地域情報プラットフォームガイドライン
第３章　技術解説

V2.8

一般財団法人全国地域情報化推進協会

目次

修正履歴(ガイドライン第3章V2.7からV2.8)

項番	該当ページ	修正概要	備考
1	全頁	ページヘッダ修正(V2.7 → V2.8)	

３．技術解説

３．１　PF 通信機能

本節では、PF 通信機能（プラットフォーム通信機能）に関する補足的な説明等を提供する。以下のような構成となっている。

(1) HTTP 通信と通信セキュリティ（3.1.1 節）
(2) SOAP 通信と通信モデル（3.1.2 節）
(3) 高信頼性通信機能（3.1.3 節）
(4) 通信において設定すべき項目（3.1.4 節）

上記（1）では、プラットフォーム通信標準仕様の「２．２　HTTP 通信と通信セキュリティ」に対する補足事項を説明する。

(2) は、プラットフォーム通信標準仕様の「２．３　SOAP 通信と通信モデル」に対応しており、SOAP、電子封筒形式に関する簡単な説明のほか、添付ファイルがある場合の電子封筒形式やデータ交換システムパターンの使い分けの目安を示す。

(3) は、プラットフォーム通信標準仕様の「２．４　高信頼性通信機能」に対応しており、高信頼性通信機能の説明、2 つの仕様が存在することによる問題点等を示す。

(4) では、サービス同士が通信できるために設定すべき項目を示す。

３．１．１　HTTP 通信と通信セキュリティ

（１）　インターネットプロトコル

PF 通信の基盤となる通信プロトコルとしては、現在のインターネットの基盤プロトコルである IPv4（Internet Protocol version 4）を採用する。IPv4 では、ネットワークに接続する各機器に割り当てられる IP アドレスが 32 ビットであり、ネットワーク接続機器の増大とともに、IP アドレスの枯渇が危惧されている。その問題を解決するために、128 ビットのアドレスをもつ IPv6（Internet Protocol version 6）が策定されている。

現在 IPv4 から IPv6 への移行が徐々に進んでいるという段階であり、総務省でも、電子政府システムの IPv6 対応を進めるにあたり、各府省における計画策定の際に参考とすべき内容をまとめた「電子政府システムの IPv6 対応に向けたガイドライン」を策定し、公表している。

地域情報プラットフォームにおいても、今後、自治体、民間での普及状態を見て、IPv6 の採用を検討する。

- 参考：「電子政府システムの IPv6 対応に向けたガイドラインの策定について」
  http://www.soumu.go.jp/s-news/2007/070402_5.html

（２）　転送プロトコル

PF 通信のアプリケーション層の通信プロトコルとしては、Web サーバ・Web ブラウザ間の通信で広く使われている、HTTP（Hypertext Transfer Protocol）1.1 を採用する。

ただし、地域情報プラットフォームの中の補助的な機能において、HTTP 以外の通信プロトコルを採用する場合がある（時刻同期機能における NTP（Network Time Protocol）の使用など）。

（３）　通信セキュリティ

セキュリティに関する詳細については、プラットフォーム通信標準仕様の「５．プラットフォーム通信標準における認証・認可・セキュリティ機能」および本ガイドラインの「３．４　セキュリティ」、「３．５　PF 共通機能　認証・認可・セキュリティ」を参照のこと。

３．１．２　SOAP 通信と通信モデル

（１）　SOAP 通信

サービス連携実現には、SOAP（Simple Object Access Protocol）1.1 を採用する。

SOAP は XML（Extensible Markup Language）をベースとしており、SOAP Envelope という電子封筒形式を規定している。SOAP Envelope は SOAP Header と SOAP Body から構成されており、SOAP Body に送信するメッセージ本体を置き、SOAP Header にメッセージの処理に関連する付加的な情報を置く。

図３．１．１　SOAP Envelope

SOAP 1.1 はまた、HTTP のリクエスト・レスポンスに SOAP Envelope を載せて通信する方法（バインディング）も規定している。

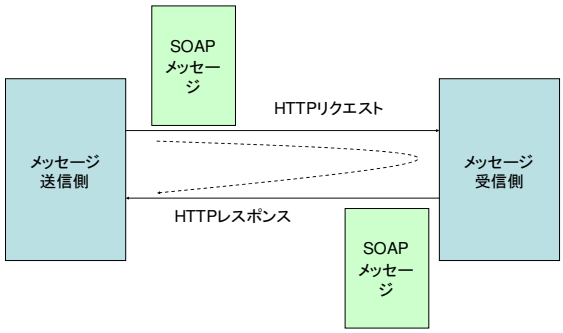


図３．１．２　SOAP の HTTP へのバインディング

SOAP の利点の１つとして、下位通信プロトコルに依存しないという点がある。現在の主流は下位プロトコルとして HTTP を使用することであるが、今後必要に応じて他のプロトコルを採用することが可能である。

SOAP の他の利点として、様々な機能を提供する Web サービスの仕様のスタックを利用できる点がある。ミドルウェアが標準仕様のスタックの実装を提供することによって、セキュリティ、高信頼性等の機能を利用できるようになる。地域情報プラットフォームにおいては、仕様の標準化状況、実装の普及状況、相互接続性の実証状況、地域情報プラットフォームにおける要件等を考慮して採用を検討する。

SOAP 1.1 は標準化団体で審議された仕様ではないが、Web サービスの基本仕様として広く利用されており、WS-I（Web Services Interoperability Organization）で策定された Basic Profile において、

仕様の曖昧な部分の明確化などがなされている。W3C（World Wide Web Consortium）においてすでに SOAP 1.2 が標準化されているが、現在の普及状況から SOAP 1.1 を採用する。併せて、Basic Profile 1.0 も採用する。

（２）　電子封筒形式

SOAP では SOAP Envelope と呼ばれる電子封筒形式を規定しているが、PF 通信においては用途に応じて 2 つの電子封筒形式を規定する。1 つは、ビジネス電文や業務ユニット間のデータ交換で使用するビジネスメッセージ用の電子封筒形式で、これは SOAP Body にプラットフォーム通信標準仕様で定める共通ヘッダとメッセージ本文を記述するものである。もう 1 つは、モニタリング機能等システム管理のためのシステムメッセージ用の電子封筒形式で、これは SOAP Body にメッセージ本文をそのまま記述するものである。PF 通信機能は 2 つの電子封筒形式の両方をサポートできる必要がある。

（３）　添付ファイル

添付ファイルは、以下の場合に利用する。

- 申請・届出において、申請書、もしくは届出書以外に、手続きに必要な書類を送信する場合
- 申請・届出等を受け付け、処理を行った団体から、申請者に交付物を送信する場合

添付ファイルの利用イメージを下図に示す。

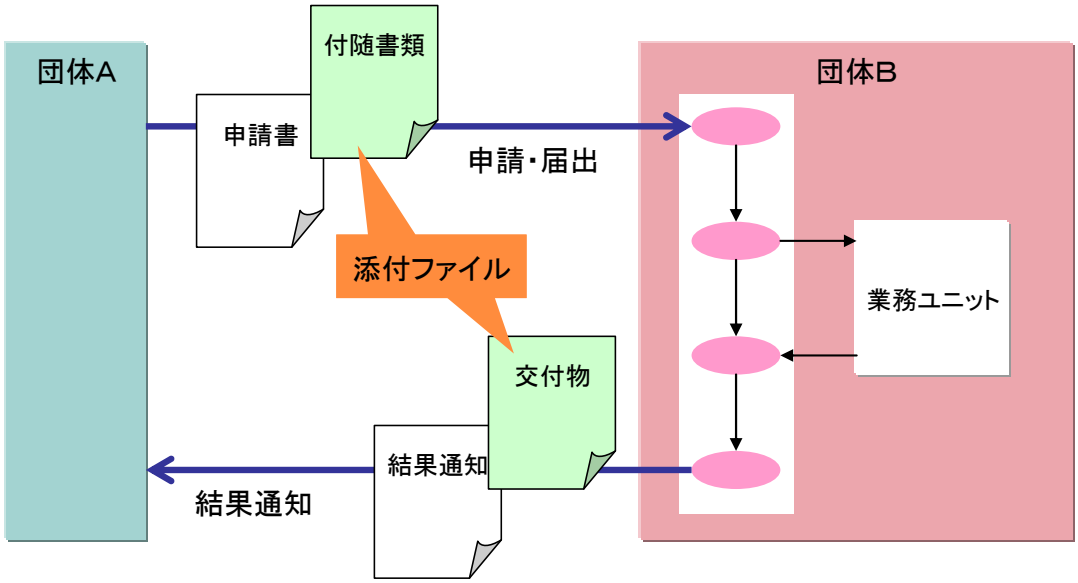


図３．１．３　添付ファイルの利用イメージ

添付ファイル実装時には、書類のサイズに留意する必要がある。

- 申請書の想定サイズ　：　数キロから数十 Kbyte（署名つき）×ワンストップサービスにおける手続き数（結果通知は申請書と同等もしくはそれ以下のサイズと想定される）
- 交付物の想定サイズ　：　数百 Kbyte から 1Mbyte

上記の想定サイズから、添付ファイルを含む書類サイズは、数 Mbyte オーダを想定する必要がある。

なお、システム的なリソースの限界があり、無限に大きい添付ファイルの交換は、現実的でない。このため、上記の要件（数 Mbyte オーダ）以下を推奨サイズとする。

PF 通信において、添付ファイルの取り扱いはオプションであるが、添付ファイルを扱うときのために、メッセージ本体格納型とメッセージへの添付型の 2 種類の電子封筒形式を規定する。

メッセージ本体格納型の場合、〈添付書類〉タグの〈添付書類内容〉タグの中に添付ファイルの内容を Base64 エンコーディングして入れる。〈添付書類名称〉タグには、「住民票」など、添付ファイルの中身を示す文字列を入れ、〈添付書類ファイル名称〉タグにはファイルとして処理する場合のファイル名を入れる。〈添付書類参照情報〉タグは使用しない。

メッセージへの添付型の場合、SwA（SOAP Messages with Attachments）仕様に基づき、MIME（Multipurpose Internet Mail Extension）パートに添付ファイルを置き、メッセージ本文から〈添付書類〉タグを使用して参照する。〈添付書類ファイル名称〉タグには、同時に送信する添付ファイル内でユニークとなる英数字のファイル名称を設定する。また、〈添付書類参照情報〉タグに、対象の添付ファイルを格納する MIME パートの Content-ID ヘッダに対応する cid URI を格納する。〈添付書類内容〉タグは使用しない。MIME ヘッダの情報と〈添付書類〉タグの情報の関係を図３．１．４に示す。

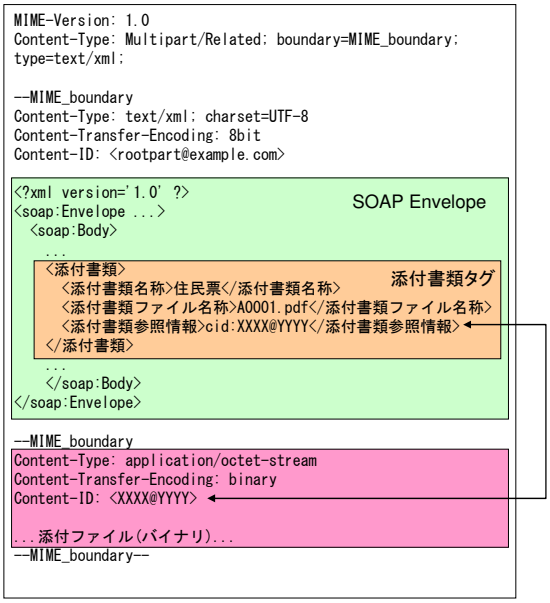


図３．１．４　メッセージへの添付型

メッセージ本体格納型は添付ファイルを Base64 エンコーディングするため、送信するメッセージのサイズが大きくなってしまう。そのため添付ファイルの容量が少ない場合のみ使用することが望ましい。特にサイト間の通信においては通信容量の小さいネットワークを経由する可能性もあるため、メッセージへの添付（SwA）型を使用することを推奨する。使い分けの目安を表３．１．１に示す。

表３．１．１　添付ファイルがある場合の電子封筒形式採用基準

	添付方式	採用基準
1	メッセージ本体格納型	添付ファイル容量が 1Mbyte 未満の場合
2	メッセージへの添付（SwA）型	添付ファイル容量が 1Mbyte 以上の場合

（４）　　データ交換システムパターン

PF 通信においては、データの受け渡し方法（本文内埋め込み、MIME で添付、統合 DB 機能利用）と BPM 機能との連携の有無の組み合わせで、6 種類のデータ交換システムパターンが考えられる。しかし、現在 BPM 機能の動作記述のために採用している WS-BPEL（Web Services Business Process Execution Language）仕様が添付ファイルの処理について規定していないため、BPM 機能と連携した本文と添付（MIME で添付）型は採用しない。したがって、PF 通信機能で採用するのは、仕様書の記述通り 5 種類のデータ交換システムパターンとなる。

統合 DB 機能経由のデータ交換システムパターン[Type3]、[Type5]については、統合 DB 機能自体がサイト内での利用を前提としたものであるため、サイト間では使用しない。

表３．１．２　データ交換システムパターン

	BPM機能との連携なし	BPM機能との連携あり
本文内埋め込み	[Type1] サイト内、サイト間	[Type4] サイト内、サイト間
MIMEで添付	[Type2] サイト内、サイト間	採用せず
統合DB機能を利用	[Type3] サイト内のみ	[Type5] サイト内のみ

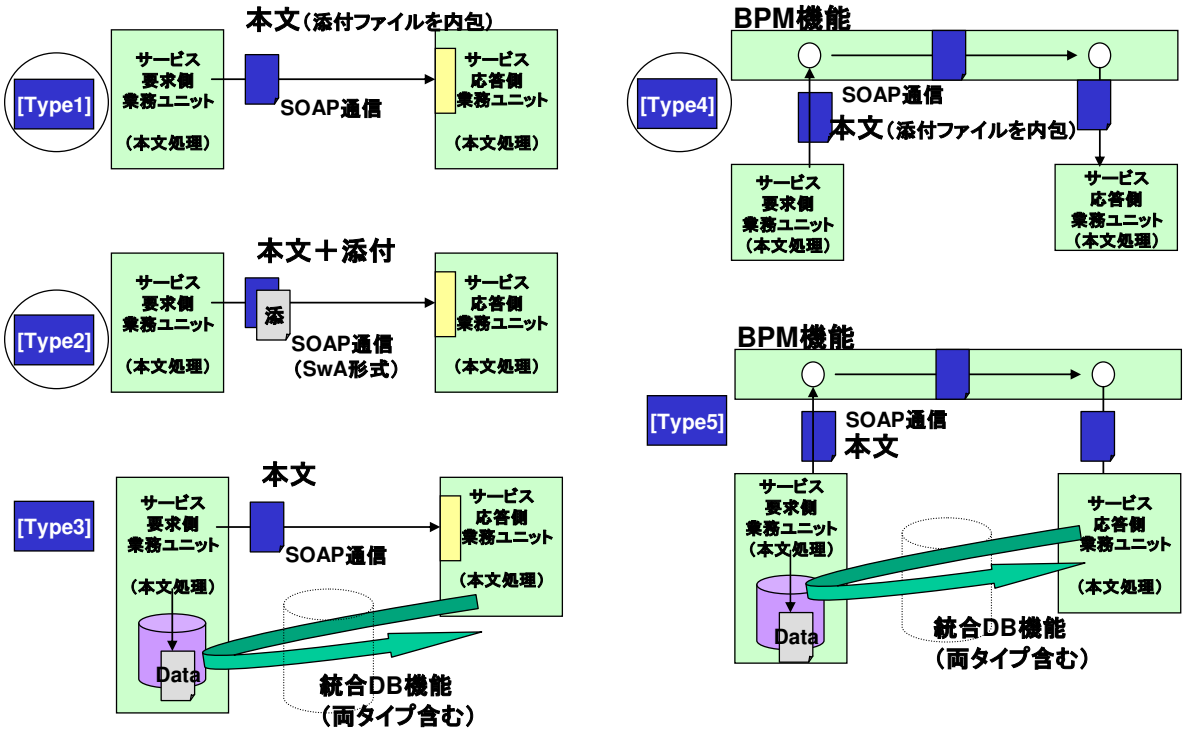


図３．１．５　データ交換システムパターン

本文内埋め込み型の[Type1]、[Type4]と MIME で添付型の[Type2]はそれぞれ添付ファイルがある場合の電子封筒形式のメッセージ本体格納型とメッセージへの添付（SwA）型に対応しており、BPM 機能がある場合を除き、添付ファイルがある場合の電子封筒形式の採用基準を目安に使い分ける。

