Production Planning and Scheduling (PPS) Version 2.0

Working Draft 06

11 Apr 2014

Specification URIs:

This version:

http://docs.oasis-open.org/pps/pps/v1.0/csprd01/pps-v2.0-d01.docx (Authoritative)

Previous version:

目次

はじめに 4
◆スコープ 4
◆方針 4
用語の説明 5
◆Application domain profile（業務プロファイル） 5
◆Implement profile（実装プロファイル） 5
◆Transaction 5
◆Domain document（業務ドキュメント） 5
◆Domain object（業務オブジェクト） 5
◆Domain property（業務プロパティ） 6
◆Messaging model 6
◆Primitive Domain object（プリミティブ業務オブジェクト） 6
データ連携の方法 7
メッセージ交換の概要 7
メッセージ交換モデルの形式 7
通知（Notify） 8
追加（Add） 8
修正（Change） 9
削除（Remove） 10
同期（Sync） 10

照会（Get） 11
同期処理のイベント通知 12
トランザクション処理 12
メッセージの構造 14
メッセージの構造 14
トランザクション情報 15
連携ドキュメント情報 17
連携データの構造 19
業務プロファイル 23
業務プロファイルの定義 23
業務ドキュメントの定義情報 25
業務オブジェクトの定義 25
業務プロパティの定義 26
業務プロファイルの拡張 27
実装プロファイル 28
実装プロファイルの構造 28
実装ドキュメントの定義 30
実装アクション 31
実装プロパティ 31
メッセージ交換方法 33
トランザクション処理 33
エラー情報の設定方法 33
連携ドキュメントの処理方法 34
◆メッセージ受信者の義務 34
トランザクションの返信 35
同期の場合のメッセージ交換パターン 36
連携プログラムの処理 37
◆追加（Add）処理 37
◆修正（Change）処理 38
◆削除（Remove）処理 39
◆照会（Get）処理 40
業務プロファイルの照会 41
実装プロファイルの交換 41
◆回答（Show）処理 42
同期と事象通知 44
同期要求トランザクションの内容 44
同期確認トランザクションの内容 45
事象通知トランザクションの内容 46

事象監視トランザクションの内容 46
監視可能な事象の定義 48
業務オブジェクト 54
業務オブジェクトの構造 54
◆プリミティブ要素 55
◆関係属性要素 56
◆複合属性要素 57
◆単純属性要素 58
◆データ要素 60
◆管理要素 60
Conformance 方法 62

はじめに

◆スコープ

この仕様書では、生産計画やスケジューリングなど、企業組織における意思決定に関連する業務アプリケーションが、相互に連携するためにメッセージ交換する場合において、交換するメッセージの作成方法およびその内容の解釈方法と、メッセージ交換ルールを定める。

メッセージ交換方法としてこの仕様書で規定する内容は、アプリケーションプログラムが認識できるレイヤーのものであり、より低レベルに位置づけられる通信プログラムのレイヤーの形式を制約するものではない。通信プログラムのレイヤーにおける通信方式は、HTTP やその他のプロトコルなどが想定されるが、この仕様書では何も規定しない。

受け取ったメッセージへの返信方法に関して、同期あるいは非同期のいずれかで行うかについてはプログラムの実装方法の問題として扱うものとし、の仕様書では既定しない。

◆方針

この仕様書の記述は、業務アプリケーションが独自にもつ業務オブジェクトモデルあるいは業務データスキーマに影響されない。

ただし、データベースの実装方法として、一般的な RDB のみならず、クラウド型のデータベースへの対応も意識した設計とする。

メッセージ交換の方式は、粗結合型とし、それぞれの業務アプリケーションが、連携先の業務アプリケーションがおかれた状況（文脈）に依存しないものとする。つまり、連携相手の業務アプリケーションがどのようなどのような機能をもち、どのような処理の途中であるとしても、同一の業務アクティビティに対して同様な手順でメッセージを送受信できなければならない。

用語の説明

◆Application domain profile（業務プロファイル）

メッセージを交換する双方の業務アプリケーションがメッセージ内容の解釈のための前提とする情報。１つの業務プロファイルをあるグループの属する複数の業務アプリケーションで共有することで、そのグループに属する業務アプリケーションがお互いにメッセージ交換可能となる。業務プロファイルによって、ドメインやグループ内であらかじめオブジェクトの定義などを設定しておく。これにより、メッセージの XML 要素のどこに値を指定するかがわかる。現状は、PSLX プロファイルがある。

◆Implement profile（実装プロファイル）

それぞれの企業組織におけるそれぞれの業務アプリケーションの実装の状況を定義した情報。Application Profile の中で、どの部分を実装しているかを各プログラム単位で宣言するもの。対象とする業務プロファイルと、業務プロファイル内で対応可能な業務オブジェクト、業務ドキュメントおよび業務プロパティを指定する。さらに、各オブジェクト、各項目などを業務アプリケーションが独自にどのような名前をつけて呼んでいるかも示す。

◆Transaction

業務アプリケーションがメッセージを処理する単位。トランザクション内でエラーがおこったときには、ロールバックし、処理前の状態にもどす。トランザクション内で複数種類の異なる業務ドキュメントを扱うことができる。業務ドキュメント間の関係を定義できる。１回のトランザクションが扱う情報の単位は、一般に業務で利用している伝票や帳票に相当する。

◆Domain document（業務ドキュメント）

実際の業務が扱う情報の単位。１種類の１つ以上の業務オブジェクトで構成される。業務ドキュメントの単位で、追加、修正、削除といった操作を定義できる。

◆Domain object（業務オブジェクト）

業務で扱うデータの種類。計画、注文、取引先、品目、資源、プロセス、ロット、タスク、作業、事象
[テキストを入力]

の１０種類のプリミティブ業務オブジェクトを基とした派正クラスとして定義する。あらかじめ業務プロパティが定義されている。

◆Domain property（業務プロパティ）

業務オブジェクト、あるいは業務ドキュメントがもつことができる属性（値をもつ項目名）の種類を定義した情報。属性のデータ型、デフォルト値、必須かどうか、主キーの区別などをあらかじめ定義する。

◆Messaging model

複数の業務アプリケーション間でメッセージをやりとりするパターンを規定したもの。下位の通信レベルのメッセージ交換の単位ではなく、業務アプリケーションとしての視点からみたメッセージ交換の最小単位となる。

◆Primitive Domain object（プリミティブ業務オブジェクト）

Domain object のなかで基本的な型となるものであり、計画、注文、取引先、品目、資源、プロセス、ロット、タスク、作業、事象の１０個。すべての業務オブジェクトはこれらのプリミティブの派生形となる。

データ連携の方法

メッセージ交換の概要

本仕様書が対象とする課題は、業務アプリケーションが、他の業務アプリケーションと、業務オブジェクトを介して連携するしくみを構成することである。そのために、一方の業務アプリケーションが、他方の業務アプリケーションへ連携データをメッセージとして送信する。ここで、双方の業務アプリケーションがもつ業務オブジェクトに対して、それらの共通の概念モデルとなる概念オブジェクト（ドメインオントロジー）を定義することで、その概念オブジェクトから派生した連携データは、双方にとって理解可能なものとなる。

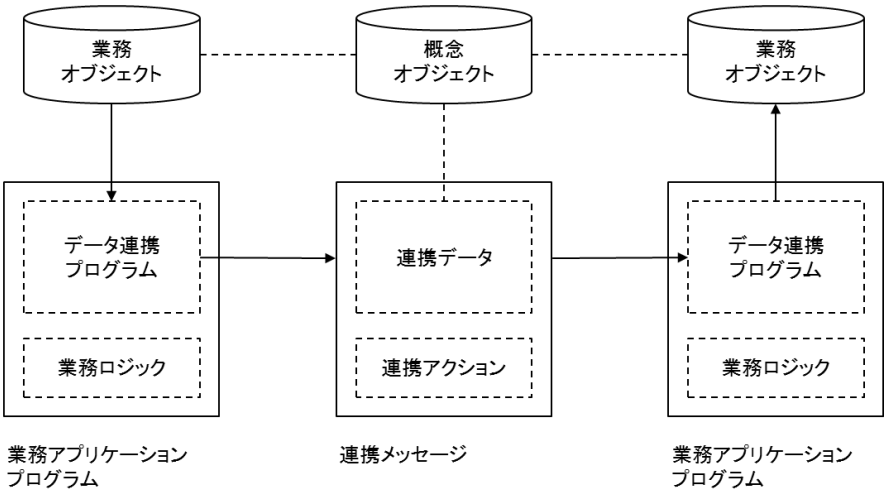


図１　業務アプリケーション間のデータ連携

メッセージ交換モデルの形式

メッセージ交換モデルは、リクエスト&レスポンス型とする。ここでは、要求者が応答者へリクエストメッセージを送信し、それに対応して、応答者が、ただちに、レスポンスメッセージを要求者へ返信する。ただし、応答者は、メッセージ受信者の義務に従って、リクエストメッセージに対応して、レスポンスメッセージを返信しない場合もある。

連携アクションとして、以下の表に示すものが定義されている。

種類	説明

通知（Notify）	業務オブジェクトまたは業務プロパティを（１つまたは複数）通知する
追加（Add）	業務オブジェクトまたは業務プロパティの追加を（１つまたは複数）要求する
修正（Change）	業務オブジェクトまたは業務プロパティの修正を要求する
削除（Remove）	業務オブジェクトまたは業務プロパティの削除を要求する
照会（Get）	条件を指定してその時点で一致する業務オブジェクトまたは業務プロパティを（１つまたは複数）照会する
回答（Show）	Get に対応した照会内容を通知する
同期（Sync）	条件に一致した時点での情報の通知を要求する（非同期の照会モデル）
確認（Confirm）	dd, Change, Remove, Notify, Sync にたいする処理結果を通知する

メッセージ交換モデルとして、これらの連携アクションを組合せることで、以下に示す連携パターンを定義することができる。以下に、それぞれの連携パターンについて、その手順を示す。

通知（Notify）

通知パターンでは、要求者である業務アプリケーションが、連携データを送信し、応答者である業務アプリケーションがその連携データを受信する。この際、要求メッセージの内容に応じて、確認メッセージを要求された場合には、応答者である業務アプリケーションは、確認メッセージを作成し、要求者へ返信しなければならない。なお、通知パターンは、後述する同期パターンに対応した事象発生時の通魏にも利用される。

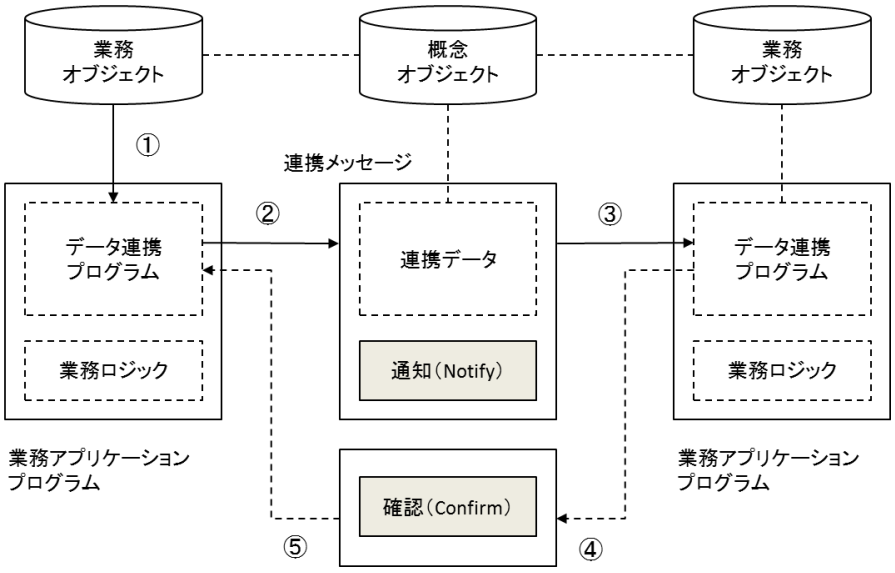


図２　連携データの通知パターン

追加（Add）

追加パターンでは、要求者である業務アプリケーションが、連携データを送信し、応答者である業務ア

プリケーションがその連携データを受信し、業務オブジェクトに登録する。この際、要求メッセージの内容に応じて、確認メッセージを要求された場合には、応答者である業務アプリケーションは、確認メッセージを作成し、要求者へ返信しなければならない。

