA オブジェクトから複数の TxInfAnsSts タグに含まれる総合振込依頼に記述された仕向支店名と被仕向銀行番号を参照する。

```
// 仕向支店名 (総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]" + "/OrgnlTxRef/DbtrAgt/BrnchId/");
try {
    out.println("Id["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
        .getTxInfAndSts().get(0).getOrgnlTxRef().getDbtrAgt()
        .getBrnchId().getNm() + "]");
} catch(NullPointerException e) {
    out.println("[No Nm]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts]");
}

// 被仕向銀行番号 (総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]" + "/OrgnlTxRef/CdtrAgt/FinInstnId/ClrSysMmbId/");
try {
    out.println("MmbId["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
        .getTxInfAndSts().get(0).getOrgnlTxRef().getCdtrAgt()
        .getFinInstnId().getClrSysMmbId().getMmbId() + "]");
} catch(NullPointerException e) {
    out.println("[No MmbId]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts]");
}
```

次に JAVA オブジェクトから複数の TxInfAnsSts タグに含まれる総合振込依頼に記述された被仕向銀行名と手形交換所番号を参照する。

```
// 被仕向銀行名(総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]" + "/OrgnlTxRef/CdtrAgt/FinInstnId/");
try {
    out.println("Nm["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
            .getTxInfAndSts().get(0).getOrgnlTxRef().getCdtrAgt()
            .getFinInstnId().getNm() + "]");
} catch(NullPointerException e) {
    out.println("[No Nm]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts]");
}

// 手形交換所番号(総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]" + "/OrgnlTxRef/CdtrAgt/FinInstnId/Othr/");
try {
    out.println("Id["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
            .getTxInfAndSts().get(0).getOrgnlTxRef().getCdtrAgt()
            .getFinInstnId().getOthr().getId() + "]");
} catch(NullPointerException e) {
    out.println("[No Id]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts]");
}
```

次に JAVA オブジェクトから複数の TxInfAnsSts タグに含まれる総合振込依頼に記述された被仕向支店番号と被仕向支店名を参照する。

```
// 被仕向支店番号(総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]" + "/OrgnlTxRef/CdtrAgt/BrnchId/");
try {
    out.println("Id["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
        .getTxInfAndSts().get(0).getOrgnlTxRef().getCdtrAgt()
        .getBrnchId().getId() + "]");
} catch(NullPointerException e) {
    out.println("[No Id]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts]");
}

// 被仕向支店名(総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]" + "/OrgnlTxRef/CdtrAgt/BrnchId/");
try {
    out.println("Id["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
        .getTxInfAndSts().get(0).getOrgnlTxRef().getCdtrAgt()
        .getBrnchId().getNm() + "]");
} catch(NullPointerException e) {
    out.println("[No Nm]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts]");
}
```

次に JAVA オブジェクトから複数の TxInfAnsSts タグに含まれる総合振込依頼に記述された受取人名と法人マイナンバーとも呼ばれる受取人法人番号を参照する。

```
// 受取人名 (総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]/OrgnlTxRef/Cdtr/");
try {
    out.println("Nm["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
            .getTxInfAndSts().get(0).getOrgnlTxRef()
            .getCdtr().getNm() + "]);
} catch (NullPointerException e) {
    out.println("[No Nm]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts]");
}

// 受取人法人番号 (法人マイナンバー) (総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]/OrgnlTxRef/Cdtr/Id/OrgId/Othr[0]/");
try {
    out.println("Id["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
            .getTxInfAndSts().get(0).getOrgnlTxRef().getCdtr().getId()
            .getOrgId().getOthr().get(0).getId() + "]);
} catch (NullPointerException e) {
    out.println("[No Id]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts/Othr]");
}
```

次に JAVA オブジェクトから複数の TxInfAnsSts タグに含まれる総合振込依頼に記述された法人マイナンバーとも呼ばれる受取人法人番号の概要コードと顧客コード 1 を参照する。

```
// 受取人法人番号(法人マイナンバー) 概要コード(総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]/OrgnlTxRef/Cdtr/Id/OrgId/Othr[0]/SchmeNm/");
try {
    out.println("Cd["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
            .getTxInfAndSts().get(0).getOrgnlTxRef().getCdtr().getId()
            .getOrgId().getOthr().get(0).getSchmeNm().getCd() + "]);
} catch (NullPointerException e) {
    out.println("[No Cd]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts/Othr]");
}

// 顧客コード 1 (総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]/OrgnlTxRef/Cdtr/Id/OrgId/Othr[1]/");
try {
    out.println("Id["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
            .getTxInfAndSts().get(0).getOrgnlTxRef().getCdtr().getId()
            .getOrgId().getOthr().get(1).getId() + "]);
} catch (NullPointerException e) {
    out.println("[No Id]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts/Othr]");
}
```

次に JAVA オブジェクトから複数の TxInfAnsSts タグに含まれる総合振込依頼に記述された顧客コード 1 概要名と顧客コード 2 を参照する。

```
// 顧客コード 1 概要名 (総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]/OrgnlTxRef/Cdtr/Id/OrgId/Othr[1]/SchmeNm/");
try {
    out.println("Prtry["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
            .getTxInfAndSts().get(0).getOrgnlTxRef().getCdtr().getId()
            .getOrgId().getOthr().get(1).getSchmeNm().getPrtry() + "]");
} catch (NullPointerException e) {
    out.println("[No Prtry]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts/Othr]");
}

// 顧客コード 2 (総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]/OrgnlTxRef/Cdtr/Id/OrgId/Othr[2]/");
try {
    out.println("Id["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
            .getTxInfAndSts().get(0).getOrgnlTxRef().getCdtr().getId()
            .getOrgId().getOthr().get(2).getId() + "]");
} catch (NullPointerException e) {
    out.println("[No Id]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts/Othr]");
}
```

次に JAVA オブジェクトから複数の TxInfAnsSts タグに含まれる総合振込依頼に記述された顧客コード 2 概要名と受取人口座番号を参照する。

```
// 顧客コード 2 概要名 (総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]/OrgnlTxRef/Cdtr/Id/OrgId/Othr[2]/SchmeNm/");
try {
    out.println("Prtry["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
            .getTxInfAndSts().get(0).getOrgnlTxRef().getCdtr().getId()
            .getOrgId().getOthr().get(2).getSchmeNm().getPrtry() + "]);
} catch (NullPointerException e) {
    out.println("[No Prtry]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts/Othr]");
}

// 受取人口座番号 (総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]/OrgnlTxRef/CdtrAcct/Id/Othr/");
try {
    out.println("Id["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
            .getTxInfAndSts().get(0).getOrgnlTxRef().getCdtrAcct().getId()
            .getOthr().getId() + "]);
} catch (NullPointerException e) {
    out.println("[No Id]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts]");
}
```

最後にJAVAオブジェクトから複数の TxInfAnsSts タグに含まれる総合振込依頼に記述された受取人預金種目を参照する。

```
// 受取人預金種目 (総合振込依頼)
out.println(
    "/Document/CstmrPmtStsRpt/OrgnlPmtInfAndSts[0]/TxInfAndSts["
        + i + "]/OrgnlTxRef/CdtrAcct/Tp/");
try {
    out.println("Prtry["
        + doc.getCstmrPmtStsRpt().getOrgnlPmtInfAndSts().get(0)
        .getTxInfAndSts().get(0).getOrgnlTxRef().getCdtrAcct()
        .getTp().getPrtry() + "]");
} catch (NullPointerException e) {
    out.println("[No Prtry]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No OrgnlPmtInfAndSts/TxInfAndSts]");
}
}
}
```


(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/Owner/Id/OrgId/Othr[0]/");
try {
    out.println("Id[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getAcct().getOwner().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/Owner/Id/OrgId/Othr[0]/SchmeNm/");
try {
    out.println("Cd[" + doc.getBkToCstmrAcctRpt().getRpt().get(0)
        .getAcct().getOwner().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).isRvslInd() ? "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().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]");
}

}

}
```

(4) 振込入金通知(ISO20022 Camt.054)の読み込み方

xjc コマンドで Camt.054.001.02 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_054_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 Camt054 {
    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.054.001.02 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.054 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.054.001.02.xml");
try {
    SchemaFactory sFactory1
        = SchemaFactory.newInstance(
            XMLConstants.W3C_XML_SCHEMA_NS_URI);
    Schema schema1
        = sFactory1.newSchema(new File("camt.054.001.02.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/BkToCstmrDbtCdtNtfctn/GrpHdr/");
try {
    out.println("MsgId["
        + doc.getBkToCstmrDbtCdtNtfctn().getGrpHdr().getMsgId() + "]);
} catch (NullPointerException e) {
    out.println("[No MsgId]");
}

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

次に JAVA オブジェクトから通知 ID、通知情報作成日時、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。

```
// 通知 ID
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/");
try {
    out.println("ID["
        + doc.getBkToCstmrDbtCdtNtfctn().getNtfctn().get(0).getId() + "]");
} catch(NullPointerException e) {
    out.println("[No ID]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Ntfctn]");
}

// 通知情報作成日時
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/");
try {
    out.println("CreDtTm[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().get(0).getCreDtTm() + "]");
} catch(NullPointerException e) {
    out.println("[No CreDtTm]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Ntfctn]");
}

// 勘定日(自)
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/FrToDt");
try {
    out.println("FrDtTm[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().get(0).getFrToDt().getFrDtTm() + "]");
} catch(NullPointerException e) {
    out.println("[No FrDtTm]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Ntfctn]");
}
```

次に JAVA オブジェクトから勘定日(至)、口座番号、口座番号、口座名を参照する。

```
// 勘定日(至)
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/FrToDt");
try {
    out.println("ToDtTm[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().get(0).getFrToDt().getToDtTm() + "]");
} catch(NullPointerException e) {
    out.println("[No ToDtTm]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Ntfctn]");
}

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

// 預金種目
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Acct/Tp/");
try {
    out.println("Prtry[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().get(0).getAcct().getTp().getPrtry() + "]");
} catch(NullPointerException e) {
    out.println("[No Prtry]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Ntfctn]");
}

// 口座名
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Acct/");
try {
    out.println("Nm[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().get(0).getAcct().getNm() + "]");
} catch(NullPointerException e) {
    out.println("[No Nm]");
}
```

次に JAVA オブジェクトから受取人法人番号(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、銀行コードを参照する。

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

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

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

次に JAVA オブジェクトから銀行名、支店コード、支店名を参照する。

```
// 銀行名
out.println(
    "/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Acct/Svcr/FinInstnId/");
try {
    out.println("Nm[" + doc.getBkToCstmrDbtCdtNtfctn().getNtfctn().get(0)
        .getAcct().getSvcr().getFinInstnId().getNm() + "]");
} catch(NullPointerException e) {
    out.println("[No Nm]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Ntfctn]");
}

// 支店コード
out.println(
    "/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Acct/Svcr/BrnchId/");
try {
    out.println("Id[" + doc.getBkToCstmrDbtCdtNtfctn().getNtfctn().get(0)
        .getAcct().getSvcr().getBrnchId().getId() + "]");
} catch(NullPointerException e) {
    out.println("[No Id]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Ntfctn]");
}

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

次に JAVA オブジェクトから振込合計件数、振込合計金額、取消合計件数を参照する。

```
// 振込合計件数
out.println(
"/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/TxSummry/TtlCdtNtries/");
try {
    out.println("NbOfNtrie[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().get(0).getTxSummry().getTtlCdtNtries()
        .getNbOfNtries() + "]");
} catch(NullPointerException e) {
    out.println("[No NbOfNtrie]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Ntfctn]");
}

// 振込合計金額
out.println(
"/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/TxSummry/TtlCdtNtries/");
try {
    out.println("Sum[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().get(0).getTxSummry().getTtlCdtNtries().getSum() + "]");
} catch(NullPointerException e) {
    out.println("[No Sum]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Ntfctn]");
}

// 取消合計件数
out.println(
"/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/TxSummry/TtlDbtNtries/");
try {
    out.println("NbOfNtrie[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().get(0).getTxSummry().getTtlDbtNtries()
        .getNbOfNtries() + "]");
} catch(NullPointerException e) {
    out.println("[No NbOfNtrie]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Ntfctn]");
}
```


次に JAVA オブジェクトから取消合計金額、金額を参照する。取消合計金額までがヘッダーに記述され、以降は振込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。

```
// 取消合計金額
out.println(
"/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/TxSummry/TtlDbtNtries/");
try {
    out.println("Sum[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().get(0).getTxSummry().getTtlDbtNtries().getSum() + "]");
} catch(NullPointerException e) {
    out.println("[No Sum]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Ntfctn]");
}

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

    out.println("¥n--" + i + "--¥n");

// 金額
out.println(
"/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry[" + i + "]/");
try {
    out.println("Amt[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().get(0).getNtry().get(i).getAmt().getValue() + "]");
    out.println("Ccy[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().get(0).getNtry().get(i).getAmt().getCcy() + "]");
} catch(NullPointerException e) {
    out.println("[No Amt]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Ntfctn/Ntry]");
}
```

次に JAVA オブジェクトから取消区分、取消通知区分、取引ステータスを参照する。

```
// 取消区分
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry["
    + i + "]/");
try {
    out.println("CdtDbtInd[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().get(0).getNtry().get(i).getCdtDbtInd() + "]);
} catch(NullPointerException e) {
    out.println("[No CdtDbtInd]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Ntfctn/Ntry]");
}

// 取消通知区分
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry["
    + i + "]/");
try {
    out.println("RvsInd[" + (doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().get(0).getNtry().get(i).isRvslInd() ? "true" : "false") + "]);
} catch(NullPointerException e) {
    out.println("[No RvsInd]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Ntfctn/Ntry]");
}

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

次に JAVA オブジェクトから取消区分、取消通知区分、取引ステータスを参照する。

```
// 取消区分
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry["
    + i + "]/");
try {
    out.println("CdtDbtInd[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().get(0).getNtry().get(i).getCdtDbtInd() + "]);
} catch(NullPointerException e) {
    out.println("[No CdtDbtInd]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Ntfctn/Ntry]");
}

// 取消通知区分
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry["
    + i + "]/");
try {
    out.println("RvsInd[" + (doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().get(0).getNtry().get(i).isRvslInd() ? "true" : "false") + "]);
} catch(NullPointerException e) {
    out.println("[No RvsInd]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Ntfctn/Ntry]");
}

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

次に JAVA オブジェクトから勘定日、起算日、取引明細種別ドメインコードを参照する。

```
// 勘定日
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry["
    + i + "]/BookgDt/");
try {
    out.println("Dt[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().get(0).getNtry().get(i).getBookgDt().getDt() + "]);
} catch (NullPointerException e) {
    out.println("[No Dt]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No Ntfctn/Ntry]");
}

// 起算日
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry["
    + i + "]/ValDt/");
try {
    out.println("Dt[" + doc.getBkToCstmrDbtCdtNtfctn().getNtfctn()
        .get(0).getNtry().get(i).getValDt().getDt() + "]);
} catch (NullPointerException e) {
    out.println("[No Dt]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No Ntfctn/Ntry]");
}

// 取引明細種別ドメインコード - "PMNT"
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry["
    + i + "]/BkTxCd/Domn/");
try {
    out.println("Cd[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().get(0).getNtry().get(i)
        .getBkTxCd().getDomn().getCd() + "]);
} catch (NullPointerException e) {
    out.println("[No Cd]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No Ntfctn/Ntry]");
}
```

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、取引明細識別番号(振込依頼人発行)を参照する。

```
// 取引明細種別系列コード - "RCDT"
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry["
    + i + "]/BkTxCd/Domn/Fmly/");
try {
    out.println("Cd[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().get(0).getNtry().get(i)
        .getBkTxCd().getDomn().getFmly().getCd() + "]);
} catch(NullPointerException e) {
    out.println("[No Cd]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Ntfctn/Ntry]");
}

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

// 取引明細識別番号(振込依頼人発行)
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry["
    + i + "]/NtryDtls[0]/TxDtls[0]/Refs/");
try {
    out.println("EndToEndId[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().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 Ntfctn/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、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 Number"
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry["
    + i + "]/NtryDtls[0]/TxDtls[0]/Refs/Prtry/");
try {
    out.println("Tp[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().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 Ntfctn/Ntry/NtryDtls/TxDtls]");
}

// 照会番号
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry["
    + i + "]/NtryDtls[0]/TxDtls[0]/Refs/Prtry/");
try {
    out.println("Ref[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().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 Ntfctn/Ntry/NtryDtls/TxDtls]");
}

// 取引明細種別ドメインコード - "PMNT"
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry["
    + i + "]/NtryDtls[0]/TxDtls[0]/BkTxCd/Domn/");
try {
    out.println("Cd[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().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 Ntfctn/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、振込依頼人名を参照する。

```
// 取引明細種別系列コード - "RCDT"
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry["
    + i + "]/NtryDtls[0]/TxDtls[0]/BkTxCd/Domn/Fmly/");
try {
    out.println("Cd[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().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 Ntfctn/Ntry/NtryDtls/TxDtls]");
}

// 取引明細種別サブ系列コード - "DMCT"
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry["
    + i + "]/NtryDtls[0]/TxDtls[0]/BkTxCd/Domn/Fmly/");
try {
    out.println("SubFmlyCd[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().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 Ntfctn/Ntry/NtryDtls/TxDtls]");
}

// 振込依頼人名
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry["
    + i + "]/NtryDtls[0]/TxDtls[0]/RltdPties/Dbtr/");
try {
    out.println("Nm[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().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 Ntfctn/Ntry/NtryDtls/TxDtls]");
}
```

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