サイト内においては統合 DB 機能を利用するデータ交換システムパターン[Type3]、[Type5]が利用可能である。統合 DB 機能を利用するかどうかについては他の検討要素もあるが、次の用途で統合 DB 機能によるデータ交換システムパターンを推奨する。

① メッセージ通信の負荷軽減
- 全データがメッセージに含まれて交換される負荷を軽減する。

② データ統合（交換手段）の一元化
- 提供側（情報源）を意識しないで統合 DB 機能経由のデータ交換を実現する。
- 利用側は統合 DB 機能だけを利用すればよいので負荷軽減できる。

③ 既存システムのデータ活用
- 地域情報プラットフォーム未対応の既存システムのデータも統合 DB 機能経由で活用できる。

④ 高度なデータ活用
- 統合 DB 機能の応用的な使い方として、複数の提供側業務ユニットに跨るデータの統合結果を活用できる（「３．２．７．４　複数の情報源の複雑な統合」を参照）。

なお、統合 DB 機能の詳細については、アーキテクチャ標準仕様の「４．５．４　統合 DB 機能」および本ガイドラインの「３．２　統合 DB 機能」を参照のこと。

３．１．３　高信頼性通信機能

高信頼性通信機能は SOAP 通信処理の信頼性を向上させるものであり、メッセージの送達保証（失敗時再送）、重複排除、順序保証の機能を持つ。送信側アプリケーションは高信頼性通信処理系にメッセージを渡すだけで、送信側と受信側の高信頼性通信処理系の間のメッセージ交換により、メッセージは高信頼で受信側アプリケーションに渡される。高信頼性通信処理系間では受領確認メッセージ（これは、プラットフォーム通信標準仕様の「６．プラットフォーム通信における MEP と異常系処理」のメッセージ交換パターン（MEP）における受領 Ack とは異なる）送付やメッセージ再送等の処理を行うことによって信頼性を高めるが、アプリケーションがその通信プロトコルや処理を意識する必要はない。

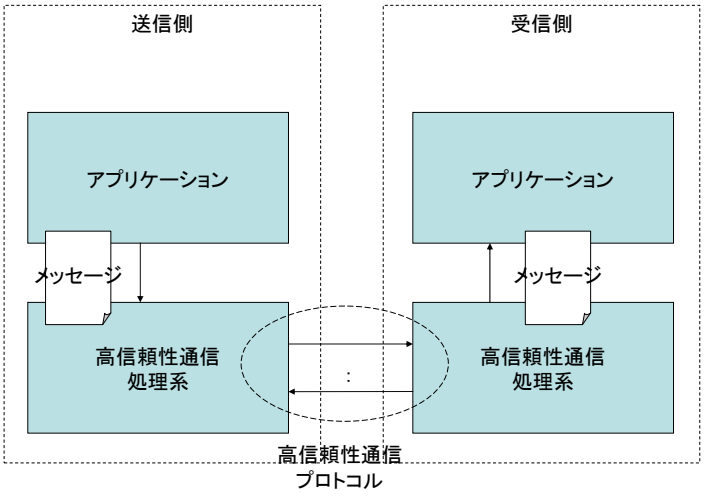


図３．１．６　高信頼性通信機能

以下に、高信頼性通信機能に含まれる個々の機能を説明する。

- 送達保証:送信側高信頼性通信処理系が受信側高信頼性通信処理系から受領確認メッセージを受け取るまでメッセージを再送することにより、メッセージの送達を保証する。受信側がメッセージを受け取っても送信側が受領確認メッセージを受け取れない場合に受信側が同じメッセージを複数回受け取る可能性が残る。これは、最低1回はメッセージが受領されることを保証するため At Least Once とも表現される。
- 重複排除:受信側高信頼性通信処理系が、同一メッセージを複数受け取った場合に受信側アプリケーションに一度だけメッセージを渡すことによって、メッセージの重複が起こらないことを保証する。これは、最高1回しかメッセージが受領されないことを保証するため At Most Once とも表現される。
- 送達保証+重複排除:この2つを組み合わせることによって、送信側アプリケーションが送ったメッセージが確実に重複なしに受信側アプリケーションに達することを保証する。これは、正確に1回だけメッセージが受領されることを保証するため Exactly Once とも表現される。
- 順序保証:送信側から受信側に複数メッセージが送られるときに、送信側アプリケーションが送った順序で受信側アプリケーションがメッセージを受け取ることを保証する。これは In Order とも表現される。なお、順序保証は機能的に送達保証と重複排除を含んでいる。

これまでも、確実なメッセージの送信が必要な場合においては、アプリケーションレベルでの送達確認等の手段により対応してきたが、ここにあげた仕様に基づいたシステムを実現することにより、アプリケーションよりも下位のレイヤで高信頼性な通信を行うことが可能となる。ただし、通信に回復不能な致命的な障害が発生した場合においては最終的にはこれまでどおりのアプリケーションによる対応は不可欠である。

SOAP を高信頼にする仕様としては、共に OASIS (Organization for the Advancement of Structured Information Standards) において標準化された次の2つが存在する。

- WS-Reliability (Web Services Reliability) 1.1 (WS-R) [2004 年 11 月に OASIS Standard として承認]
- WS-ReliableMessaging (Web Services Reliable Messaging) 1.1 (WS-RM) [2007 年 6 月に OASIS Standard として承認]

これらの標準仕様は、送達保証、重複排除、順序保証の機能としては同等であるが、通信プロトコルが異なっており、現在のところ相互接続できない。そのため、PF 通信機能としての高信頼性通信機能の相互接続性を保証するためには、両仕様をサポートすることを必須とするか、何れかの仕様に一本化することが必要となる。一本化する場合、サイト内の通信に限定すれば、サイト毎にどちらか一方の仕様を選択して利用することが可能である。しかし、将来的にサイト間通信も行うことを考慮すると、サイト間の仕様と同じことが望ましい。サイト内の仕様とサイト間の仕様が異なる場合、サイト内とサイト間を繋ぐサービスにおいて両仕様をサポートするなどの対応が必要となる。

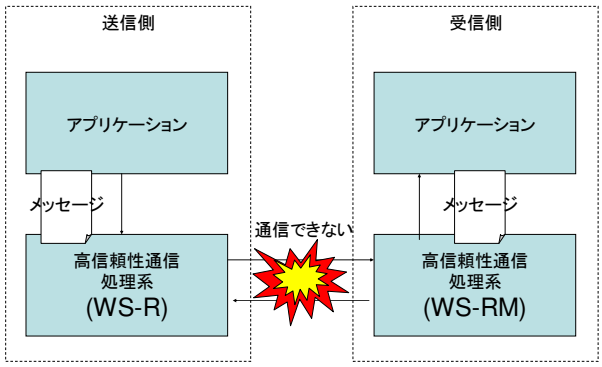


図３．１．７　WS-R と WS-RM の相互接続

以上の説明は WS-R と WS-RM とが相互接続できないという前提とした場合であり、今後、標準化団体において、もしくは、地域情報プラットフォームの仕様として、2 つの仕様の実装同士を相互接続するための仕様を作成することによって、両仕様の混在が可能になることも考えられる。2 つの仕様の実装同士を相互接続するための仕様の例としては、以下のような通信プロトコル変換ゲートウェイのようなものが考えられるが、その実現可能性も含めて検討が必要である。

図３．１．８　WS-R/WS-RM 接続ゲートウェイ

地域情報プラットフォーム標準仕様 V1.0 を策定した時点では、WS-R のみが唯一の標準仕様であったためこれを採用したが、V2.0 の策定段階において WS-RM が標準仕様として完成した。2 つの標準仕様の並存には上で説明したような問題があるため、V2.0 では高信頼性通信機能に関して仕様化していない。高信頼性通信は、これから利用が進む技術であり、相互接続性の確認や普及の状況を注視していく必要がある。現在の段階で高信頼性通信機能を採用する場合は、上記問題点を理解し、リスクをもって採用すること。

なお、プラットフォーム通信標準仕様および本ガイドラインでは、上位層における MEP（メッセージ交換パターン）を規定しており、障害を検出した場合の対応方針についても明記している。MEP は高信頼性通信機能の上位に位置付けられるため、高信頼性通信機能の有無に係わらず、通信上で発生した障害は統一的に処理される。高信頼性通信機能を利用すると、回復可能な障害は高信頼性通信機能のレベルで吸収され、上位層では回復不能な致命的な障害を扱うことになる。プラットフォーム通信標準仕様の「６．プラットフォーム通信における MEP と異常系処理」を参照のこと。

３．１．４　通信において設定すべき項目

サービスが他のサービスと通信（連携）するためには、通信先のサービスの各種情報を設定しておく必要がある。設定すべき項目としては、以下のようなものが考えられる。

- HTTP のタイムアウト
- 本文メッセージの最大サイズ（最大バイト数など）
- 各サービスのエンドポイント URL（Uniform Resource Locator）（DNS（Domain Name System）への設定）
- 認証機能の設定
- 認証方式の選択（SSL（Secure Sockets Layer）クライアント認証、HTTP ベーシック認証、等）
- 認証情報の交換（信頼できる CA（Certification Authority）情報や ID やパスワード、等）認証機能の設定

また、採用するオプション機能に応じて、そのオプション機能に固有の設定が必要になる場合もある。

これらの情報は通信するサービス間で何らかの方法で交換される必要がある。将来的にはサービスレジストリ機能等を利用して交換されることが想定されるが、現時点では、サービスの管理者間の情報交換などにより手動で設定することが現実的である。

３．２　統合 DB 機能

統合 DB 機能は、業務ユニット間で必要となるデータを統合的に管理することにより、自治体内における業務ユニット間のデータ連携を効率的に行う機能である。

従来における業務ユニット間のデータ交換は、個別に開発・運用が行われており、標準化していないため、業務ユニットの差し替えが困難であった。

統合 DB 機能は、各業務ユニットで分散管理されているデータを統合して、標準化された手段（業務ユニット間のデータ連携のインタフェース）により必要とする業務ユニットに提供する。これにより、データ交換の側面から業務ユニットの差し替えを容易にしている。（図 3.2.1）

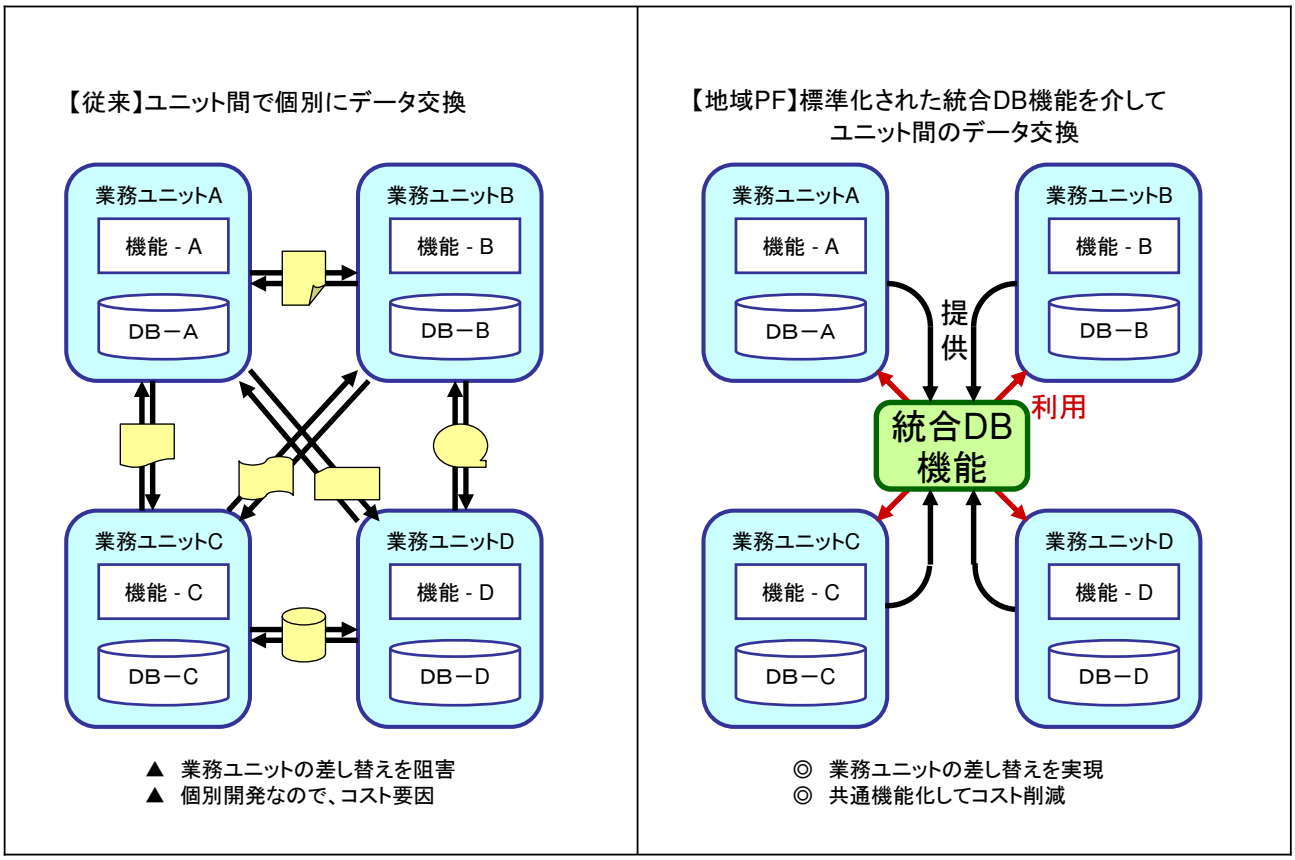


図３．２．１ 統合 DB 機能におけるデータ交換

自らデータを管理（保持）して、他の業務ユニットにデータを提供する役割を持つ業務ユニットを「提供側業務ユニット」、他の業務ユニットのデータを利用（参照）する業務ユニットを「利用側業務ユニット」と呼ぶ。これは、データ交換における業務ユニットの役割により区別するものであり、図 3.2.1 でも解るように、１つの業務ユニットが、「提供側業務ユニット」として他の業務ユニットにデータの提供を行い、「利用側業務ユニット」として他の業務ユニットのデータを利用するのが一般的である。

統合 DB 機能の視点からは、統合 DB 機能にデータを提供する役割を持つ「提供側業務ユニット」、統合 DB 機能を利用する「利用側業務ユニット」と、統合 DB 機能に関する役割により使い分けている。