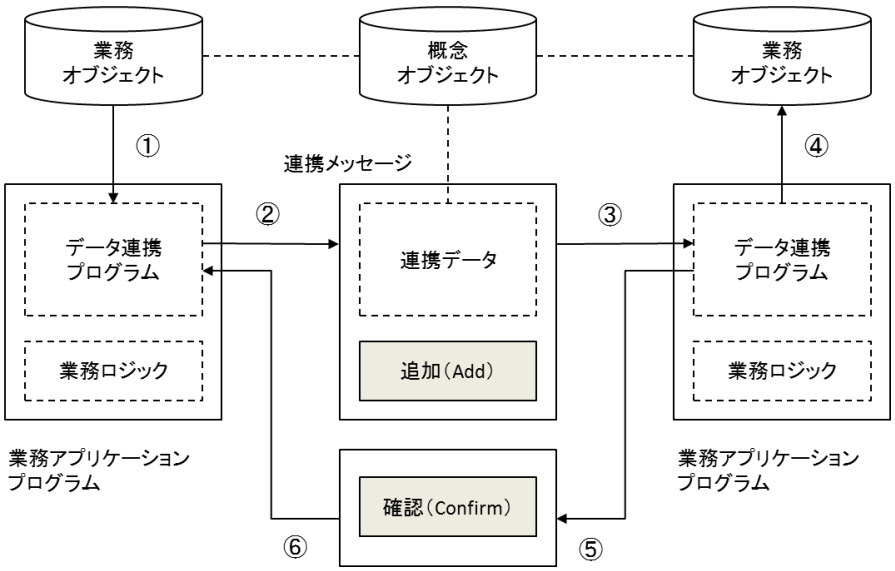


図２　連携データの追加パターン

修正（Change）

修正パターンでは、要求者である業務アプリケーションが、連携データを送信し、応答者である業務アプリケーションが業務オブジェクトから、その連携データに対応するデータを受信し、連携データの内容にあわせて修正する。この際、要求メッセージの内容に応じて、確認メッセージを要求された場合には、応答者である業務アプリケーションは、確認メッセージを作成し、要求者へ返信しなければならない。

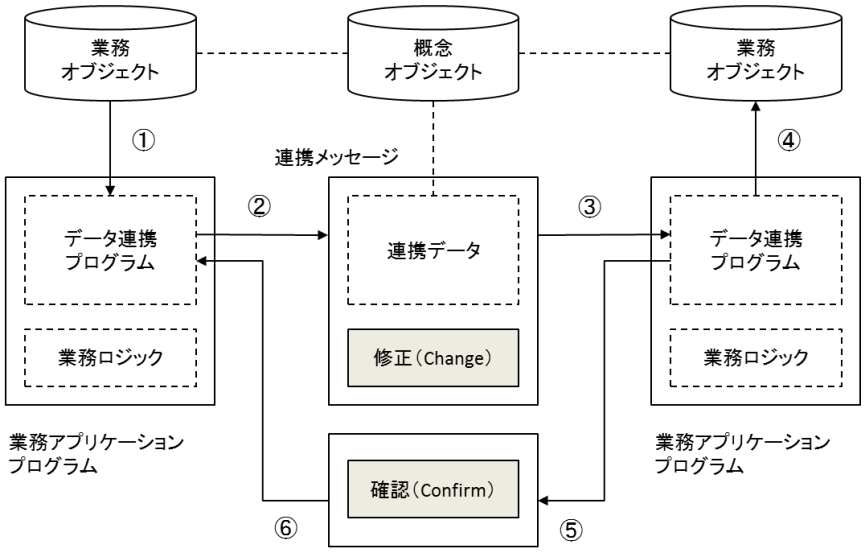


図２　連携データの修正パターン

削除（Remove）

削除パターンでは、要求者である業務アプリケーションが、連携データとして削除する連携データの ID を送信し、応答者である業務アプリケーションが業務オブジェクトから、その連携データに対応するデータを削除する。この際、要求メッセージの内容に応じて、確認メッセージを要求された場合には、応答者である業務アプリケーションは、確認メッセージを作成し、要求者へ返信しなければならない。

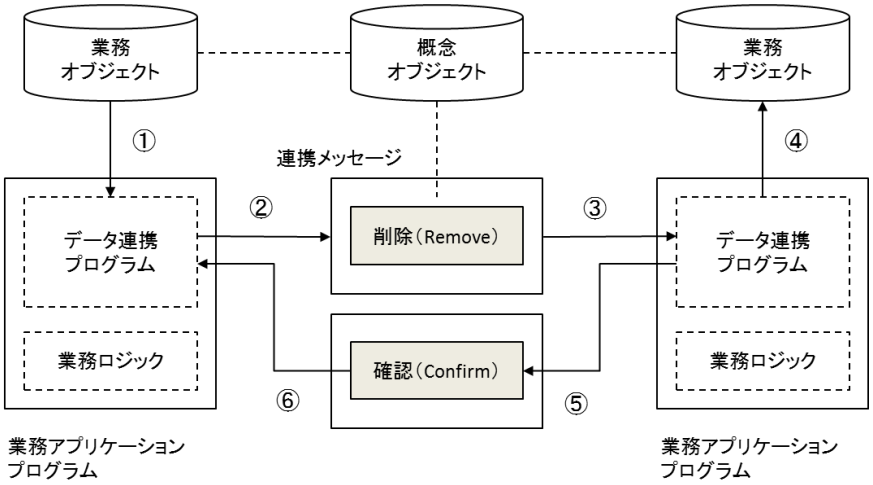


図２　連携データの削除パターン

同期（Sync）

同期パターンでは、要求者である業務アプリケーションが、連携データとして同期したい業務オブジェ

クトと同期形式を候補のなかから指定し、応答者である業務アプリケーションがその同期依頼情報を首里または拒否する。この際、要求メッセージの内容に応じて、確認メッセージを要求された場合には、応答者である業務アプリケーションは、同期依頼の受付状況を表す確認メッセージを作成し、要求者へ返信しなければならない。なお、実際に同期のためのメッセージが送信されるのは、指定した条件に適合した時点であり、別途、通知パターンによって対象とする業務オブジェクトのデータ ID が送信される。

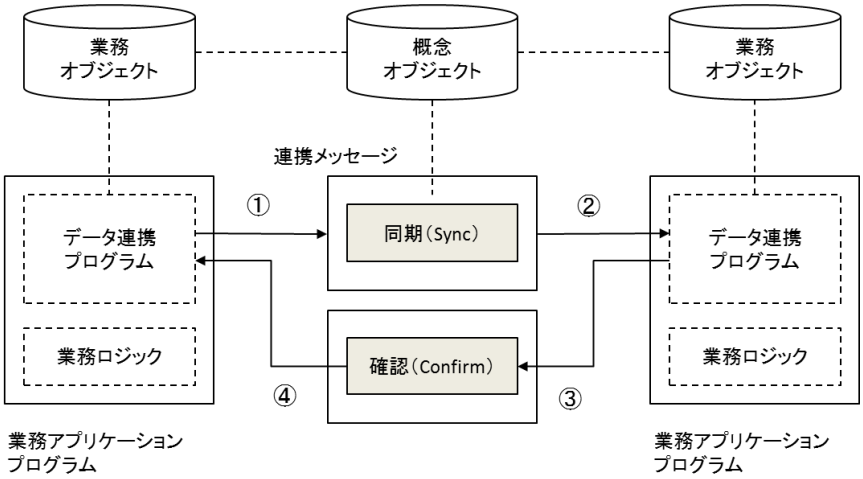


図２　連携データの同期パターン

照会（Get）

照会パターンでは、要求者である業務アプリケーションが、連携データとして照会する業務オブジェクトや照会条件を送信し、応答者である業務アプリケーションが、対応するデータを業務オブジェクトの中から検索し、結果をリストとして要求者へ返信する。返信メッセージの内容として、連携データが設定される。もしエラーなどがあった場合は、その内容を返信しなければならない。

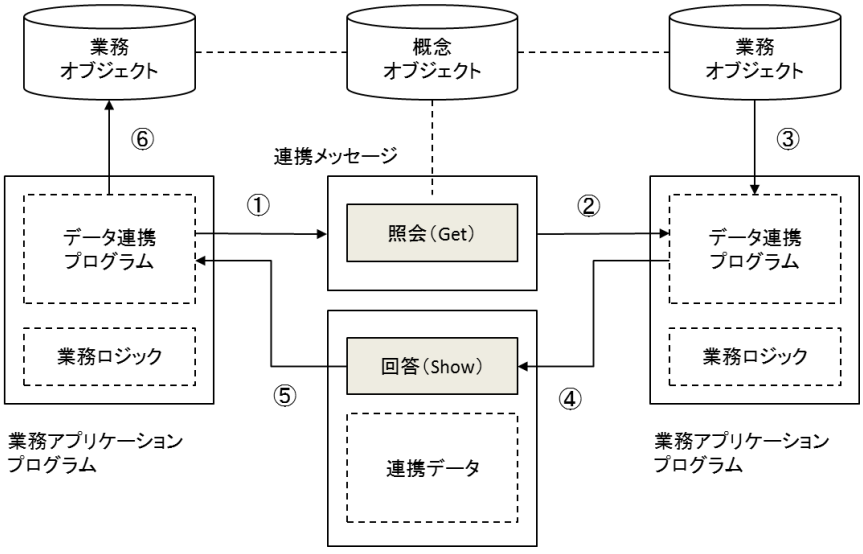


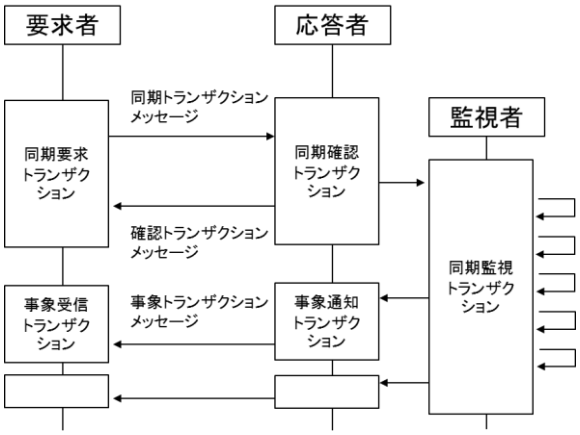
図２　連携データの照会パターン

同期処理のイベント通知

事象通知は、サービス提供者（事象通知者）が、あらかじめ監視対象としている業務オブジェクトおよびその業務プロパティについて、特定の条件を満たした場合に、その事象の識別番号と、対象オブジェクトデータ ID を一律に通知する。通知を受けたい依頼者は、事象通知者に対して、通知依頼を要求しておく。

イベント通知サービスを行なう業務アプリケーションは、通知可能な事象を、実装プロファイルで定義する。ここで定義した事象の通知サービスを、他の業務プログラムが、Sync ドキュメントによって依頼された場合は、イベント通知サービスを提供する。業務アプリケーションは、対応する事象が発生した時点で、依頼者に通知しなければならない。

同期－応答パターンでは､処理種別が通知依頼(Sync)である業務ドキュメントを受け取った業務アプリケーションは、内部のデータベースに含まれる業務オブジェクトについて、特定の条件が成り立つかどうかを監視し、条件が成立した場合に、依頼された相手に、その事実を Notify ドキュメントによって通知しなければならない。



図○　同期－監視パターン

トランザクション処理

送信者である業務アプリケーションから、受信者である業務アプリケーションに対して往信するメッセージには、連携操作が追加（Add)、修正（Change)、削除（Remove）である場合に限り、複数の連携アクションを１つのエッセージに含めることができる。

連携操作が通知（Notify）、照会（Get）、同期（Sync）である場合は、同一種類の連携操作を 1 つのメッセージ内に複数含めることができる。

メッセージの構造

メッセージの構造

本仕様に準拠したメッセージ交換で、交換する内容は以下の構造をもったデータでなければならない。
◇メッセージとは、通信プログラムが業務アプリケーション間で、一回の送信または受信でやりとりする単位である。メッセージ本体には、すくなくとも１つのトランザクション情報を含んでいなければならない。メッセージ本体には、複数のトランザクション情報をもつことができる。
◇ヘッダ情報は、メッセージ交換に必要な情報であり、主に通信プログラムが利用する情報である。
◇トランザクション情報は、業務プログラムが一回のトランザクションで処理する単位の情報である。このトランザクション情報については、その形式と内容について本仕様にて規定する内容にしたがわなければならない。

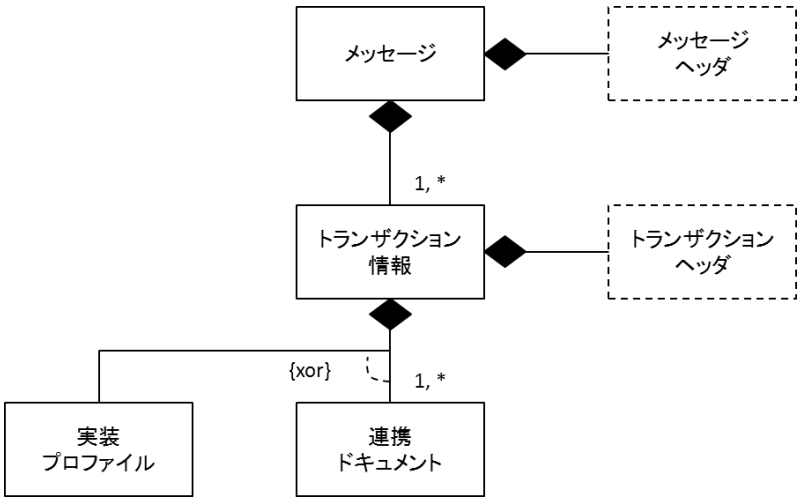


図１　メッセージ交換内容

◇メッセージヘッダとして、データ連携プログラムは、以下の属性を解釈できなければならない。また、必須である属性については、すべてのメッセージがその値を持たなければならない。