```
// 振込依頼人コード
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry["
    + i + "]/NtryDtls[0]/TxDtls[0]/RltdPties/Dbtr/Id/OrgId/Othr[0]/");
try {
    out.println("Id[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().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 Ntfctn/Ntry/NtryDtls/TxDtls/Othr]");
}

// 振込依頼人概要コード - "BANK"
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry[" + i
    + "]/NtryDtls[0]/TxDtls[0]/RltdPties/Dbtr/Id/OrgId/Othr[0]/SchmeNm/");
try {
    out.println("Cd[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().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 Ntfctn/Ntry/NtryDtls/TxDtls/Othr]");
}

// 振込依頼人法人番号(法人マイナンバー)
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry["
    + i + "]/NtryDtls[0]/TxDtls[0]/RltdPties/Dbtr/Id/OrgId/Othr[1]/");
try {
    out.println("Id[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().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 Ntfctn/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)概要コード、仕向銀行名、仕向支店名を参照する。

```
// 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/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry[" + i
+ "]/NtryDtls[0]/TxDtls[0]/RltdPties/Dbtr/Id/OrgId/Othr[1]/SchmeNm/");
try {
    out.println("Cd[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().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 Ntfctn/Ntry/NtryDtls/TxDtls/Othr]");
}

// 仕向銀行名
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry[" + i
+ "]/NtryDtls[0]/TxDtls[0]/RltdAgts/DbtrAgt/FinInstnId/");
try {
    out.println("Nm[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().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 Ntfctn/Ntry/NtryDtls/TxDtls]");
}

// 仕向支店名
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry[" + i
+ "]/NtryDtls[0]/TxDtls[0]/RltdAgts/DbtrAgt/BrnchId/");
try {
    out.println("Nm[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().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 Ntfctn/Ntry/NtryDtls/TxDtls]");
}
```

次に JAVA オブジェクトから金融 EDI を特定する EDI 情報を参照する。更に金融 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。

```
// EDI 情報
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry[" + i
        + i + "]/NtryDtls[0]/TxDtls[0]/RltdRmtInf[0]/");
try {
    out.println("RmtId[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().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 Ntfctn/Ntry/NtryDtls/TxDtls/RltdRmtInf]");
}

// 金融 EDI 情報
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry[" + i
        + "]/NtryDtls[0]/TxDtls[0]/RmtInf");
try {
    List<String> ustrds = null;
    ustrds = doc.getBkToCstmrDbtCdtNtfctn().getNtfctn().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) + "]");
        }
    }
}
```

XML の場合は MIME ヘッダーの確認と Base64 のデコードを行う。なお、if-else の単純なネストとならなかったためインデントが異なることに注意されたい。

```

        } 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) + "]");
                } 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 Ntfctn/Ntry/NtryDtls/TxDtls/Ustrd]");
        }
    }

```

次に JAVA オブジェクトからうち他店券金額、うち他店券金額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/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry[" + i
    + "]/NtryDtls[1]/TxDtls[0]/AmtDtls/");
try {
    out.println("Amt[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().get(0).getNtry().get(i).getNtryDtls().get(1)
        .getTxDtls().get(0).getAmtDtls().getTxAmt()
        .getAmt().getValue() + "]);
    out.println("Ccy[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().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 Ntfctn/Ntry/NtryDtls/TxDtls]");
}

// うち他店券金額情報種別ドメインコード - "PMNT"
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry[" + i
    + "]/NtryDtls[1]/TxDtls[0]/BkTxCd/Domn/");
try {
    out.println("Cd[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().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 Ntfctn/Ntry/NtryDtls/TxDtls]");
}
```

次に JAVA オブジェクトからうち他店券金額情報種別系列コード、うち他店券金額情報種別サブ系列コードを参照する。

```
// うち他店券金額情報種別系列コード - "RCHQ"
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry[" + i
    + "]/NtryDtls[1]/TxDtls[0]/BkTxCd/Domn/Fmly/");
try {
    out.println("Cd[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().get(0).getNtry().get(i).getNtryDtls().get(1)
        .getTxDtls().get(0).getBkTxCd().getDomn()
        .getFmly().getCd() + "]);
} catch(NullPointerException e) {
    out.println("[No Cd]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Ntfctn/Ntry/NtryDtls/TxDtls]");
}

// うち他店券金額情報種別サブ系列コード - "CCHQ"
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry[" + i
    + "]/NtryDtls[1]/TxDtls[0]/BkTxCd/Domn/Fmly/");
try {
    out.println("SubFmlyCd[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().get(0).getNtry().get(i).getNtryDtls().get(1)
        .getTxDtls().get(0).getBkTxCd().getDomn()
        .getFmly().getSubFmlyCd() + "]);
} catch(NullPointerException e) {
    out.println("[No SubFmlyCd]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Ntfctn/Ntry/NtryDtls/TxDtls]");
}
```

次に JAVA オブジェクトから仕向金融機関指示情報を参照する。ここまでが個々の振込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。

```
// 仕向金融機関指示情報
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/Ntry["
    + i + "]/");
try {
    out.println("AddtlNtryInf[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().get(0).getNtry().get(i).getAddtlNtryInf() + "]);
} catch(NullPointerException e) {
    out.println("[No AddtlNtryInf]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Ntfctn/Ntry]");
}

}

out.println("¥n--  --¥n");

// 金融機関指示情報
out.println("/Document/BkToCstmrDbtCdtNtfctn/Ntfctn[0]/");
try {
    out.println("AddtlRptInf[" + doc.getBkToCstmrDbtCdtNtfctn()
        .getNtfctn().get(0).getAddtlNtfctnInf() + "]);
} catch(NullPointerException e) {
    out.println("[No AddtlNtfctnInf]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Ntfctn]");
}

}

}
```

(5) 総合振込依頼制御情報 (ISO20022 Bah.001) の作り方

xjc コマンドで Head.001.001.01 の XML スキーマから生成された関数と共に、Bah.001 のメッセージに該当する JAVA オブジェクト生成する。JAXB の marshaller を使って XML 形式の Bah.001 のメッセージに変換する。

まず準備としてビジネスヘッダーを 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やランタイムルーチンをインポートする。

次にビジネスアプリケーションヘッダールートを作成する。

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

import iso.std.iso._20022.tech.xsd.head_001_001.*;

import java.io.File;
import java.io.StringWriter;
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 javax.xml.datatype.XMLGregorianCalendar;
import javax.xml.datatype.DatatypeFactory;

public class Bah001OUT {
    public static void main(String[] args) {

        // ビジネスアプリケーションヘッダールート
        AppHdr hdr = new AppHdr();
        Party9Choice fr = new Party9Choice();
        hdr.setFr(fr);
```

次に当方情報、当方識別子、通信制御概要情報、通信制御情報、ファイル制御詳細情報、ファイルアクセスキー、ファイル制御概要情報を作成する。

```
// 当方情報
PartyIdentification42 frOrgId = new PartyIdentification42();
fr.setOrgId(frOrgId);
Party10Choice frId = new Party10Choice();
frOrgId.setId(frId);
OrganisationIdentification7 frOId = new OrganisationIdentification7();
frId.setOrgId(frOId);
GenericOrganisationIdentification1 frGOI1
    = new GenericOrganisationIdentification1();
frOId.getOthr().add(frGOI1);

// 当方識別子
frGOI1.setId(new CCPSW0.putCC("03000000019999").putPSW("abcd")
    .Unparse());
// CCPSW ccpsw = new CCPSW0();
// ccpsw.setCC("03000000019999");
// ccpsw.setPSW("abcd");
// frGOI1.setId(ccpsw.Unparse());

// 通信制御概要情報
OrganisationIdentificationSchemeName1Choice oisn1
    = new OrganisationIdentificationSchemeName1Choice();
frGOI1.setSchmeNm(oisn1);

// 通信制御情報
oisn1.setPrtry("Communication ValidationCode");

// ファイル制御詳細情報
GenericOrganisationIdentification1 frGOI2
    = new GenericOrganisationIdentification1();
frOId.getOthr().add(frGOI2);

// ファイルアクセスキー
frGOI2.setId("XXXXXXXXXXXXX");

// ファイル制御概要情報
OrganisationIdentificationSchemeName1Choice oisn2
    = new OrganisationIdentificationSchemeName1Choice();
frGOI2.setSchmeNm(oisn2);
```


次にファイル制御情報、相手方情報、相手方識別子、相手方接続情報、ファイル名補助情報(オプション)、メッセージ ID、制御情報を作成する。

```
// ファイル制御情報
oisn2.setPrtry("FileControl ValidationCode");

// 相手方情報
Party9Choice to = new Party9Choice();
hdr.setTo(to);
BranchAndFinancialInstitutionIdentification5 toFIId
    = new BranchAndFinancialInstitutionIdentification5();
to.setFIId(toFIId);
FinancialInstitutionIdentification8 toFId
    = new FinancialInstitutionIdentification8();
toFIId.setFinInstnId(toFId);
GenericFinancialIdentification1 toGFI
    = new GenericFinancialIdentification1();
toFId.setOthr(toGFI);

// 相手方識別子
toGFI.setId(new CCPSW().putCC("06000000019999").putPSW("wxyz")
    .Unparse());

// 相手方接続情報
toGFI.setIssr(new Issr().putIP("192.0.2.1").putTel("03000000000").Unparse());

// ファイル名補助情報(オプション)
hdr.setBizMsgIdr("xxxx012345678901234567890123456789");

// メッセージ ID
hdr.setMsgDefIdr("pain.001.001.03");
// hdr.setMsgDefIdr("camt.052.001.02");
// hdr.setMsgDefIdr("camt.054.001.02");

// 制御情報
hdr.setBizSvc(new BizSvc().putMode("TST").putFileName("5020xxxxxyyzw")
    .putCharCode("0").putConnect("").putResend("").Unparse());
```

次に通信年月日時分秒を作成する。

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 head.001.001.01.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。

```
// 通信年月日時分秒
try {
    XMLGregorianCalendar xmlGregCal = DatatypeFactory.newInstance()
        .newXMLGregorianCalendar(2018, 3, 20, 15, 14, 13, 0, 0);
    hdr.setCreDt(xmlGregCal);
} catch (Exception e) {
    e.printStackTrace();
}

StringWriter stringWriter = new StringWriter();
try {
    SchemaFactory sFactory
        = SchemaFactory.newInstance(
            XMLConstants.W3C_XML_SCHEMA_NS_URI);
    Schema schema = sFactory.newSchema(new File("head.001.001.01.xsd"));
    JAXBContext context = JAXBContext.newInstance(AppHdr.class);
    Marshaller marshaller = context.createMarshaller();
    marshaller.setSchema(schema);
    marshaller.setEventHandler(new MyValidationEventHandler());

    marshaller.marshal(hdr, stringWriter);

    //marshaller.setProperty(Marshaller.JAXB_FRAGMENT, true);
    marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);

    //System.out.print("<?xml version='1.0'>");
    marshaller.marshal(hdr, System.out);

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

制御情報をモード、ファイル名、文字コード、接続形態区分、再送有無を組み合わせて記述するためのクラスを作成している。

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

public class BizSvc {

    protected String mode;
    protected String fileName;
    protected String charCode;
    protected String connect;
    protected String resend;
    protected Pattern p;

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

    BizSvc(String mode, String fileName, String charCode,
           String connect, String resend) {
        this.p = Pattern.compile("([^\:]*):([^\:]*):([^\:]*):([^\:]*):([^\:]*");
        this.mode = mode;
        this.fileName = fileName;
        this.charCode = charCode;
        this.connect = connect;
        this.resend = resend;
    }

    public String Unparse() {
        return this.mode + ":" + this.fileName + ":" + this.charCode
            + ":" + this.connect + ":" + this.resend;
    }
}
```

制御情報を記述するためのクラスの続きを示す。

```
public void setMode(String v) {
    this.mode = v;
}

public BizSvc putMode(String v) {
    this.mode = v;
    return this;
}

public String getMode() {
    return this.mode;
}

public void setFileName(String v) {
    this.fileName = v;
}

public BizSvc putFileName(String v) {
    this.fileName = v;
    return this;
}

public String getFileName() {
    return this.fileName;
}

public void setCharCode(String v) {
    this.charCode = v;
}

public BizSvc putCharCode(String v) {
    this.charCode = v;
    return this;
}

public String getCharCode() {
    return this.charCode;
}
```

制御情報を記述するためのクラスの続きを示す。

```
public void setConnect(String v) {
    this.connect = v;
}

public BizSvc putConnect(String v) {
    this.connect = v;
    return this;
}

public String getConnect() {
    return this.connect;
}

public void setResend(String v) {
    this.resend = v;
}

public BizSvc putResend(String v) {
    this.resend = v;
    return this;
}

public String getResend() {
    return this.resend;
}
}
```

当方識別子、相手方識別子となるセンター確認コードとパスワードは組み合わせて記述するためのクラスを作成している。

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

public class CCPSW {

    protected String cc;
    protected String psw;
    protected Pattern p;

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

    CCPSW(String cc, String psw) {
        this.p = Pattern.compile("([^\:]*):([^\:]*)");
        this.cc = cc;
        this.psw = psw;
    }

    public String Unparse() {
        return this.cc + ":" + this.psw;
    }

    public CCPSW Parse(String v) {
        Matcher m = p.matcher(v);
        if (m.find()){
            this.cc = m.group(1);
            this.psw = m.group(2);
        }
        return this;
    }
}
```

センター確認コードとパスワードは組み合わせて記述するためのクラスの続きを示す。

```
public void setCC(String v) {
    this.cc = v;
}

public CCPSW putCC(String v) {
    this.cc = v;
    return this;
}

public String getCC() {
    return this.cc;
}

public void setPSW(String v) {
    this.psw = v;
}

public CCPSW putPSW(String v) {
    this.psw = v;
    return this;
}

public String getPSW() {
    return this.psw;
}
}
```

相手方接続情報となる IP アドレスと電話番号を組み合わせて記述するためのクラスを作成している。

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

public class Issr {

    private String ip;
    private String tel;
    private static Pattern p = Pattern.compile("(d+¥¥.d+¥¥.d+¥¥.d+):(d+)");

    Issr() {
    }

    Issr(String ip, String tel) {
        this.ip = ip;
        this.tel = tel;
    }

    public String Unparse() {
        return this.ip + ":" + this.tel;
    }

    public Issr Parse(String v) {
        Matcher m = p.matcher(v);
        if (m.find()){
            this.ip = m.group(1);
            this.tel = m.group(2);
        }
        return this;
    }
}
```


相手方接続情報を記述するためのクラスの続きを示す。

```
public void setIP(String v) {
    this.ip = v;
}

public Issr putIP(String v) {
    this.ip = v;
    return this;
}

public String getIP() {
    return this.ip;
}

public void setTel(String v) {
    this.tel = v;
}

public Issr putTel(String v) {
    this.tel = v;
    return this;
}

public String getTel() {
    return this.tel;
}

}
```

(6) 入出金取引明細制御情報 (ISO20022 Bah.001) の作り方

総合振込依頼制御情報と同様に xjc コマンドで Head.001.001.01 の XML スキーマから生成された関数と共に、Bah.001 のメッセージに該当する JAVA オブジェクト生成する。JAXB の marshaller を使って XML 形式の Bah.001 のメッセージに変換する。

メッセージ ID に camt.052.001.02 を設定することが、総合振込依頼制御情報と唯一の異なる。

```
// メッセージ ID
// hdr.setMsgDefIdr("pain.001.001.03");
hdr.setMsgDefIdr("camt.052.001.02");
// hdr.setMsgDefIdr("camt.054.001.02");
```

メッセージ ID の設定以外は同様のため省略する。

(7) 振込入金通知依頼制御情報 (ISO20022 Bah.001) の作り方

総合振込依頼制御情報と同様に xjc コマンドで Head.001.001.01 の XML スキーマから生成された関数と共に、Bah.001 のメッセージに該当する JAVA オブジェクト生成する。JAXB の marshaller を使って XML 形式の Bah.001 のメッセージに変換する。

メッセージ ID に camt.054.001.02 を設定することが、総合振込依頼制御情報と唯一の異なる。

```
// メッセージ ID
// hdr.setMsgDefIdr("pain.001.001.03");
// hdr.setMsgDefIdr("camt.052.001.02");
hdr.setMsgDefIdr("camt.054.001.02");
```

メッセージ ID の設定以外は同様のため省略する。

(8) 総合振込結果制御情報 (ISO20022 Bah.001) の読み方

xjc コマンドで Head.001.001.01 の XML スキーマから生成された関数、読み込んだ XML を検証するためのランタイムルーチン等をインポートする。

まず出力用ストリームの文字コードを UTF-8 に指定する。次に Head.001.001.01 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 Bah.001 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。

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

import javax.xml.bind.JAXBContext;
import javax.xml.bind.Unmarshaller;
import javax.xml.XMLConstants;
import javax.xml.validation.Schema;
import javax.xml.validation.SchemaFactory;
import iso.std.iso._20022.tech.xsd.head_001_001.*

public class Bah001IN {
    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;
        }

        File file = new File("head.001.001.01.xml");
        AppHdr hdr = new AppHdr();
        try {
            SchemaFactory sFactory = SchemaFactory.newInstance(
                XMLConstants.W3C_XML_SCHEMA_NS_URI);
            Schema schema = sFactory.newSchema(new File("head.001.001.01.xsd"));
            JAXBContext context = JAXBContext.newInstance(AppHdr.class);
            Unmarshaller unmarshaller = context.createUnmarshaller();
            unmarshaller.setSchema(schema);
            hdr = (AppHdr) unmarshaller.unmarshal(file);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