３．２．１　統合 DB 機能の基礎

統合 DB 機能はオプションであり、方式として「公開用 DB 方式」と「共通インタフェース方式」を選択できる。

最初に統合 DB 機能に共通の基礎概念を解説し、方式や実装に依存する部分は、次の項で解説する。

・方式に依存する解説 →　「3.2.2 統合 DB 機能の方式と選択基準」

・実装に依存する解説 →　「3.2.3 統合 DB 機能の実装モデル」

３．２．１．１　統合 DB 機能の使われ方

アーキテクチャ標準仕様で定義している統合 DB 機能の使われ方 3 種類について解説する。

◇A：独立型（業務ユニット間の SOAP 通信を伴わないで、統合 DB によるデータ交換のみ）

◇B：本文とリファレンス（統合 DB 機能の検索キーを交換）型

◇C：BPM 機能と連携した本文とリファレンス（統合 DB 機能の検索キーを交換）型

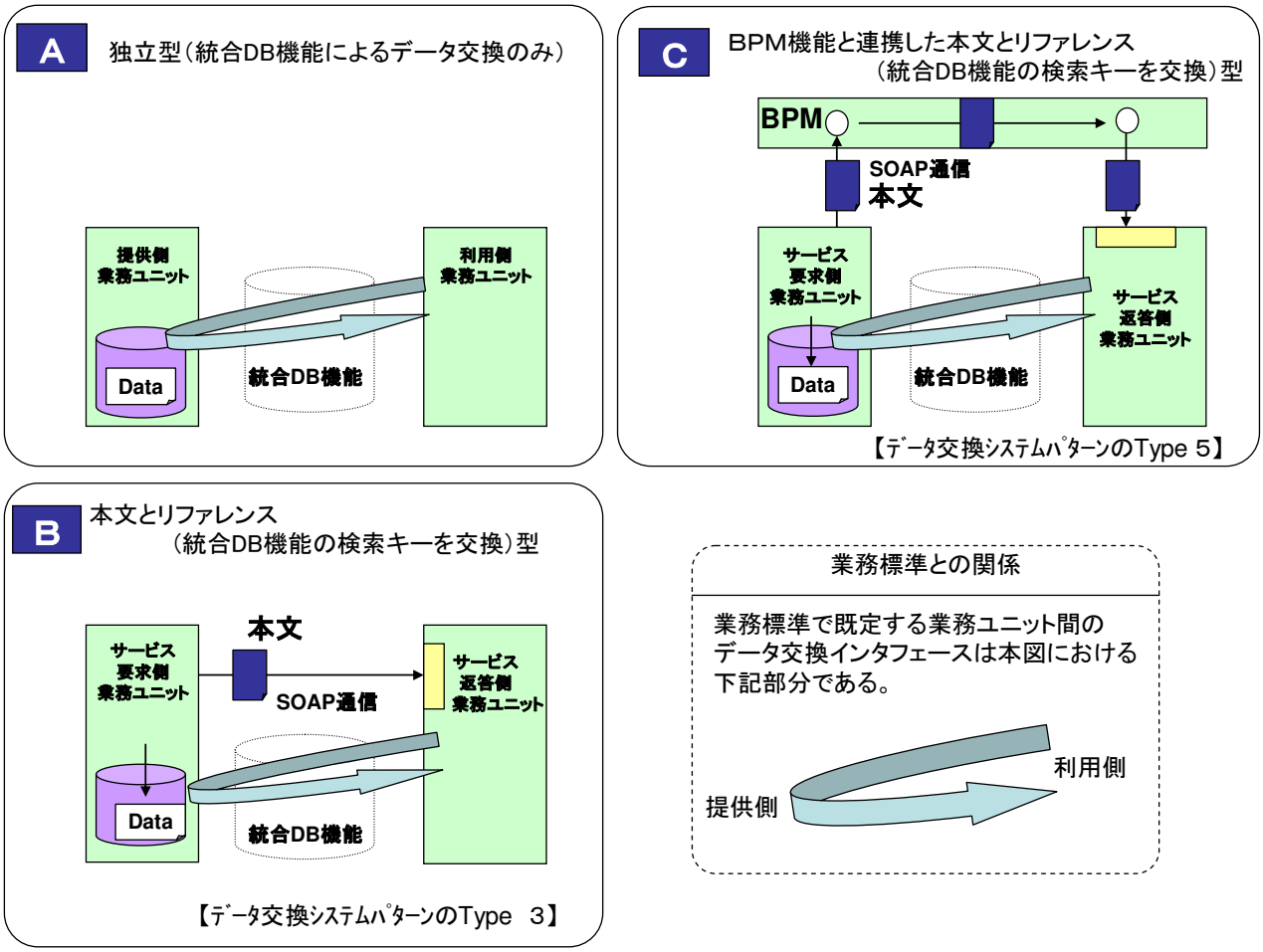


図３．２．２ 統合 DB 機能によるデータ交換方式

【A】「独立型（統合 DB 機能によるデータ交換のみ）」

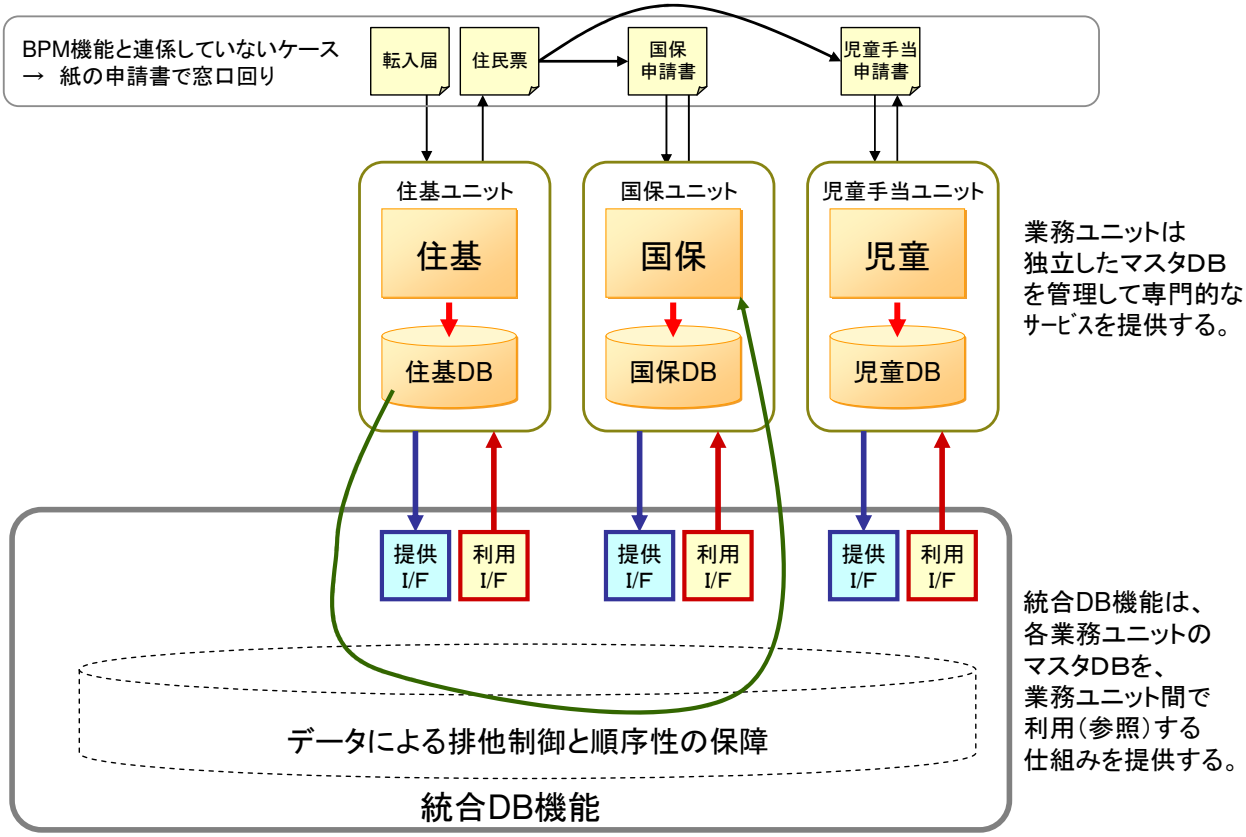


図３．２．３　統合 DB 機能の使い方（A 方式）

A の方式は、SOAP による業務ユニット間連係を行わない独立した業務ユニット間において、統合 DB 機能を介したデータ交換を行うものである。特にこの型は、完全には地域情報 PF に対応していない既存の業務ユニットにおいて、他の業務ユニットのデータを活用する手段として採用することができるので、段階的な地域情報 PF 対応を実現する手段として効果的である。

住基ユニットにおける転入届の処理が完了して住民票が出力されるのを待って、次の国保ユニットへの処理を依頼するなど、人の判断による運用により全体の順序性保障を行うことができる方式であり、情報システムと人との役割を整理して、全体として矛盾の無い運用を行う必要がある。

【B】「本文とリファレンス（統合 DB 機能の検索キーを交換）型」（Type 3）

サービス要求側の業務ユニットは自身が管理するデータを更新（または登録）した後、SOAP 通信として、リファレンス（統合 DB 機能の検索キー）を付与した依頼メッセージをサービス応答側の業務ユニットに送信する。サービス応答側の業務ユニットは、依頼メッセージを受け取ると、その中のリファレンスを参照して統合 DB 機能を検索することにより、サービス要求側の必要なデータを取得して、依頼された処理を実行することができる。

B の方式では、BPM 機能による業務の流れ制御が行われないため、各業務ユニットは、統合 DB 機能を介して得ることができる他の業務ユニットの状態（データ）を確認しながら、担当する業務を矛盾無く遂行する必要がある。

例えば、図 3.2.4 の住基ユニットにおいて転入届が処理され、住基 DB に当該住民の情報が正常に登録完了しており、統合 DB 機能を介して住基 DB の該当データが参照できることを確認した後で、国保ユニットは国民健康保険の受付処理を行うことにより、順序性を保障することが求められる。

ここで行う順序性の保障は、業務ユニットの処理として実現するものであり、統合 DB 機能は、必要な情報を利用可能にするという意味で、他の方式と役割や機能が変わるものではない。

B の方式における順序性保障は確実で影響範囲が狭い 3.2.1.2(2)を推奨する。

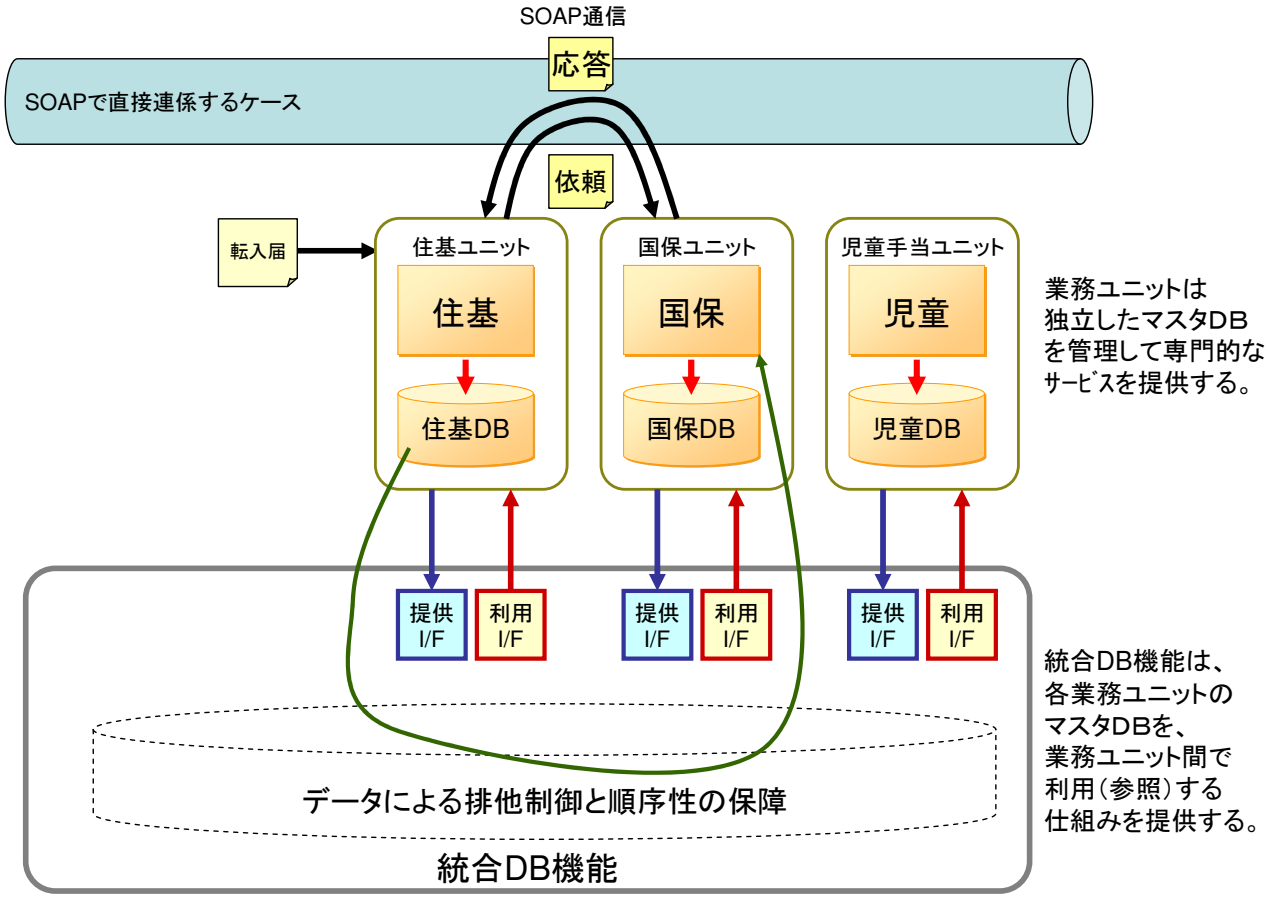


図３．２．４ 統合 DB 機能の使い方（B 方式）

【C】「BPM 機能と連携した本文とリファレンス（統合 DB 機能の検索キーを交換）型」(Type 5)

統合 DB 機能は、利用側業務ユニットから直接利用（参照）されるため、BPM 機能から直接制御されることはないが、C 方式の通信では、業務ユニット間の業務の流れを、BPM 機能が制御しているため、データの排他制御や更新順序性の保障などは、BPM 機能の制御下で実現されることとなる。
例えば図 3.2.5 において、住基ユニットは、依頼された転入手続きを実行して、自ら管理している住基 DB に転入者の基本情報を登録した後、BPM 機能に処理完了を通知する。
次に BPM 機能から依頼を受けた国保ユニットは、転入手続きが完了して、住基ユニット内の住基 DB に登録された転入者の基本データを、統合 DB 機能の利用 I/F を通して参照することが可能になる。

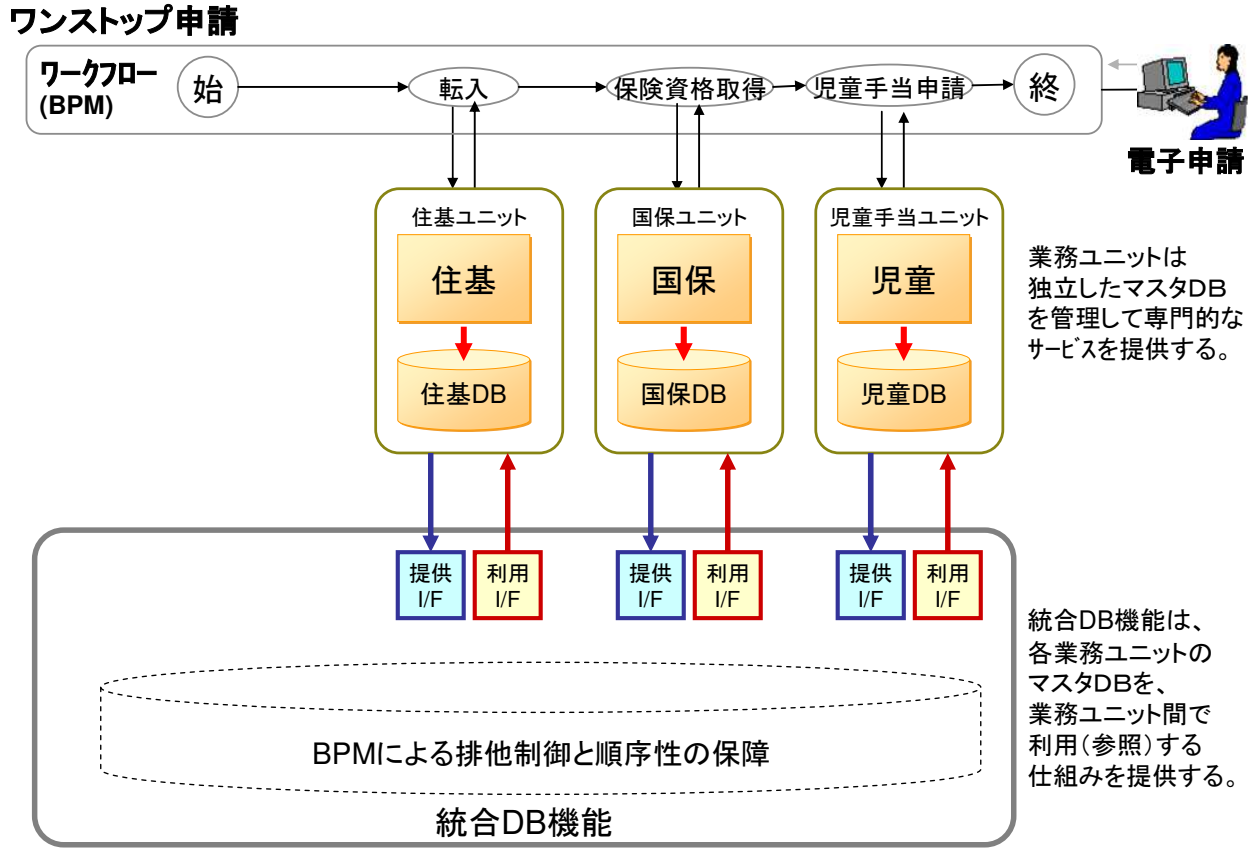


図３．２．５ 統合 DB 機能の使い方(C 方式)