表　メッセージヘッダ属性

属性名	説明	必須
id		○
sender		○
security		
create		○

description		

業務アプリケーションが送信または受信するメッセージのトップの要素名は、本仕様書では Message を用いるが、変更可能である。トップ要素の構造は、以下のＸＭＬスキーマによって正しく解釈できなければならない。

＝メッセージ要素の構造の制約＝

```
<xsd:complexType name="MessageType">
  <xsd:sequence>
    <xsd:element ref="Transaction" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:string" use="required"/>
  <xsd:attribute name="sender" type="xsd:string"/>
  <xsd:attribute name="security" type="xsd:string"/>
  <xsd:attribute name="create" type="xsd:dateTime"/>
  <xsd:attribute name="description" type="xsd:string"/>
</xsd:complexType>
```

以下に、メッセージ例を示す。

＝メッセージ記述例＝

```
<Message
  id="0001"
  sender="hosei"
  security="alskdjf;lasdjflaksjdfalk"
  create="2013-02-23T14:23:54"
  description="これはメッセージです。">
  <Transaction id="aaa">
       …
  </Transaction>
</Message>
```

トランザクション情報

トランザクション情報は、データ連携プログラムがまとめて処理をする単位である。
トランザクション情報には、１つの実装プロファイルか、または１つ以上の業務ドキュメントが含まれる。
実装プロファイルとは、業務プログラムが本仕様書にある機能のどの部分までを、どのような内容についてサポートしているかについての情報である。
業務ドキュメント情報は、業務における追加、修正、削除、照会といった１つのアクションに対応する情報であり、特定のコンテキストに対応したデータの集まりである。

表　トランザクションヘッダ

属性名	説明	必須

id		○
type		○
confirm		
connection		○
result		
create		
description		
profile		

トランザクション要素の構成は、以下のＸＭＬスキーマで検証可能なものでなければならない。

＝ドランザクション要素の構造（ＸＭＬスキーマ）＝

```
<xsd:element name="Transaction">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element ref="ImplementProfile"/>
      <xsd:element ref="Document" maxOccurs="unbounded"/>
      <xsd:element ref="Error"/>
    </xsd:choice>
    <xsd:attribute name="id" type="xsd:string" use="required"/>
    <xsd:attribute name="type" type="xsd:string"/>
    <xsd:attribute name="confirm" type="xsd:string"/>
    <xsd:attribute name="connection" type="xsd:string"/>
    <xsd:attribute name="result" type="xsd:string"/>
    <xsd:attribute name="create" type="xsd:dateTime"/>
    <xsd:attribute name="description" type="xsd:string"/>
    <xsd:attribute name="profile" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

以下に、トランザクションメッセージの例を示す。

＝ドランザクション要素の例＝

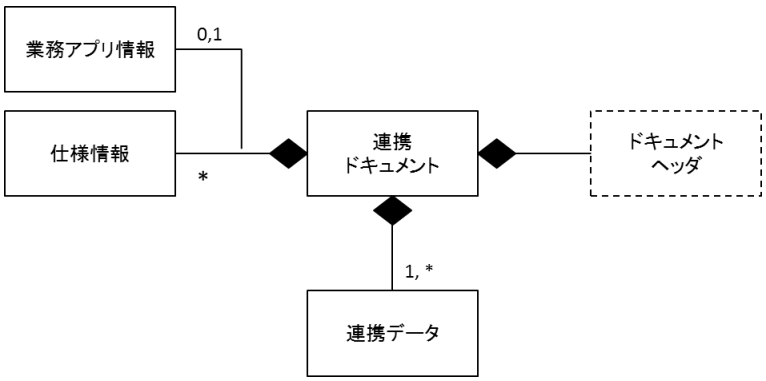
```
<Transaction id="0001">
  <Document action="Add">
    <Order id=" 201" ><Item id="A01"><Qty value="1" unit="個"/></Item></Order>
    <Order id=" 202" /><Item id=" A05" ><Qty value=" 3"  unit=" 個" /></Item></Order>
  </Document>
</Transaction>
```

＝トランザクションがエラーとなった場合の例＝

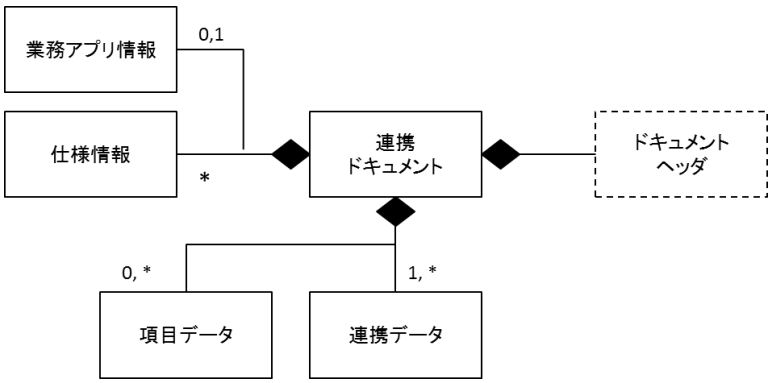
```
<Transaction id=" 0001"  result=" ERROR" >
  <Document action=" Add" >
    <Error type=" 404" >該当製品が存在しません。Id=" A05" </Error>
  </Document>
</Transaction>
```

連携ドキュメント情報

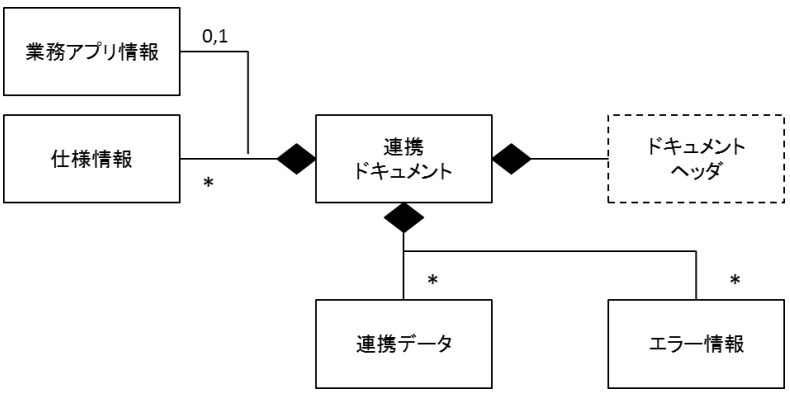
通知（Notify）および追加（Add）、修正（Change）、削除（Remove）、同期（Sync）メッセージの業務ドキュメント情報は、ヘッダ情報と１つ以上の連携データで構成される。



照会（Get）メッセージの業務ドキュメント情報は、ヘッダ情報と連携データと項目データで構成される。連携データが複数ある場合は、OR の関係としてそれぞれの検索結果のデータをマージする。



返信メッセージである回答（Show）や確認（Confirm）メッセージの業務ドキュメント情報は、ヘッダ情報と複数の連携データで構成される。



連携ドキュメントは、業務プロファイルによって、あらかじめその業務上の意味定義がなされている。連携ドキュメント名は、業務プロファイルで定義した種類以外を設定してはならない。

要求側の連携ドキュメントには、１つ以上の連携データを設定しなければならない。応答側が作成する連携ドキュメントには、エラー情報を設定するか、１つ以上の連携データを設定しなければならない。

連携データは、あらかじめ業務プロファイルによって定義された項目と値のペアによって構成される。１つの業務ドキュメントの下位には、2 種類以上の連携データを設定してはならない。ただし、トランザクション要素の下位に、種類の異なる種類の連携データをもつ業務ドキュメントを設定することができる。

業務ドキュメントは、その送信者と id によって、ユニークに識別可能でなければならない。

表　ドキュメントヘッダの属性

属性名	説明	必須
id		○
name		○
ref	対応する業務ドキュメント	
action		
option		
event		
namespace		
create		
description		

連携データは、以下のいずれかの内容を表すものでなければならない。業務オブジェクトとは、業務上の対象問題を構成する基本要素となるオブジェクトであり、オントロジーとして定義されるものである。関係オブジェクトとは、業務オブジェクト間の関係を定義する情報である。特性オブジェクトとは、業務オブジェクトの内容を定義する情報である。

連携ドキュメント情報は、Document 要素によって記述され、その構造は、以下のＸＭＬスキーマで検証可能でなければならない。

１つの連携ドキュメント情報に含まれる連携データの種類は、１種類でなければならない。つまり、Document 要素の下位にある連携データの XML 要素はすべて同じでなければならない。また、その要素に type 属性がある場合は、すべて同一の値でなければならない。

```
<xsd:element name="Document">
```

```
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="Error" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="App" minOccurs="0"/>
      <xsd:element ref="Spec" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Condition" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Selection" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Property" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:choice minOccurs="0">
        <xsd:element ref="Party" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="Plan" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="Order" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="Item" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="Resource" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="Process" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="Lot" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="Task" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="Operation" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="Event" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="Compose" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="Produce" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="Consume" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="Assign" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="Relation" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="Location" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="Capacity" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="Progress" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="Spec" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="Start" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="End" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="Price" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="Cost" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:choice>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:string" use="required"/>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="ref" type="xsd:string"/>
    <xsd:attribute name="action" type="xsd:string"/>
    <xsd:attribute name="option" type="xsd:string"/>
    <xsd:attribute name="event" type="xsd:string"/>
    <xsd:attribute name="namespace" type="xsd:string"/>
    <xsd:attribute name="create" type="xsd:dateTime"/>
    <xsd:attribute name="description" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

連携データの構造

連携データは、業務オブジェクト、関係オブジェクト、特性オブジェクトのいずれかとなる。以下の図に、それぞれの要素の下位に設定可能な XML 要素を示す。

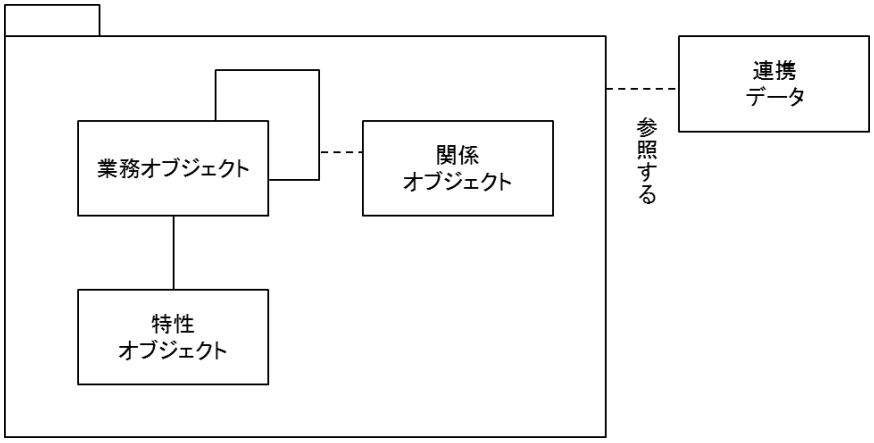


図３　連携データ情報

連携データを表記するための XML 要素は、プリミティブ要素、関係属性要素、複合属性要素、単純属性要素、データ要素、そして管理要素にグループ化できる。

表１　連携ドキュメントを構成する連携データの種類

区分	キーワード	要素種類	説明
業務オブジェクト	Party	プリミティブ要素	
	Plan		
	Order		
	Item		
	Resource		
	Process		
	Lot		
	Task		
	Operation		
	Event		
関係オブジェクト	Compose	関係属性要素	
	Produce		
	Consume		
	Assign		
	Relation		
特性オブジェクト	Location	複合属性要素	
	Capacity		
	Progress		
	Spec		
	Start	単純属性要素	

	End		
	Price		
	Cost		

業務オブジェクトを表記するには、プリミティブ要素を用いる。関係オブジェクトを表記するには、関係属性要素を用いなければならない。特性オブジェクトを表記するには、複合属性要素、単純属性要素を用いなければならない。

連携データは、Document 要素の下位に記述する。連携データの種類は、業務プロファイルにあらかじめ定義されていなければならない。

ドキュメント要素の下位には、プリミティブ要素、関係属性要素、複合属性要素、単純属性要素のいずれかの要素 1 種類が、1 つ以上設定される。異なる要素を 1 つの連携ドキュメントに設定してはならない。