次に JAVA オブジェクトからビジネスメッセージ ID、メッセージ ID、処理結果コード、通信年月日時分秒、当方識別子（依頼）を参照する。当該要素が無い場合、NullPointerException、IndexOutOfBoundsException の例外が発生することに注意されたい。

```
// ビジネスメッセージ ID
out.println("/AppHdr/");
try {
    out.println("BizMsgIdr[" + hdr.getBizMsgIdr() + "]");
} catch (NullPointerException e) {
    out.println("[No BizMsgIdr]");
}

// メッセージ ID
out.println("/AppHdr/");
try {
    out.println("MsgDefIdr[" + hdr.getMsgDefIdr() + "]");
} catch (NullPointerException e) {
    out.println("[No MsgDefIdr]");
}

// 処理結果コード
out.println("/AppHdr/");
try {
    out.println("BizSvc[" + hdr.getBizSvc() + "]");
} catch (NullPointerException e) {
    out.println("[No BizSvc]");
}

// 通信年月日時分秒
out.println("/AppHdr/");
try {
    out.println("CreDt[" + hdr.getCreDt() + "]");
} catch (NullPointerException e) {
    out.println("[No CreDt]");
}
```

次に JAVA オブジェクトから当方識別子(依頼)、相手方識別子(依頼)、相手方接続情報、ファイル名補助情報(オプション)(依頼)を参照する。

```
// 当方識別子(依頼)
out.println("/AppHdr/Rltd/Fr/OrgId/Id/OrgId/Othr/");
try {
    out.println("Id[" + hdr.getRltd().getFr().getOrgId().getId().getOrgId()
        .getOthr().get(0).getId() + "]");
} catch(NullPointerException e) {
    out.println("[No Id]");
} catch(IndexOutOfBoundsException e) {
    out.println("[No Othr]");
}

// 相手方識別子(依頼)
out.println("/AppHdr/Rltd/To/FIId/FinInstnId/Othr/");
try {
    out.println("Id[" + hdr.getRltd().getTo().getFIId().getFinInstnId()
        .getOthr().getId() + "]");
} catch(NullPointerException e) {
    out.println("[No Id]");
}

// 相手方接続情報
out.println("/AppHdr/Rltd/To/FIId/FinInstnId/Othr/");
try {
    out.println("Issr[" + hdr.getRltd().getTo().getFIId().getFinInstnId()
        .getOthr().getIssr() + "]");
} catch(NullPointerException e) {
    out.println("[No Issr]");
}

// ファイル名補助情報(オプション)(依頼)
out.println("/AppHdr/Rltd/");
try {
    out.println("BizMsgIdr[" + hdr.getBizMsgIdr() + "]");
} catch(NullPointerException e) {
    out.println("[No BizMsgIdr]");
}
```

次に JAVA オブジェクトからメッセージ ID (依頼)、制御情報 (依頼)、通信年月日時分秒 (依頼) を参照する。

```
// メッセージ ID (依頼)
out.println("/AppHdr/Rltd/");
try {
    out.println("MsgDefIdr[" + hdr.getRltd().getMsgDefIdr() + "]");
} catch (NullPointerException e) {
    out.println("[No MsgDefIdr]");
}

// 制御情報 (依頼)
out.println("/AppHdr/Rltd/");
try {
    out.println("BizSvc[" + hdr.getRltd().getBizSvc() + "]");
} catch (NullPointerException e) {
    out.println("[No BizSvc]");
}

// 通信年月日時分秒 (依頼)
out.println("/AppHdr/Rltd/");
try {
    out.println("CreDt[" + hdr.getRltd().getCreDt() + "]");
} catch (NullPointerException e) {
    out.println("[No CreDt]");
}

}
}
```

(9) 入出金取引明細結果制御情報 (ISO20022 Bah.001) の読み方

総合振込依頼結果制御情報と同様に xjc コマンドで Head.001.001.01 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 Bah.001 のメッセージを JAVA オブジェクトに変換する。JAVA オブジェクトから各情報項目を参照する。

メッセージIDの参照結果が camt.052.001.02 となることが、総合振込依頼制御情報と唯一の異なるが、プログラムは同じのため省略する。

(10) 振込入金通知結果制御情報 (ISO20022 Bah.001) の読み方

総合振込依頼結果制御情報と同様に xjc コマンドで Head.001.001.01 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 Bah.001 のメッセージを JAVA オブジェクトに変換する。JAVA オブジェクトから各情報項目を参照する。

メッセージIDの参照結果が camt.054.001.02 となることが、総合振込依頼制御情報と唯一の異なるが、プログラムは同じのため省略する。

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:unfact: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、unfact 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。

(3) 国連 CEFAC 支払案内メッセージの作り方

金融 EDI 情報は、商流情報から抽出された情報項目を国連 CEFAC 支払案内メッセージの形式に格納し、XML 化を行う。商流情報項目が国連 CEFAC 支払案内メッセージのどこに該当するかは、国連 CEFAC 支払案内メッセージ設計書を参照されたい。

まず準備として国連 CEFAC 支払案内メッセージを JAVA オブジェクトに変換するための xjc コマンドで CrossIndustryRemittanceAdvice.xsd の XML スキーマから関数を生成する。生成した関数やランタイムルーチンをインポートする。

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

import javax.xml.bind.JAXBContext;
import javax.xml.bind.Marshaller;
import javax.xml.bind.Unmarshaller;
import javax.xml.XMLConstants;
import javax.xml.validation.Schema;
import javax.xml.validation.SchemaFactory;
import javax.xml.datatype.XMLGregorianCalendar;
import javax.xml.datatype.DatatypeFactory;
import un.unece.uncefact.data.standard.crossindustryremittanceadvice._11.*;

import java.math.BigDecimal;
import java.util.Base64;
import java.util.GregorianCalendar;
```

まず、以下の XML に当たる JAVA オブジェクト群(データ区分、業界区分)を生成する。

```
<CrossIndustryRemittanceAdvice
  xmlns="urn:un:unece:uncefact:data:standard:CrossIndustryRemittanceAdvice:11">
  <CIExchangedDocumentContext>
    <BusinessProcessSpecifiedCIDocumentContextParameter>
      <ID>01</ID>
    </BusinessProcessSpecifiedCIDocumentContextParameter>
    <SubsetSpecifiedCIDocumentContextParameter>
      <ID>JPSFEDI</ID>
    </SubsetSpecifiedCIDocumentContextParameter>
  </CIExchangedDocumentContext>
  ...
</CrossIndustryRemittanceAdvice>
```

最初に CIExchangedDocumentContext オブジェクトを作成し、更にデータ区分、業界区分を作成する。

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

        CrossIndustryRemittanceAdvice cira =
            new CrossIndustryRemittanceAdvice();
        CIExchangedDocumentContextType cedc =
            new CIExchangedDocumentContextType();
        CIDocumentContextParameterType cdcp1 =
            new CIDocumentContextParameterType();
        IDType id1 = new IDType();

        cira.setCIExchangedDocumentContext(cedc);
        cedc.setBusinessProcessSpecifiedCIDocumentContextParameter(cdcp1);
        cdcp1.setID(id1);

        // データ区分
        id1.setValue("01");

        CIDocumentContextParameterType cdcp2 =
            new CIDocumentContextParameterType();
        cedc.setSubsetSpecifiedCIDocumentContextParameter(cdcp2);

        // 業界区分
        IDType id2 = new IDType();
        cdcp2.setID(id2);
        id2.setValue("JPSFEDI");
    }
}
```

次に、以下の XML に当たる JAVA オブジェクト群(支払通知番号、支払通知発行日)を生成する。

```
<CrossIndustryRemittanceAdvice ...>
...
  <CIReExchangedDocument>
    <ID>AA</ID>
    <IssueDateTime>
      <DateTime>2018-03-02T20:11:10+09:00</DateTime>
    </IssueDateTime>
  ...
</CrossIndustryRemittanceAdvice>
```

CIRExchangedDocument オブジェクトを作成する。支払通知番号 AA を記述する。AA は例であり意味は無い。支払通知発行日を作成する。

```
CIRExchangedDocumentType ced = new CIRExchangedDocumentType();
cira.setCIRExchangedDocument(ced);

// 支払通知番号
IDType id3 = new IDType();
ced.setID(id3);
id3.setValue("AA");

// 支払通知発行日
DateTimeType dt1 = new DateTimeType();
ced.setIssueDateTime(dt1);
try {
    XMLGregorianCalendar xmlGregCal1 = DatatypeFactory.newInstance()
        .newXMLGregorianCalendar(new GregorianCalendar());
    dt1.setDateTime(xmlGregCal1);

    xmlGregCal1.setYear(2018);
    xmlGregCal1.setMonth(3);
    xmlGregCal1.setDay(2);
    xmlGregCal1.setTimezone(540);
    xmlGregCal1.setTime(20, 11, 10);

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

次に、以下の XML に当たる JAVA オブジェクト群(支払内容)を生成する。

```
<CrossIndustryRemittanceAdvice ...>
...
  <CIRExchangedDocument>
    ...
    <IncludedCINote>
      <Content>note</Content>
    </IncludedCINote>
  </CIRExchangedDocument>
...
</CrossIndustryRemittanceAdvice>
```

支払内容を作成する。

```
// 支払内容
CINoteType cin1 = new CINoteType();
ced.getIncludedCINote().add(cin1);
TextType tt1 = new TextType();
cin1.setContent(tt1);
tt1.setValue("note");
```

次に、以下の XML に当たる JAVA オブジェクト群(支払番号、支払日時)を生成する。

```
<CrossIndustryRemittanceAdvice ...>
...
  <CIRHTradeSettlementPayment>
    <EndToEndID>1000000000000:2018-03-02:0123456789</EndToEndID>
    <RequestedExecutionDateTime>
      <DateTime>2018-03-02T20:11:10+09:00</DateTime>
    </RequestedExecutionDateTime>
  ...
</CrossIndustryRemittanceAdvice>
```

支払番号、支払日時を作成する。

```
CIRHTradeSettlementPaymentType ctsp =
    new CIRHTradeSettlementPaymentType();
cira.setCIRHTradeSettlementPayment(ctsp);

// 支払番号
IDType id4 = new IDType();
ctsp.setEndToEndID(id4);
id4.setValue("1000000000000:2018-03-02:0123456789");

// 支払日時
DateTimeType dt2 = new DateTimeType();
ctsp.setRequestedExecutionDateTime(dt2);
try {
    XMLGregorianCalendar xmlGregCal2 = DatatypeFactory.newInstance()
        .newXMLGregorianCalendar(new GregorianCalendar());
    dt2.setDateTime(xmlGregCal2);

    xmlGregCal2.setYear(2018);
    xmlGregCal2.setMonth(3);
    xmlGregCal2.setDay(2);
    xmlGregCal2.setTimezone(9 * 60);
    xmlGregCal2.setTime(20, 11, 10);

    // System.out.println(xmlGregCa2);
} catch (Exception e) {
    e.printStackTrace();
}
```

次に、以下の XML に当たる JAVA オブジェクト群(受取人企業法人コード、受取人企業名)を生成する。

```
<CrossIndustryRemittanceAdvice ...>
...
  <CIRHTradeSettlementPayment>
    ...
    <SpecifiedCIRHSupplyChainTradeSettlement>
      <PayeeCITradeParty>
        <GlobalID>10000000000001</GlobalID>
        <Name languageID="ja">受取人企業名</Name>
        ...
      </PayeeCITradeParty>
    ...
  </CrossIndustryRemittanceAdvice>
```

受取人企業法人コード、受取人企業名を作成する。

```
CIRHSupplyChainTradeSettlementType chscts =
    new CIRHSupplyChainTradeSettlementType();
ctsp.getSpecifiedCIRHSupplyChainTradeSettlement().add(chscts);

CITradePartyType ctp1 = new CITradePartyType();
chscts.setPayeeCITradeParty(ctp1);

// 受取人企業法人コード
IDType id5 = new IDType();
ctp1.setGlobalID(id5);
id5.setValue("10000000000001");

// 受取人企業名
TextType tt2 = new TextType();
ctp1.setName(tt2);
tt2.setValue("受取人企業名");
tt2.setLanguageID("ja");
```


次に、以下の XML に当たる JAVA オブジェクト群(受取人企業連絡先電話番号)を生成する。

```
<CrossIndustryRemittanceAdvice ...>
...
  <CIRHTradeSettlementPayment>
    ...
    <SpecifiedCIRHSupplyChainTradeSettlement>
      <PayeeCITradeParty>
        ...
        <DefinedCITradeContact>
          <TelephoneCIUniversalCommunication>
            <CompleteNumber>06-0000-0000</CompleteNumber>
          </TelephoneCIUniversalCommunication>
        </DefinedCITradeContact>
      </PayeeCITradeParty>
    ...
  </CrossIndustryRemittanceAdvice>
```

受取人企業連絡先電話番号を作成する。

```
// 受取人企業連絡先電話番号
CITradeContactType ctc1 = new CITradeContactType();
ctp1.setDefinedCITradeContact(ctc1);
CIUniversalCommunicationType cuc1 =
    new CIUniversalCommunicationType();
ctc1.setTelephoneCIUniversalCommunication(cuc1);
TextType tt3 = new TextType();
cuc1.setCompleteNumber(tt3);
tt3.setValue("06-0000-0000");
```

次に、以下の XML に当たる JAVA オブジェクト群(支払人企業法人コード、支払人企業名)を生成する。

```
<CrossIndustryRemittanceAdvice ...>
...
  <CIRHTradeSettlementPayment>
    ...
    <SpecifiedCIRHSupplyChainTradeSettlement>
      ...
      <PayerCITradeParty>
        <GlobalID>10000000000000</GlobalID>
        <Name languageID="ja">支払人企業名</Name>
        ...
      </PayerCITradeParty>
    ...
  </CrossIndustryRemittanceAdvice>
```

支払人企業法人コード、支払人企業名を作成する。

```
CITradePartyType ctp2 = new CITradePartyType();
chscs.setPayerCITradeParty(ctp2);

// 支払人企業法人コード
IDType id6 = new IDType();
ctp2.setGlobalID(id6);
id6.setValue("10000000000000");

// 支払人企業名
TextType tt4 = new TextType();
ctp2.setName(tt4);
tt4.setValue("支払人企業名");
tt4.setLanguageID("ja");
```

次に、以下の XML に当たる JAVA オブジェクト群(支払人企業連絡先電話番号)を生成する。

```
<CrossIndustryRemittanceAdvice ...>
...
  <CIRHTradeSettlementPayment>
    ...
    <SpecifiedCIRHSupplyChainTradeSettlement>
      ...
      <PayerCITradeParty>
        ...
        <DefinedCITradeContact>
          <TelephoneCIUniversalCommunication>
            <CompleteNumber>03-0000-0000</CompleteNumber>
          </TelephoneCIUniversalCommunication>
        </DefinedCITradeContact>
      </PayerCITradeParty>
    ...
  </CrossIndustryRemittanceAdvice>
```

支払人企業連絡先電話番号を作成する。

```
// 支払人企業連絡先電話番号
CITradeContactType ctc2 = new CITradeContactType();
ctp2.setDefinedCITradeContact(ctc2);
CIUniversalCommunicationType cuc2 =
    new CIUniversalCommunicationType();
ctc2.setTelephoneCIUniversalCommunication(cuc2);
TextType tt5 = new TextType();
cuc2.setCompleteNumber(tt5);
tt5.setValue("03-0000-0000");
```

次に、以下の XML に当たる JAVA オブジェクト群(支払合計金額、金額相殺項目理由コード)を生成する。

```
<CrossIndustryRemittanceAdvice ...>
...
  <CIRHTradeSettlementPayment>
    ...
    <SpecifiedCIRHSupplyChainTradeSettlement>
      ...
      <SpecifiedCIRHTradeSettlementMonetarySummation>
        <PaymentTotalAmount currencyID="JPY">
          10900
        </PaymentTotalAmount>
        <ApplicableCIRHSpecifiedBalanceOut>
          <ReasonCode>1</ReasonCode>
          ...
        </ApplicableCIRHSpecifiedBalanceOut>
      </SpecifiedCIRHTradeSettlementMonetarySummation>
    ...
  </CIRHTradeSettlementPayment>
</CrossIndustryRemittanceAdvice>
```

CIRHTradeSettlementMonetarySummation オブジェクトを作成する。支払合計金額、金額相殺項目理由コードを記述する。

```
CIRHTradeSettlementMonetarySummationType chtsms =
    new CIRHTradeSettlementMonetarySummationType();
chscs.setSpecifiedCIRHTradeSettlementMonetarySummation(chtsms);

// 支払合計金額
AmountType at1 = new AmountType();
chtsms.setPaymentTotalAmount(at1);
at1.setValue(new BigDecimal(10900));
at1.setCurrencyID("JPY");

// 金額相殺項目理由コード
CIRHSpecifiedBalanceOutType csbo1 = new CIRHSpecifiedBalanceOutType();
chtsms.getApplicableCIRHSpecifiedBalanceOut().add(csbo1);
CodeType rc1 = new CodeType();
csbo1.setReasonCode(rc1);
rc1.setValue("1");
```

次に、以下の XML に当たる JAVA オブジェクト群(金額相殺項目理由、相殺金額)を生成する。

```
<CrossIndustryRemittanceAdvice ...>
...
  <CIRHTradeSettlementPayment>
    ...
    <SpecifiedCIRHSupplyChainTradeSettlement>
      ...
      <SpecifiedCIRHTradeSettlementMonetarySummation>
        ...
        <ApplicableCIRHSpecifiedBalanceOut>
          ...
          <ReasonDescription languageID="ja">
            金額相殺理由
          </ReasonDescription>
          <CalculatedAmount currencyID="JPY">100</CalculatedAmount>
        </ApplicableCIRHSpecifiedBalanceOut>
      </SpecifiedCIRHTradeSettlementMonetarySummation>
    ...
  </CrossIndustryRemittanceAdvice>
```

金額相殺項目理由、相殺金額を記述する。