ここで重要な留意点は、次の３点である。

①業務ユニット内で、処理（トランザクション）が矛盾なく完結していること。
即ち、提供側業務ユニット（住基ユニット）は、自己が管理するデータベース（住基 DB）へのデータ登録、統合 DB 機能に対するデータ提供など、全ての処理が完了したのを確認した後で、BPM 機能に処理完了の通知を行う必要がある。

②BPM 機能は、前の処理完了を確認して、次の処理要求を発行する順次制御を行うこと。

③統合 DB 機能は、提供側業務ユニットの必要なデータについて、その内容を即時反映して、最新の内容を利用側業務ユニットに応答することが求められる。
これらを遵守することで、BPM 機能による処理（ワンストップサービスによる申請など）を矛盾なく完結することができる。

３．２．１．２　統合 DB 機能における順序性の保障

統合 DB 機能を利用する業務システム間のデータ交換では、適切な順序性保障によって、正しいデータが交換できるようにする必要がある。

一般的な順序性の保障方法としては概ね次の 3 つが考えられる。

（１）データに基づく保障

利用側業務ユニットが、処理開始前に統合 DB 機能を介して必要なデータが参照可能であることを確認する方法である。新規登録されたデータについてはデータの有無で確認できるが、更新、削除のケースは、処理中フラグや、処理完了番号など、業務ユニット側の制御に必要な項目を統合 DB 機能が標準で持つデータに追加して制御する必要がある。

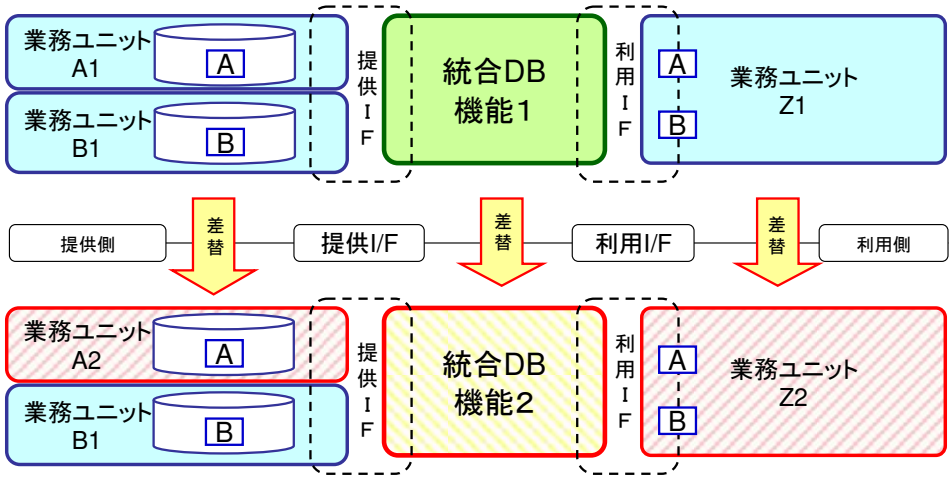
（２）処理完了情報による保障

提供側業務ユニットの処理完了報告を確認したあとで、利用側業務ユニットに依頼メッセージを送信する方法である。このとき、提供側業務ユニットは、統合 DB 機能を介して、自身が管理している該当データを、利用側業務ユニットから利用できる状態になったことを確認した後で、処理完了報告を行うことが求められる。

（３）運用による保障

予め、提供側業務ユニットの処理結果について、統合 DB 機能を介して利用できる状態になるまでの処理時間を決めておき、決められた時間を経過したことを確認して次の利用側業務ユニットの処理を開始する方法である。例えば、提供側業務ユニットの処理結果を、夜間のバッチ処理で統合 DB 機能に反映することを、運用前提として決めておき、次の利用側業務ユニットの処理は、翌日に行うように制御する方法である。

３．２．１．３　統合 DB 機能における差し替えの考え方



□従来：差し替えにはDB及びアプリケーションの改修が必要　→　**改善したい。**

ここを狙う　□現実：部品化されたオプションを選択して、設定変更などを行えば、差し替えられる。
既存システムを含めた実情に最適な方式を選択できる。

□理想：何もしないで差し替えられる。　→　**広範囲の標準化が必要で、硬直化する恐れがある。**

図３．２．６　業務ユニットおよび統合 DB 機能の差し替え

統合 DB 機能を標準化する利点の１つは、「業務ユニットおよび統合 DB 機能の差し替え（互換性）を容易にすること。」である。

何もしないで（コスト 0 で）各ユニットの差し替えを実現することが理想であるが、理想の実現には広範囲の詳細な標準化が必要となり、硬直化やコストアップの要因になる。

そこで、現実的な解として、「差し替えとは、できるだけ最小限のコストでユニットの差し替えが可能なこと。」を目指している。

地域情報 PF に対応することにより「提供側業務ユニット」、「統合 DB 機能」、「利用側業務ユニット」の替え性を高めることができる。

３．２．１．４　統合 DB 機能の役割と標準化ポイント

統合 DB 機能および関連する業務ユニットの役割と標準化ポイントについて図３．２．７により解説する。

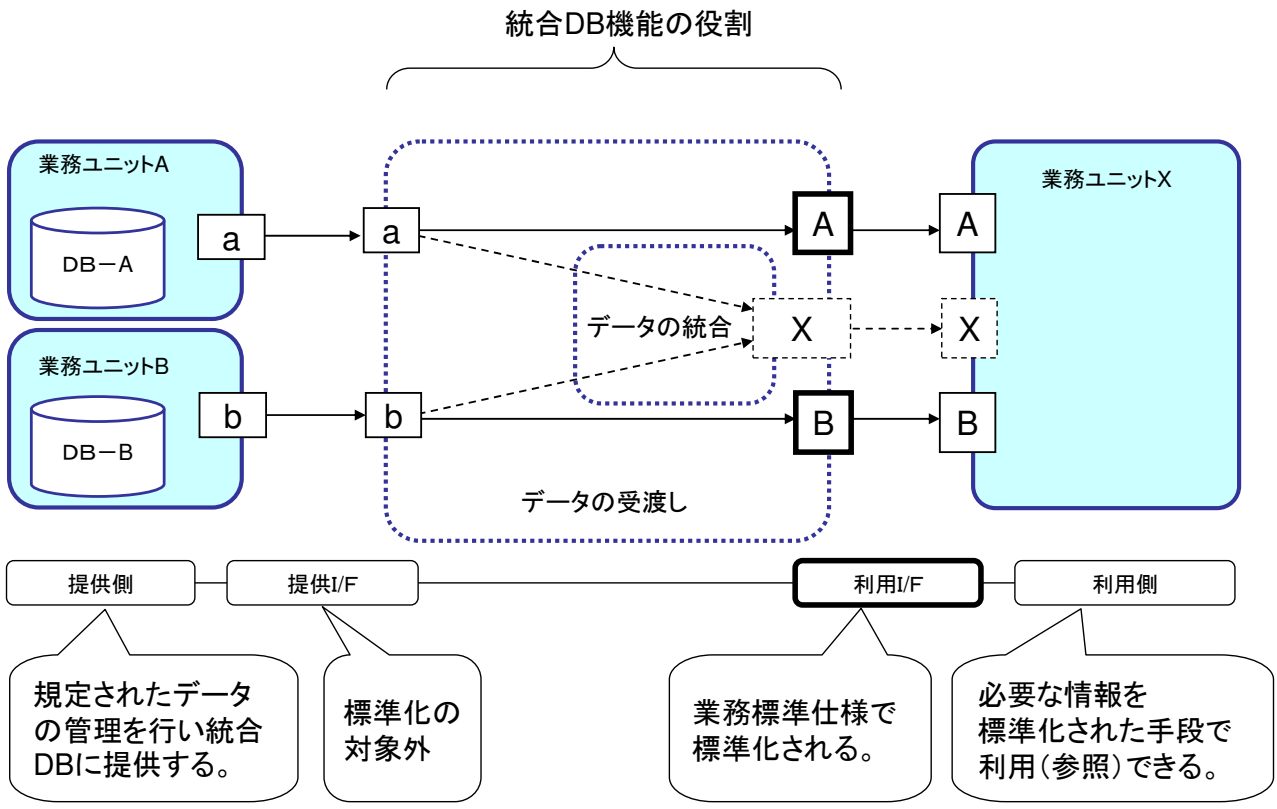


図３．２．７　統合 DB 機能の役割と標準化ポイント

（１）提供側業務ユニット

統合 DB 機能にデータを提供する業務ユニット（以下提供側または提供側業務ユニット）は、「自治体業務アプリケーションユニット標準仕様」で規定するデータについて責任を持って管理を行い、統合 DB 機能にデータを提供する。
提供側におけるデータの管理方法（データモデルなど）は規定していないので、業務ユニットに最適な管理方法を採用することができる。

（２）提供 I/F

提供側から統合 DB 機能にデータを提供するインタフェース（提供 I/F）は、データモデル（データ構造）を含めて標準で規定していないので、提供側業務ユニットと統合 DB 機能との組み合わせにより、最適なインタフェースを採用することができる。但し、業務ユニットおよび統合 DB 機能の差し替え性を高めるために、統合 DB 機能の方式毎の提供 I/F について選択肢を推奨しているので参考にして頂きたい。（詳細は 3.2.2.2～3 節で解説する。）

（３）統合 DB 機能

統合 DB 機能の役割は、提供側業務ユニットを意識することなく、提供側業務ユニットで管理されているデータを利用側業務ユニットから利用するための「データの受け渡し機能」である。

統合 DB 機能におけるデータの管理方法は規定していない。特に統合 DB 機能の方式や採用するデータモデルは、提供 I/F などへの影響が大きいので慎重に設計するべきである（3.2.5 参照）。

一般的に統合 DB 機能は、複数のデータを組み合わせて必要なデータを利用するための「データ統合機能」を持っているが、この機能（図 3.2.7 の利用 I/F である X）については標準化していない。（統合 DB 機能の応用として 3.2.7.4 節で解説する）

（４）利用 I/F

統合 DB 機能を利用側業務ユニットから利用するためのインタフェース（利用 I/F）は、統合 DB 機能の方式に関わらず、「自治体業務アプリケーションユニット標準仕様」で標準化する SOAP による連係インタフェースを統合 DB 機能が実現する必要がある（必須）。利用 I/F として、SQL の採用も可能だが、インタフェースやデータモデルを標準で規定していないので、差し替え性に留意する必要がある。

（５）利用側業務ユニット

利用側業務ユニットは、他の業務ユニットで管理されている必要な情報を統合 DB 機能から取得して利用することができる。差し替え性を高めるために、他の業務ユニットで管理されているデータの参照は、統合 DB 機能がサポートする地域情報 PF 標準に準拠した利用 I/F(SOAP)により行うことを推奨する。

３．２．２　統合 DB 機能の方式と選択基準

業務ユニット間のデータ交換手段として選択できる次の 3 つの方式について、選択基準と併せて解説する。いずれの方式においても、利用側システムが必要とする正本データまたは、副本データが取得できることを提供側システムと確認する必要がある。

※ここでは、正本データを業務システムが持つデータとし、副本データを統合 DB 機能が持つデータとする。

（１）業務ユニット間の直接連携
（２）公開用 DB 方式の統合 DB 機能
（３）共通インタフェース方式の統合 DB 機能

３．２．２．１　業務ユニット間の直接連携

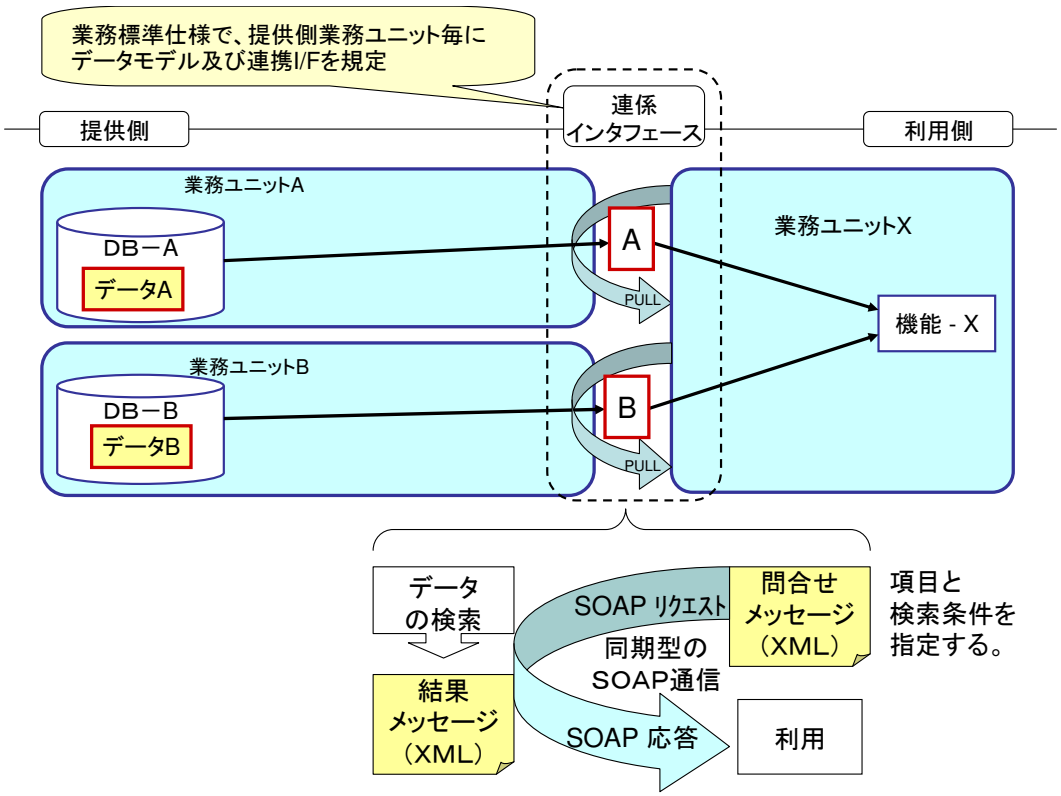


図３．２．８ 業務ユニット間の直接連携

統合 DB 機能を利用しないで、利用側システムが直接個々の提供側システムにアクセスして、必要なデータを取得する形態である。

地域情報プラットフォームの「自治体業務アプリケーションユニット標準仕様」では、各業務ユニット毎に同期型の SOAP 通信（サービス）として、データ連携のインタフェースを規定しており、地域情報 PF に準拠した提供側業務ユニットはこのインタフェースを実装している。

このインタフェースは、統合 DB 機能が実現する利用 I/F と一致しているので、統合 DB 機能を利用しなくても業務ユニット間のデータ交換が可能である。

直接連係の利点および留意事項は次の通りである。

【利　点】
（１）正本からデータを取得できる。
（２）統合 DB 機能が不要なので、システム構成が簡素化される。

【留意点】
（１）利用側業務ユニットが、依頼先（SOAP の宛先）を意識する必要がある。

（２）文字コード系やデータ表現などを一致させる必要がある。
対策：これらの留意点は、SOAP 宛先の変換（ルーティング）、メッセージの変換などを行うように構成する対策により回避できる場合がある。

（３）提供側業務ユニットの負荷が増大する。
対策：提供業務ユニット側の負荷に余裕がない場合は、データベースのレプリケーションや、CPU の増強などの対策を実施する必要がある。

（４）統合 DB 機能の応用（3.2.7 節参照）には対応できない。
対策：利用側業務ユニットが標準外の統合 DB 利用 I/F（SQL など）を利用しないように管理する必要がある。