プリミティブ要素の下位には、関係属性要素、複合属性要素、単純属性要素、データ要素の中から 1 種類以上が設定される。ただし、1 つのプリミティブ要素の下位に、要素名と type 属性の組合せについて重複があってはならない。

関係属性要素の下位には、1 つのプリミティブ要素および、複合属性要素、単純属性要素、データ要素の中から 1 種類以上が設定される。ただし、1 つの関係属性要素の下位に、要素名と type 属性の組合せについて重複があってはならない。

複合属性要素の下位には、1 つのプリミティブ要素および、単純属性要素、データ要素の中から 1 種類以上が設定される。ただし、1 つの複合属性要素の下位に、要素名と type 属性の組合せについて重複があってはならない。

単純属性要素の下位には、1 つのプリミティブ要素および、データ要素の中から 1 種類以上が設定される。ただし、1 つの単純属性要素の下位に、要素名と type 属性の組合せについて重複があってはならない。

業務オブジェクトは、プリミティブ要素に対応する Party、Plan, Order, Item, Resource,　Process, Lot, Operation, Task, Event のいずれかの要素によって表現しなければならない。

以下の表に、XML による記述の場合の構造を示す。表でグレーの部分は、上位要素に対して下位要素を設定してはならない。

グループ	要素名	下位要素						
		ドキュメント要素	プリミティブ要素	関係属性要素	複合属性要素	単純属性要素	データ要素	管理要素
ドキュメント要素	Document		*					
プリミティブ要素	Party, Order, Plan, Item,Resource, Process, Lot, Task,Operation, Event			1				0,1
関係属性要素	Compose, Produce, Consume, Assign, Relation		1		1			0,1
複合属性要素	Location, Capacity, Progress, Spec		1			1		0,1
単純属性要素	Start, End, Price, Cost		1				1	0,1
データ要素	Qty, Char, Time							
管理要素	Priority, Display, Description, Author, Date							

業務アプリケーションがもつ特性オブジェクトや関係オブジェクトの内容は、業務オブジェクトを連携データとする場合も、メッセージに含めることができる。したがって、選択する業務ドキュメントの種類によって、XML 上の異なる位置に値が設定されることになる。たとえば、作業の開始時刻は、作業指示を表す業務オブジェクトの中で設定する場合と、開始時刻を表す特性オブジェクトとして連携ドキュメントに設定する場合とが選択可能である。

タイプ１

```
<Document id=" aa" >
  <Order id=" 001" >
    <Start><Time value=" 2013-11-21T12:10:00>
  </Order>
</Document>
```

タイプ２

```
<Document id=" bb" >
  <Start>
    <Order id=" 001" >
  </Start>
</Document>
```

業務プロファイル

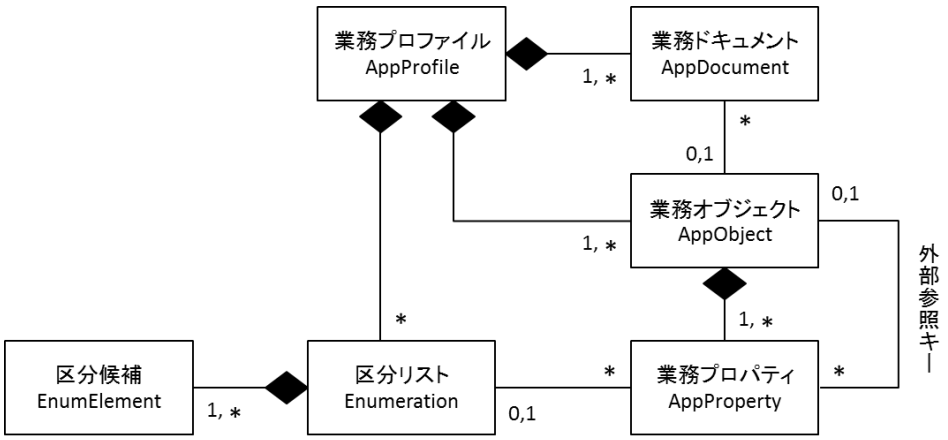
業務プロファイルの定義

業務プロファイルとは、異なる複数の業務プログラム間で、業務オブジェクトの構造が異なる場合に、双方の連携プログラムが理解可能な共通の業務オブジェクトを連携オブジェクトとして定義したものである。

業務プロファイルは、実際に業務プログラムがメッセージを交換するのに先立って定義されなければならない。メッセージ交換を行なう業務プログラムは、業務トランザクション単位で指定された業務プロファイルをあらかじめ入手可能でなければならない。

メッセージ交換を行なう業務プログラムは、そのメッセージがどの業務プロファイルにもとづいたものであるかを指定しなければならない。業務プロファイルの指定は、業務トランザクション単位に行うことができる。同一の業務トランザクションに含まれる業務ドキュメントは、同一の業務プロファイルを前提としていなければならない。

業務プロファイルは、以下の図に示すように、業務ドキュメント、業務オブジェクト、区分リストによって構成される。



業務プロファイルは、name で指定される名称と、namespace で指定される名前空間によって、グローバルに識別可能でなければならない。

業務プロファイルの属性

属性名	説明	必須

[テキストを入力]

name		○
base	プロファイル拡張をおこなった場合に、拡張前のプロファイル名。なお namespace は同じでなければならない。	
location		
prefix		
namespace		○
create		
description		

業務プロファイルは、AppProfile 要素によって記述しなければならない。また、以下の XML スキーマによって解釈可能でなければならない。

```
<xsd:element name="AppProfile">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="Enumeration" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="AppObject" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="AppDocument" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="base" type="xsd:string"/>
    <xsd:attribute name="location" type="xsd:string"/>
    <xsd:attribute name="prefix" type="xsd:string"/>
    <xsd:attribute name="namespace" type="xsd:string"/>
    <xsd:attribute name="create" type="xsd:string"/>
    <xsd:attribute name="description" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

業務プロファイルの記述例を以下に示す。

```
<(message)>
    <AppProfile>
        <AppObject name=“注文伝票”
               primitive=“Order”>
            <AppProperty
                name=“項目名”
                dataType=“string”
                path=“対応する値の位置” />
        </AppObject>
        <AppDocument name=“注文情報”
          object=“注文” />
    </AppProfile>
</(message)>
```

業務ドキュメントの定義情報

業務ドキュメントは、name によって、その名称を指定しなければならない。名称は、業務プロファイルの中でユニークなものでなければならない。

業務ドキュメントは、以下の属性をもつ。

業務ドキュメントの属性

属性名	説明	必須
name		○
object		
category		
description		

業務ドキュメントは、AppDocument によって記述しなければならない。また、以下の XML スキーマによって解釈可能でなければならない。

```
<xsd:element name="AppDocument">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="object" type="xsd:string"/>
    <xsd:attribute name="category" type="xsd:string"/>
    <xsd:attribute name="description" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

属性 object の値には、業務オブジェクトの名称以外に、業務オブジェクトの属性も指定できる？
業務プロファイルによって、業務ドキュメントの構造を規定するには、・・・・？？

業務オブジェクトの定義

業務オブジェクトは、name によって、その名称を指定しなければならない。名称は、業務プロファイルの中でユニークなものでなければならない。

連携オブジェクトの属性

属性名	説明	必須
name		○
primitive		
description		

[テキストを入力]

業務オブジェクトは、AppObject によって記述しなければならない。また、以下の XML スキーマによって解釈可能でなければならない。

```
<xsd:element name="AppObject">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="AppProperty" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="primitive" type="xsd:string" use="required"/>
    <xsd:attribute name="description" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

注）AppProperty の minOccurs=”0”は削除した。

業務プロパティの定義

業務プロパティは、name によって、その名称を指定しなければならない。名称は、業務オブジェクトの中でユニークなものでなければならない。

業務プロパティが、他の業務オブジェクトの主キーを外部参照している場合には、extlink の値に、同一業務プロファイル内で定義されている業務オブジェクト名を設定する。この場合は、指定した業務オブジェクトの主キーへの外部参照キーであることを示す。

メッセージ交換時には、外部参照キーに対応する業務オブジェクトのデータが、同一トランザクション要素内に存在する場合、そのデータを外部参照先とする。また、それ以外の場合で、相手プログラムがもつ業務オブジェクトのデータが対応する場合には、そのデータを外部参照先とする。それ以外の場合は、外部参照先のデータは存在しないものとし、エラーとしてはならない。

属性名	説明	必須
name		
path		
multiple		
key		
extlink	外部参照キー	
enumeration		
dataType		
use		
description		

業務プロパティは、AppProperty によって記述しなければならない。また、以下の XML スキーマによって解釈可能でなければならない。

```
<xsd:element name="AppProperty">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsd:string"/>
    <xsd:attribute name="path" type="xsd:string"/>
    <xsd:attribute name="multiple" type="xsd:string"/>
    <xsd:attribute name="key" type="xsd:string"/>
    <xsd:attribute name="extlink" type="xsd:string"/>
    <xsd:attribute name="enumeration" type="xsd:string"/>
    <xsd:attribute name="dataType" type="xsd:string"/>
    <xsd:attribute name="use" type="xsd:string"/>
    <xsd:attribute name="description" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

業務プロファイルの拡張

ひとつの業務プロファイルから、新たな別の業務プロファイルに拡張する場合、以下の操作が許される。拡張後の業務プロファイルは、拡張もとの業務プロファイルを base 属性によって指定しなければならない。また、その場合に、両者の名前空間は同じでなければならない。

拡張方法	説明
業務ドキュメントの追加	
業務オブジェクトの追加	
業務プロパティの追加	

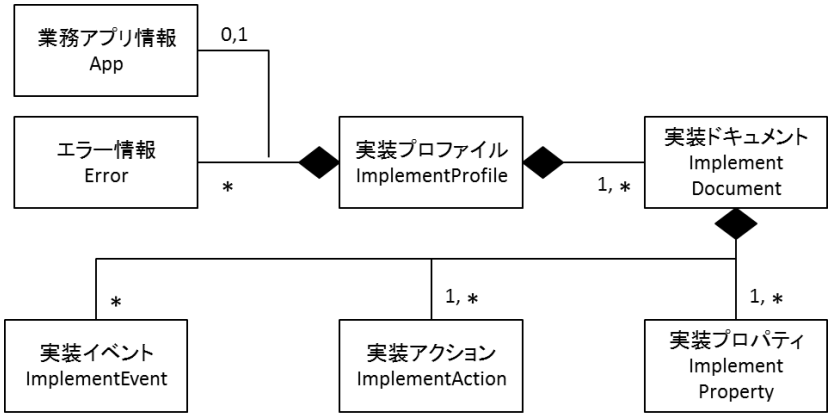
つまり、拡張前の業務プロファイルにたがってデータ連携を行なうプログラムは、拡張後の業務プロファイルをベースにデータ連携を行っているプログラムに対して、旧来の方法で連携できなければならない。

実装プロファイル

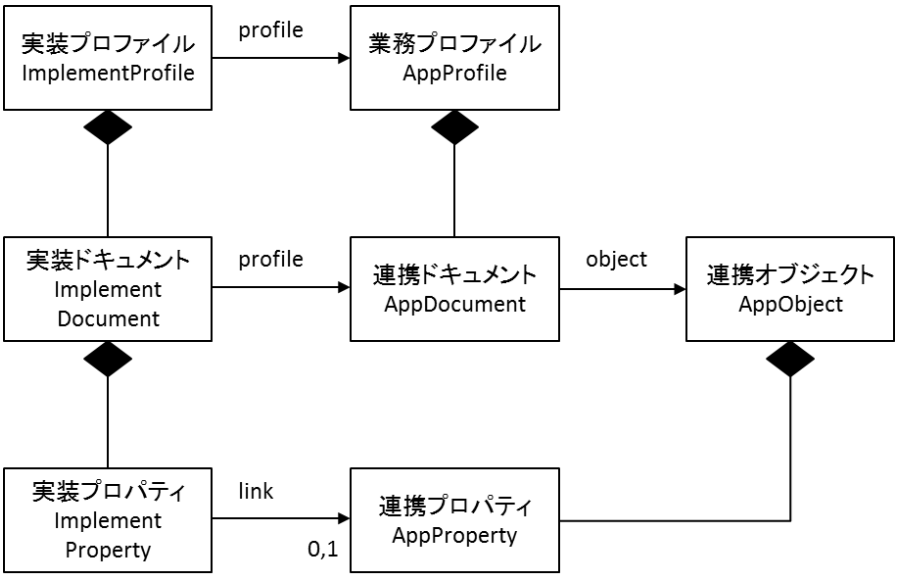
実装プロファイルの構造

実装プロファイル（Implement Profile）には、対象ドメインにおいて、個々のアプリケーションがもつ機能（ケーパビリティ）を定義しなければならない。実装プロファイルは、あらかじめ定義された業務プロファイルに対応して、業務アプリケーションに実装された機能を表したものである。