```
// 金額相殺項目理由
TextType tt6 = new TextType();
csbo1.setReasonDescription(tt6);
tt6.setValue("金額相殺理由");
tt6.setLanguageID("ja");

// 相殺金額
AmountType at2 = new AmountType();
csbo1.setCalculatedAmount(at2);
at2.setValue(new BigDecimal("100"));
at2.setCurrencyID("JPY");
```

次に、以下の XML に当たる JAVA オブジェクト群(税額(支払全体の合計税額)、税区分(支払全体の税区分、全て同じ税区分の場合)、税率(支払全体の税率、全て同率の場合))を生成する。

```
<CrossIndustryRemittanceAdvice ...>
...
  <CIRHTradeSettlementPayment>
    ...
    <SpecifiedCIRHSupplyChainTradeSettlement>
      ...
      <ApplicableCITradeTax>
        <CalculatedAmount>900</CalculatedAmount>
        <TypeCode>EXC</TypeCode>
        <CalculatedRate>0.09</CalculatedRate>
      </ApplicableCITradeTax>
    </SpecifiedCIRHSupplyChainTradeSettlement>
  </CIRHTradeSettlementPayment>
...
</CrossIndustryRemittanceAdvice>
```

税額(支払全体の合計税額)、税区分(支払全体の税区分、全て同じ税区分の場合)、税率(支払全体の税率、全て同率の場合)を記述する。

```
// 税額(支払全体の合計税額)
CITradeTaxType ctt1 = new CITradeTaxType();
chscts.setApplicableCITradeTax(ctt1);
AmountType at3 = new AmountType();
ctt1.setCalculatedAmount(at3);
at3.setValue(new BigDecimal("900"));

// 税区分(支払全体の税区分、全て同じ税区分の場合)
CodeType tc1 = new CodeType();
ctt1.setTypeCode(tc1);
tc1.setValue("EXC");

// 税率(支払全体の税率、全て同率の場合)
RateType rt1 = new RateType();
ctt1.setCalculatedRate(rt1);
rt1.setValue(new BigDecimal("0.09"));
```

次に、以下の XML に当たる JAVA オブジェクト群(行番号、支払内容)を生成する。

```
<CrossIndustryRemittanceAdvice ...>
...
  <CIRTSupplyChainTradeTransaction>
    <AssociatedCIRDocumentLineDocument>
      <LineID>L01</LineID>
      <IncludedCINote>
        <Content languageID="ja">支払内容</Content>
      </IncludedCINote>
    </AssociatedCIRDocumentLineDocument>
  ...
</CrossIndustryRemittanceAdvice>
```

CIRTSupplyChainTradeTransaction オブジェクトを生成し、行番号、支払内容を記述する。

```
CIRTSupplyChainTradeTransactionType csctt =
    new CIRTSupplyChainTradeTransactionType();
cira.getCIRTSupplyChainTradeTransaction().add(csctt);

// 行番号
CIRDocumentLineDocumentType cdld1 =
    new CIRDocumentLineDocumentType();
csctt.setAssociatedCIRDocumentLineDocument(cdld1);
IDType lid = new IDType();
cdld1.setLineID(lid);
lid.setValue("L01");

// 支払内容
CINoteType cn1 = new CINoteType();
TextType tt7 = new TextType();

cdld1.getIncludedCINote().add(cn1);
cn1.setContent(tt7);
tt7.setValue("支払内容 1");
tt7.setLanguageID("ja");
```

次に、以下の XML に当たる JAVA オブジェクト群(請求先企業法人コード、請求先企業名)を生成する。

```
<CrossIndustryRemittanceAdvice ...>
...
  <CIRTSupplyChainTradeTransaction>
    ...
    <ApplicableCIRTSupplyChainTradeAgreement>
      <BuyerCITradeParty>
        <GlobalID>10000000000002</GlobalID>
        <Name languageID="ja">請求先企業名</Name>
      </BuyerCITradeParty>
    </ApplicableCIRTSupplyChainTradeAgreement>
    ...
  </CIRTSupplyChainTradeTransaction>
</CrossIndustryRemittanceAdvice>
```

CIRTSupplyChainTradeAgreement オブジェクトを生成し、請求先企業法人コード、請求先企業名を記述する。

```
CIRTSupplyChainTradeAgreementType cscta =
    new CIRTSupplyChainTradeAgreementType();
csctt.setApplicableCIRTSupplyChainTradeAgreement(cscta);
CITradePartyType ctp3 = new CITradePartyType();
cscta.setBuyerCITradeParty(ctp3);
IDType ida = new IDType();

// 請求先企業法人コード
ctp3.setGlobalID(ida);
ida.setValue("10000000000002");

// 請求先企業名
TextType tta = new TextType();
ctp3.setName(tta);
tta.setValue("請求先企業名");
tta.setLanguageID("ja");
```


次に、以下の XML に当たる JAVA オブジェクト群(締日)を生成する。

```
<CrossIndustryRemittanceAdvice ...>
...
  <CIRTSupplyChainTradeTransaction>
    ...
    <ApplicableCIRTSupplyChainTradeSettlement>
      <ClosingBookDueDateTime>
        <DateTime>2018-02-25T00:00:00+09:00</DateTime>
      </ClosingBookDueDateTime>
      ...
    </ApplicableCIRTSupplyChainTradeSettlement>
    ...
  </CIRTSupplyChainTradeTransaction>
</CrossIndustryRemittanceAdvice>
```

CIRTSupplyChainTradeSettlement オブジェクトを生成し、締日を記述する。

```
CIRTSupplyChainTradeSettlementType ctscts =
    new CIRTSupplyChainTradeSettlementType();
csctt.setApplicableCIRTSupplyChainTradeSettlement(ctscts);

// 締日
DateTimeType dt4 = new DateTimeType();
ctscts.setClosingBookDueDateTime(dt4);
try {
    XMLGregorianCalendar xmlGregCal4 = DatatypeFactory.newInstance()
        .newXMLGregorianCalendar(new GregorianCalendar());
    dt4.setDateTime(xmlGregCal4);

    xmlGregCal4.setYear(2018);
    xmlGregCal4.setMonth(2);
    xmlGregCal4.setDay(25);
    xmlGregCal4.setTimezone(9 * 60);
    xmlGregCal4.setTime(0, 0, 0);

    // System.out.println(xmlGregCa2);
} catch (Exception e) {
    e.printStackTrace();
}
```

次に、以下の XML に当たる JAVA オブジェクト群(入金予定日)を生成する。

```
<CrossIndustryRemittanceAdvice ...>
...
  <CIRTSupplyChainTradeTransaction>
    ...
    <ApplicableCIRTSupplyChainTradeSettlement>
      ...
      <ScheduledPaymentDateTime>
        <DateTime>2018-03-03T00:00:00+09:00</DateTime>
      </ScheduledPaymentDateTime>
      ...
    </ApplicableCIRTSupplyChainTradeSettlement>
    ...
  </CIRTSupplyChainTradeTransaction>
</CrossIndustryRemittanceAdvice>
```

入金予定日を記述する。

```
// 入金予定日
DateTimeType dt5 = new DateTimeType();
ctscts.setScheduledPaymentDateTime(dt5);
try {
    XMLGregorianCalendar xmlGregCal5 = DatatypeFactory.newInstance()
        .newXMLGregorianCalendar(new GregorianCalendar());
    dt5.setDateTime(xmlGregCal5);

    xmlGregCal5.setYear(2018);
    xmlGregCal5.setMonth(3);
    xmlGregCal5.setDay(3);
    xmlGregCal5.setTimezone(9 * 60);
    xmlGregCal5.setTime(0, 0, 0);

    // System.out.println(xmlGregCa2);
} catch (Exception e) {
    e.printStackTrace();
}
```

次に、以下の XML に当たる JAVA オブジェクト群 (請求先連絡担当者、請求先連絡先部門) を生成する。

```
<CrossIndustryRemittanceAdvice ...>
  ...
  <CIRTSupplyChainTradeTransaction>
    ...
    <ApplicableCIRTSupplyChainTradeSettlement>
      ...
      <InvoiceeCITradeParty>
        <DefinedCITradeContact>
          <PersonName>請求先連絡担当者</PersonName>
          <DepartmentName>請求先連絡担当者</DepartmentName>
          ...
        </DefinedCITradeContact>
      </InvoiceeCITradeParty>
      ...
    </ApplicableCIRTSupplyChainTradeSettlement>
    ...
  </CIRTSupplyChainTradeTransaction>
</CrossIndustryRemittanceAdvice>
```

CITradeParty オブジェクトを生成し、請求先連絡担当者、請求先連絡先部門を記述する。

```
CITradePartyType ctp4 = new CITradePartyType();
ctscts.setInvoiceeCITradeParty(ctp4);
CITradeContactType ctc3 = new CITradeContactType();
ctp4.setDefinedCITradeContact(ctc3);

// 請求先連絡担当者
TextType ttb = new TextType();
ctc3.setPersonName(ttb);
ttb.setValue("請求先連絡担当者");

// 請求先連絡先部門
TextType ttc = new TextType();
ctc3.setDepartmentName(ttc);
ttc.setValue("請求先連絡部門");
```

次に、以下の XML に当たる JAVA オブジェクト群(請求先電話番号)を生成する。

```
<CrossIndustryRemittanceAdvice ...>
...
  <CIRTSupplyChainTradeTransaction>
    ...
    <ApplicableCIRTSupplyChainTradeSettlement>
      ...
      <InvoiceeCITradeParty>
        <DefinedCITradeContact>
          ...
          <TelephoneCIUniversalCommunication>
            <CompleteNumber>06-0000-0000</CompleteNumber>
          </TelephoneCIUniversalCommunication>
        </DefinedCITradeContact>
      </InvoiceeCITradeParty>
    ...
  </ApplicableCIRTSupplyChainTradeSettlement>
  ...
</CIRTSupplyChainTradeTransaction>
</CrossIndustryRemittanceAdvice>
```

請求先電話番号を記述する。

```
// 請求先電話番号
CIUniversalCommunicationType cuc3 =
    new CIUniversalCommunicationType();
ctc3.setTelephoneCIUniversalCommunication(cuc3);
TextType ttd = new TextType();
cuc3.setCompleteNumber(ttd);
ttd.setValue("06-0000-0000");
```

次に、以下の XML に当たる JAVA オブジェクト群(支払金額(明細))を生成する。

```
<CrossIndustryRemittanceAdvice ...>
...
  <CIRTSupplyChainTradeTransaction>
    ...
    <ApplicableCIRTSupplyChainTradeSettlement>
      ...
      <SpecifiedCIRTTradeSettlementMonetarySummation>
        <PaymentTotalAmount>100000</PaymentTotalAmount>
      </SpecifiedCIRTTradeSettlementMonetarySummation>
    </ApplicableCIRTSupplyChainTradeSettlement>
    ...
  </CIRTSupplyChainTradeTransaction>
</CrossIndustryRemittanceAdvice>
```

支払金額(明細)を記述する。

```
// 支払金額(明細)
CIRTTradeSettlementMonetarySummationType cttsms =
    new CIRTTradeSettlementMonetarySummationType();
ctscts.setSpecifiedCIRTTradeSettlementMonetarySummation(cttsms);
AmountType at8 = new AmountType();
cttsms.setPaymentTotalAmount(at8);
at8.setValue(new BigDecimal("100000"));
```

次に、以下の XML に当たる JAVA オブジェクト群(納品伝票番号)を生成する。

```
<CrossIndustryRemittanceAdvice ...>
...
  <CIRTSupplyChainTradeTransaction>
    ...
    <IncludedCIRLSupplyChainTradeLineItem>
      <AssociatedCIRDocumentLineDocument>
        <LineID>D000</LineID>
      </AssociatedCIRDocumentLineDocument>
    ...
    </IncludedCIRLSupplyChainTradeLineItem>
  ...
</CIRTSupplyChainTradeTransaction>
</CrossIndustryRemittanceAdvice>
```

CIRLSupplyChainTradeLineItem オブジェクトを生成し、納品伝票番号を記述する。

```
CIRLSupplyChainTradeLineItemType clsctli =
    new CIRLSupplyChainTradeLineItemType();
csctt.getIncludedCIRLSupplyChainTradeLineItem().add(clsctli);

// 納品伝票番号
CIRDocumentLineDocumentType cdld2 =
    new CIRDocumentLineDocumentType();
clsctli.setAssociatedCIRDocumentLineDocument(cdld2);
IDType idb = new IDType();
cdld2.setLineID(idb);
idb.setValue("D000");
```

次に、以下の XML に当たる JAVA オブジェクト群(発注番号)を生成する。

```
<CrossIndustryRemittanceAdvice ...>
...
  <CIRLSupplyChainTradeTransaction>
    ...
    <IncludedCIRLSupplyChainTradeLineItem>
      ...
      <SpecifiedCIRLSupplyChainTradeSettlement>
        <ReferencedCIRLSupplyChainTradeTransaction>
          <ApplicableCIRLSupplyChainTradeAgreement>
            <BuyerOrderReferencedCIRReferencedDocument>
              <IssuerAssignedID>200</IssuerAssignedID>
            </BuyerOrderReferencedCIRReferencedDocument>
            ...
          </ApplicableCIRLSupplyChainTradeAgreement>
          ...
        </ReferencedCIRLSupplyChainTradeTransaction>
      </SpecifiedCIRLSupplyChainTradeSettlement>
    </IncludedCIRLSupplyChainTradeLineItem>
    ...
  </CIRLSupplyChainTradeTransaction>
</CrossIndustryRemittanceAdvice>
```

CIRLSupplyChainTradeSettlement オブジェクトを生成し、発注番号を記述する。

```
CIRLSupplyChainTradeSettlementType clscts =
    new CIRLSupplyChainTradeSettlementType();
clsctli.getSpecifiedCIRLSupplyChainTradeSettlement().add(clscts);

CIRLSupplyChainTradeTransactionType clsctt =
    new CIRLSupplyChainTradeTransactionType();
clscts.setReferencedCIRLSupplyChainTradeTransaction(clsctt);
CIRLSupplyChainTradeAgreementType clscta =
    new CIRLSupplyChainTradeAgreementType();
clsctt.setApplicableCIRLSupplyChainTradeAgreement(clscta);

// 発注番号
CIRReferencedDocumentType crd3 = new CIRReferencedDocumentType();
clscta.setBuyerOrderReferencedCIRReferencedDocument(crd3);
IDType idc = new IDType();
crd3.setIssuerAssignedID(idc);
idc.setValue("200");
```

次に、以下の XML に当たる JAVA オブジェクト群(単価、受注番号)を生成する。

```
<CrossIndustryRemittanceAdvice ...>
...
  <CIRTSupplyChainTradeTransaction>
    ...
    <IncludedCIRLSupplyChainTradeLineItem>
      ...
      <SpecifiedCIRLSupplyChainTradeSettlement>
        <ReferencedCIRLSupplyChainTradeTransaction>
          <ApplicableCIRLSupplyChainTradeAgreement>
            ...
            <OrderPriceProductCITradePrice>
              <ChargeAmount>500</ChargeAmount>
            </OrderPriceProductCITradePrice>
            <SellerOrderReferencedCISupplyChainTradeDocument>
              <IssuerAssignedID>300</IssuerAssignedID>
            </SellerOrderReferencedCISupplyChainTradeDocument>
          </ApplicableCIRLSupplyChainTradeAgreement>
          ...
        </ReferencedCIRLSupplyChainTradeTransaction>
      </SpecifiedCIRLSupplyChainTradeSettlement>
    </IncludedCIRLSupplyChainTradeLineItem>
    ...
  </CIRTSupplyChainTradeTransaction>
</CrossIndustryRemittanceAdvice>
```

単価、受注番号を記述する。

```
// 単価
CITradePriceType ctp1 = new CITradePriceType();
clscta.setOrderPriceProductCITradePrice(ctp1);
AmountType atg = new AmountType();
ctp1.setChargeAmount(atg);
atg.setValue(new BigDecimal("500"));

// 受注番号
CISupplyChainTradeDocumentType crd4 = new CISupplyChainTradeDocumentType();
clscta.setSellerOrderReferencedCISupplyChainTradeDocument(crd4);
IDType idd = new IDType();
crd4.setIssuerAssignedID(idd);
idd.setValue("300");
```


次に、以下の XML に当たる JAVA オブジェクト群(数量、納入番号)を生成する。

```
<CrossIndustryRemittanceAdvice ...>
...
  <CIRTSupplyChainTradeTransaction>
    ...
    <IncludedCIRLSupplyChainTradeLineItem>
      ...
      <SpecifiedCIRLSupplyChainTradeSettlement>
        <ReferencedCIRLSupplyChainTradeTransaction>
          ...
          <ApplicableCIRLSupplyChainTradeDelivery>
            <BilledQuantity>10</BilledQuantity>
            <RequestedDeliveryCISupplyChainEvent>
              <ID>400</ID>
            ...
          </ReferencedCIRLSupplyChainTradeTransaction>
        </SpecifiedCIRLSupplyChainTradeSettlement>
      </IncludedCIRLSupplyChainTradeLineItem>
    ...
  </CIRTSupplyChainTradeTransaction>
</CrossIndustryRemittanceAdvice>
```

CIRLSupplyChainTradeDelivery オブジェクトを生成し、数量、納入番号を記述する。