（５）提供側業務ユニットが稼動していないタイミングではデータ交換できない。
対策：提供側業務ユニットのデータ提供機能を分離して常時稼動するように構成することが必要になるが、この対策は統合 DB 機能を導入することと等しいため、統合 DB 機能の導入を検討することを推奨する。

コラム３．２．１「同期型の SOAP 通信」

図 3.2.8 に示すように、利用側の連係インタフェースは、同期型の SOAP 通信として規定している。（詳細は通信標準 6.2.3 を参照）

同期型の SOAP 通信とは、サービス要求側から発行された SOAP による依頼メッセージに対して、サービス応答側は、依頼された処理が完全に完了するのを待って、処理結果を SOAP による応答メッセージとしてサービス要求側に返信する方式である。平行動作による効率化には制約を受けるが、逐次性が確保され、順番に決められた処理が実行される利点がある。

３．２．２．２　公開用 DB 方式の統合 DB 機能

「公開用 DB 方式」の統合 DB 機能は、統合 DB 機能として物理的な DBMS に公開対象の二次データ（提供側から提供されたデータ）を保持し、この二次データを介してデータの受け渡しを行う方式である。

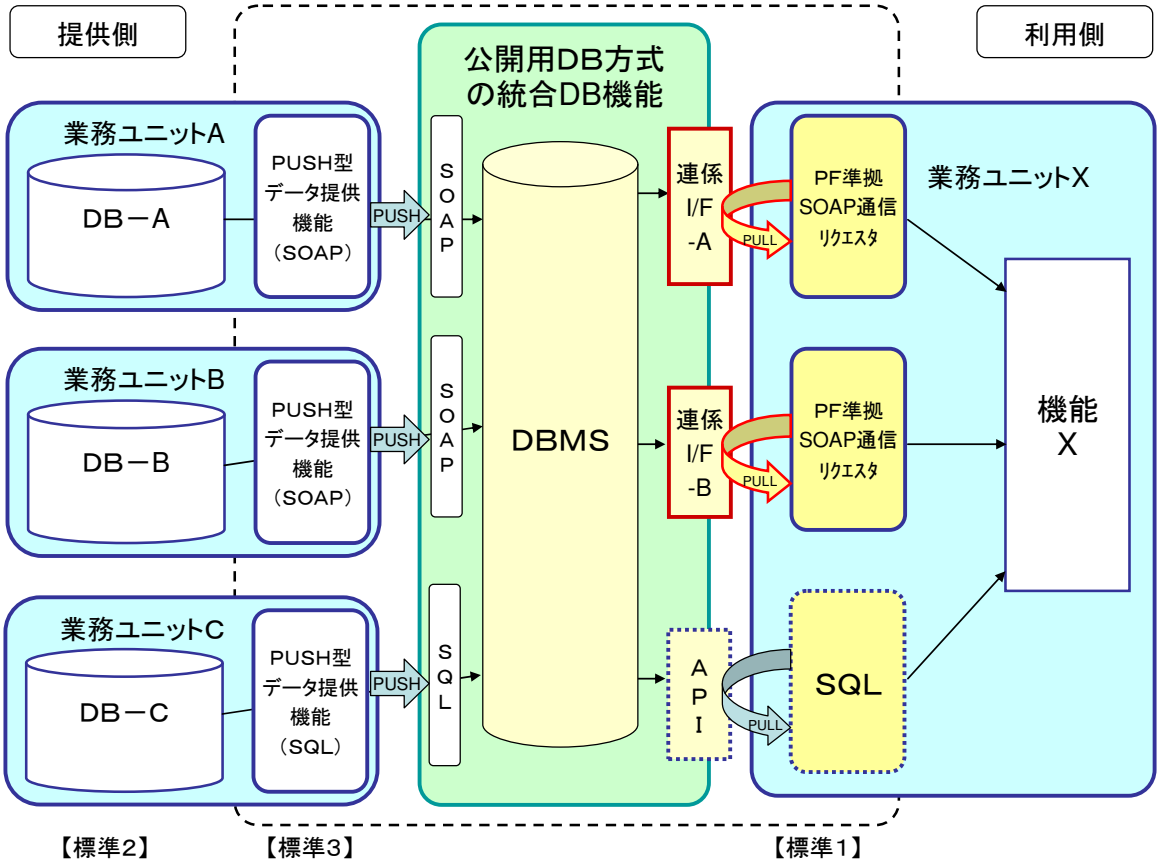


図３．２．９ 公開用 DB 方式の統合 DB 機能

（１）各ユニットの役割と留意点

ユニット毎に具体的な動作および留意点について解説する。

①提供側業務ユニット

提供側業務ユニットは、PUSH 型データ提供機能によって、自己が管理するデータに更新が発生したときに統合 DB 機能(DBMS)の該当する二次データを更新する。このデータ更新は、利用側業務ユニットなど、他の機能とは無関係に提供側業務ユニットの制御により実行される。

公開用 DB 方式の統合 DB 機能では統合 DB 機能内の物理的な DBMS で管理されている二次データを提供側業務ユニットが更新する必要があるため、同時更新（提供側業務ユニット内のデータ更新と同時に統合 DB 機能内の二次データも更新すること）は理論的に不可能であり、更新タイミングについて(3)に示す考慮が必要である。

二次データの更新（提供）手段は規定していないので、レプリケーション、SOAP、DBMS のアクセス手段(SQL)など、提供側業務ユニットと統合 DB 機能の実装によって可能な範囲で、任意の方式を採用できるが、差し替え性を考慮して次の点に留意すべきである。

◇SOAP を採用する場合
PUSH 型の SOAP インタフェースは標準で規定していないが、データモデル（XML タグ構成）は、「自治体業務アプリケーションユニット標準仕様」で標準化する SOAP によるインタフェースが参考になる。標準で規定している SOAP インタフェースとは制御方向が逆になるので注意が必要である。
即ち、「自治体業務アプリケーションユニット標準仕様」は PULL 型であるのに対して、公開用 DB 方式の統合 DB 機能における提供側業務ユニットに求められる二次データの更新機能は PUSH 型になる。（コラム 3.2.2 参照）

◇SQL を採用する場合
SQL は統合 DB 機能で用いられる DBMS で規定されるインタフェースを使用して実行され、SQL 自身も DBMS に依存した仕様になるので、差し替え性を考慮すると、そのインタフェースは、ODBC、JDBC などの標準化されたものを採用し、SQL も DBMS に依存しない範囲で使用するべきである。(3.2.7.3(3)参照)

・レプリケーションによる実現
SQL の代わりに、DBMS のレプリケーション機能により実現する方法もある。
レプリケーション機能は DBMS ベンダにより提供される場合が多いので、提供側業務ユニットで採用している DBMS と、統合 DB 機能で採用する DBMS との組み合わせ（相性）を考慮する必要がある。

②利用側業務ユニット
利用側業務ユニットは、提供側業務ユニットとは無関係に、利用 I/F を介して統合 DB 機能に検索要求を送信し、結果データを受け取る。即ち、利用側業務ユニットは、統合 DB 機能の有無および統合 DB 機能の方式に関わらず、標準化された利用 I/F によるデータ取得を実現できる。標準外で統合 DB 機能の利用 I/F として SQL を採用する場合は、統合 DB 機能で採用されている DBMS がサポートしている SQL インタフェースとなるケースが多い。この場合、インタフェースおよびデータモデルは標準化していないので、利用側業務ユニットの対応と差し替え性に留意する必要がある。

③統合 DB 機能
統合 DB 機能は利用側業務ユニットからの要求に応じて、統合 DB 機能内で管理している二次データの検索を行い、利用 I/F を介して利用側業務ユニットに検索結果データを返す。
公開用 DB 方式の統合 DB 機能は、物理的な DBMS で二次データの集中管理を行うため、性能に関して特に配慮が必要である。(3.2.3.1 参照)

（２）仕様に関する留意点
公開用 DB 方式の統合 DB 機能を採用する場合に必要となる仕様として次のものがある。

【PUSH 型データ提供機能】　前項の①に示す機能が該当する。
◇提供側業務ユニットは、採用する統合 DB 機能に適合した PUSH 型データ提供機能を有すること。
◇統合 DB 機能は、対象とする各提供側業務ユニットからの PUSH 型データ提供機能を受けて、統合 DB 機能内の DBMS を更新する機能を提供側業務ユニットに公開すること。
ここで、提供側業務ユニットと統合 DB 機能の提供 I/F が適合していることが重要であり、システム要件として具体的に決定すべきである。特に統合 DB 機能側のデータモデル（表モデル）は標準化していないので配慮が必要である。(3.2.3.1 参照)

コラム３．２．２「PULL 型のデータ提供と PUSH 型のデータ提供」

FROM 側（提供側）で管理するデータを TO 側（統合 DB 機能）に提供する場合の制御方式として、公開用 DB 方式で使用する PUSH 型と、共通インタフェース方式で使用する PULL 型がある。

◇PUSH 型のデータ提供

FROM 側に「PUSH 型データ提供機能」を持ち、データ提供機能が呼び出されると、①で該当データの取得を行い、TO 側に関係なく取得したデータを TO 側に提供する方式である。

FROM 側の制御によりデータ提供を行うことができる利点がある一方、TO 側は、提供されたデータ（二次データ）の実体を保管する必要が生じる。

インタフェースとして SOAP を採用する場合は、TO 側に②で提供されたデータを受け取って、DBMS を更新する「PUSH 型データ提供機能」が必要になるが、SQL を採用する場合は、FROM 側のデータ提供機能から直接 TO 側の DBMS を更新できるので、③のデータ提供機能は不要である。

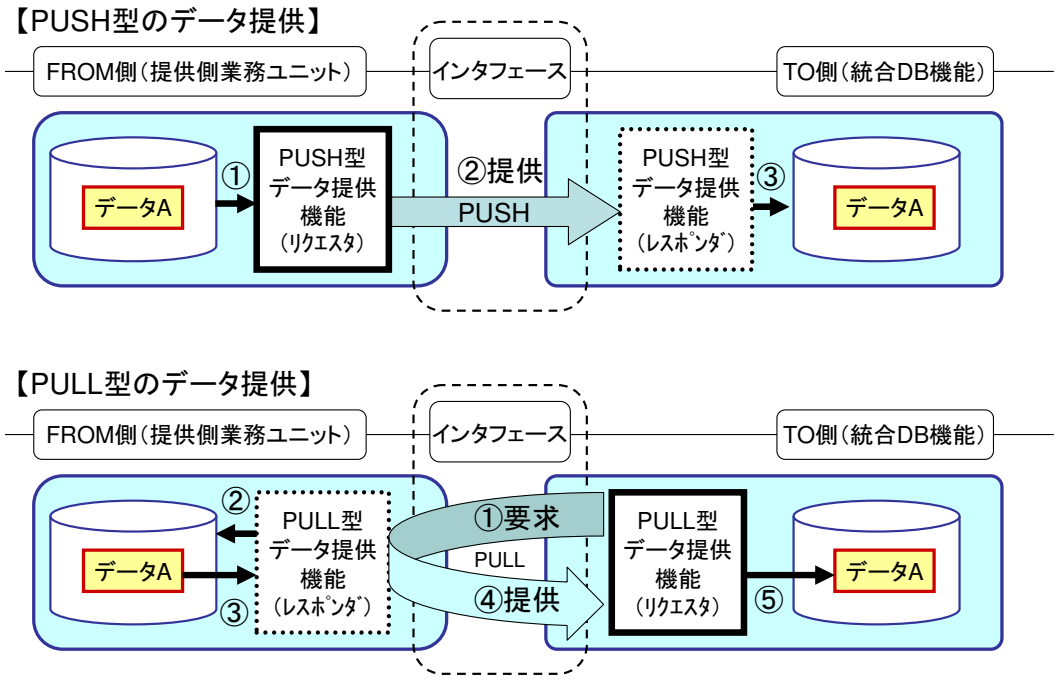


図３．２．１０ PUSH 型と PULL 型のデータ提供

◇PULL 型のデータ提供

TO 側に「データ提供機能」を持ち、TO 側のデータ提供機能からの要求①に従って、FROM 側が要求されたデータを提供④する方式である。

TO 側の制御により必要なタイミングで必要なデータだけを利用できるので、TO 側における二次データの保管が不要になる利点がある。

インタフェースとして SOAP を採用する場合は、FROM 側に、要求①されたデータの検索②を行い、検索結果③を TO 側に提供④する「PULL 型データ提供機能」が必要になるが、SQL を採用する場合は、TO 側のデータ利用機能から直接 FROM 側の DBMS を検索してデータを取得することができるので、FROM 側のデータ提供機能は不要である。

（３）PUSH 型データ提供機能におけるデータ更新タイミング

提供側業務ユニット内のデータが、統合 DB 機能内の DBMS で管理している二次データに反映されるまでの時間的な遅れ（データ更新タイミング）が問題になる。

PUSH 型データ提供機能におけるデータ更新タイミングは、データの性質や利用側の要件、通信方式などを考慮して、データ毎に規定するべきであるが、システム全体の方式や性能に影響する重要な要件である。

通信方式として「3 型：本文とリファレンス型」を採用するケースでは、3.2.1.2 に示す順序性保障手段を講じることによって、遅延が許されるデータについては遅延更新が可能である。

一方、通信方式として「5 型：BPM 機能と連係した本文とリファレンス型」を採用するケースでは、BPM 機能によるフロー制御の都合上、データ更新タイミングとして即時更新が求められる。

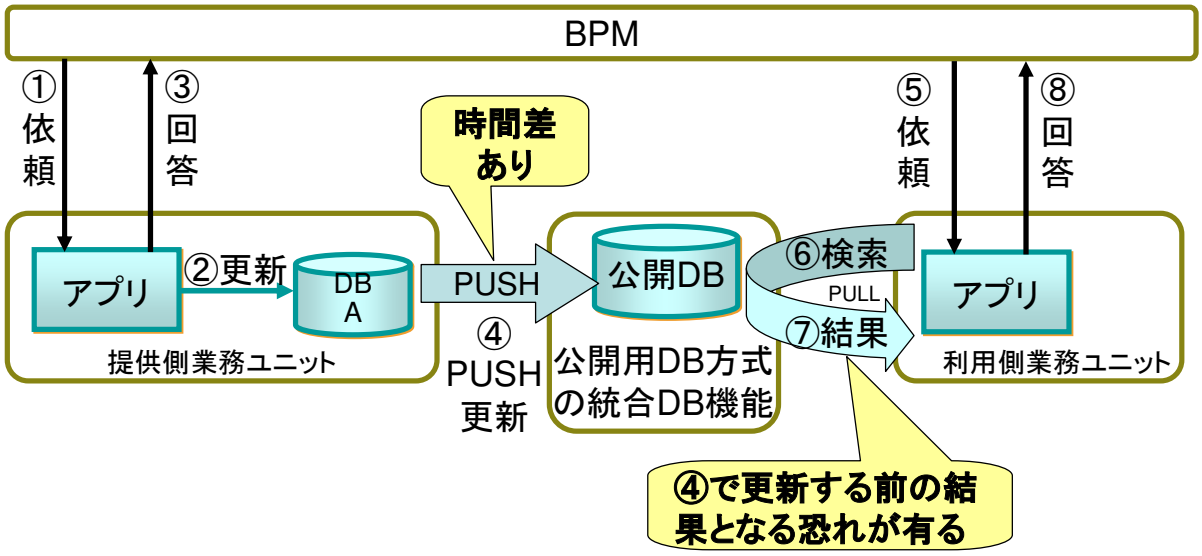


図３．２．１１ PUSH 型更新タイミングの課題

【課題】

公開用 DB 方式の統合 DB 機能における PUSH 型データ提供機能は、方式上、統合 DB 機能で保持している二次データの更新に一定の時間（遅延）が必要であるため、次のような問題が発生する恐れがある。（図 3.2.11）

提供側業務ユニットは BPM 機能からの依頼①に対して処理を実行し、自身のデータを更新②して、該当データのリファレンス情報を BPM に処理結果として応答③する。

提供側業務ユニットは、更新されたデータを公開 DB に PUSH 型で提供④（統合 DB 機能の DBMS を更新）する。

BPM はその応答③を受けて、対象データのリファレンス情報を含んだ次の処理依頼⑤を利用側業務ユニットに送る。

利用側業務ユニットは、BPM から受け取った依頼⑤で指定されたリファレンス情報を参照して、統合 DB 機能に検索依頼⑥を送信することにより、その結果⑦を受け取る。