実装プロファイルは、業務アプリケーションが実装した実装ドキュメントの内容が表記されていなければならない。また、業務ドキュメントには、実装プロパティ、実装アクション、実装イベントが表記されていなければならない。なお、実装イベントの定義については、別章（〇章）にて規定する。



実装プロファイルには、対応する業務プロファイルを指定しなければならない。また、実装ドキュメントには、対応する業務オブジェクトを指定しなければならない。実装プロパティは、対応する業務プロパティが存在する場合は、それを指定しなければならない。



実装プロファイルには、ヘッダ情報として、以下の内容を表記する。

実装プロファイルの属性

属性名	説明	必須
id		
name		
action		
profile	業務プロファイル名	○
location		
namespace		
create		
description		

◆実装プロファイルの表記

実装プロファイルは、ImplementProfile によって記述しなければならない。また、以下の XML スキーマによって解釈可能でなければならない。

```
<xsd:element name="ImplementProfile">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="Error" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="App" minOccurs="0"/>
      <xsd:element ref="ImplementDocument" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:string"/>
    <xsd:attribute name="name" type="xsd:string"/>
    <xsd:attribute name="action" type="xsd:string"/>
    <xsd:attribute name="profile" type="xsd:string"/>
```

[テキストを入力]

```
      <xsd:attribute name="location" type="xsd:string"/>
      <xsd:attribute name="namespace" type="xsd:string"/>
      <xsd:attribute name="create" type="xsd:dateTime"/>
      <xsd:attribute name="description" type="xsd:string"/>
    </xsd:complexType>
  </xsd:element>
```

＝記述例＝

```
<(message)>
    <ImplementProfile>
        <ImplementDocument type=“注文書”>
            <ImplementProperty
                name=“項目名”
                dataType=“string”
                path=“対応する値の位置” />
        </ImplementDocument>
    </ImplementProfile>
</(message)>
```

実装ドキュメントの定義

業務ドキュメントには、ヘッダ情報として、以下の内容を表記する。

属性名	説明	必須
name		
title		
option		
profile	業務ドキュメント名	○
location		
namespace		
description		

◆実装ドキュメントの定義

実装ドキュメントは、ImplementDocument によって記述しなければならない。また、以下の XML スキーマによって解釈可能でなければならない。

```
  <xsd:element name="ImplementDocument">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="ImplementAction" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="ImplementProperty" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="ImplementEvent" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:string" use="required"/>
      <xsd:attribute name="title" type="xsd:string"/>
```

[テキストを入力]

```
    <xsd:attribute name="option" type="xsd:string"/>
    <xsd:attribute name="profile" type="xsd:string"/>
    <xsd:attribute name="location" type="xsd:string"/>
    <xsd:attribute name="namespace" type="xsd:string"/>
    <xsd:attribute name="description" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

実装アクション

業務アクションには、ヘッダ情報として、以下の内容を表記する。

属性名	説明	必須
action		○
~~level~~	（廃止：互換性用）	
role		
description		

◆実装されたアクションの定義

実装アクションは、ImplementAction によって記述しなければならない。また、以下の XML スキーマによって解釈可能でなければならない。

```
<xsd:element name="ImplementAction">
  <xsd:complexType>
    <xsd:attribute name="action" type="xsd:string" use="required"/>
    <xsd:attribute name="level" type="xsd:int"/>
    <xsd:attribute name="role" type="xsd:string"/>
    <xsd:attribute name="description" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

実装プロパティ

業務プロパティには、ヘッダ情報として、以下の内容を表記する。

属性名	説明	必須
name		
title		
extend		
link	業務プロパティ名	

[テキストを入力]

~~multiple~~	（廃止：互換性用）	
path	データの値が記述されているパス(業務データ要素の位置を基準とする)	
dataType		
enumeration		
type		
use		
description		

◆業務ドキュメントの項目

実装プロパティは、ImplementProperty によって記述しなければならない。また、以下の XML スキーマによって解釈可能でなければならない。

```
<xsd:element name="ImplementProperty">
  <xsd:complexType>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="title" type="xsd:string"/>
    <xsd:attribute name="extend" type="xsd:string"/>
    <xsd:attribute name="link" type="xsd:string"/>
    <xsd:attribute name="multiple" type="xsd:string"/>
    <xsd:attribute name="path" type="xsd:string"/>
    <xsd:attribute name="dataType" type="xsd:string"/>
    <xsd:attribute name="enumeration" type="xsd:string"/>
    <xsd:attribute name="type" type="xsd:string"/>
    <xsd:attribute name="use" type="xsd:string"/>
    <xsd:attribute name="description" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

メッセージ交換方法

トランザクション処理

複数のトランザクション情報をもつメッセージ本体を受け取った業務プログラムは、そこにあるトランザクション情報に対応するトランザクションを、その順で実行しなければならない。もし、ひとつのトランザクションの実行が失敗した場合は、それ以降のトランザクションを実行してはならない。

メッセージの受信側では、ひとつのトランザクション内で、複数の連携ドキュメントを処理する場合、その中で、エラーが発生した場合は、トランザクション内のすべての実行を取消し、トランザクションを実行する前の状態にもどさなければならない。

返信メッセージ内には、ヘッダ情報として成功／失敗の区分を error 属性の値として設定しなければならない。返信でエラーがある場合は、対応する業務ドキュメント要素の下位に、エラー要素を設定しなければならない。

同じ ID をもったメッセージを複数回送信した場合、受信側は、２回目のメッセージを無視しなければならない。また、同じ ID をもった業務トランザクション情報を複数回送信した場合、受信側は、２回目のトランザクションを無視しなければならない。いちど送信したトランザクションを取り消すことはできない。

エラー情報の設定方法

応答者は、受け取った情報にエラーがある場合、あるいは処理中にエラーが発生した場合は、エラー情報を Error 要素に設定しなければならない。エラーの識別は、以下の表にあるとおりとしなければならない。

表○　エラー番号と説明

エラー番号	説明
404	該当するオブジェクトがない

連携ドキュメントの処理方法

連携ドキュメント情報にあるヘッダ情報には、その連携ドキュメントの処理方法（アクション）を指定しなければならない。

要求者のトランザクション情報に含まれる Document 要素に設定されるアクションは、Add, Change, Remove, Notify, Sync, Get のいずれかでなければならない。一方、応答者のトランザクション情報に含まれる Document 要素に設定されるアクションは、Show, Confirm のいずれかでなければならない。

トランザクション情報には、処理方法（アクション）が指定された連携ドキュメントを複数もつことができる。ただし、それらの連携ドキュメントは、同一種類の処理でなければならない。

トランザクションの分類が同じ要求アクションをもつ連携ドキュメントは、同一のトランザクション要素内に設定することができる。たとえば、Add アクションと Change アクションの連携ドキュメントを同じトランザクション内に設定することができる。この場合は、処理が失敗した場合に応答側のプログラムではロールバックすることになる。トランザクションの分類が異なる要求アクションをもつ連携ドキュメントは、同一のトランザクション要素内に設定してはならない。

種類	往信	返信
通知型	通知（Notify）	確認（Confirm）
照会型	照会（Get）	回答（Show）
処理型	追加（Add）、修正（Change）、削除（Remove）	確認（Confirm）
同期型	同期（Sync）	確認（Confirm）

連携ドキュメントを受けとった業務アプリケーションは、そこで指定された処理方法（アクション）の種別に対応した処理を実行しなければならない。対応する処理は、○章にて規定する。

トランザクション情報が応答者のものである場合には、対応する要求トランザクションの内容に対する返信以外の情報を含めてはならない。

◆メッセージ受信者の義務

要求者から連携ドキュメントを受け取った業務アプリケーションは、メッセージ受信者の義務に従って、返信メッセージを要求者に返信するかを判断し、必要に応じて返信しなければならない。

要求者の連携ドキュメントのアクションが Get である場合は、応答者は処理結果を連携ドキュメントとして返信しなければならない。この場合のアクションは Show としなければならない。

[テキストを入力]

要求者の連携ドキュメントのアクションが Add, Change, Remove, Notify, Sync のいずれかである場合は、応答者は要求者の返信要求のキーワードにしたがって、以下の表にしたがい、処理結果を連携ドキュメントとして返信するかを判断しなければならない。返信する場合のアクションは Confirm としなければならない。

要求者への返信は、トランザクション単位で行わなければならない。返信者は受け取ったトランザクション情報に含まれる連携ドキュメントに対して、１つ以上の返信が必要な場合は、トランザクション情報を要求者に返信しなければならない。

表２　受信に対する Confirm アクションの返信方法

キーワード	説明
Never	返信しない
OnError	エラーとなった場合のみ返信する
Always	常に返信する

Confirm メッセージ、Show メッセージなど、応答メッセージに対応する応答は行ってはならない。応答メッセージにエラーがあった場合でも、その事実は応答者に伝えずに、そのメッセージ交換は終了する。

要求者は、応答メッセージがない場合に、要求メッセージを再送することができる。ただし、この場合は、要求メッセージの業務トランザクションＩＤ、連携ドキュメントＩＤを、前回のものと同一にしなければならない。

トランザクションの返信

応答メッセージにある１つのトランザクション要素には、異なる複数の要求トランザクションの内容にある連携ドキュメントの応答が含まれてはならない。要求メッセージにある１つのトランザクションに対応した返信内容は、返信メッセージにある１つのトランザクション要素内に設定されていなければならず、複数に分けてはならない。

同一のメッセージ要素に含まれるトランザクション情報は、異なるメッセージによって返信することができる。

通知型のトランザクションの場合、連携ドキュメントすべての返信区分が Never（返信しない）となっている場合、トランザクション要素を返信しなくてもよい。また、返信メッセージの内容として、返信するトランザクションがない場合は、応答者は、返信メッセージを送らなくてもよい。

返信のための複数のトランザクションを１つのメッセージ要素にまとめることができる。また、１つのメッセージ要素に設定された複数の要求トランザクションに対応する返信を、複数のメッセージ要素に分けて返信することができる。

同期の場合のメッセージ交換パターン

同期型（Sync）のトランザクションは、その完了後、その後に続く 0 回以上の通知型（Notify）と欄ザックションによりイベント通知を行なう。この場合、イベント通知を行なうトランザクション ID は、対応する同期型（Sync）トランザクションの ID と同じ出なければならない。

場合、Sync パターンの応答者は、その後、特定の条件が整った時点で、Notify メッセージを、Sync メッセージの送信者にメッセージを送信する。この Notify メッセージの送信は、指定回数、指定期間、または Sync メッセージの送信者からサービス停止を要求する Sync メッセージを再度受信するまで繰り返される。同期に対応する通知メッセージの送信方法は、〇章で規定する。

パターン名	業務アプリ A		業務アプリ B	説明
Sync パターン	Sync 送信	→	Sync 受信	サービス開始を依頼する場合に送信する。
	Confirm 受信	←	Confirm 送信	
Notify パターン	Notify 受信	←	Notify 送信	該当するイベントが起きた場合に送信する。
	No action			
Sync パターン（オプショナル）	Sync 送信	→	Sync 受信	サービス停止を依頼する場合に送信する。
	Confirm 受信	←	Confirm 送信	

連携プログラムの処理

◆追加（Add）処理

処理種別が追加(Add)である連携ドキュメントを受け取った業務アプリケーションは、そこに含まれる連携オブジェクトを、内部のデータベースに追加しなければならない。ただし、追加する連携データについて、すでに存在する内部データベースのデータと id が重複している場合は、追加せずにエラーとしなければならない。また、key が指定されている場合は、key の値を ID として利用しなければならない。

＝追加の例＝

```
<Transaction id=" 123"  sender=" hosei" >
<Document id=" 301"  name=“受注情報”  action=“Add” >
<Order id=“001” >
    <Spec type=“顧客名”  ><Char value=“西岡” /></Spec>
    <Item name=“なべ” ><Qty value=“10”  unit=“個” /></Item>
</Order>
</Document>
```

＝受信側は、追加をおこなった連携オブジェクトの id を返す。

```
<Transaction id=" 456"  sender=" apson"  ref=" 123"  result=" Success" >
<Document id=" 444"  ref=" 301"  name=“受注情報”  action=“Confirm” >
<Order id=“001” />
</Document>
</Transaction>
```

＝追加の例＝

```
<Transaction id="123" sender="hosei">
<Document id="301" name="生産実績" action="Add">
<Produce id="002">
  <Item id="A001"/>
  <Qty value="10" unit="Kg"/>
  <Time value=" 2013-03-10T12:30:12" />
</Produce>
<Produce id="003">
  <Item id="D001"/>
  <Qty value="23"/>
  <Time value="2013-03-10T14:13:45"/>
</Produce>
</Document>
</Transaction>
```

＝追加の例（業務プロパティの利用）＝