```
CIRLSupplyChainTradeDeliveryType clsctd =
    new CIRLSupplyChainTradeDeliveryType();
clsctd.setApplicableCIRLSupplyChainTradeDelivery(clsctd);

// 数量
QuantityType qt = new QuantityType();
clsctd.setBilledQuantity(qt);
qt.setValue(new BigDecimal("10"));

CISupplyChainEventType csce = new CISupplyChainEventType();
clsctd.setRequestedDeliveryCISupplyChainEvent(csce);

// 納入番号
IDType ide = new IDType();
csce.setID(ide);
ide.setValue("400");
```

次に、以下の XML に当たる JAVA オブジェクト群(納入日時)を生成する。

```
<CrossIndustryRemittanceAdvice ...>
...
  <CIRTSupplyChainTradeTransaction>
    ...
    <IncludedCIRLSupplyChainTradeLineItem>
      ...
      <SpecifiedCIRLSupplyChainTradeSettlement>
        <ReferencedCIRLSupplyChainTradeTransaction>
          ...
          <OccurrenceDateTime>
            <DateTime>2018-02-20T00:00:00+09:00</DateTime>
          </OccurrenceDateTime>
          </RequestedDeliveryCISupplyChainEvent>
          </ApplicableCIRLSupplyChainTradeDelivery>
          ...
        ...
      </CIRTSupplyChainTradeTransaction>
    </CrossIndustryRemittanceAdvice>
```

納入日時を記述する。

```
// 納入日時
DateTimeType dt6 = new DateTimeType();
csce.setOccurrenceDateTime(dt6);
try {
    XMLGregorianCalendar xmlGregCal6 = DatatypeFactory.newInstance()
        .newXMLGregorianCalendar(new GregorianCalendar());
    dt6.setDateTime(xmlGregCal6);

    xmlGregCal6.setYear(2018);
    xmlGregCal6.setMonth(2);
    xmlGregCal6.setDay(20);
    xmlGregCal6.setTimezone(9 * 60);
    xmlGregCal6.setTime(0, 0, 0);

    // System.out.println(xmlGregCa2);
} catch (Exception e) {
    e.printStackTrace();
}
```

次に、以下の XML に当たる JAVA オブジェクト群 (製品コード、製品名) を生成する。

```
<CrossIndustryRemittanceAdvice ...>
...
  <CIRTSupplyChainTradeTransaction>
    ...
    <IncludedCIRLSupplyChainTradeLineItem>

      <SpecifiedCIRLSupplyChainTradeSettlement>
        <ReferencedCIRLSupplyChainTradeTransaction>

          <IncludedCITradeProduct>
            <BuyerAssignedID>xxx-yyy-zzz</BuyerAssignedID>
            <Name languageID="ja">製品名</Name>
          </IncludedCITradeProduct>
        </ReferencedCIRLSupplyChainTradeTransaction>
      </SpecifiedCIRLSupplyChainTradeSettlement>
    </IncludedCIRLSupplyChainTradeLineItem>
    ...
  </CIRTSupplyChainTradeTransaction>
</CrossIndustryRemittanceAdvice>
```

CITradeProduct オブジェクトを生成し、製品コード、製品名を記述する。

```
CITradeProductType ctp2 = new CITradeProductType();
clsett.setIncludedCITradeProduct(ctp2);

// 製品コード
IDType idf = new IDType();
ctp2.setBuyerAssignedID(idf);
idf.setValue("xxx-yyy-zzz");

// 製品名
TextType ttf = new TextType();
ctp2.setName(ttf);
ttf.setValue("製品名");
ttf.setLanguageID("ja");
```

次に、以下の XML に当たる JAVA オブジェクト群(参照文書種別(契約書)、契約名)を生成する。

```
<CrossIndustryRemittanceAdvice ...>
...
  <CIRTSupplyChainTradeTransaction>
    ...
    <AssociatedCISReferencedDocument>
      <ReferenceTypeCode>315</ReferenceTypeCode>
      <Name languageID="ja">契約名</Name>
    </AssociatedCISReferencedDocument>
    ...
  </CIRTSupplyChainTradeTransaction>
</CrossIndustryRemittanceAdvice>
```

CISReferencedDocument オブジェクトを生成し、参照文書種別(契約書)、契約名を記述する。

```
CISReferencedDocumentType crd1 = new CISReferencedDocumentType();
csctt.getAssociatedCISReferencedDocument().add(crd1);

// 参照文書種別(契約書)
CodeType rc2 = new CodeType();
crd1.setReferenceTypeCode(rc2);
rc2.setValue("315");

// 契約名
TextType tt9 = new TextType();
crd1.setName(tt9);
tt9.setValue("契約名");
tt9.setLanguageID("ja");
```

次に、以下の XML に当たる JAVA オブジェクト群(参照文書種別(契約書)、契約名)を生成する。

```
<CrossIndustryRemittanceAdvice ...>
  ...
  <CIRTSupplyChainTradeTransaction>
    ...
    <AssociatedCISReferencedDocument>
      <IssuerAssignedID>1000000000001:2018-02-03:101</IssuerAssignedID>
      <IssueDateTime>
        <DateTime>2018-02-03T00:00:00+09:00</DateTime>
      </IssueDateTime>
      <ReferenceTypeCode>380</ReferenceTypeCode>
    </AssociatedCISReferencedDocument>
  </CIRTSupplyChainTradeTransaction>
</CrossIndustryRemittanceAdvice>
```

2 つ目の CISReferencedDocument オブジェクトを生成し、請求書番号を記述する。

```
CISReferencedDocumentType crd2 = new CISReferencedDocumentType();
csctt.getAssociatedCISReferencedDocument().add(crd2);

// 請求書番号
IDType id7 = new IDType();
crd2.setIssuerAssignedID(id7);
id7.setValue("1000000000001:2018-02-03:101");
```

引き続き、請求書発行日、参照文書種別(請求書)を記述する。

```
// 請求書発行日
DateTimeType dt3 = new DateTimeType();
crd2.setIssueDateTime(dt3);
try {
    XMLGregorianCalendar xmlGregCal3 = DatatypeFactory.newInstance()
        .newXMLGregorianCalendar(new GregorianCalendar());
    dt3.setDateTime(xmlGregCal3);

    xmlGregCal3.setYear(2018);
    xmlGregCal3.setMonth(2);
    xmlGregCal3.setDay(3);
    xmlGregCal3.setTimezone(9 * 60);
    xmlGregCal3.setTime(0, 0, 0);

    // System.out.println(xmlGregCa2);
} catch (Exception e) {
    e.printStackTrace();
}

// 参照文書種別(請求書)
CodeType rc3 = new CodeType();
crd2.setReferenceTypeCode(rc3);
rc3.setValue("380");
```

次に XML 化を図り、XML スキーマ検証も行う。XML スキーマ検証では出力チェック用の XML スキーマを用い、標準としては任意でもシステム上必要な項目を必須としたチェックを行う厳しめのチェックを行う。ここでは、整形した出力方法と、改行の無い出力方法を示している。

```
StringWriter stringWriter = new StringWriter();
try {
    JAXBContext context = JAXBContext
        .newInstance(CrossIndustryRemittanceAdvice.class);
    Marshaller marshaller = context.createMarshaller();
    SchemaFactory sFactory = SchemaFactory
        .newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);
    Schema schema = sFactory.newSchema(
        new File("CrossIndustryRemittanceAdvice.xsd"));
    marshaller.setSchema(schema);
    marshaller.setEventHandler(new MyValidationEventHandler());

    // 整形出力
    PrintStream out = new PrintStream(System.out, true, "UTF-8");
    marshaller.setProperty(Marshaller.JAXB_FRAGMENT, false);
    // 出力対象により XML ヘッダーの扱いが変わることに注意
    // System.out.print("<?xml version='1.0'>");
    marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);
    marshaller.marshal(cira, out);

    // 改行無し出力
    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
    marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, false);
    marshaller.marshal(cira, stringWriter);
    String edi = stringWriter.toString();
    out.println(edi);
} catch (Exception e) {
    e.printStackTrace();
}
}
```

(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]");
}
```

次に支払内容、支払番号、支払日時を参照する。

```
// 支払内容
out.println("CIRExchangedDocument/IncludedCINote[0]/");
try {
    out.println("Content[" + cira.getCIRExchangedDocument()
        .getIncludedCINote().get(0).getContent().getValue() + "]");
} catch (NullPointerException e) {
    out.println("[No Content]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No IncludedCINote]");
}

// 支払番号
out.println("CIRHTradeSettlementPayment/");
try {
    out.println("EndToEndID[" + cira
        .getCIRHTradeSettlementPayment().getEndToEndID()
        .getValue() + "]");
} catch (NullPointerException e) {
    out.println("[No EndToEndID]");
}

// 支払日時
out.println("CIRHTradeSettlementPayment/");
try {
    out.println("RequestedExecutionDateTime/DateTime["
        + cira.getCIRHTradeSettlementPayment()
        .getRequestedExecutionDateTime().getDateTime() + "]");
} catch (NullPointerException e) {
    out.println("[No RequestedExecutionDateTime/DateTime]");
}
```

次に受取人企業法人コード、受取人企業名を参照する。

```
// 受取人企業法人コード
out.println("CIRHTradeSettlementPayment"
    + "/SpecifiedCIRHSupplyChainTradeSettlement[0]"
    + "/PayeeCITradeParty/");
try {
    out.println("GlobalID[" + cira.getCIRHTradeSettlementPayment()
        .getSpecifiedCIRHSupplyChainTradeSettlement().get(0)
        .getPayeeCITradeParty().getGlobalID().getValue() + "]");
} catch (NullPointerException e) {
    out.println("[No GlobalID]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No SpecifiedCIRHSupplyChainTradeSettlement]");
}

// 受取人企業名
out.println("CIRHTradeSettlementPayment"
    + "/SpecifiedCIRHSupplyChainTradeSettlement[0]"
    + "/PayeeCITradeParty/");
try {
    out.println("Name[" + cira.getCIRHTradeSettlementPayment()
        .getSpecifiedCIRHSupplyChainTradeSettlement().get(0)
        .getPayeeCITradeParty().getName().getValue() + "]");
} catch (NullPointerException e) {
    out.println("[No Name]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No SpecifiedCIRHSupplyChainTradeSettlement]");
}
```

次に受取企業連絡先電話番号、支払人企業法人コードを参照する。

```
// 受取企業連絡先電話番号
out.println("CIRHTradeSettlementPayment"
    + "/SpecifiedCIRHSupplyChainTradeSettlement[0]"
    + "/PayeeCITradeParty/DefinedCITradeContact"
    + "/TelephoneCIUniversalCommunication/");
try {
    out.println("CompleteNumber["
        + cira.getCIRHTradeSettlementPayment()
        .getSpecifiedCIRHSupplyChainTradeSettlement().get(0)
        .getPayeeCITradeParty().getDefinedCITradeContact()
        .getTelephoneCIUniversalCommunication()
        .getCompleteNumber().getValue() + "]");
} catch (NullPointerException e) {
    out.println("[No CompleteNumber]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No SpecifiedCIRHSupplyChainTradeSettlement]");
}

// 支払人企業法人コード
out.println("CIRHTradeSettlementPayment/"
    + "SpecifiedCIRHSupplyChainTradeSettlement[0]"
    + "/PayerCITradeParty/");
try {
    out.println("GlobalID[" + cira.getCIRHTradeSettlementPayment()
        .getSpecifiedCIRHSupplyChainTradeSettlement().get(0)
        .getPayerCITradeParty().getGlobalID().getValue() + "]");
} catch (NullPointerException e) {
    out.println("[No GlobalID]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No SpecifiedCIRHSupplyChainTradeSettlement]");
}
```

次に支払人企業名、支払企業連絡先電話番号を参照する。

```
// 支払人企業名
out.println("CIRHTradeSettlementPayment/"
    + "SpecifiedCIRHSupplyChainTradeSettlement[0]"
    + "/PayerCITradeParty/");
try {
    out.println("Name[" + cira.getCIRHTradeSettlementPayment()
        .getSpecifiedCIRHSupplyChainTradeSettlement().get(0)
        .getPayerCITradeParty().getName().getValue() + "]");
} catch (NullPointerException e) {
    out.println("[No Name]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No SpecifiedCIRHSupplyChainTradeSettlement]");
}

// 支払企業連絡先電話番号
out.println("CIRHTradeSettlementPayment"
    + "/SpecifiedCIRHSupplyChainTradeSettlement[0]"
    + "/PayerCITradeParty/DefinedCITradeContact"
    + "/TelephoneCIUniversalCommunication/");
try {
    out.println("CompleteNumber["
        + cira.getCIRHTradeSettlementPayment()
        .getSpecifiedCIRHSupplyChainTradeSettlement().get(0)
        .getPayerCITradeParty().getDefinedCITradeContact()
        .getTelephoneCIUniversalCommunication()
        .getCompleteNumber().getValue() + "]");
} catch (NullPointerException e) {
    out.println("[No CompleteNumber]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No SpecifiedCIRHSupplyChainTradeSettlement]");
}
```

次に支払合計金額、金額相殺項目理由コードを参照する。

```
// 支払合計金額
out.println("CIRHTradeSettlementPayment"
    + "/SpecifiedCIRHSupplyChainTradeSettlement[0]"
    + "/SpecifiedCIRHTradeSettlementMonetarySummation/");
try {
    out.println("PaymentTotalAmount[" + cira
        .getCIRHTradeSettlementPayment()
        .getSpecifiedCIRHSupplyChainTradeSettlement().get(0)
        .getSpecifiedCIRHTradeSettlementMonetarySummation()
        .getPaymentTotalAmount().getValue() + "]");
} catch (NullPointerException e) {
    out.println("[No PaymentTotalAmount]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No SpecifiedCIRHSupplyChainTradeSettlement]");
}

// 金額相殺項目理由コード
out.println("CIRHTradeSettlementPayment"
    + "/SpecifiedCIRHSupplyChainTradeSettlement[0]"
    + "/SpecifiedCIRHTradeSettlementMonetarySummation"
    + "/ApplicableCIRHSpecifiedBalanceOut/");
try {
    out.println("ReasonCode["
        + cira.getCIRHTradeSettlementPayment()
        .getSpecifiedCIRHSupplyChainTradeSettlement().get(0)
        .getSpecifiedCIRHTradeSettlementMonetarySummation()
        .getApplicableCIRHSpecifiedBalanceOut().get(0)
        .getReasonCode().getValue() + "]");
} catch (NullPointerException e) {
    out.println("[No ReasonCode]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No SpecifiedCIRHSupplyChainTradeSettlement"
        + "/ApplicableCIRHSpecifiedBalanceOut]");
}
```

次に金額相殺項目理由、相殺金額を参照する。

```
// 金額相殺項目理由
out.println("CIRHTradeSettlementPayment"
    + "/SpecifiedCIRHSupplyChainTradeSettlement[0]"
    + "/SpecifiedCIRHTradeSettlementMonetarySummation"
    + "/ApplicableCIRHSpecifiedBalanceOut/");
try {
    out.println("ReasonDescription[" + cira
        .getCIRHTradeSettlementPayment()
        .getSpecifiedCIRHSupplyChainTradeSettlement().get(0)
        .getSpecifiedCIRHTradeSettlementMonetarySummation()
        .getApplicableCIRHSpecifiedBalanceOut().get(0)
        .getReasonDescription().getValue() + "]");
} catch (NullPointerException e) {
    out.println("[No ReasonDescription]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No SpecifiedCIRHSupplyChainTradeSettlement"
        + "/ApplicableCIRHSpecifiedBalanceOut]");
}

// 相殺金額
out.println("CIRHTradeSettlementPayment"
    + "/SpecifiedCIRHSupplyChainTradeSettlement[0]"
    + "/SpecifiedCIRHTradeSettlementMonetarySummation"
    + "/ApplicableCIRHSpecifiedBalanceOut/");
try {
    out.println("CalculatedAmount[" + cira
        .getCIRHTradeSettlementPayment()
        .getSpecifiedCIRHSupplyChainTradeSettlement().get(0)
        .getSpecifiedCIRHTradeSettlementMonetarySummation()
        .getApplicableCIRHSpecifiedBalanceOut().get(0)
        .getCalculatedAmount().getValue() + "]");
} catch (NullPointerException e) {
    out.println("[No CalculatedAmount]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No SpecifiedCIRHSupplyChainTradeSettlement"
        + "/ApplicableCIRHSpecifiedBalanceOut]");
}
```


次に税額(支払全体の合計税額)、税区分(支払全体の税区分、全て同じ税区分の場合)を参照する。

```
// 税額(支払全体の合計税額)
out.println("CIRHTradeSettlementPayment"
    + "/SpecifiedCIRHSupplyChainTradeSettlement[0]"
    + "/ApplicableCITradeTax/");
try {
    out.println("CalculatedAmount[" + cira
        .getCIRHTradeSettlementPayment()
        .getSpecifiedCIRHSupplyChainTradeSettlement().get(0)
        .getApplicableCITradeTax().getCalculatedAmount()
        .getValue() + "]");
} catch (NullPointerException e) {
    out.println("[No CalculatedAmount]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No SpecifiedCIRHSupplyChainTradeSettlement]");
}

// 税区分(支払全体の税区分、全て同じ税区分の場合)
out.println("CIRHTradeSettlementPayment"
    + "/SpecifiedCIRHSupplyChainTradeSettlement[0]"
    + "/ApplicableCITradeTax/");
try {
    out.println("TypeCode[" + cira
        .getCIRHTradeSettlementPayment()
        .getSpecifiedCIRHSupplyChainTradeSettlement().get(0)
        .getApplicableCITradeTax().getTypeCode()
        .getValue() + "]");
} catch (NullPointerException e) {
    out.println("[No TypeCode]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No SpecifiedCIRHSupplyChainTradeSettlement]");
}
```