この時、④により統合 DB 機能に提供側 DB の更新データが反映される前のタイミングで⑥の統合 DB 機能の検索が実行さると、⑦の結果として④で更新される前のデータが返されるため、利用側が古いデータを使った処理となり、処理結果に矛盾を生じる課題がある。

【解決策】

図 3.2.12 で示すように、③と④の順序を入れ換えることで、この課題は解決できる。

即ち、提供側業務ユニットで自身のデータを更新②した直後に、統合 DB 機能への PUSH 型データ提供③を実行して、その完了を確認後に BPM への応答④を行うように構成することにより、③の完了後に⑤の依頼が発行されるので、⑦の結果が最新であることを保障できる。

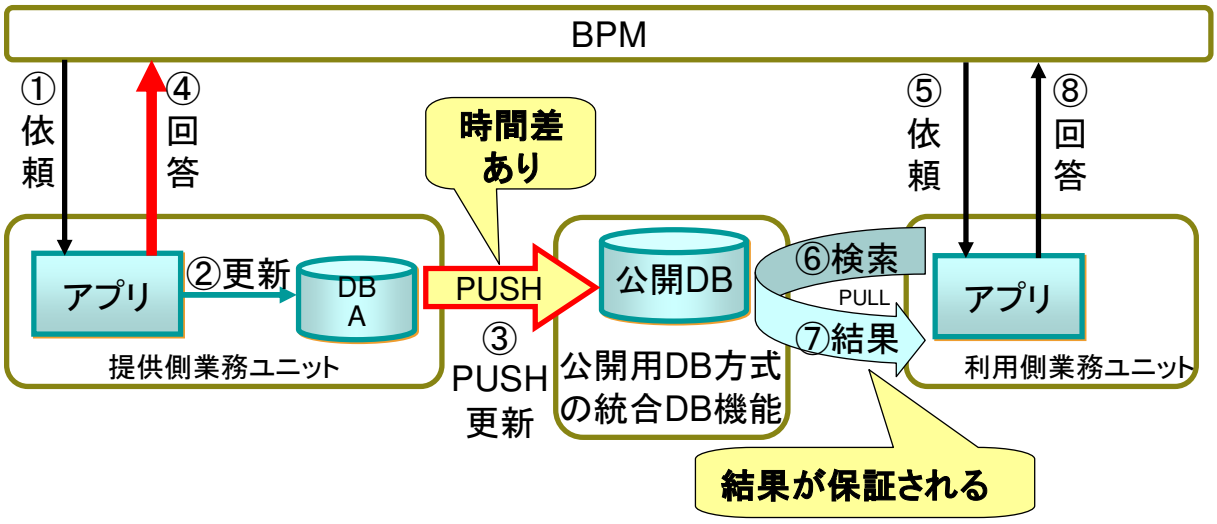


図３．２．１２ PUSH 型更新タイミングの解決策

この場合、③の PUSH 型更新機能の完了が遅れると、提供側業務ユニットの処理完了④も遅れるため、全体の処理遅延につながる。

従って、③の PUSH 型更新タイミングは、可能な限り即時に実行されるべきであり、許容される処理（遅延）時間を要件として規定すべきである。

この許容される処理（遅延）時間は、数秒～十数秒とするのが妥当だが、あまり小さい時間を指定すると実現が困難になるので、考慮すべきである。

許容時間を調達仕様書に明記するか、不整合を起こさない妥当な時間内で処理できること等を調達仕様書に明記すると良い。

より厳密なデータ更新の制御方式として「2 フェーズコミットメント」がある。

「2 フェーズコミットメント」は、データ更新を「準備」と「実行」の 2 段階に分けて、複数の DB を矛盾なく更新する手法であり、提供側業務ユニットに閉じた制御は次の手順になる。（括弧中の数字は図 3.2.12 の数字と一致する）

・提供側業務ユニットは、自己の DB の更新準備（該当データのロックなど）を行う。
・提供側業務ユニットは、統合 DB 機能の更新準備を行う。
・提供側業務ユニットは、自己および統合 DB 機能の更新準備が完了したのを確認する。
・提供側業務ユニットは、自己および統合 DB 機能の更新を実行する（②③）。
・提供側業務ユニットは、自己および統合 DB 機能の更新が成功したことを確認して BPM への応答を行う(④)。このとき、何れかのデータ更新でエラーが発生した場合には、両方のデータを元に戻した上で、リトライするかエラーを返す。

この方式は、統合 DB 機能が「更新準備」機能をサポートしている場合（一般的に SQL インタフェースを採用する場合）に採用することができる。

３．２．２．３　共通インタフェース方式の統合 DB 機能

「共通インタフェース方式」の統合 DB 機能とは、統合 DB 機能内に物理的な DBMS による二次データの保持を必須とせず、提供側業務ユニットから公開される情報を利用側業務ユニットが必要とする情報として仮想的にアクセスするためのプログラムとして構成された統合 DB 機能である。

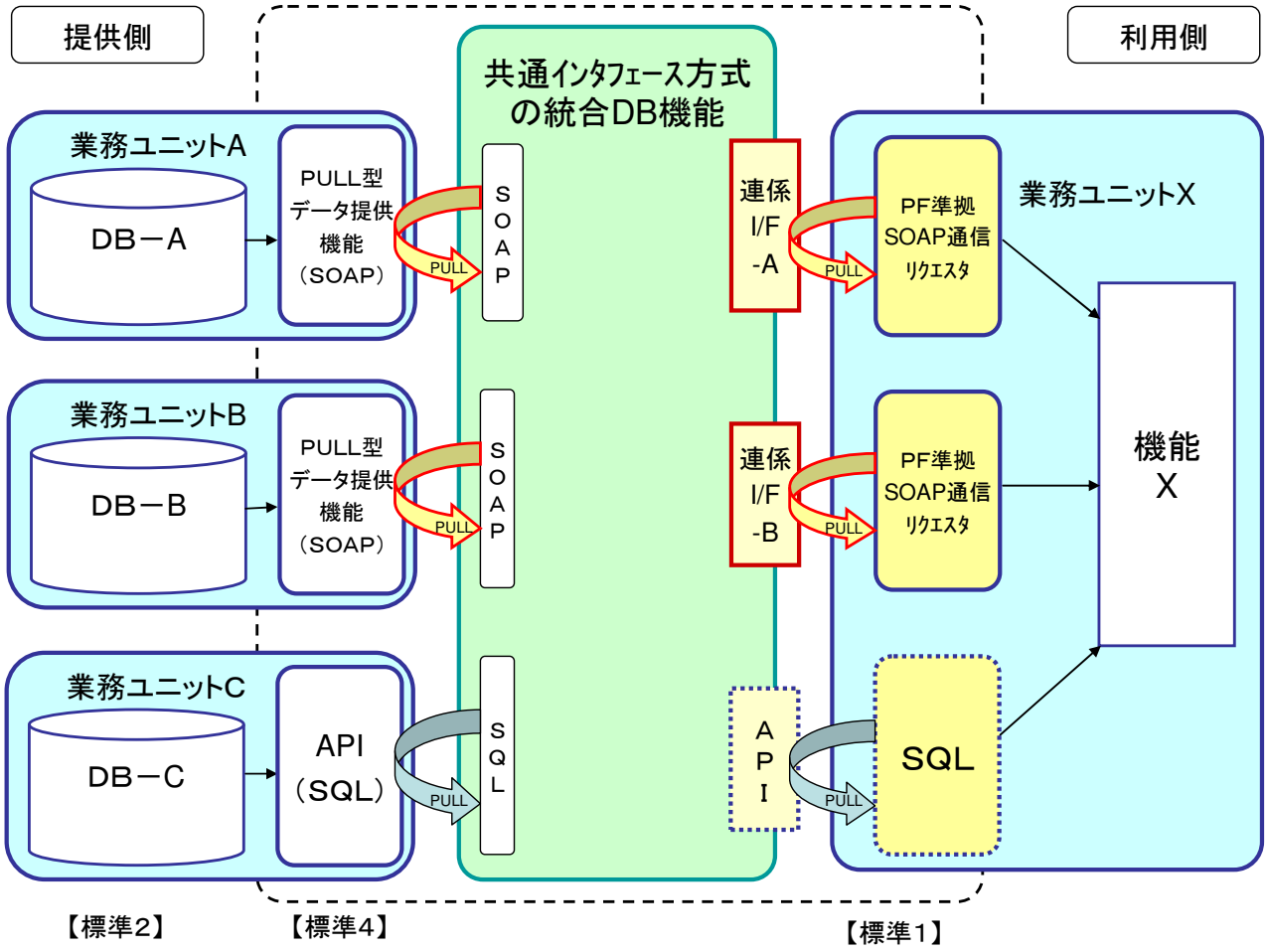


図３．２．１３ 共通インタフェース方式の統合 DB 機能

（１）各ユニットの役割と留意点

ユニット毎に具体的な動作および留意点について解説する。

①提供側業務ユニット

提供側業務ユニットは、PULL 型のデータ提供手段によって、自己が管理する規定のデータを統合 DB 機能に公開する。ここで、公開とは、検索・参照する権限と手段を提供することである。

データの提供手段は規定していないので、SOAP、DBMS のアクセス手段(SQL)など、提供側業務ユニットと統合 DB 機能の実装によって可能な範囲で、任意の方式を採用できるが、差し替え性を考慮して次の点に留意すべきである。

◇SOAP を採用する場合

統合 DB 機能の利用側ＩＦを「自治体業務アプリケーションユニット標準仕様」で標準化する SOAP

によるインタフェースに準拠させることにより、提供側業務ユニットが標準でサポートしている SOAP によるデータ提供機能をそのまま使うことができる。
地域情報 PF 標準以外の SOAP 仕様を採用することは推奨しない。

◇SQL を採用する場合
SQL は提供側業務ユニットに採用されている DBMS でサポートされるインタフェースを使用して実行され、SQL 自身も DBMS に依存した仕様になるので、差し替え性を考慮すると、そのインタフェースは、ODBC、JDBC などの標準化されたものを採用し、SQL も DBMS に依存しない範囲で使用するべきである。（3.2.7.3(3)参照）

②利用側業務ユニット
利用側業務ユニットは、提供側業務ユニットとは無関係に、利用 I/F を介して統合 DB 機能に検索要求を送信し、結果データを受け取る。即ち、利用側業務ユニットは、統合 DB 機能の有無および統合 DB 機能の方式に関わらず、利用 I/F によるデータ取得が実現できる。

統合 DB 機能の利用 I/F として SQL を採用する場合は、統合 DB 機能独自のインタフェースとなるケースが多い。この場合、インタフェースおよびデータモデルは標準化していないので、利用側業務ユニットの対応に留意する必要がある。（ODBC、JDBC などを推奨）

③統合 DB 機能
統合 DB 機能は利用側業務ユニットからの要求に応じて、提供側業務ユニットにアクセスを行い、必要なデータを収集して、利用側業務ユニットが求めるデータ形式のデータとして、利用 I/F を介して利用側業務ユニットに検索結果を返す。

（２）必要となる仕様
共通インタフェース方式の統合 DB 機能を採用する場合に必要となる仕様として次のものがある。

【PULL 型データ提供機能】　上記の①の機能が該当する。
◇提供側業務ユニットは、採用する統合 DB 機能に適合した PULL 型データ提供機能を有すること。
◇統合 DB 機能は、対象とする各提供側業務ユニットから提供される PULL 型データ提供機能により、提供側業務ユニットにアクセスを行い、利用側業務ユニットが必要なデータを取得すること。

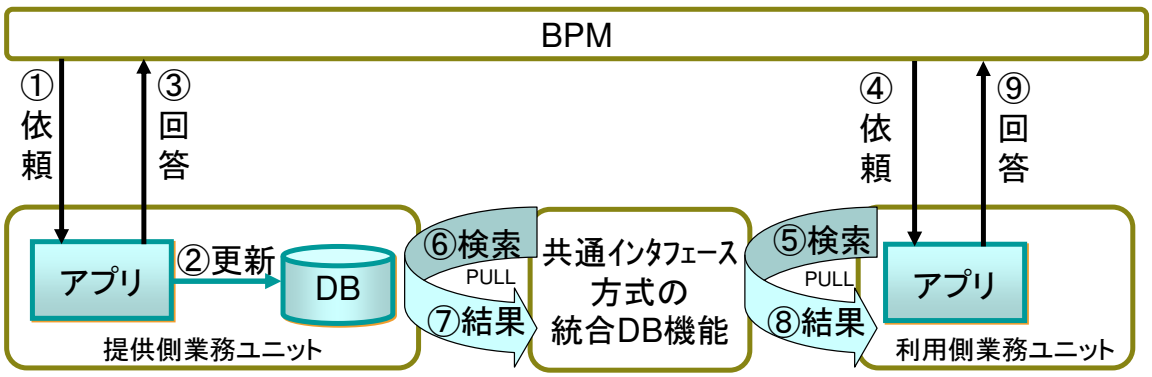


図３．２．１４ 共通インタフェース方式の統合 DB 機能におけるデータ更新タイミング

ここで、提供側業務ユニットと統合 DB 機能の提供 I/F が適合していることが重要であり、システム要件として具体的に指定すべきである。

本方式の統合 DB 機能は、利用側から要求された最新のデータを提供側からオンデマンドで取得するので、リアルタイム性（情報の鮮度）が高いという特徴がある。（図 3.2.14）
従って、公開用 DB 方式の統合 DB 機能で課題とされたデータ鮮度に関するタイミング問題は存在しないので、特別な制御は不要である。

本方式の欠点として、統合 DB 機能の検索アクセスが提供側業務ユニットの性能に影響する可能性がある点や、提供側業務ユニットの停止時に統合 DB 機能の利用も止まる点が指摘されている。これらの欠点は提供側業務ユニットと統合 DB 機能の間にレプリカを置くことにより回避できる。

コラム３．２．３　「DBMS のレプリケーション機能」

DBMS のレプリケーション機能とは、マスタ表と同じ表（レプリカ）を別に作成して、その間の同期を制御する機能である。

初期状態としてマスタ表のコピーをレプリカとして作成しておく。
マスタ表が更新①されると、更新ログ（どのような更新が行われたのかを記録したデータ）が抽出②されてレプリカ側に転送③される。
レプリカ側では受け取った更新ログをレプリカ（表）に適用④する。
通常、この制御を更新トランザクション（アプリケーションからの DB の処理単位）毎に行うことにより、レプリカをマスタ表とほぼ同じ状態に保つことができる。

特徴として、レプリカとマスタ表は 1:1 となり、双方にキー項目が必要であることが多い。

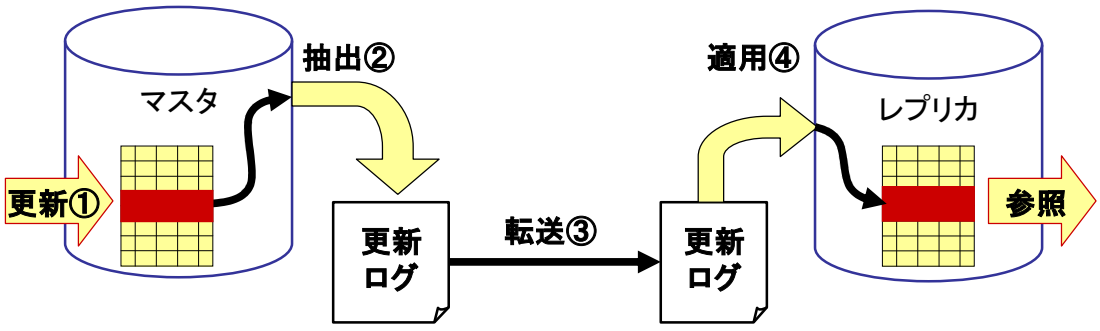


図３．２．１５ DBMS のレプリケーション機能

３．２．２．４　統合 DB 機能の方式とインタフェースのまとめ

以上で解説した統合 DB 機能の方式について表 3.2.1 に整理する。また、統合 DB 機能の方式毎の業務ユニットとのインタフェースについて、図 3.2.16 で整理する。