```
<Transaction id="123" sender="hosei">
<Document id="301" name="生産実績" action="Add">
```

```
<Produce id="002">
  <Property type="name" value="A001"/>
  <Property type="produce_value" value="10" />
  <Property type="produce_time" value="2013-03-10T12:30:12"/>
</Produce>
<Produce id=" 003" >
  <Property type="name" value="D001"/>
  <Property type="produce_value" value="23" />
  <Property type="produce_time" value="2013-03-10T14:10:45"/>
</Produce>
</Document>
</Transaction>
```

◆修正（Change）処理

処理種別が修正(Change)である連携ドキュメントを受け取った業務アプリケーションは、そこに含まれる連携オブジェクトの内容にしたがって、内部のデータベースにすでに存在するオブジェクトの内容を修正しなければならない。

修正する対象は id で明示的に指定しなければならない。ただし、key が指定されている場合は、その値を ID として利用しなければならない。対象オブジェクトが存在しない場合はエラーとしなければならない。対象オブジェクトが一意に決定できない場合はエラーとしなければならない。

対象とする連携オブジェクトまたは業務プロパティに定義された連携データについて、実際に修正されるのは、業務プロファイルにおいて定義した項目の値に対応する部分となる。たとえば、以下の例では、Spec/Char@value の値である”西岡”、Item@name の値である”なべ”、Item/Qty@value の値である 10 が修正対象となる。Item/Qty@unit の値である”個”は修正値として認識されない。これは、業務プロファイルにおける受注情報の項目に、こうした相対パスが項目定義の中で設定されているかどうかによる。

＝　修正メッセージ例（１）＝

```
<Transaction id="123" sender="hosei">
<Document id="301" name="受注情報" action="Change">
<Order id="001">
    <Spec type="顧客名"><Char value="西岡"/></Spec>
    <Item name="なべ"><Qty value="10" unit="個"/></Item>
</Order>
</Document>
</Transaction>
```

＝　修正メッセージ例（２）＝

```
<Transaction id="123" sender="hosei">
<Document id="301" name="生産実績" action="Change">
<Produce id="002">
  <Item id="A001"/>
  <Qty value="10" unit="Kg"/>
  <Time value="2013-03-10T12:30:12"/>
```

```
</Produce>
</Document>
</Transaction>
```

◆削除（Remove）処理

処理種別が削除(Remove)である連携ドキュメントを受け取った業務アプリケーションは、指定された連携オブジェクトに対応するデータを、内部のデータベースから削除しなければならない。ただし、論理的に、内部のデータベースに存在しない状態であれば、実際にデータを削除しなくてもよい。

削除する対象は id で明示的に指定する。ただし、key が指定されている場合は、その値を ID として利用しなければならない。ID を指定しない場合は、なにもせず削除件数がなかったものとして処理しなければならない。

◆照会（Get）処理

照会対象となる業務データを条件で指定する場合には、データ要素の condition 属性を利用する。照会を行なう連携ドキュメントにおいて、連携データが複数ある場合には、OR 条件として処理しなければならない。また、同一の連携データに複数のデータ要素があり、複数の condition 属性が設定されている場合には、それらの条件は AND として処理されなければならない。

連携プロパティには、照会する業務プロパティの項目を指定しなければならない。連携プロパティが設定されていない場合には、ID のみからなる連携データを返信しなければならない。

もし、連携プロパティの type 属性が all の場合は、返信側の業務アプリケーションが実装プロファイルの中で指定しているすべての業務プロパティを返信しなければならない。ただし、値が NULL の場合は返信を省略することができる。(要検討)

要求側の業務ドキュメントの連携プロパティに、ソート条件、最大件数、ページ番号の指定がある場合には、それに対応した返信をしなければならない。なお、最大件数、ページ番号は、先頭の連携プロパティの設定を優先しなければならない。

＝例：

```
<Transaction>
  <Document id="678" action="Get" name="Product" limit="100" page="1">
    <Item type="pps:Material">
      <Spec name="Product Name"><Char value="製品 A"/></Spec>
    </Item>
    <Property type="pps:Price"/>
  </Document>
</Transaction>
```

指定された識別項目名が理解できない場合には、受信者はエラーを返さなければならない。

＝例２：100 円以上のなべありますか？ そのなべのメーカー名と納期を知りたい。

```
<Document action="Get">
<Item >
   <Spec type="メーカー"><Char type="かな"/></Spec>
   <Event type="納期"><Time /></Event>
</Item>
<Condition>
<Item id="なべ"/>
    <Price ><Qty value="100" condition="GE"/></Price>
</Item>
<Condition>
</Document>
```

業務プロファイルの照会

できる？

＝業務プロファイル照会メッセージ＝

```
<Message>
  <Transaction id="aaa">
    <AppProfile action="Get"/>
  </Transaction>
</Message>
```

実装プロファイルの交換

業務アプリケーションは、以下の実装プロファイル照会に対応して、自分の実装プロファイルを返信しなければならない。また、業務プロファイルは、別途入手可能でなければならない。

＝実装プロファイル照会メッセージ＝

```
<Message>
  <Transaction id="aaa">
    <ImplementProfile action="Get"/>
  </Transaction>
</Message>
```

＝応答メッセージ記述例＝

```
<Message>
  <Transaction id="aaa">
    <ImplementProfile action="Show" appPfofile="www.pslx.org/2009/v01">
    ...実装プロファイルの内容
    </ImplementProfile>
    <ImplementProfile action="Show" appPfofile="www.pslx.org/2014/v02">
    ...実装プロファイルの内容
    </ImplementProfile>
  </Transaction>
</Message>
```

業務アプリケーションが、複数の業務プロファイルに対応している場合には、それぞれについて実装プロファイルの内容を返信する。もし、トランザクション要素に profile 属性が設定されている場合には、該当する業務プロファイルに対応する実装プロファイルを返すかエラーを返さなければならない。

実装プロファイルには、対応する業務ドキュメント、業務オブジェクトなどのメッセージ内容に関する情報と、可能なサービス内容に関する情報が記述される。業務アプリケーションは、実装プロファイルで記述した内容の範囲にない要求について、意味解釈不能、あるいはサービス対象外のエラーメッセージを返信するか、無視することができる。

◆回答（Show）処理

照会処理要素内にある Condition 要素によって限定されたオブジェクトのリストを返す。存在しない場合は空のリストとして返す。Condition がない場合は、すべての候補を返す。

返信オブジェクト数の最大数を指定するには、limit 属性と page 属性を利用する。ソート条件にしたがった順にカウントする。

該当するオブジェクトの内容は、指定があった業務プロパティの値を返す。照会する業務プロパティの指定がない場合は、対象オブジェクトまたは対象プロパティの id のみを返す。

対象オブジェクトの指定
対象項目の指定

返信する連携ドキュメントまたは業務プロパティの内容は、Property 要素で照会された場合は、Property 要素によってその値を返す。

```
<Transaction>
  <Document action="Show" name="Product" limit="100" page="1" ref="678">
    <Item id="00123" type="pps:Material">
      <Spec name="Product Name"><Char value="製品Ａ"></Spec>
      <Price><Qty value="2500"></Price>
    </Item>
    <Item id="00351" type="pps:Material">
      <Spec name="Product Name"><Char value="製品Ｇ"></Spec>
      <Price><Qty value="3450"></Price>
    </Item>
  </Document>
</Transaction>
```

```
<Transaction>
  <Document action="Show" name="Product" limit="100" page="1" ref="678">
    <Item id="00123" type="pps:Material">
      <Property name="Spec name="Product Name"><Char value="製品Ａ"></Spec>
      <Price><Qty value="2500"></Price>
    </Item>
    <Item id="00351" type="pps:Material">
      <Spec name="Product Name"><Char value="製品Ｇ"></Spec>
      <Price><Qty value="3450"></Price>
    </Item>
  </Document>
</Transaction>
```

Property を用いた回答ドキュメントの例
→内容を追加する

[テキストを入力]

◆アプリケーション拡張用（App）

アプリケーション間でトランザクションに関する情報を任意に設定できる。アプリケーション拡張（App）は、Document 要素の下に 1 つ設定することができる。この要素以下の構造および内容については、規定はない。

◆選択条件の指定（Condition）

照会において､対象オブジェクトを選択するための情報｡指定には Property 要素を用いることができ、Condition 要素が複数ある場合は OR 条件とし。1 つの Condition 要素内に、Property 要素または、指定データの値が複数ある場合は AND 条件としてオブジェクトを選択する。

◆対象項目の指定（Property）

照会において、対象オブジェクトの項目を指定するための要素。ただし、Property で指定する項目名はアプリケーションプロファイルの AppProperty においてあらかじめ定義されていなければならない。

同期と事象通知

同期要求トランザクションの内容

応答者であるサーバーに特定の事象に対応したイベント通知を依頼するには、同期トランザクションによって同期トランザクションメッセージを作成し、応答者へ送信する。イベントは、連携ドキュメントがもつ連携データの値によって発生するので、連携ドキュメントの一部として定義される。
◇要求するイベント通知は、応答者が実装プロファイルにおいて定義したものでなければならない。
◇同期トランザクションメッセージには、要求するイベント名と、要求者の中でユニークな要求番号を付加しなければならない。
◇同期トランザクションメッセージにおける Sync ドキュメントでは、業務ドキュメント名、イベント識別記号、通知の戻り先のＩＤ（通知先のアドレスと、自分自身のＩＤ)、送付方法（ＳＭＴＰ，ＨＴＴＰなどの区別）有効期限を指定する。

例：依頼者が事象通知者へ通知を依頼するためのメッセージ

サービス依頼者→サービス提供者

```
<Transaction id="1002" sender="user01">
<Document id="E123" name="機械状態" event="機械停止" action="Sync" option="start"/>
</Transaction>
```

注）option="start"はなくてもよい（デフォルト)。

サービス依頼者が、サービスの中断（または停止）を要求する場合、以下のように、処理区分の値を stop とする。

依頼者→サービス提供者（中断)

```
<Transaction id="1002" sender="user01">
<Document id="567" name="機械状態" event="機械停止" ref="E123" action="Sync" option="stop"/>
</Transaction>
```

依頼したサービスが実際に正常に動作しているかを照会する（実際にイベントが発生しているかの照会も含む)

例A）以前に依頼したサービス依頼（E123）が動いていますか？

サービス依頼者→サービス提供者

```
<Transaction id="1006" sender="user01">
<Document id="E456" name="機械状態" event="機械停止" action="Get" ref=" E123"/>
</Transaction>
```

注）option="start"はなくてもよい（デフォルト）。

同期確認トランザクションの内容

同期トランザクションメッセージによって、同期要求（Sync ドキュメント）を受け取ったイベント通知サーバーは、その依頼を受け付けた場合には、受付記号を送信者へ返さなければならない。なお、この受付記号は、依頼者が通知サービスの状態を照会する場合、依頼をキャンセルあるいは中断／再開する場合に利用することができる。

上記の回答（確認メッセージ）

サービス提供者→サービス依頼者

```
<Transaction id="2008" sender="apsom" ref="1002" >
<Document id="444" name="機械状態" event="機械停止" ref="E123" action="Confirm"/>
</Transaction>
```

サービス提供者→サービス依頼者（サービスできない場合）

```
<Transaction id="2008" sender="apsom" ref="1002" >
<Document id="444" name="機械状態" event="機械停止" ref="E123" action="Confirm">
  <Error name="該当するサービスは現在停止しています。"/>
</Document>
</Transaction>
```

◇イベント通知要求メッセージに、返信要求に係らず、依頼の受付可否を Confirm メッセージによって通知しなければならない。

要求者からイベント通知要求を受け取ったサーバーは、イベント通知リストに要求者を追加し、該当するイベントが発生したら、ただちに通知トランザクションを実行する。

このために、サービス提供者側は、イベントごとに、その対象となる項目（１つまたはリスト形式）の値と併せて、以下のリストを常時保持しておく必要がる。

１．依頼者のユーザ名（user01 に対応）、
２．依頼番号（E123 に対応）
３．依頼日時

上記の情報は、外部からの照会に対して、回答できるようにする。（要、具体的な方法検討）

事象通知サービスが正常に動いているかどうかを依頼者から問合わせがあった場合には、いかのような返信をしなければならない。

例Ａに対する回答）

サービス提供者→サービス依頼者

[テキストを入力]

```
<Transaction id="2009" sender="apsom" ref="1006" >
<Document id="447" name="機械状態" event="機械停止" ref="E123" action="Confirm"
  description="動作中"/>
</Transaction>
```

事象通知トランザクションの内容

◇イベントが発生した場合に、要求者に対して通知する Notify メッセージの内容には、イベント名とともに Sync メッセージで指定された要求番号を設定しなければならない。
◇この Notify では、連携オブジェクト、業務プロパティの内容は返さずに、イベントＩＤと連携オブジェクト（業務プロパティ）id、および発生時刻を返す。
詳細な内容については、通知をうけた側が再度照会メッセージを送ることで対応する。
◇イベント通知メッセージには、各イベントに対応して、時刻タグを設定しなければならない。これは、イベント通知の時刻ではなく、イベントに対応する状態になった時刻とする。

◆サービス提供者が、サービスの中断（または停止）を通知する場合

サービス提供者→依頼者（中断）