次に税率(支払全体の税率、全て同率の場合)、行番号を参照する。行番号以降は請求書毎になるので、繰り返し請求書を参照している。

```
// 税率(支払全体の税率、全て同率の場合)
out.println("CIRHTradeSettlementPayment"
    + "/SpecifiedCIRHSupplyChainTradeSettlement[0]"
    + "/ApplicableCITradeTax/");
try {
    out.println("CalculatedRate[" + cira
        .getCIRHTradeSettlementPayment()
        .getSpecifiedCIRHSupplyChainTradeSettlement().get(0)
        .getApplicableCITradeTax().getCalculatedRate()
        .getValue() + "]");
} catch (NullPointerException e) {
    out.println("[No CalculatedRate]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No SpecifiedCIRHSupplyChainTradeSettlement]");
}

for(int i = 0; i < cira.getCIRTSupplyChainTradeTransaction().size();
    i++) {
    out.println("¥n--" + i + "--¥n");

    // 行番号
    out.println("CIRTSupplyChainTradeTransaction[" + i
        + "]/AssociatedCIRDocumentLineDocument/");
    try {
        out.println("LineID[" + cira
            .getCIRTSupplyChainTradeTransaction().get(i)
            .getAssociatedCIRDocumentLineDocument()
            .getLineID().getValue() + "]");
    } catch (NullPointerException e) {
        out.println("[No LineID]");
    }
}
```

次に支払内容（請求書毎）を参照する。請求書と契約書に関しては `AssociatedCIReferencedDocument` オブジェクトを繰り返し使い参照する。参照種別で請求書か契約書を判別する。なお、請求書または契約書の関連情報の XML 上の位置は後ろになるが、順序に関わらず参照可能である。

```
// 支払内容(請求書毎)
out.println("CIRTSupplyChainTradeTransaction[" + i
    + "]/AssociatedCIRDocumentLineDocument"
    + "/IncludedCINote[0]/");
try {
    out.println("Content[" + cira
        .getCIRTSupplyChainTradeTransaction().get(i)
        .getAssociatedCIRDocumentLineDocument()
        .getIncludedCINote().get(0).getContent().getValue() + "]);
} catch (NullPointerException e) {
    out.println("[No Content]");
} catch (IndexOutOfBoundsException e) {
    out.println("[No IncludedCINote]");
}

for(int j = 0;
    j < cira.getCIRTSupplyChainTradeTransaction().get(i)
        .getAssociatedCIReferencedDocument().size(); j++) {

    // 請求書番号(請求書の場合)
    out.println("CIRTSupplyChainTradeTransaction[" + i
        + "]/AssociatedCIReferencedDocument[" + j + "]/");
    try {
        out.println("IssuerAssignedID[" + cira
            .getCIRTSupplyChainTradeTransaction().get(i)
            .getAssociatedCIReferencedDocument().get(j)
            .getIssuerAssignedID().getValue() + "]);
    } catch (NullPointerException e) {
        out.println("[No IssuerAssignedID]");
    }
}
```

次に支請求書発行日(請求書の場合)、参照文書種別、契約名(契約書の場合)を参照する。

```
// 請求書発行日(請求書の場合)
out.println("CIRTSupplyChainTradeTransaction[" + i
    + "]/AssociatedCISuppliedDocument[" + j + "]/");
try {
    out.println("IssueDateTime/DateTime[" + cira
        .getCIRTSupplyChainTradeTransaction().get(i)
        .getAssociatedCISuppliedDocument().get(j)
        .getIssueDateTime().getDateTime() + "]/");
} catch (NullPointerException e) {
    out.println("[No IssueDateTime/DateTime]");
}

// 参照文書種別
out.println("CIRTSupplyChainTradeTransaction[" + i
    + "]/AssociatedCISuppliedDocument[" + j + "]/");
try {
    out.println("ReferenceTypeCode[" + cira
        .getCIRTSupplyChainTradeTransaction().get(i)
        .getAssociatedCISuppliedDocument().get(j)
        .getReferenceTypeCode().getValue() + "]/");
} catch (NullPointerException e) {
    out.println("[No ReferenceTypeCode]");
}

// 契約名(契約書の場合)
out.println("CIRTSupplyChainTradeTransaction[" + i
    + "]/AssociatedCISuppliedDocument[" + j + "]/");
try {
    out.println("Name[" + cira
        .getCIRTSupplyChainTradeTransaction().get(i)
        .getAssociatedCISuppliedDocument().get(j)
        .getName().getValue() + "]/");
} catch (NullPointerException e) {
    out.println("[No Name]");
}
}
```

次に請求先企業法人コード、請求先企業名、締日を参照する。

```
// 請求先企業法人コード
out.println("CIRTSupplyChainTradeTransaction[" + i
    + "]/ApplicableCIRTSupplyChainTradeAgreement"
    + "/BuyerCITradeParty/");
try {
    out.println("GlobalID[" + cira
        .getCIRTSupplyChainTradeTransaction().get(i)
        .getApplicableCIRTSupplyChainTradeAgreement()
        .getBuyerCITradeParty().getGlobalID().getValue() + "]);
} catch (NullPointerException e) {
    out.println("[No GlobalID]");
}

// 請求先企業名
out.println("CIRTSupplyChainTradeTransaction[" + i
    + "]/ApplicableCIRTSupplyChainTradeSettlement"
    + "/BuyerCITradeParty/");
try {
    out.println("Name[" + cira
        .getCIRTSupplyChainTradeTransaction().get(i)
        .getApplicableCIRTSupplyChainTradeAgreement()
        .getBuyerCITradeParty().getName().getValue() + "]);
} catch (NullPointerException e) {
    out.println("[No Name]");
}

// 締日
out.println("CIRTSupplyChainTradeTransaction[" + i
    + "]/ApplicableCIRTSupplyChainTradeSettlement");
try {
    out.println("ClosingBookDueDateTime/DateTime[" + cira
        .getCIRTSupplyChainTradeTransaction().get(i)
        .getApplicableCIRTSupplyChainTradeSettlement()
        .getClosingBookDueDateTime().getDateTime() + "]);
} catch (NullPointerException e) {
    out.println("[No ClosingBookDueDateTime/DateTime]");
}
```

次に入金予定日、請求先連絡担当者、請求先連絡先部門を参照する。

```
// 入金予定日
out.println("CIRTSupplyChainTradeTransaction[" + i
    + "]/ApplicableCIRTSupplyChainTradeSettlement/");
try {
    out.println("ScheduledPaymentDateTime/DateTime[" + cira
        .getCIRTSupplyChainTradeTransaction().get(i)
        .getApplicableCIRTSupplyChainTradeSettlement()
        .getScheduledPaymentDateTime().getDateTime() + "]);
} catch (NullPointerException e) {
    out.println("[No ScheduledPaymentDateTime/DateTime]");
}

// 請求先連絡担当者
out.println("CIRTSupplyChainTradeTransaction[" + i
    + "]/ApplicableCIRTSupplyChainTradeSettlement"
    + "/InvoiceeCITradeParty/DefinedCITradeContact/");
try {
    out.println("PersonName[" + cira
        .getCIRTSupplyChainTradeTransaction().get(i)
        .getApplicableCIRTSupplyChainTradeSettlement()
        .getInvoiceeCITradeParty().getDefinedCITradeContact()
        .getPersonName().getValue() + "]);
} catch (NullPointerException e) {
    out.println("[No PersonName]");
}

// 請求先連絡先部門
out.println("CIRTSupplyChainTradeTransaction[" + i
    + "]/ApplicableCIRTSupplyChainTradeSettlement"
    + "/InvoiceeCITradeParty/DefinedCITradeContact/");
try {
    out.println("DepartmentName[" + cira
        .getCIRTSupplyChainTradeTransaction().get(i)
        .getApplicableCIRTSupplyChainTradeSettlement()
        .getInvoiceeCITradeParty().getDefinedCITradeContact()
        .getDepartmentName().getValue() + "]);
} catch (NullPointerException e) {
    out.println("[No DepartmentName]");
}
```

次に請求先電話番号、支払金額(明細)を参照する。ここまでが請求書毎の記述となる。
IncludedCIRLSupplyChainTradeLineItem オブジェクトは複数の納品伝票が記述されるため、繰り返し参照する。

```
// 請求先電話番号
out.println("CIRTSupplyChainTradeTransaction[" + i
    + "]/ApplicableCIRTSupplyChainTradeSettlement"
    + "/InvoiceeCITradeParty/DefinedCITradeContact"
    + "/TelephoneCIUniversalCommunication/");
try {
    out.println("CompleteNumber[" + cira
        .getCIRTSupplyChainTradeTransaction().get(i)
        .getApplicableCIRTSupplyChainTradeSettlement()
        .getInvoiceeCITradeParty().getDefinedCITradeContact()
        .getTelephoneCIUniversalCommunication()
        .getCompleteNumber().getValue() + "]");
} catch (NullPointerException e) {
    out.println("[No CompleteNumber]");
}

// 支払金額(明細)
out.println("CIRTSupplyChainTradeTransaction[" + i
    + "]/ApplicableCIRTSupplyChainTradeSettlement"
    + "/SpecifiedCIRTTradeSettlementMonetarySummation/");
try {
    out.println("PaymentTotalAmount[" + cira
        .getCIRTSupplyChainTradeTransaction().get(i)
        .getApplicableCIRTSupplyChainTradeSettlement()
        .getSpecifiedCIRTTradeSettlementMonetarySummation(
    )
        .getPaymentTotalAmount().getValue() + "]");
} catch (NullPointerException e) {
    out.println("[No PaymentTotalAmount]");
}

for(int ln = 0;
    ln < cira.getCIRTSupplyChainTradeTransaction().get(i)
        .getIncludedCIRLSupplyChainTradeLineItem().size();
    ln++) {
    out.println("¥n==" + ln + "=="¥n");
}
```

次に納品伝票番号、発注番号を参照する。

```
// 納品伝票番号
out.println("CIRTSupplyChainTradeTransaction[" + i
    + "]/IncludedCIRLSupplyChainTradeLineItem["+ ln + "]/"
    + "AssociatedCIRDocumentLineDocument/");
try {
    out.println("LineID[" + cira
        .getCIRTSupplyChainTradeTransaction().get(i)
        .getIncludedCIRLSupplyChainTradeLineItem()
        .get(ln)
        .getAssociatedCIRDocumentLineDocument()
        .getLineID()
        .getValue() + "]);
} catch (NullPointerException e) {
    out.println("[No LineID]");
}

// 発注番号
out.println("CIRTSupplyChainTradeTransaction[" + i
    + "]/getIncludedCIRLSupplyChainTradeLineItem["
    + ln + "]"
    + "/SpecifiedCIRLSupplyChainTradeSettlement[0]"
    + "/ReferencedCIRLSupplyChainTradeTransaction"
    + "/ApplicableCIRLSupplyChainTradeAgreement"
    + "/BuyerOrderReferencedCIRReferencedDocument/");
try {
    out.println("IssuerAssignedID[" + cira
        .getCIRTSupplyChainTradeTransaction().get(i)
        .getIncludedCIRLSupplyChainTradeLineItem()
        .get(ln)
        .getSpecifiedCIRLSupplyChainTradeSettlement()
        .get(0)
        .getReferencedCIRLSupplyChainTradeTransaction()
        .getApplicableCIRLSupplyChainTradeAgreement()
        .getBuyerOrderReferencedCIRReferencedDocument()
        .getIssuerAssignedID().getValue() + "]);
} catch (NullPointerException e) {
    out.println("[No IssuerAssignedID]");
} catch (IndexOutOfBoundsException e) {
    out.println(
        "[No SpecifiedCIRLSupplyChainTradeSettlement]");
}
```


次に納品伝票番号、発注番号を参照する。

```
// 受注番号
out.println("CIRTSupplyChainTradeTransaction[" + i
    + "]/getIncludedCIRLSupplyChainTradeLineItem["
    + ln + "]"
    + "/SpecifiedCIRLSupplyChainTradeSettlement[0]"
    + "/ReferencedCIRLSupplyChainTradeTransaction"
    + "/ApplicableCIRLSupplyChainTradeAgreement"
    + "/SellerOrderReferencedCIRReferencedDocument/");
try {
    out.println("IssuerAssignedID[" + cira
        .getCIRTSupplyChainTradeTransaction().get(i)
        .getIncludedCIRLSupplyChainTradeLineItem()
        .get(ln)
        .getSpecifiedCIRLSupplyChainTradeSettlement()
        .get(0)
        .getReferencedCIRLSupplyChainTradeTransaction()
        .getApplicableCIRLSupplyChainTradeAgreement()
        .getSellerOrderReferencedCIRReferencedDocument()
        .getIssuerAssignedID().getValue() + "]");
} catch (NullPointerException e) {
    out.println("[No IssuerAssignedID]");
} catch (IndexOutOfBoundsException e) {
    out.println(
        "[No SpecifiedCIRLSupplyChainTradeSettlement]");
}
```

次に単価を参照する。

```
// 単価
out.println("CIRTSupplyChainTradeTransaction[" + i
    + "]/getIncludedCIRLSupplyChainTradeLineItem["
    + ln + "]"
    + "/SpecifiedCIRLSupplyChainTradeSettlement[0]"
    + "/ReferencedCIRLSupplyChainTradeTransaction"
    + "/ApplicableCIRLSupplyChainTradeAgreement"
    + "/OrderPriceProductCITradePrice/");
try {
    out.println("ChargeAmount[" + cira
        .getCIRTSupplyChainTradeTransaction().get(i)
        .getIncludedCIRLSupplyChainTradeLineItem()
        .get(ln)
        .getSpecifiedCIRLSupplyChainTradeSettlement()
        .get(0)
        .getReferencedCIRLSupplyChainTradeTransaction()
        .getApplicableCIRLSupplyChainTradeAgreement()
        .getOrderPriceProductCITradePrice()
        .getChargeAmount().getValue() + "]");
} catch (NullPointerException e) {
    out.println("[No ChargeAmount]");
} catch (IndexOutOfBoundsException e) {
    out.println(
        "[No SpecifiedCIRLSupplyChainTradeSettlement]");
}
```

次に数量を参照する。

```
// 数量
out.println("CIRTSupplyChainTradeTransaction[" + i
    + "]/getIncludedCIRLSupplyChainTradeLineItem["
    + ln + "]"
    + "/SpecifiedCIRLSupplyChainTradeSettlement[0]"
    + "/ReferencedCIRLSupplyChainTradeTransaction"
    + "/ApplicableCIRLSupplyChainTradeDelivery/");
try {
    out.println("BilledQuantity[" + cira
        .getCIRTSupplyChainTradeTransaction().get(i)
        .getIncludedCIRLSupplyChainTradeLineItem()
        .get(ln)
        .getSpecifiedCIRLSupplyChainTradeSettlement()
        .get(0)
        .getReferencedCIRLSupplyChainTradeTransaction()
        .getApplicableCIRLSupplyChainTradeDelivery()
        .getBilledQuantity().getValue() + "]");
} catch (NullPointerException e) {
    out.println("[No BilledQuantity]");
} catch (IndexOutOfBoundsException e) {
    out.println(
        "[No SpecifiedCIRLSupplyChainTradeSettlement]");
}
```

次に納入番号を参照する。

```
// 納入番号
out.println("CIRTSupplyChainTradeTransaction[" + i
    + "]/getIncludedCIRLSupplyChainTradeLineItem["
    + ln + "]"
    + "/SpecifiedCIRLSupplyChainTradeSettlement[0]"
    + "/ReferencedCIRLSupplyChainTradeTransaction"
    + "/ApplicableCIRLSupplyChainTradeDelivery"
    + "/RequestedDeliveryCISupplyChainEvent/");
try {
    out.println("ID[" + cira
        .getCIRTSupplyChainTradeTransaction().get(i)
        .getIncludedCIRLSupplyChainTradeLineItem()
        .get(ln)
        .getSpecifiedCIRLSupplyChainTradeSettlement()
        .get(0)
        .getReferencedCIRLSupplyChainTradeTransaction()
        .getApplicableCIRLSupplyChainTradeDelivery()
        .getRequestedDeliveryCISupplyChainEvent()
        .getID().getValue() + "]");
} catch (NullPointerException e) {
    out.println("[No ID]");
} catch (IndexOutOfBoundsException e) {
    out.println(
        "[No SpecifiedCIRLSupplyChainTradeSettlement]");
}
```

次に納入日時を参照する。

```
// 納入日時
out.println("CIRTSupplyChainTradeTransaction[" + i
    + "]/getIncludedCIRLSupplyChainTradeLineItem["
    + ln + "]"
    + "/SpecifiedCIRLSupplyChainTradeSettlement[0]"
    + "/ReferencedCIRLSupplyChainTradeTransaction"
    + "/ApplicableCIRLSupplyChainTradeDelivery"
    + "/RequestedDeliveryCISupplyChainEvent/");
try {
    out.println("OccurrenceDateTime/DateTime[" + cira
        .getCIRTSupplyChainTradeTransaction().get(i)
        .getIncludedCIRLSupplyChainTradeLineItem()
        .get(ln)
        .getSpecifiedCIRLSupplyChainTradeSettlement()
        .get(0)
        .getReferencedCIRLSupplyChainTradeTransaction()
        .getApplicableCIRLSupplyChainTradeDelivery()
        .getRequestedDeliveryCISupplyChainEvent()
        .getOccurrenceDateTime().getDateTime() + "]");
} catch (NullPointerException e) {
    out.println("[No OccurrenceDateTime/DateTime]");
} catch (IndexOutOfBoundsException e) {
    out.println(
        "[No SpecifiedCIRLSupplyChainTradeSettlement]");
}
```