表３．２．１ 統合 DB 機能の方式比較

No.	項目	公開用 DB 方式	共通インタフェース方式
1	統合 DB 機能 ・二次データの保持 ・データ容量	◇提供された二次データを保持する。 ・二次データの総量を管理できること。	◇二次データの保持は不要 ・一度に大量のデータを利用する際は メモリ容量に注意
	提供機能の動作	◇統合 DB に保持しているデータを検索して返す。	◇利用用側からの要求に応じて 必要なデータを収集
	利用 I/F	□SOAP(PF 標準のプロバイダ機能)　必須 □SQL(統合 DB の DBMS に依存)	□SOAP（PF 標準のプロバイダ機能）　必須 □SQL
	提供 I/F	□SOAP(ﾌﾟﾛﾊﾞｲﾀﾞ機能) □SQL(統合 DB の DBMS に依存)	□SOAP（PF 標準のﾘｸｴｽﾀ機能） □SQL(提供側 DBMS の RDB に対応)
	利用データの鮮度	◇提供されたデータの鮮度に依存する。	◇提供側業務ユニットの最新データを利用可能
	順序制御	◇提供側の順序制御が必要	◇順序制御は不要
	可用性	◇二次データのバックアップ、復旧対策が必要 ◇提供元の停止時も運転可能	◇二次データを持たないので並列運転に適する ◇提供元の停止時の運転不可 → レプリカで回避
	主な実現技術	◇DBMS	◇EII、フェデレーション
2	提供側業務ユニット	◇PUSH 型のデータ提供機能で 統合 DB を更新 □提供 I/F = SOAP →統合 DB □提供 I/F = SQL	◇PULL 型のデータ提供機能で 統合 DB にデータ公開 □提供 I/F = SOAP(PF 標準仕様) □提供 I/F = SQL
	運用負荷	◇提供機能の実行負荷が有る。	◇統合 DB からの検索負荷が有る → レプリカで回避
3	利用側業務ユニット	□SOAP（PF 標準仕様） 推奨 □SQL(統合 DB の DBMS に依存)	□SOAP（PF 標準仕様） 推奨 □SQL

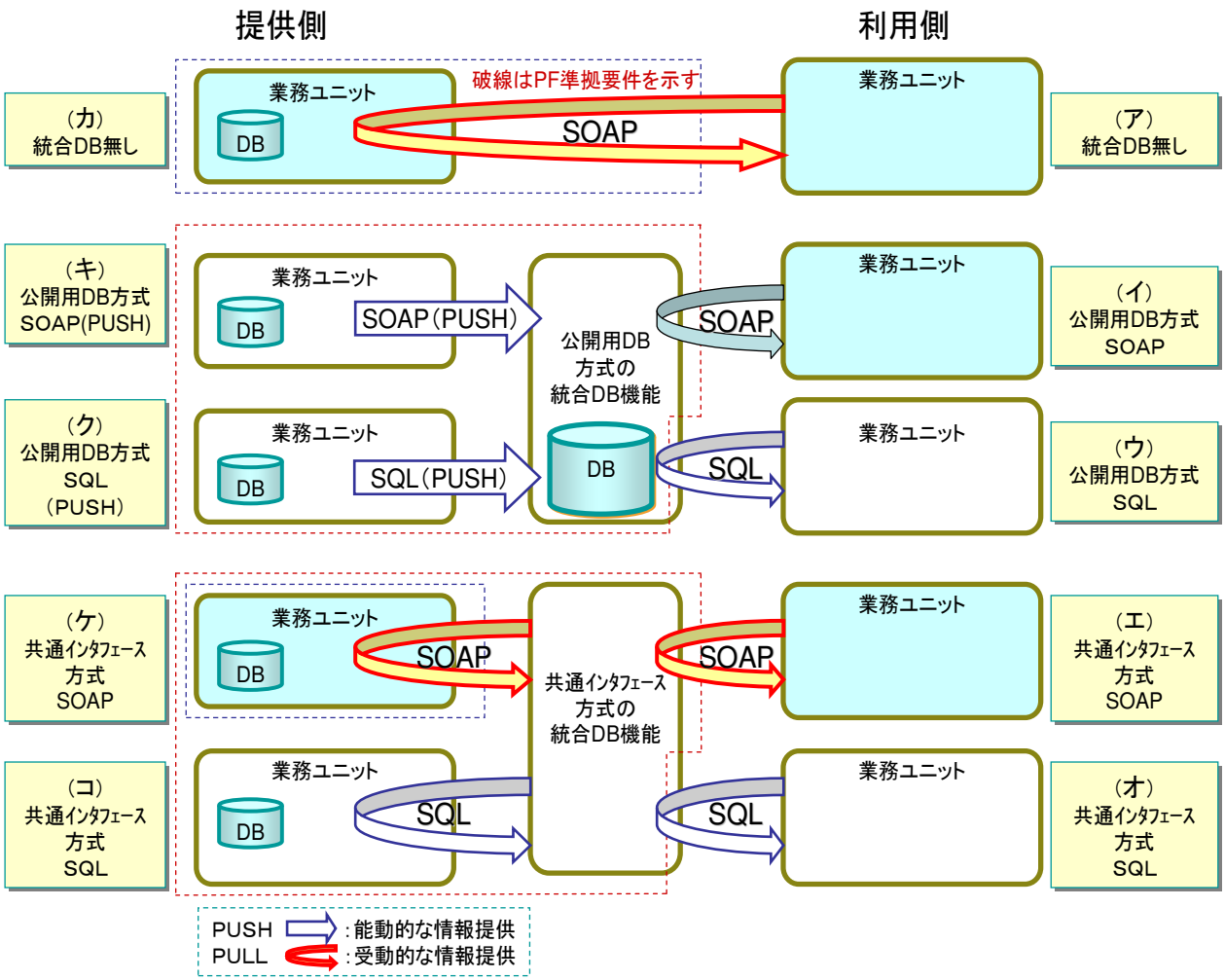


図３．２．１６ 統合 DB 機能の方式とインタフェースのまとめ

図３．２．１６は、中央に業務ユニット間のデータ交換方式（3 種）をあらわし、右側に利用側業務ユニットと統合 DB 機能間のインタフェース方式（ア～オ）、左側に提供側業務ユニットと統合 DB 機能間のインタフェース方式（カ～コ）を示している。

標準で規定しているのは、利用側業務ユニットから利用される SOAP インタフェース（ア、イ、エ）であり、他は最適な方式を選択することができる。

利用 I/F として SQL（ウ）（オ）も採用可能だが、差し替え性は自己責任となる。
提供側業務ユニットは上記の SOAP インタフェースをサポートしていることが必須であり、統合 DB 機能など、アダプタと組み合わせた実現も可能である。従って、破線で示している機能により地域情報 PF の準拠性を満足することができる。

提供側業務ユニットから見ると、統合 DB 機能無し（カ）と、共通インタフェース方式の SOAP（ケ）は同じであり、地域情報 PF 標準に準拠した業務ユニットはそのままで適用できる。

（キ）（ク）（コ）は統合 DB 機能と提供側業務ユニットの方式に依存する部分となる。
既存システム（地域情報 PF 未対応）の業務ユニットを提供側とする用途には、データの公開のみで済む（コ）が最も導入が容易である。

標準では、実情に合った方式の選択が可能なように必要最小限の規定をしているので、方式の選定に際しては、各方式の特徴を理解した上でシステム全体として整合性のある最適な構成になるように留意すべきである。

３．２．３　統合 DB 機能の実装モデル

「公開用 DB 方式」と「共通インタフェース方式」の 2 つの方式について、実際に統合 DB 機能を実装する際の参考となる事項を解説する。

３．２．３．１　公開用 DB 方式の統合 DB 機能の実装

公開用 DB 方式の統合 DB 機能について実装パターンを図３．２．１７に示す。

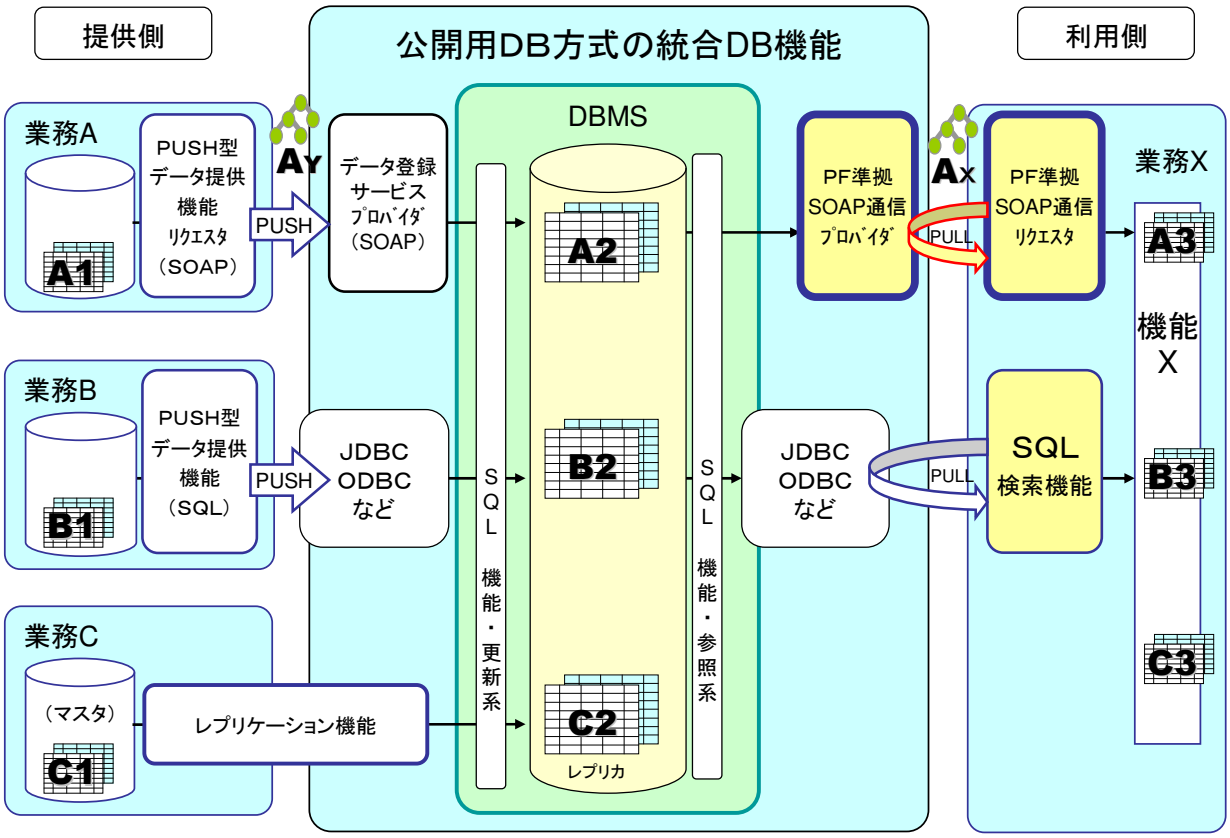


図３．２．１７ 公開用 DB 方式の統合 DB 機能の実装パターン

汎用的な事項は 3.2.2.2 で解説済みなので、ここではデータモデルおよび具体的な実装における留意点を中心に解説する。

（１）統合 DB 機能

統合 DB 機能は物理的な RDBMS（リレーショナルデータベース管理システム）を使用して実装するのが一般的である。XML データベースによる実装も可能であるが、その場合の提供 I/F、利用 I/F の実装は SOAP であることが多い。

DBMS で保管する二次データは、表として管理されるが、地域情報 PF で標準化しているのは SOAP(XML)即ちツリー構造なので、二次データのデータ構造は標準化したツリーのデータ構造（図 3.2.17 の Ax）を表現できる表構造（図 3.2.17 の A2）である必要があり、一度決定すると変更の影響が大きいので、慎重に決めるべきである。一般的に１つのツリー構造データを表現するには、複数の表が必要になる。（詳細は 3.2.5 参照）

（２）利用 I/F と利用側業務ユニット

利用側の SOAP インタフェースは、統合 DB 機能で実装することが必須である。具体的には、統合 DB 機能側の機能として地域情報 PF 準拠の SOAP 通信を提供するプロバイダ機能を実装する。このプロバイダ機能は、利用側からの依頼（リクエスト）に応じて、統合 DB 機能で管理している該当表(A2)の検索を行い、検索結果(A2)をツリー構造(Ax)に変換して利用側に応答（レスポンス）する。

また、利用側業務ユニットは、SOAP インタフェースによる統合 DB 機能の利用を行う場合、統合 DB 機能がサポートする SOAP 通信を呼び出す機能（リクエスタ）を実装することになる。このリクエスタ機能では、ツリー構造(Ax)として得たデータを業務ユニット X 内のデータモデル(A3)に変換して利用するのが一般的である。

利用側のインタフェースとして SQL を採用する場合には、統合 DB 機能で採用している RDBMS でサポートされる SQL インタフェース（JDBC や ODBC など）を利用側業務ユニットが直接使用して SQL による統合 DB 機能の検索を行うのが一般的な実装となる。ツリー構造を介さないので、データ構造の変換が不要であり、統合 DB 機能側に特別な機能を実装する必要がないので、統合 DB 機能が簡素化でき、高速なインタフェースを実現することができる利点がある。一方、利用側で統合 DB 機能のデータ構造を意識して、統合 DB 機能で採用されている RDBMS に依存する SQL インタフェースを使用する必要があるので、差し替え性が阻害される恐れがある点が欠点である。

（３）提供 I/F と提供側業務ユニット

提供 I/F として、SOAP を採用する場合は、統合 DB 機能側の提供 I/F として統合 DB 機能に対するデータ登録（更新）を実現する SOAP 通信によるプロバイダ機能（データ登録サービス）を実装する。このプロバイダ機能は、提供側からの依頼（リクエスト）に応じて、依頼内容(Ay)を表構造(A2)に変換し、統合 DB 機能で管理している該当表(A2)に対するデータ追加、更新、削除などを行い、その結果を提供側に応答（レスポンス）する。

また、提供側業務ユニットは、統合 DB 機能がサポートするデータ登録サービスを呼び出す機能（リクエスタ）を実装する。このリクエスタ機能では、提供側で管理している該当表(A1)を統合 DB 機能が規定するツリー構造(Ay)に変換して統合 DB 機能のデータ登録サービスに依頼することになる。

提供 I/F として SQL を採用する場合は、統合 DB 機能で採用している RDBMS によりサポートされる SQL インタフェース（JDBC や ODBC など）を提供側業務ユニットが直接使用して SQL による統合 DB 機能へのデータ登録・更新・削除を行うのが一般的な実装となる。ツリー構造を介さないので、表とツリー間のデータ構造変換が不要であり、統合 DB 機能側に特別な機能を実装する必要がないので、統合 DB 機能が簡素化でき、高速なインタフェースを実現することができる利点がある。

一方、提供側と統合 DB 機能のデータモデルは異なるので、提供側は統合 DB 機能のデータ構造を意識して、統合 DB 機能で採用されている RDBMS に依存する SQL インタフェースを使用する必要があるので、差し替え性が阻害される恐れがある点が欠点である。

提供 I/F の選択肢として DBMS がサポートするレプリケーションを使用することも考えられる。この方法は、提供側業務ユニットが管理する DBMS と、統合 DB 機能の DBMS をレプリケーション機能で繋ぐことにより、提供側業務ユニットは、自己の DBMS の更新に専念することができるので、実装が容易になる。（コラム 3.2.3 参照）

この場合､提供側と統合 DB 機能のデータ構造が異なるケースが多いので､レプリケーション機能がデータ構造変換を行う必要があり、一般的なレプリケーション（同じ構造の表を複製する）と異なるので注意する必要がある。また、統合 DB 機能および提供側業務ユニットが同じ DBMS を採用するなど、レプリケーション機能がサポートできる組み合わせであることを確認する必要がある。

（４）性能への配慮

統合 DB 機能を物理的な DBMS で実現する性質上、統合 DB 機能に対する提供側および利用側からのアクセスが集中して、性能が劣化する恐れがあるので、性能への配慮が特に重要である。