```
<Transaction id="2004" sender="apsom">
<Document id="324" name="機械状態" event="機械停止" ref="E123" action="Notify" option="stop"
  description="明日の８時まで停止します。"/>
</Transaction>
```

事象監視トランザクションの内容

イベント通知サービス提供者は、あらかじめ設定されたパラメータの値を監視し、条件に対応する状況となったら、事象通知トランザクションを起動する。

例：サービス提供者→サービス依頼者（イベント発生時）

```
<Transaction id="456" sender="apsom" ref="123" >
<Document id="865" name="機械状態" event="機械停止" ref="E123" action="Notify" >
  <Property name="稼働状態" value="故障"><Time value=" 2013-05-06T13:45:12" /></Property>
</Document>
</Transaction>
```

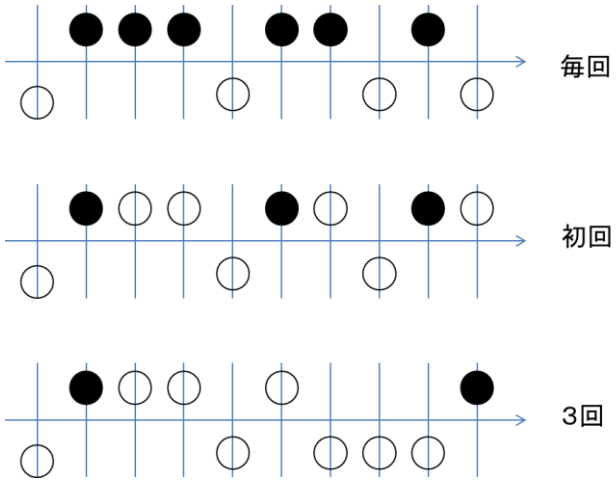
```
<Transaction id="456" sender="apsom" ref="123" >
<Document id="865" name="機械状態" event="機械停止" ref="E123" action="Notify">
  <Event id="R003">
    <Resource id="R003"/><Spec name="稼働状態"><Char value="故障"/></Resource>
    <Time value=" 2013-05-06T13:45:12" />
  </Event>
```

```
</Document>
</Transaction>
```

イベントの監視方法として、監視サイクルを定義することができる。監視サイクルとは、イベント発生の有無を判定するために、指定されたパラメータを確認するタイミング（トリガー）を決定する方法である。監視サイクルは、基準時刻からの特定の時間幅を指定する場合と、第三者のプログラムからイベントを受け取った場合などがある。

監視サイクルのひとつの監視タイミングにおいて、実際にイベントが発生しているかどうかは、定義された条件を対象パラメータに適用させることで決定される。
条件が設定されていない場合は、イベント発生のすべてのタイミングで事象通知トランザクションを起動する。
ただし、送信パターンを設定することで、以下の図のように、実際に事象通知メッセージを送信する頻度をすくなくすることができる。

以下の図では、水平線よりも上部にある○は、条件を満たす場合（イベント発生）を意味する。黒丸は、実際にメッセージを送信するタイミングである。以下の図の“毎回”、“初回”、“３回”といった指定は、実装プロファイルにて、サービス提供者側が設定する。



図○　イベント発生の判定方法の設定

返信のタイミング（サイクル）や期限などは、応答者側が、あらかじめ ImplementProfile にて指定する。

例：事象通知者がもつ実装プロファイル

サービス依頼者（user001）→サービス提供者（apsom）

```
<Transaction id="1001" sender="user01" >
  <ImplementProfile action="Get"/>
</Transaction>
```

例１）「MachineStatus において、稼働状態の値が、故障となったら、機械停止というイベントを通知する」場合。この際に、稼働状態と装置番号をあわせて通知する。

サービス提供者（apsom）→サービス依頼者（user001）

```
<Transaction id="2001" sender="apsom" ref="1001" >
  <ImplementProfile action="Show">
    <ImplementDocument name="MachineStatus" title="機械状態">
      <ImplementProperty name="稼働状態"/>
      <ImplementProperty name="装置番号"/>
      <ImplementEvent name="機械停止" >
        <Property name="稼働状態" />
        <Property name="装置番号" />
        <Condition><Property name="稼働状態"><Char value="故障"/></Property></Condition>
      </ImplementEvent>
    </ImplementDocument>
  </ImplementProfile>
</Transaction>
```

例２）「MachineStatus において、装置 03 の稼働状態の値が、故障となったら、機械停止というイベントを通知する」場合・・・条件に装置番号を加える。

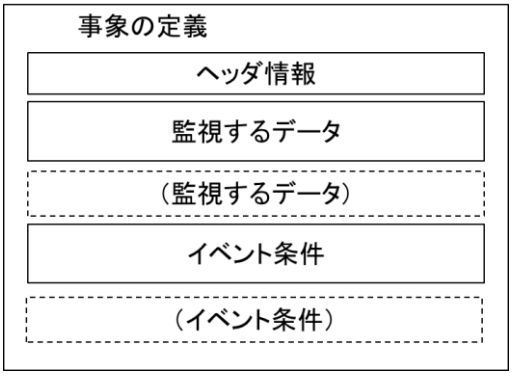
例３）「MachineStatus において、稼働状態の値が、3 以上となったら、機械異常というイベントを通知する」場合

サービス提供者（apsom）→サービス依頼者（user001）

```
<Transaction id="2001" sender="apsom" ref="1001" >
  <ImplementProfile action="Show">
    <ImplementDocument name="MachineStatus" title="機械状態">
      <ImplementProperty name="稼働状態"/>
      <ImplementProperty name="設備名"/>
      <ImplementEvent name="機械停止" >
        <Property name="稼働状態"/>
        <Condition><Property name="稼働状態"><Qty value="3" condition="min"/></Property></Condition>
      </ImplementEvent>
    </ImplementDocument>
  </ImplementProfile>
</Transaction>
```

監視可能な事象の定義

イベント通知サービスを行なう業務アプリケーションが、通知可能な事象を定義する。提供可能なイベント通知サービスの内容は、以下のイベント定義情報によって記述する。



図○　イベント定義情報

◇イベント定義情報は、１つ以上の監視対象の情報（０はＯＫ？）と、イベント発生を識別するイベント条件を記述しなければならない。
◇イベント定義情報は、ImplementEvent 要素によって記述しなければならない。以下のスキーマによって解釈可能でなければならない。

```
<xsd:element name="ImplementEvent">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="App" minOccurs="0"/>
      <xsd:element ref="Condition" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Selection" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Property" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="type" type="xsd:string"/>
    <xsd:attribute name="cycle" type="xsd:duration"/>
    <xsd:attribute name="start" type="xsd:dateTime"/>
    <xsd:attribute name="expire" type="xsd:dateTime"/>
    <xsd:attribute name="description" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

事象通知の方法は、サービス提供者側が事前に業務プロファイルの中で設定する。設定する内容は以下の項目がある。

項目名	実装名	説明
事象名	ImplementEvent/@name	事象の識別名。サービス提供者内でユニークであること
事象の分類	ImplementEvent/@type	変化（追加、修正、削除）／閾値／条件／差分
事象判定区分		事象とみなす判定区分（０または正の整数）
確認サイクル	ImplementEvent/@cycle	期間型（デフォルト１分）
確認サイクル単位		月、週、日
サービス開始日時	ImplementEvent/@start	確認サイクルの起点

サービス終了日時	ImplementEvent/@expire	監視終了日時
最大回数	ImplementEvent/@maxCount	デフォルトで１回
最大期間	ImplementEvent/@maxDuration	
備考	ImplementEvent/@description	事象の説明（任意のテキスト）
条件式	ImplementEvent/Condition	事象発生とみなす条件式
監視項目	ImplementEvent/Property	監視する項目（属性）名

◆事象名

事象は、サービス提供者ＩＤ、ドキュメントＩＤ、そして事象名によって識別されます。

◆監視項目

業務アプリが監視する項目をサービス提供者側が指定します。１つの事象に複数の項目（Property）を設定することができます。監視対象でなくても、定数であって、計算上利用するものはここに定義しておきます。

◆事象発火

監視対象の項目およびその他の項目と定数で定義される条件式が満たされた場合を事象発火状態とします。条件は Condition 要素によって、ＡＮＤ、ＯＲの関係を定義できます。

◆事象発生とみなす判定区分

監視サイクルに対して、条件が満たされた都度、イベントの通知をするかどうかを指定します。以下のタイプがあります。

タイプ	説明
毎回	条件を満たす領域にある場合に毎回通知します。＝０
初回	条件を満たす領域に入った最初の回で通知します。＝１
Ｎ回	最低Ｎ回条件外となった後に、条件を満たす領域に入った最初の回で通知します。＝Ｎ

閾値ではなく、変化量でみる場合

例）前回と値がことなった場合に通知する

例）前回と値が５以上異なった場合は通知する

閾値がある場合

```
<Condition><Property name="稼働状態"><Qty value="3" condition="min"/></Property></Condition>
```

前回と異なる場合に通知（閾値がない：前回値の場合）

```
<Condition><Property name="稼働状態"><Qty  value="前回の値"  condition="NE"/>
</Property></Condition>
```

[テキストを入力]

前回よりもプラスマイナス１０

```
<ImplementEvent type="prev">
    <Condition value= "10"  type="GE"><Property name="稼働状態"/><</Condition>
    <Condition value="-10"  type="LE"><Property name="稼働状態"/></Condition>
</ImplementEvent>
```

例）他の値(項目Ａと項目Ｂの合計)よりも大きい場合（項目間の比較)。項目を２つ Condition の下位に定義すること。順番があり。

```
        <Condition value="0" type="DIF">
            <Property name="項目Ａ"/>
            <Property name="項目Ｂ"/>
        </Condition>
```

条件のＡＮＤが書けなくなった。(A=10 and B=5 の場合）

```
        <Condition> →旧バージョン
            <Property name="項目Ａ"><Qty value="10" /><Qty value="5" /></Property>
        </Condition>
```

ＡＮＤの場合は type=”and”を Condition 要素に設定し囲む

```
        <Condition type="and">
          <Condition  value="10"><Property name="項目Ａ" /></Condition>
          <Condition  value="5"><Property name="項目Ｂ" /></Condition>
        </Condition>
```

◇対象パラメータの条件を設定するには、以下の要素を利用します。条件を表すＸＭＬは以下のスキーマで定義できなければなりません。

```
  <xsd:element name="Condition">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Property" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="id" type="xsd:string"/>
      <xsd:attribute name="wildcard" type="xsd:string"/>
      <xsd:attribute name="value" type="xsd:string"/>
      <xsd:attribute name="version" type="xsd:string"/>
    </xsd:complexType>
  </xsd:element>
```

```
  <xsd:element name="Property">
    <xsd:complexType>
      <xsd:choice>
        <xsd:element ref="Qty" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="Char" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="Time" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:choice>
```

[テキストを入力]

```
    <xsd:attribute name="type" type="xsd:string"/>
    <xsd:attribute name="name" type="xsd:string"/>
    <xsd:attribute name="path" type="xsd:string"/>
    <xsd:attribute name="value" type="xsd:string"/>
    <xsd:attribute name="sort" type="xsd:string"/>
    <xsd:attribute name="calc" type="xsd:string"/>
    <xsd:attribute name="display" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

利用パターン：

実施中のサービスを終了する際のメッセージ（サービス提供側⇒サービス依頼側）

実施中のサービスを中断、再開する際のメッセージ（(サービス提供側⇒サービス依頼側）

過去に依頼したサービスを中断する際のメッセージ（サービス依頼側⇒サービス提供側）

依頼したサービスが実施されているかを確認するメッセージ（サービス依頼側⇒サービス提供側）

＝スキーマ

```
<xsd:element name="ImplementEvent">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="App" minOccurs="0"/>
      <xsd:element ref="Condition" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Selection" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Property" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="type" type="xsd:string"/>
    <xsd:attribute name="cycle" type="xsd:duration"/>
    <xsd:attribute name="start" type="xsd:dateTime"/>
    <xsd:attribute name="expire" type="xsd:dateTime"/>
    <xsd:attribute name="description" type="xsd:string"/>
  </xsd:complexType>
</xsd:element>
```

◆参考