次に製品コードを参照する。

```
// 製品コード
out.println("CIRTSupplyChainTradeTransaction[" + i
    + "]/getIncludedCIRLSupplyChainTradeLineItem["
    + ln + "]"
    + "/SpecifiedCIRLSupplyChainTradeSettlement[0]"
    + "/ReferencedCIRLSupplyChainTradeTransaction"
    + "/IncludedCITradeProduct/");
try {
    out.println("BuyerAssignedID[" + cira
        .getCIRTSupplyChainTradeTransaction().get(i)
        .getIncludedCIRLSupplyChainTradeLineItem()
        .get(ln)
        .getSpecifiedCIRLSupplyChainTradeSettlement()
        .get(0)
        .getReferencedCIRLSupplyChainTradeTransaction()
        .getIncludedCITradeProduct()
        .getBuyerAssignedID().getValue() + "]");
} catch (NullPointerException e) {
    out.println("[No BuyerAssignedID]");
} catch (IndexOutOfBoundsException e) {
    out.println(
        "[No SpecifiedCIRLSupplyChainTradeSettlement]");
}
```

次に製品名を参照する。else 以降は、名前空間名が国連 CEFACT Remittance Advice の名前空間名でなかった時の処理をする。

```

        // 製品名
        out.println("CIRTSupplyChainTradeTransaction[" + i
            + "]/getIncludedCIRLSupplyChainTradeLineItem["
            + ln + "]"
            + "/SpecifiedCIRLSupplyChainTradeSettlement[0]"
            + "/ReferencedCIRLSupplyChainTradeTransaction"
            + "/IncludedCITradeProduct/");
        try {
            out.println("Name[" + cira
                .getCIRTSupplyChainTradeTransaction().get(i)
                .getIncludedCIRLSupplyChainTradeLineItem()
                .get(ln)
                .getSpecifiedCIRLSupplyChainTradeSettlement()
                .get(0)
                .getReferencedCIRLSupplyChainTradeTransaction()
                .getIncludedCITradeProduct()
                .getName().getValue() + "]");
        } catch (NullPointerException e) {
            out.println("[No Name]");
        } catch (IndexOutOfBoundsException e) {
            out.println(
                "[No SpecifiedCIRLSupplyChainTradeSettlement]");
        }
        out.println("¥n==  ==¥n");
    }
    out.println("¥n--  --¥n");
}

} else {

    // 不明な名前空間名の処理
    out.println("Unknown XML: namespace[" + ns + "]");
}
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

(5) 国連 CEFACT 基礎データ記述方法

① 金額データ(Amount)型記述方法

金額データ(Amount)型は金額を表現する数値を指定するために使用する。通貨単位を明示的または暗黙的に指定する。

金額データ型には、補足情報として通貨コードと通貨コードリストの版情報を付加することが出来る。通貨コードは XML 上では CurrencyID 属性、通貨コードリスト版情報は CurrencyCodeListVersionID 属性として記述する。通貨コードは ISO4217 あるいは国連 CEFACT 推奨 (UN/ECE Recommendation) の 9 番で規定されている 3 文字コードを用いる。

補足情報名	XML属性名	値
通貨コード	CurrencyID	ISO4217 あるいは国連 CEFACT 推奨の 9 番 (UN/ECE Recommendation 9) で規定されている 3 文字コード
通貨コードリストの版情報	CurrencyCodeListVersionID	

金額データ型の値は XML スキーマの decimal 型で記述する。

例えばタグ名を Amount とし、通貨コードは円 (JPY)、通貨コードリスト版情報は指定しないとした XML 形式で 100 円を記述すると以下の様になる。

```
<Amount CurrencyID="JPY">100</Amount>
```

規定上は通貨コード、通貨コードリスト版情報は任意記述であるが、金額データ型の場合、誤解を防ぐために通貨コードの記述を推奨する。しかし、XML スキーマのデフォルトまたは必須として指定されている場合は、記述が無くても明らかなため省略してもよい。

② バイナリーオブジェクト型記述方法

バイナリーオブジェクト型は画像等の仕様化されたデータを指定するために使用する。データ処理が可能な様にフォーマット等の仕様を明示的または暗黙的に指定する。

バイナリーオブジェクト型には、補足情報としてフォーマット情報、MIME コード、エンコーディングコード、文字コード、URL、ファイル名を付加することが出来る。

フォーマット情報は XML 上では Format 属性、MIME コードは mimeType 属性、エンコーディングコードは encodingCode 属性、文字コードは characterSetCode 属性、URL は url 属性、ファイル名は filename 属性として記述する。

MIME コードは、エンコーディングコード、文字コード、ファイル名は RFC2045、RFC2046、RFC2047 に定められているコードや記述方法を用いる。

補足情報名	XML属性名	値
フォーマット情報	Format	
MIME コード	mimeCode	RFC2045 (RFC2231 、 RFC6532) 、
エンコーディングコード	encodingCode	RFC2046 (RFC3676 、 RFC5174 、
文字コード	characterSetCode	RFC6657) 、 RFC2047 (RFC2231) に定められているコードや記述方法を用いる。
ファイル名	filename	RFC2048 (RFC6838、RFC4285) に定められている様に IANA が登録管理しているコードも参照されたい。
URL	url	

バイナリーオブジェクト型の値は XML 上では XML スキーマの base64binary 型で記述する。Base64binary 型は base64 エンコーディングされた文字列である。

例えばタグ名を BinaryObject とし、MIME コードを text/xml とし、エンコーディングコードを、フォーマット情報、文字コード、URL、ファイル名は指定しないとした XML 形式で値「abcdefghij」、即ち base64 エンコード結果を「YWJjZGVmZ2hpag==」すると以下の様になる。

```
<BinaryObject mimeCode="text/xml">YWJjZGVmZ2hpag==</BinaryObject>
```

規定上はフォーマット情報、MIME コード、エンコーディングコード、文字コード、URL、ファイル名は任意記述であるが、バイナリーオブジェクト型の場合、誤解を防ぐために MIME コード、エンコーディングコードの記述を推奨する。しかし、XML スキーマのデフォルトまたは必須として指定されている場合は、記述が無くても明らかなため省略してもよい。

③ コード(Code)型記述方法

コード(Code)型は符号化された文字列を含む値に使用する。コード値を理解するためのコードリストの情報が明示的または暗黙的に指定する。

コード型には、補足情報としてコードリストの識別値と名前、コードリスト管理者の識別値、名前と版情報、コード名、言語識別子、URI、コードリスト体系 URI を付加することが出来る。コードリストの識別値は、XML 上では listID 属性、コードリストの名前は listName 属性、コードリスト管理者の識別値は listAgencyID 属性、コードリスト管理者の名前は listAgencyName 属性、コードリスト管理者の版情報は listVersionID 属性、コード名は name 属性、言語識別子は languageID 属性、URI は listURI 属性、コードリスト体系 URI は listSchemeURI 属性として記述する。

コードリスト管理者の識別値は原則として UN/EDIFACT のデータエレメント 3055(コード管理機関)コードリストで規定されているコードを用いる。コードリスト管理者の版情報は原則として UN/EDIFACT のデータエレメント 3055(コード管理機関)コードリストの版情報の値を用いるが、記述されていない場合は最新判とする。コードリストの識別子は URL 等で指定することも出来るが、コードリスト管理者が管理するコードリストの識別子を用いることを推奨する。

補足情報名	XML属性名	値
コードリストの識別値	listID	
コードリストの名前	listName	
コードリスト管理者の識別値	listAgencyID	原則として UN/EDIFACT のデータエレメント 3055(コード管理機関)コードリストで規定されているコードを用いる。
コードリスト管理者の名前	listAgencyName	
コードリスト管理者の版情報	listVersionID	原則として UN/EDIFACT のデータエレメント 3055(コード管理機関)コードリストの版情報の値を用いるが、記述されていない場合は最新判とする。
コード名	name	
言語識別子	languageID	
URI	listURI	
コードリスト体系 URI	listSchemeURI	

コード値は補足情報で指定されるコードリストの値とする。コード型の値は XML 上では XML スキーマの token 型で記述する。原則としてコード値の値に制限されるため、XML スキーマで検証することが出来る。

例えばタグ名を Code とし、コードリストの識別値は 3055、コードリスト管理者の識別値は 6、コードリストの名前、コードリスト管理者の名前と版情報、コード名、URI、コードリスト体系 URI は指定しないとした XML 形式でコード値 402 を記述すると以下の様になる。なお、コードリスト管理者の識別値は 6 は UN/ECE(国連 CEFAC)を表し、コードリストの識別値 3055 は国連 CEFAC が管轄する UN/EDIFACT のコード管理機関コードリストを表し、コード値 402 はコード管理機関コードリストに登録されている国税庁になる。

```
<Code listID="3055" listAgencyID="6">402</Code>
```

規定上はコードリストの識別値、名前と版情報、コードリスト管理者の識別値と名前、コード名、URI、コードリスト体系 URI は任意記述であるが、コードリストの識別値とコードリスト管理者の識別値の記述を推奨する。しかし、XML スキーマのデフォルトまたは必須として指定されている場合は、記述が無くても明らかなため省略してもよい。

④ 日付・時刻データ型記述方法

日付・時刻データ型は日付と時刻を含む値に使用する。日付だけ、時刻だけ、日付と時刻の両方の 3 種類の指定方法がある。その他にフォーマットを陽に指定して記述する方法がある。

日付だけを指定する際には形式 "YYYY-MM-DD" を用いる。YYYY は年を、MM は月を、DD は日を表し、全コンポーネントが必須である。

コード値は補足情報で指定されるコードリストの値とする。コード型の値は XML スキーマの token 型で記述する。原則としてコード値の値に制限され、XML スキーマで検証することが出来る。

例えばタグ名を StartDate とし、2002 年 9 月 24 日を XML 形式で記述すると以下の様になる。

```
<StartDate><Date>2002-09-24</Date></StartDate>
```

タイムゾーンの指定には、次のように UTC 時間の日付の後ろに "Z" を追加して入力する。

```
<StartDate><Date>2002-09-24Z</Date></StartDate>
```

または、次のように UTC 日付の後ろにオフセットとして正または負の時間を追加して指定することができる。

```
<StartDate><Date>2002-09-24-06:00</Date></StartDate> または
<StartDate><Date>2002-09-24+06:00</Date></StartDate>
```

時刻だけを指定する際には形式 "hh:mm:ss" を用いる。hh は時間を、mm は分を、ss は秒を表し、全コンポーネントが必須である。

例えばタグ名を StartTime とし、9 時 30 分 10 秒を XML 形式で記述すると以下のようになる。

```
<StartTime><Time>09:30:10</Time></StartTime>
```

例えば 9 時 30 分 10.5 秒の様な 1 秒未満の値がある時刻を XML 形式で記述することも出来て、以下のようになる。

```
<StartTime><Time>09:30:10.5</Time></StartTime>
```

タイムゾーンの指定には、次のように UTC の場合は時刻の後ろに "Z" を追加する。

```
<StartTime><Time>09:30:10Z</Time></StartTime>
```

または、次のように時刻の後ろにオフセットとして正または負の時間を追加して指定することができる。

```
<StartTime><Time>09:30:10-06:00</Time></StartTime> または
<StartTime><Time>09:30:10.5+06:00</Time></StartTime>
```

日付と時刻を指定する際には、形式 "YYYY-MM-DDThh:mm:ss" を用いる。YYYY は年を表し、MM は月を表し、DD は日を表し、T は必須な時刻セクションの開始を表し、hh は時間を表し、mm は分を表し、ss は秒を表し、注:全コンポーネントが必須である。

例えばタグ名を Start とし、2002 年 9 月 24 日 9 時 30 分 10 秒および 2002 年 9 月 24 日 9 時 30 分 10.5 秒を XML 形式で記述すると以下のようになる。

```
<Start><Date>2002-09-24T09:30:10</Date></Start>
<Start><Date>2002-09-24T09:30:10.5</Date></Start>
```

タイムゾーンの指定には、次のように UTC の場合は日時の後ろに "Z" を追加する。

```
<Start><Date>2002-09-24T09:30:10Z</Date></Start>
```

または、次のように日時の後ろにオフセットとして正または負の時間を追加して指定する。

```
<Start><DateTime>2002-09 T09:30:10-24-06:00</DateTime></Start> または
<Start><DateTime>2002-09 T09:30:10.5-24+06:00</DateTime></Stare>
```

フォーマットを陽に指定して記述する方法は、期間データなどに用いる。

フォーマットは補足情報として指定し、XML 上では `format` 属性およびタグ `DateTimeString` を記述する。

値は XML 上では XML スキーマの `string` 型で記述する。このため XML スキーマでフォーマットの検証は行われない。

期間データ型は、時間間隔を指定するために使用する。時間間隔は、形式 "`PnYnMnDTnHnMnS`" で指定する。

`P` は期間を表し(必須)、`nY` は年数を表し、`nM` は月数を表し、`nD` は日数を表し、`T` は時刻セクションの開始を表し(時間、分、秒を指定する場合は必須)、`nH` は時間数を表し、`nM` は分数を表し、`nS` は秒数を表す。

例えばタグ名を `Duration` とし、期間 5 年間、期間 5 年 2 か月と 10 日、期間 5 年 2 か月 10 日と 15 時間、期間 15 時間を XML 形式で記述すると以下の様になる。

```
<Duration format="PnY">
  <DateTimeString>P5Y</DateTimeString>
</Duration>

<Duration format="PnYnMnD">
  <DateTimeString>P5Y2M10D</DateTimeString>
</Duration>

<Duration format="PnYnMnDTnH ">
  <DateTimeString>P5Y2M10DT15H</DateTimeString>
</Duration>

<Duration format=" PTnH ">
  <DateTimeString>PT15H</DateTimeString>
</Duration>
```

`P` の前にマイナスを付けると負の期間を指定できる。

例えばタグ名を Duration とし、10 日前から今日までの期間を XML 形式で記述すると以下の様になる。

```
<Duration format="PnD">
  <DateTimeString>-P10D</DateTimeString>
</Duration>
```

⑤ ID データ(Identifier)型記述方法

ID コード(Identifier)型は実体に対して符号化された文字列を対応させる際に使用する。ID コード値を理解するための ID コードリストの情報を明示的または暗黙的に指定する。

ID データ型には、補足情報として ID リストの識別値と名前、ID リスト管理者の識別値、名前と版情報、ID リストの URI、ID リスト体系 URI を付加することが出来る。ID リストの識別値は、XML 上では schemeID 属性、ID リストの名前は schemeName 属性、ID リスト管理者の識別値は schemeAgencyID 属性、ID リスト管理者の名前は schemeAgencyName 属性、ID リスト管理者の版情報は schemeVersionID 属性、ID リストの URI は schemeDataURI 属性、ID リスト体系 URI は SchemeURI 属性として記述する。

ID リスト管理者の識別値は原則として UN/EDIFACT のデータエレメント 3055(コード管理機関)コードリストで規定されているコードを用いる。ID リスト管理者の版情報は原則として UN/EDIFACT のデータエレメント 3055(コード管理機関)コードリストの版情報の値を用いるが、記述されていない場合は最新判とする。ID リストの識別値はコードリスト管理者が管理する ID リストの識別値の一覧から指定することを推奨する。

補足情報名	XML属性名	値
ID リストの識別値	schemeID	
ID リストの名前	schemeName	
ID リスト管理者の識別値	schemeAgencyID	原則として UN/EDIFACT のデータエレメント 3055(コード管理機関)コードリストで規定されているコードを用いる。
ID リスト管理者の名前	schemeAgencyName	
ID リスト管理者の版情報	schemeVersionID	原則として UN/EDIFACT のデータエレメント 3055(コード管理機関)コードリストの版情報の値を用いるが、記述されていない場合は最新判とする。
ID リストの URI	schemeDataURI	
ID リスト体系の URI	schemeURI	

ID コード値は補足情報で指定される ID リストの値とする。ID コード型の値は XML 上では XML スキーマの token 型で記述する。

例えばタグ名を ID とし、ID リストの識別値は 3055、ID リスト管理者の識別値は 6、ID リストの名前、ID リスト管理者の名前と版情報、コード名、URI、コードリスト体系 URI は指定しないとした XML 形式でコード値 402 を記述すると以下の様になる。なお、コードリスト管理者の識別値は 413 は国連 CEFAC 日本委員会を表し、ID リストの識別値 JEC0001 は国連 CEFAC 日本委員会が管理している業界区分コード表

を表し、ID コード値 JPSFEDI は金融 EDI になる。

```
<ID schemeID="JEC0001" schemeAgencyID="413">JPSFEDI</ID>
```

ID リストの識別値は、国連 CEFACT が管理または登録されているコード表および国連 CEFACT 日本委員会が管理または登録されているコード表から設定することを推奨する。なお、企業等が個別に管理している公開されないコード表の多くについても ID リストの識別値の設定方法を規定している。

規定上は ID リストの識別値と名前、ID リスト管理者の識別値、名前と版情報、ID リストの URI、ID リスト体系の URI は任意記述であるが、ID リストの識別値と ID リスト管理者の識別値の記述を推奨する。しかし、XML スキーマのデフォルトまたは必須として指定されている場合は、記述が無くても明らかなため省略してもよい。

⑥ 真偽値 (Indicator) 型記述方法

真偽値 (Indicator) 型は真と偽からなる。XML スキーマの booleana 型を使う場合とフォーマットを陽に指定して記述する方法がある。

XML スキーマの boolean 型を使う場合は、真の場合 true または 1、偽の場合 false または 0 を記述する。

例えばタグ名を Boolean とし、真を XML 形式で記述すると以下のようになる。

```
< Boolean ><Indicator>true</Indicator></Boolean>
```

フォーマットを陽に指定して記述する方法は、フォーマットを補足情報として指定し、XML 上では format 属性およびタグ IndicatorString を記述する。

値は XML 上では XML スキーマの string 型で記述する。このため XML スキーマでフォーマットの検証は行われない。

例えばタグ名を Boolean とし、フォーマットを on/off とし、on を XML 形式で記述すると以下のようになる。

```
< Boolean format="on/off"><IndicatorString>on</IndicatorString></Boolean>
```

真偽値型のフォーマットの規定は一般的で無いため、フォーマットを陽に指定する方法は相互運用性が低いことに注意されたい。

⑦ 単位付き数値 (Measure) 型記述方法

単位付き数値 (Measure) 型は、長さ 1 メートルの様に単位を持つ物理的の様な数値を指定するために使用する。単位を明示的または暗黙的に指定する。

単位付き数値型には、補足情報として単位コードと単位コードリストの版情報を付加することが出来る。単位コードは XML 上では `unitCode` 属性、単位コードリスト版情報は `unitCodeListVersionID` 属性として記述する。単位コードは国連 CEFACT 推奨 (UN/ECE Recommendation) の 20 番で規定されているコードを用いる。

補足情報名	XML属性名	値
単位コード	<code>unitCode</code>	国連 CEFACT 推奨の 20 番 (UN/ECE Recommendation 20) で規定されているコード
単位コードリストの版情報	<code>unitCodeListVersionID</code>	

単位付き数値型の値は XML スキーマの `decimal` 型で記述する。

例えばタグ名を `Measure` とし、単位コードはメートル (`MTR`)、単位コードリスト版情報は指定しないとした XML 形式で 100 メートルを記述すると以下の様になる。

```
<Measure unitCode="MTR">100</Measure>
```

規定上は単位コード、単位コードリスト版情報は任意記述であるが、単位付く数値型の場合、誤解を防ぐために単位コードの記述を推奨する。しかし、XML スキーマのデフォルトまたは必須として指定されている場合は、記述が無くても明らかなため省略してもよい。

⑧ 数値 (Numeric) 型記述方法

数値 (Numeric) 型は、単位の無い数値を指定するために使用する。補足情報としてフォーマットを付加することが出来る。

補足情報名	XML属性名	値
フォーマット	<code>format</code>	

数値型の値は XML スキーマの `decimal` 型で記述する。

例えばタグ名を `Numeric` とし、フォーマットは指定しないとした XML 形式で 100 を記述すると以下の様になる。

```
<Numeric>100</Numeric>
```

フォーマットを指定しても、XML スキーマでフォーマットの検証は行われないことに注意されたい。

⑨ 数量(Quantity)型記述方法

数量(Quantity)型は、個数 1 ダースの様に単位を持つ、数え上げる様な数値を指定するために使用する。単位を明示的または暗黙的に指定する。

数量型には、補足情報として単位コード、単位コードリストの識別値、単位コードリスト管理者の識別値、単位コードリスト管理者の名前を付加することが出来る。単位コードは XML 上では unitCode 属性、単位コードリストの識別値は unitCodeListID 属性、単位コードリスト管理者の識別値は unitCodeListAgencyID 属性として記述する。単位コードは国連 CEFACT 推奨 (UN/ECE Recommendation) の 20 番で規定されているコードを用いる。

補足情報名	XML属性名	値
単位コード	unitCode	国連 CEFACT 推奨の 20 番 (UN/ECE Recommendation 20) で規定されているコード
単位コードリストの識別値	unitCodeListID	
単位コードリストの管理者識別値	unitCodeListAgencyID	
単位コードリスト管理者名	unitCodeListAgencyName	

数量の値は XML スキーマの decimal 型で記述する。

例えばタグ名を Quantity とし、単位コードはダース(DZN)、単位コードリスト識別値、単位コードリスト管理者識別値、単位コードリスト管理者名は指定しないとした XML 形式で 10 ダースを記述すると以下の様になる。

```
<Quantity unitCode="DZN">10</Quantity>
```

規定上は単位コード、単位コードリスト識別値、単位コードリスト管理者識別値、単位コードリスト管理者名単位コードリストの識別値は任意記述であるが、数量型の場合、誤解を防ぐために単位コードの記述を推奨する。しかし、XML スキーマのデフォルトまたは必須として指定されている場合は、記述が無くても明らかなため省略してもよい。

⑩ テキスト(Text)型記述方法

テキスト(Text)型は文字列を指定するために使用する。

テキスト型には、補足情報として言語コードとロケールコードを付加することが出来る。言語コードは XML 上では languageID 属性、ロケールコードは languageLocaleID 属性として記述する。言語コードは ISO 6391 で規定されている 2 文字コードを用いる。

補足情報名	XML属性名	値
言語コード	languageID	ISO6391 で規定されている 2 文字コード
ロケールコード	languageLocaleID	

テキスト型の値は XML スキーマの string 型で記述する。

例えばタグ名を Text とし、言語コードは日本語(ja)、ロケールコードは指定しないとした XML 形式で日本語を記述すると以下のようになる。

```
<Text languageID="ja">日本語</Text>
```

規定上は言語コード、ロケールコードは任意記述である。言語コード、ロケールコードを指定しても、XML スキーマで検証するものではないことに注意されたい。

第5章 参考文献

- [1] 全国銀行協会：XML 形式 適用業務およびレコード・フォーマット, 平成 29 年 8 月,
https://www.zenginkyo.or.jp/fileadmin/res/abstract/efforts/smooth/xml/XML_news290829.pdf
- [2] ISO 20022 - Registration Authority :Pain.001.001.03 XML schema,
https://www.iso20022.org/documents/messages/1_0_version/pain/schemas/pain.001.001.03.zip
- [3] ISO 20022 - Registration Authority :Camt.052.001.02 XML schema,
https://www.iso20022.org/documents/messages/1_0_version/camt/schemas/camt.052.001.02.zip)
- [4] ISO 20022 - Registration Authority :Camt.054.001.02 XML schema,
https://www.iso20022.org/documents/messages/1_0_version/camt/schemas/camt.054.001.02.zip
- [5] ISO 20022 - Registration Authority:ISO 20022 Message Archive,
https://www.iso20022.org/message_archive.page
- [6] 経済産業省 中小企業庁：金融 EDI 情報として格納すべき商流情報の整理について, 平成 28 年 12 月 22 日, <http://www.chusho.meti.go.jp/koukai/kenkyukai/kinyuedi/2016/161222kinyuedi.pdf>
- [7] サプライチェーン情報基盤研究会：ビジネスインフラ 業界横断 EDI 仕様 V4, 平成 29 年 8 月 6 日,
http://www.caos-a.co.jp/SIPS/bizinfra/CI_Spec4.html

第6章 付録

(1) XML スキーマ

① Pain.001 の XML スキーマ(チェック用)

国際標準(Pain.001.001.03)で規定しているメッセージ仕様に合っているかを検証するために利用できる。
全銀 EDI システム向けの利用条件を盛り込んだ XML スキーマは今後作成したい。

ファイル名 : pain.001.001.03.xsd

② Pain.001 の XML スキーマ(JAXB 用)

JAXB の xjc コマンドが処理できる様にルートエレメント定義を修正した総合振込(Pain.001.001.03)用の XML スキーマ

ファイル名 : pain.001.001.03-jaxb.xsd

③ Pain.002 の XML スキーマ(チェック用)

国際標準(Pain.002.001.03)で規定しているメッセージ仕様に合っているかを検証するために利用できる。
全銀 EDI システム向けの条件を盛り込んだ XML スキーマは今後作成したい。

ファイル名 : pain.002.001.03.xsd

④ Pain.002 の XML スキーマ(JAXB 用)

JAXB の xjc コマンドが処理できる様にルートエレメント定義を修正した総合振込結果明細(Pain.002.001.03)用の XML スキーマ

ファイル名 : pain.002.001.03-jaxb.xsd

⑤ Bah.001 の XML スキーマ(チェック用)

国際標準(Head.001.001.01)で規定しているメッセージ仕様に合っているかを検証するために利用できる。全銀 EDI システム向けの条件を盛り込んだ XML スキーマは今後作成したい。

ファイル名: head.001.001.01.xsd

⑥ Bah.001 の XML スキーマ(JAXB 用)

JAXB の xjc コマンドが処理できる様にルートエレメント定義を修正した Head.001.001.01 の XML スキーマ

ファイル名: head.001.001.01-jaxb.xsd

⑦ Camt.052 の XML スキーマ(チェック用)

国際標準(Camt.052.001.02)で規定しているメッセージ仕様に合っているかを検証するために利用できる。全銀 EDI システム向けの条件を盛り込んだ XML スキーマは今後作成したい。

ファイル名: camt.052.001.02.xsd

⑧ Camt.052 の XML スキーマ(JAXB 用)

JAXB の xjc コマンドが処理できる様にルートエレメント定義を修正した入出金取引明細(Camt.052.001.02)用の XML スキーマ

ファイル名: camt.052.001.02-jaxb.xsd

⑨ Camt.054 の XML スキーマ(チェック用)

国際標準(Camt.054.001.02)で規定しているメッセージ仕様に合っているかを検証するために利用できる。全銀 EDI システム向けの条件を盛り込んだ XML スキーマは今後作成したい。

ファイル名: camt.054.001.02.xsd

⑩ Camt.054 の XML スキーマ(出力チェック用)

JAXB の xjc コマンドが処理できる様にルートエレメント定義を修正した振込入金通知 (Camt.054.001.02)用の XML スキーマ

ファイル名: camt.054.001.02-jaxb.xsd

⑪ Remittance Advice の XML スキーマ(チェック用)

商流情報項目を国連 CEFACT 支払案内 (Remittance Advice) にマッピングしたメッセージ仕様に合っているかを検証するために利用できる。

ファイル名: CrossIndustryRemittanceAdvice.xsd

⑫ Remittance Advice の XML スキーマ(JAXB 用)

JAXB の xjc コマンドが処理できる様にルートエレメント定義を修正した国連 CEFACT 支払案内の XML スキーマ。

ファイル名: CrossIndustryRemittanceAdvice-jaxb.xsd

(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		

(4) メッセージ例

① Pain.001 の例

ファイル名 : pain.001.001.03.xml

② Pain.002 の例

ファイル名 : pain.002.001.03_1.xml

③ Camt.052 の例

ファイル名 : camt.052.001.02.xml

④ Camt.054 の例

ファイル名 : camt.054.001.02.xml

⑤ Bah.001 の例

ファイル名 : head.001.001.01.xml

⑥ Remittance Advice の例

ファイル名 : CrossIndustryRemittanceAdvice.xml

(5) ソフトウェア例

① Pain.001 の例

金流情報と金融 EDI 情報から Pain.001 メッセージを作成し、XML を標準出力に出す動作検証済のプログラム

ファイル名:Pain001w.java

② Pain.002 の例

Pain.002 のメッセージを読み込み、メッセージ ID を標準出力に出す動作検証済のプログラム

ファイル名:Pain002r.java

③ Camt.052 の例

Camt.052 メッセージを読み込み、メッセージ ID と金融 EDI 情報を標準出力に出す動作検証済のプログラム

ファイル名:Camt052r.java

④ Camt.054 の例

Camt.054 メッセージを読み込み、メッセージ ID を標準出力に出す動作検証済のプログラム

ファイル名:Camt054r.java

⑤ Bah.001 の例

Bah.001 メッセージのを作成する動作検証済のプログラム

ファイル名:Bah001w.java

Bah.001 メッセージを読み込み、情報項目の値を標準出力に出す動作検証済のプログラム

ファイル名:Bah001r.java

⑥ Remittance Advice の例

Cross Industry Remittance Advice メッセージのを作成する動作検証済のプログラム

ファイル名:RAOUT.java

Cross Industry Remittance Advice メッセージを読み込み、情報項目の値を標準出力に出す動作検証済のプログラム

ファイル名:RAIN.java

(6) JAXB の使用例

まず XML スキーマから JAVA ソースコードを生成する `xjc` コマンドを使い XML スキーマから関数を生成する。

```
C:\tmp>xjc pain.001.001.03.xsd
スキーマの解析中...
スキーマのコンパイル中...
iso\std\iso\20022\tech\xsd\pain_001_001\AccountIdentification4Choice.java
iso\std\iso\20022\tech\xsd\pain_001_001\AccountSchemeName1Choice.java
...
iso\std\iso\20022\tech\xsd\pain_001_001\TaxRecordPeriod1Code.java
iso\std\iso\20022\tech\xsd\pain_001_001\package-info.java
```

名前空間から算出されたディレクトリに関数が生成される。生成されるディレクトリは名前空間名から自動的に計算される。`Pain.001.001.03` の場合以下の様になる。

名前空間名 : `urn:iso:std:iso:20022:tech:xsd:pain.001.001.03`
 ディレクトリ : `iso\std\iso\20022\tech\xsd\pain_001_001`

また JAVA のインポートで指定する場合は以下の様になる。個々のクラス名を直接指定してもよいが、XML スキーマの変更によりクラス名が変更されることもある。

```
Import iso.std.iso._20022.tech.xsd.pain_001_001.*;
```

次に `javac` コマンドでコンパイルする。

```
C:\tmp>dir iso\std\iso\20022\tech\xsd\pain_001_001
...
C:\tmp\iso\std\iso\20022\tech\xsd\pain_001_001 のディレクトリ

2018/03/07 17:52 <DIR> .
2018/03/07 17:52 <DIR> ..
2018/03/07 17:52      2,708 AccountIdentification4Choice.java
2018/03/07 17:52      2,563 AccountSchemeName1Choice.java
2018/03/07 17:52      2,546 ActiveOrHistoricCurrencyAndAmount.java
...
C:\tmp>javac iso\std\iso\20022\tech\xsd\pain_001_001\*.java
```

Pain.001.001.03 から作成された関数の例を示す。

```
//
// このファイルは、JavaTM Architecture for XML Binding(JAXB) Reference
// Implementation、v2.2.8-b130911.1802 によって生成されました
// <a href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jaxb</a>を参照してくだ
// さい
// ソース・スキーマの再コンパイル時にこのファイルの変更は失われます。
// 生成日: 2018.03.02 時間 07:55:45 PM JST
//

package iso.std.iso._20022.tech.xsd.pain_001_001;

import java.util.ArrayList;
import java.util.List;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlType;

/**
 * <p>CustomerCreditTransferInitiationV03 complex type の Java クラス。
 *
 * <p>次のスキーマ・フラグメントは、このクラス内に含まれる予期されるコンテンツを指定します。
 *
 * <pre>
 * <complexType name="CustomerCreditTransferInitiationV03">
 *   <complexContent>
 *     <restriction base="{http://www.w3.org/2001/XMLSchema}anyType">
 *       <sequence>
 *
 *                                     <element      name="GrpHdr"
type="{urn:iso:std:iso:20022:tech:xsd:pain.001.001.03}GroupHeader32"/>
 *                                     <element      name="PmtInf"
type="{urn:iso:std:iso:20022:tech:xsd:pain.001.001.03}PaymentInstructionInformation3
" maxOccurs="unbounded"/>
 *       </sequence>
 *     </restriction>
 *   </complexContent>
 * </complexType>
 * </pre>
 *
 *
 */
```


作成された関数の例の続きを示す。JAXB は原則としてタグ毎に値を取得する関数 (getter) と、値を設定する関数 (setter) を作成する。GrpHdr に対応する getGrpHdr と setGrpHdr が生成されている。

```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "CustomerCreditTransferInitiationV03", propOrder = {
    "grpHdr",
    "pmtInf"
})
public class CustomerCreditTransferInitiationV03 {

    @XmlElement(name = "GrpHdr", required = true)
    protected GroupHeader32 grpHdr;
    @XmlElement(name = "PmtInf", required = true)
    protected List<PaymentInstructionInformation3> pmtInf;

    /**
     * grpHdr プロパティの値を取得します。
     *
     * @return
     *     possible object is
     *     {@link GroupHeader32 }
     */
    public GroupHeader32 getGrpHdr() {
        return grpHdr;
    }

    /**
     * grpHdr プロパティの値を設定します。
     *
     * @param value
     *     allowed object is
     *     {@link GroupHeader32 }
     */
    public void setGrpHdr(GroupHeader32 value) {
        this.grpHdr = value;
    }
}
```

作成された関数の例の続きを示す。2 個以上の繰り返しの可能性があるタグについては、JAVA の `List` を対応させるため、`List` 型のオブジェクトを取得する関数 (`getter`) のみが生成される。最初の取得の際に空の `List` が生成される。空の `List` が生成されているオブジェクトを XML 化すると空タグが生成されるので、注意されたい。

```
/**
 * Gets the value of the pmtInf property.
 *
 * <p>
 * This accessor method returns a reference to the live list,
 * not a snapshot. Therefore any modification you make to the
 * returned list will be present inside the JAXB object.
 * This is why there is not a <CODE>set</CODE> method for the pmtInf property.
 *
 * <p>
 * For example, to add a new item, do as follows:
 * <pre>
 *     getPmtInf().add(newItem);
 * </pre>
 *
 * <p>
 * Objects of the following type(s) are allowed in the list
 * {@link PaymentInstructionInformation3 }
 *
 */
public List<PaymentInstructionInformation3> getPmtInf() {
    if (pmtInf == null) {
        pmtInf = new ArrayList<PaymentInstructionInformation3>();
    }
    return this.pmtInf;
}

}
```

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