```
<Message id="" sender="" security="" create="" description="">
  <Transaction id="" type="" confirm="" connection="" create="" description="">
    <Error>
    <ImplementProfile id="" name="" action="" profile=""
      location="" namespace="" create="" description="">
      <Error>
      <App>
      <ImplementDocument name="" title="" option="" profile=""
         location="" namespace="" desctiption="">
        <ImplementAcction action="" level="" role="" description="">
        <ImplementProperty name="" title="" extend="" link="" multiple=""
           path="" dataType="" enumeration="" type="" use="" description="">
        <ImplementEvent name="" type="" cycle="" start="" expire="" description="">
          <App>
```

[テキストを入力]

```
          <Condition id="" wildcard="" value="" version="">
            <Property type="" name="" path="" value="" sort="" calc="" display="">
              <Qty>
              <Char>
              <Time>
          <Property>

    <Document id="" name="" ref="" action="" option="" event="" namespace="" create="" description="">
      <Error>
      <App>
      <Spec>
      <Condition>
      <Property>
      <(PrimitiveElements) id="" key="" name="" parent="" type="" status="">

<AppProfile name="" base="" location="" prefix="" namespace="" create="" desctiption"">
  <AppObject>
  <AppDocument>
  <Enumeration>
```

業務オブジェクト

業務オブジェクトの構造

要検討・・・

id	要素のユニークキー。ref で参照される値。
key	実装上のＲＤＢで利用する物理的なキーシステム用
name	名称（識別用文字）
parent	階層構造（木構造）の場合の親の id
type	データモデル上のオブジェクト識別名（プロファイルで設定）システム用
status	（書込みや修正の実行結果をしめす）
apply（動詞でなくする）	（書込みや修正を強制するかどうかの識別用）??システム用
~~condition,~~ operator	条件で絞り込みをする場合の条件記号
value	値（要素によって型は異なる）
count	番号（順序など）
unit	単位系（文字列）
base	分数の分母、比較対象となる値など

	Party	Plan	Order	Item	Resource	Process	Lot	Task	Operation	Event	Compose	Produce	Consume	Assign	Relation	Location	Capacity	Progress	Spec	Start	End	Price	Cost	Priority	Display	Description	Author	Date	Qty	Char	Time
id	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U							
key	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N							
parent	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P							
name	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
type	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
status	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
apply											S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
condition																				S	S	S	S	S	S	S	S	S	S	S
value																				S	S	S	S	S	S	S	S	D	S	T
count																												N	N	N
unit																												S	S	S
base																												D	S	T

◆プリミティブ要素

プリミティブ要素は、業務アプリケーションが扱うさまざまな情報の意味を規定するためのオントロジー構成要素となる。業務アプリケーション間でメッセージ交換するすべての業務オブジェクトは、以下の１０種類のプリミティブ要素の派生形となる。計画（Plan)、注文（Order)、取引先（Party)、品目（Item)、資源（Resource)、プロセス（Process)、ロット（Lot)、タスク（Task)、作業（Operation)、事象（Event)。

属性名	説明
id	要素のユニークキー。ref で参照される値。
key	実装上のＲＤＢで利用する物理的なキーシステム用
name	名称（識別用文字)
parent	階層構造（木構造）の場合の親の id
type	データモデル上のオブジェクト識別名（プロファイルで設定）システム用
status	（書込みや修正の実行結果をしめす)

プリミティブ要素（Party、Plan、Order、Item、Resource、Process、Lot、Task、Operation、Event）の場合は、以下のＸＭＬスキーマで解釈可能でなければならない。

```
<xsd:complexType name="PrimitiveType">
  <xsd:choice minOccurs="0">
    <xsd:element ref="Property" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:sequence>
      <xsd:element ref="Compose" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Produce" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Consume" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Assign" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Relation" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Location" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Capacity" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Progress" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Spec" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Start" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="End" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Event" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Price" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Cost" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Priority" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Display" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Description" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Author" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Date" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:choice>
```

[テキストを入力]

```
  <xsd:attribute name="id" type="xsd:string" use="required"/>
  <xsd:attribute name="key" type="xsd:long"/>
  <xsd:attribute name="name" type="xsd:string"/>
  <xsd:attribute name="parent" type="xsd:string"/>
  <xsd:attribute name="type" type="xsd:string"/>
  <xsd:attribute name="status" type="xsd:string"/>
</xsd:complexType>
```

◆関係属性要素

属性名	説明
id	要素のユニークキー。ref で参照される値。
key	実装上のＲＤＢで利用する物理的なキーシステム用
name	名称（識別用文字）
parent	階層構造（木構造）の場合の親の id
type	データモデル上のオブジェクト識別名（プロファイルで設定）システム用
status	（書込みや修正の実行結果をしめす）
apply（動詞でなくする）	（書込みや修正を強制するかどうかの識別用）??システム用
~~condition,~~ operator	条件で絞り込みをする場合の条件記号
value	値（要素によって型は異なる）
count	番号（順序など）
unit	単位系（文字列）
base	分数の分母、比較対象となる値など

これに対して、関係要素（Compose、Produce、Consume、Assign、Relation）である場合は、以下のＸＭＬスキーマで解釈可能でなければならない。

```
<xsd:complexType name="RelationalType">
  <xsd:choice minOccurs="0">
    <xsd:element ref="Property" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:sequence>
      <xsd:choice minOccurs="0">
        <xsd:element ref="Party"/>
        <xsd:element ref="Plan"/>
        <xsd:element ref="Order"/>
        <xsd:element ref="Item"/>
        <xsd:element ref="Resource"/>
        <xsd:element ref="Process"/>
        <xsd:element ref="Lot"/>
        <xsd:element ref="Task"/>
        <xsd:element ref="Operation"/>
```

```
        <xsd:element ref="Event"/>
      </xsd:choice>
      <xsd:element ref="Location" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Capacity" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Progress" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Spec" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Start" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="End" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Event" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Price" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Cost" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Priority" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Display" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Description" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Author" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Date" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Qty" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Char" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Time" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:choice>
  <xsd:attribute name="id" type="xsd:string"/>
  <xsd:attribute name="key" type="xsd:long"/>
  <xsd:attribute name="name" type="xsd:string"/>
  <xsd:attribute name="type" type="xsd:string"/>
  <xsd:attribute name="status" type="xsd:string"/>
  <xsd:attribute name="apply" type="xsd:string"/>
</xsd:complexType>
```

◆複合属性要素

属性名	説明
id	要素のユニークキー。ref で参照される値。
key	実装上のＲＤＢで利用する物理的なキーシステム用
name	名称（識別用文字）
parent	階層構造（木構造）の場合の親の id
type	データモデル上のオブジェクト識別名（プロファイルで設定）システム用
status	（書込みや修正の実行結果をしめす）
apply（動詞でなくする）	（書込みや修正を強制するかどうかの識別用）??システム用
~~condition,~~ operator	条件で絞り込みをする場合の条件記号
value	値（要素によって型は異なる）
count	番号（順序など）
unit	単位系（文字列）
base	分数の分母、比較対象となる値など

複合属性要素（Location、Capacity、Progress、Spec）の場合は、以下のＸＭＬスキーマで解釈可能でなければならない。

```
<xsd:complexType name="SpecificType">
  <xsd:choice minOccurs="0">
    <xsd:element ref="Property" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:sequence>
      <xsd:choice minOccurs="0">
        <xsd:element ref="Party"/>
        <xsd:element ref="Plan"/>
        <xsd:element ref="Order"/>
        <xsd:element ref="Item"/>
        <xsd:element ref="Resource"/>
        <xsd:element ref="Process"/>
        <xsd:element ref="Lot"/>
        <xsd:element ref="Task"/>
        <xsd:element ref="Operation"/>
        <xsd:element ref="Event"/>
      </xsd:choice>
      <xsd:element ref="Start" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="End" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Event" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Price" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Cost" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Priority" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Display" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Description" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Author" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Date" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Qty" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Char" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Time" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:choice>
  <xsd:attribute name="id" type="xsd:string"/>
  <xsd:attribute name="key" type="xsd:long"/>
  <xsd:attribute name="name" type="xsd:string"/>
  <xsd:attribute name="type" type="xsd:string"/>
  <xsd:attribute name="status" type="xsd:string"/>
  <xsd:attribute name="apply" type="xsd:string"/>
</xsd:complexType>
```

◆単純属性要素

属性名	説明
id	要素のユニークキー。ref で参照される値。
key	実装上のＲＤＢで利用する物理的なキーシステム用
name	名称（識別用文字）

parent	階層構造（木構造）の場合の親の id
type	データモデル上のオブジェクト識別名（プロファイルで設定）システム用
status	（書込みや修正の実行結果をしめす）
apply（動詞でなくする）	（書込みや修正を強制するかどうかの識別用）??システム用
~~condition,~~ operator	条件で絞り込みをする場合の条件記号
value	値（要素によって型は異なる）
count	番号（順序など）
unit	単位系（文字列）
base	分数の分母、比較対象となる値など

最後に単純属性要素（Start、End、Price、Cost）の場合は、以下のＸＭＬスキーマで解釈可能でなければならない。

```
<xsd:complexType name="AccountingType">
  <xsd:choice minOccurs="0">
    <xsd:element ref="Property" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:sequence>
      <xsd:choice minOccurs="0">
        <xsd:element ref="Party"/>
        <xsd:element ref="Plan"/>
        <xsd:element ref="Order"/>
        <xsd:element ref="Item"/>
        <xsd:element ref="Resource"/>
        <xsd:element ref="Process"/>
        <xsd:element ref="Lot"/>
        <xsd:element ref="Task"/>
        <xsd:element ref="Operation"/>
        <xsd:element ref="Event"/>
      </xsd:choice>
      <xsd:element ref="Priority" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Display" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Description" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Author" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Date" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Qty" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Char" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="Time" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:choice>
  <xsd:attribute name="id" type="xsd:string"/>
  <xsd:attribute name="key" type="xsd:long"/>
  <xsd:attribute name="name" type="xsd:string"/>
  <xsd:attribute name="type" type="xsd:string"/>
  <xsd:attribute name="status" type="xsd:string"/>
  <xsd:attribute name="value" type="xsd:string"/>
  <xsd:attribute name="condition" type="xsd:string"/>
  <xsd:attribute name="apply" type="xsd:string"/>
</xsd:complexType>
```

◆データ要素

属性名	説明
id	要素のユニークキー。ref で参照される値。
key	実装上のＲＤＢで利用する物理的なキーシステム用
name	名称（識別用文字）
parent	階層構造（木構造）の場合の親の id
type	データモデル上のオブジェクト識別名（プロファイルで設定）システム用
status	（書込みや修正の実行結果をしめす）
apply（動詞でなくする）	（書込みや修正を強制するかどうかの識別用）??システム用
~~condition,~~ operator	条件で絞り込みをする場合の条件記号
value	値（要素によって型は異なる）
count	番号（順序など）
unit	単位系（文字列）
base	分数の分母、比較対象となる値など

Priority
Display
Description
Author
Date

◆管理要素

属性名	説明
id	要素のユニークキー。ref で参照される値。
key	実装上のＲＤＢで利用する物理的なキーシステム用
name	名称（識別用文字）
parent	階層構造（木構造）の場合の親の id

type	データモデル上のオブジェクト識別名（プロファイルで設定）システム用
status	（書込みや修正の実行結果をしめす）
apply（動詞でなくする）	（書込みや修正を強制するかどうかの識別用）??システム用
~~condition,~~ operator	条件で絞り込みをする場合の条件記号
value	値（要素によって型は異なる）
count	番号（順序など）
unit	単位系（文字列）
base	分数の分母、比較対象となる値など

Qty

Char

Time

Conformance 方法

修正履歴

修正点（バージョン 01）
●初期バージョン：以下の 2 つの節は仕様から削除した。
●3.4.3 Multiple property (Level 2 function)　　39
●3.4.4 Using Header element　　41
●Header
●Selection

2013/5/22（バージョン 02）
●実装レベルの考え方はなくした。
●Property 要素をもちいなくても照会ができるようにした。
●Sync-Notiry の仕様を実装可能なレベルとした。
●property を document の下位に設定可能とした
●transaction の属性に connection を追加した
●document の下位の header 要素は削除した。header 要素は implement profile に設定すべき情報であるため。

2013/10/4（バージョン 03）
●業務ドキュメントには複数種類の業務オブジェクトを設定できるという仕様を改め、1 種類とした（当初の仕様にもどした）。複数種類の業務ドキュメントを transaction 内でまとめることにした。
●Implement profile の要素の位置を message の下位ではなく、transaction の下位に移動
●transaction の属性に profile を追加。ここに業務プロファイルの識別名を記述する。
●transaction の要素に Error を追加。document レベルのエラーとトランザクションレベルのエラーを使い分けることにした。

2013/11/21（バージョン 04）
●Document の直下にある Selection を削除（Selection はすべて削除）

2014/1/23（バージョン 05）
●構成を再検討し、説明を拡充した

2014/4/11（バージョン 06）

●PSLX バージョン３仕様に対応して表記方法を整理した

●ひととおり一貫して読める形とした