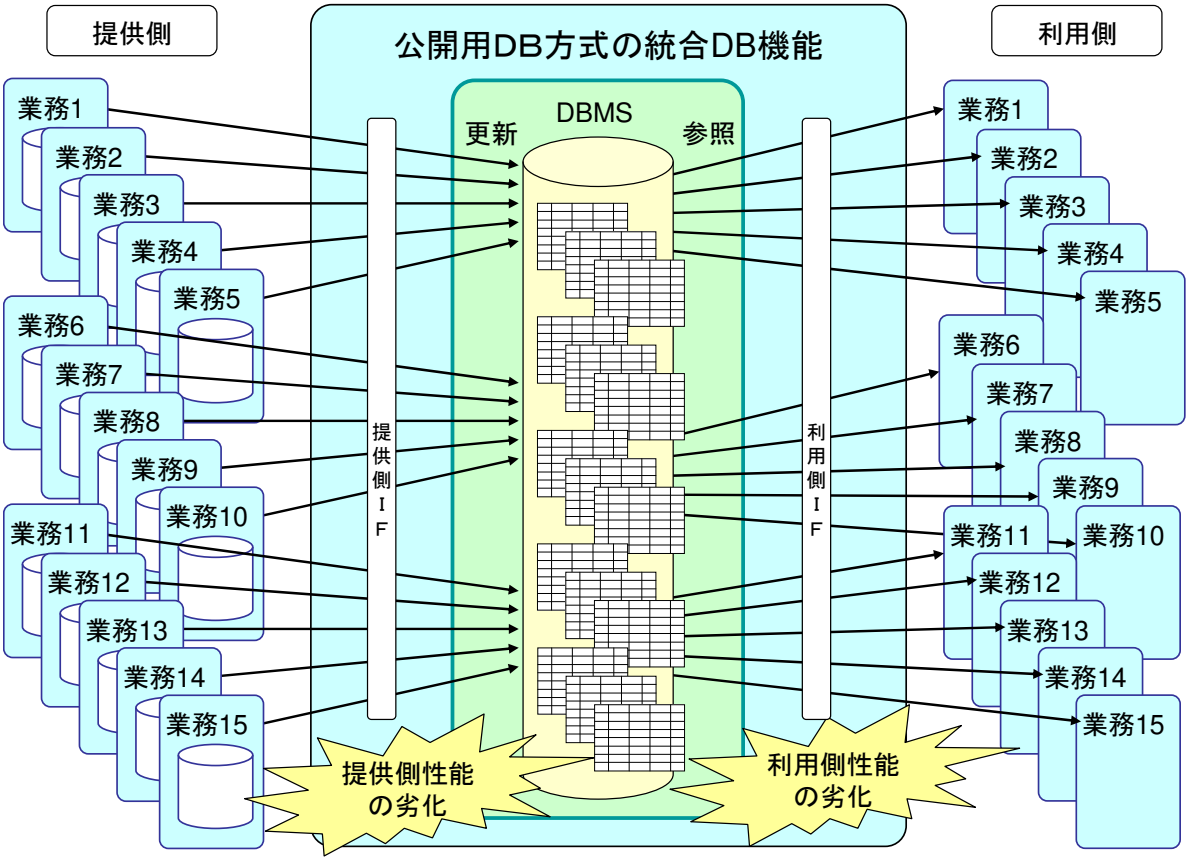


図３．２．１８ 公開用 DB 方式の統合 DB 機能の性能劣化

提供側の性能が劣化すると､統合 DB 機能への登録を待って処理完了とする提供側業務の遅延に直結し、利用側の性能が劣化すると、統合 DB 機能経由で所得したデータに基づく利用側の処理が遅延する。

一般的に、1 つの業務ユニットが提供側・利用側の両方の役割で統合 DB 機能と連係しており、提供側および利用側業務ユニットは、BPM などにより密接に関連しているので、統合 DB 機能の性能劣化は、業務システム全体の性能劣化に直結することになる。

回避策として、複数の統合 DB 機能に分割することが考えられるが、この場合、同じ二次データを複数の統合 DB 機能で管理すると、提供側は複数の統合 DB 機能に対する提供処理を行う必要があり、提供側業務ユニットの遅延に繋がるので留意すべきである。この課題は管理する二次データを複数の統合 DB 機能に分割することで回避できるが、この場合利用側が複数の統合 DB 機能を使い分ける必要が生じ、管理面で複雑になる欠点がある。

（５）統合 DB 機能のデータベース容量

統合 DB 機能で交換対象の二次データの実体を保持して管理する必要があるので､統合 DB 機能内の DBMS が管理するデータベース容量に留意する必要がある。

具体的には図 3.2.18 からも解る通り、全ての交換対象データ（二次データ）の合計容量が統合 DB 機能の DBMS で管理されるように構成する必要がある。

DBMS などの制約により、必要なデータ管理領域が確保できない場合は、統合 DB 機能を複数に分割するなど、必要な対策を講じることになるが、前項の性能など総合的にバランスのよい構成とするべきである。

コラム３．２．４　「DBMS のビュー機能」

DBMS のビュー機能は、DBMS 内で物理的なデータを管理している表（実表）とは別に、実表の見え方をビュー表として定義することにより、仮想的にビュー表として利用する機能である。利用側からは通常の表と同様にビュー表に対する検索が可能であるが、ビュー表は内部的に関係演算の結果表として定義され、ビュー表に対する検索はビューを構成する実表に対する関係演算として処理されるのが一般的である。

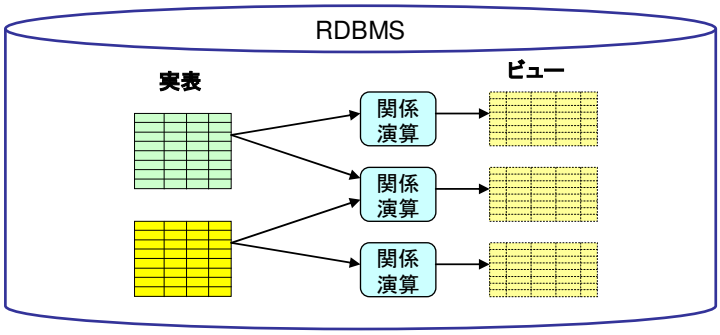


図３．２．１９ DBMS のビュー機能

３．２．３．２　共通インタフェース方式の統合 DB 機能の実装

共通インタフェース方式の統合 DB 機能は、統合 DB 機能に物理的な DBMS を必須とせず、提供側業務ユニットからのデータ提供機能が PULL 型（データを公開する機能）である特徴があり、具体的な実装として次の 3 パターンが考えられる。これらは実装が異なるだけであり、提供側または利用側から見ると全て同じ方式である。

（１）　データ統合機能を持つ DBMS による実装
（２）　データ統合ミドルウェアによる実装
（３）　専用アプリケーションによる実装

各実装パターンの仕組みと特徴は次の通り。

（１）データ統合機能を持つ DBMS による実装
図 3.2.20 は、データ統合機能を持つ RDBMS による実装パターンである。

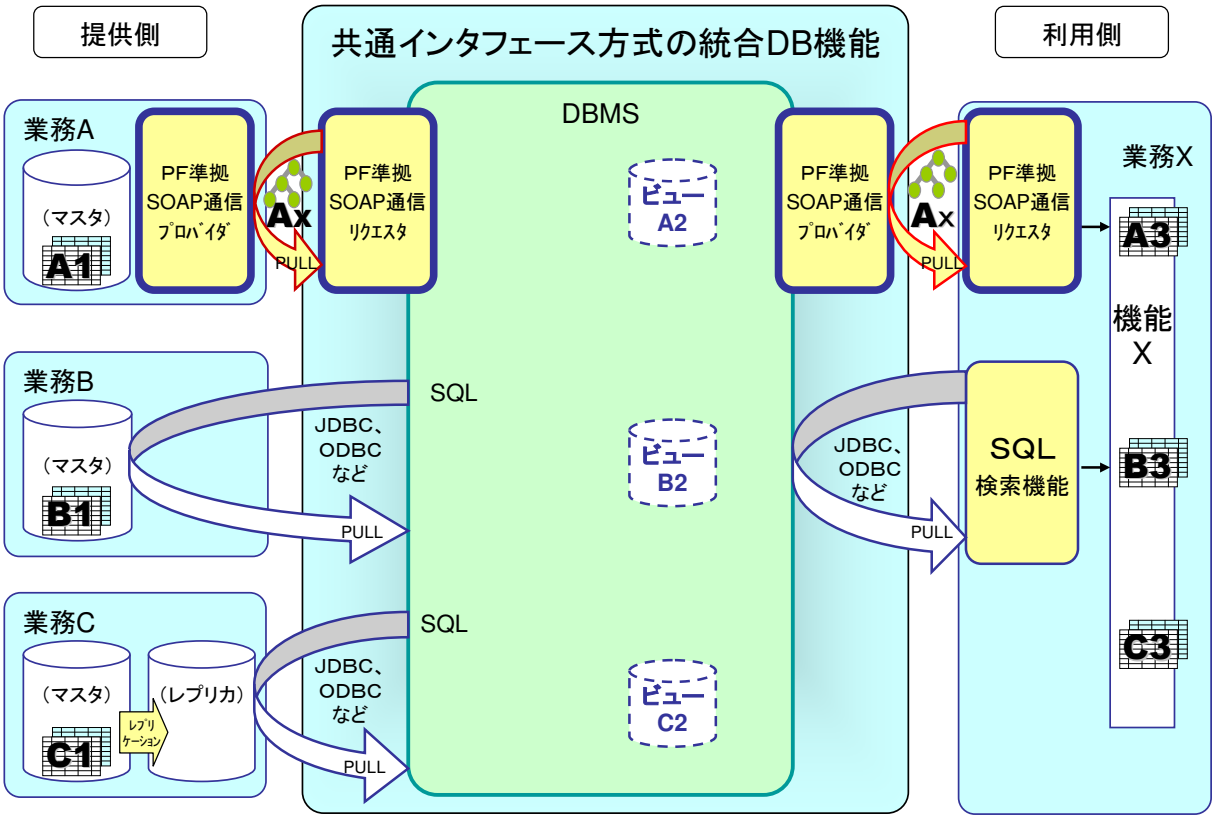


図３．２．２０ 共通インタフェース方式の統合 DB　実装パターン(1)

DBMS 製品の中には、データ統合機能として外部の DBMS に存在するデータ（外部表）についても内部の表と同様に動的なビューを提供できるものがある。
統合 DB 機能として二次データを保持せず､利用側からのリクエストに従って提供側にアクセスを行い、

PULL 型のデータ検索によりデータを獲得するので、共通インタフェース方式に分類される。
統合 DB 機能に DBMS を採用しているので、利用 I/F の実現方法は公開用 DB 方式と同様になる。
SQL による提供 I/F、利用 I/F は DBMS が標準でサポートしているケースが多い。また、SOAP による提供 I/F、利用 I/F の実装は、地域情報 PF 標準の SOAP インタフェースをアプリケーションで実装することが多い。

DBMS から直接アクセスする外部表としては、提供側業務ユニットが持つマスタ表を直接設定する方法と、マスタ表のレプリカを設定する方法がある。このレプリカは統合 DB 機能の内部に作成すことも、外部に作成することもできる。

特徴はデータ統合機能を持つ DBMS が限られている点であるが、この種の DBMS は多くの種類の DBMS を入力としてサポートするものが多いので、提供側業務ユニットの DBMS について選択肢が広いが、DBMS の組み合わせに対する検証は必要である。

（２） データ統合ミドルウェアによる実装
図３．２．２１は、データ統合ミドルウェアによる実装パターンである。

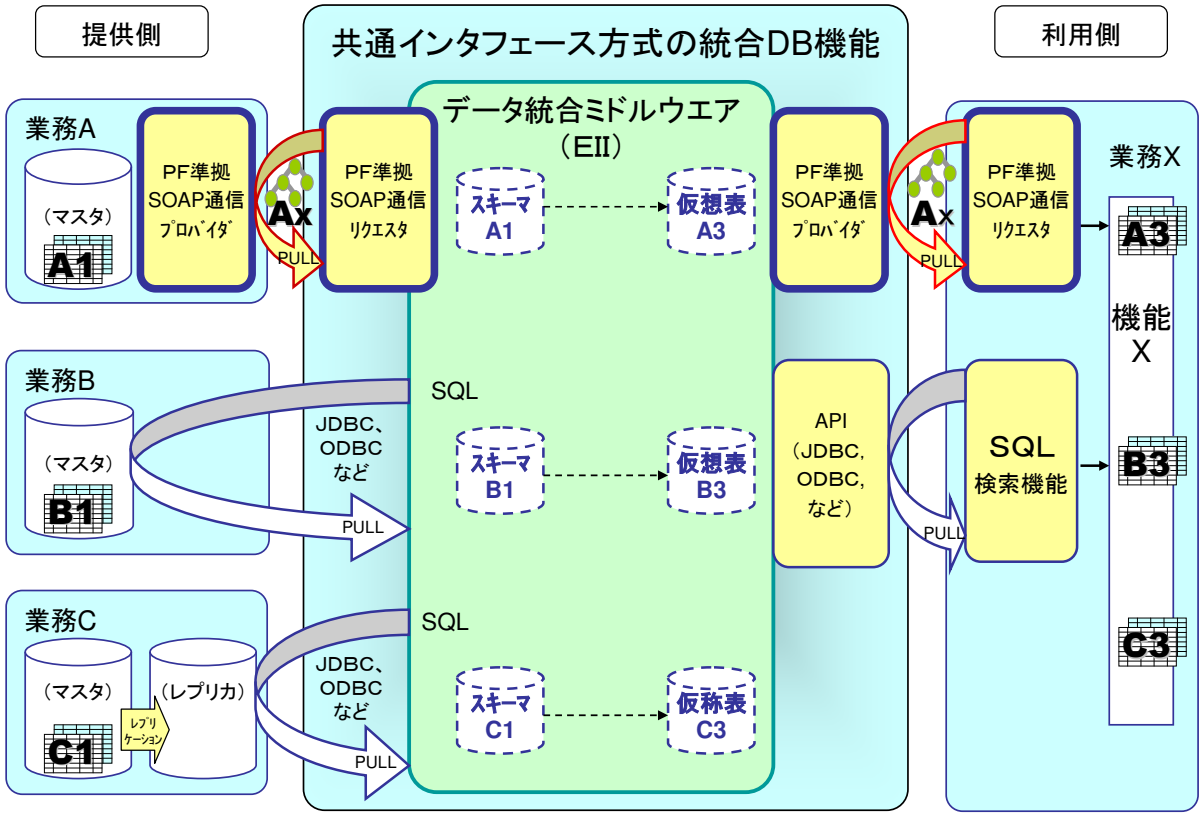


図３．２．２１ 共通インタフェース方式の統合 DB 実装パターン(2)

統合 DB 機能に DBMS を使用せず、データ統合ミドルウェアに提供側業務ユニットのスキーマと、利用側に必要なスキーマを定義して、ミドルウェア（またはそのアプリケーション）の機能として利用 I/F を実現する。具体的には、利用側からの検索要求を統合 DB 機能が受け取ると、必要な提供側業務ユニッ

トに対する検索要求に分解して提供側業務ユニットの検索を行い、その結果を統合して利用側に応答する処理が動的に行われる。
SQL による提供 I/F はミドルウェアが標準でサポートするケースが多く、SQL による利用 I/F はミドルウェアの API として提供されるケースがある。また、SOAP による提供 I/F、利用 I/F の実装は、地域情報 PF 標準の SOAP インタフェースをアプリケーションで実装することが多い。

統合 DB 機能に DBMS を持たず、検索処理が動的に行われ、検索結果は利用側が受け取った後は捨てられるので、統合 DB 機能側のデータ容量に対する配慮は不要である。
また、全てメモリ上で処理される実装では、複数のデータ統合サーバ（統合 DB 機能）を並列運用することが容易であるため、性能面で有利である一方、検索結果を処理できるだけのメモリ量に留意する必要がある。
提供側業務ユニットへの影響を抑える目的で提供側のレプリカを使用する場合は、提供側にレプリカ用の DBMS が必要になる。

本方式は、オンデマンド、リアルタイム、柔軟な統合パターン、提供側の地域情報 PF 対応が容易などの共通インタフェース DB の長所を最も引き出せる実装である。
データ統合ミドルウェアとしては、EII(Enterprise Information Integration)という分野のソフトウェアが数社から提供が始まっており、フェデレーション機能と呼ばれることもある。

（３） 専用アプリケーションによる実装

図３．２．２２は、専用のアプリケーションによる実装パターンである。

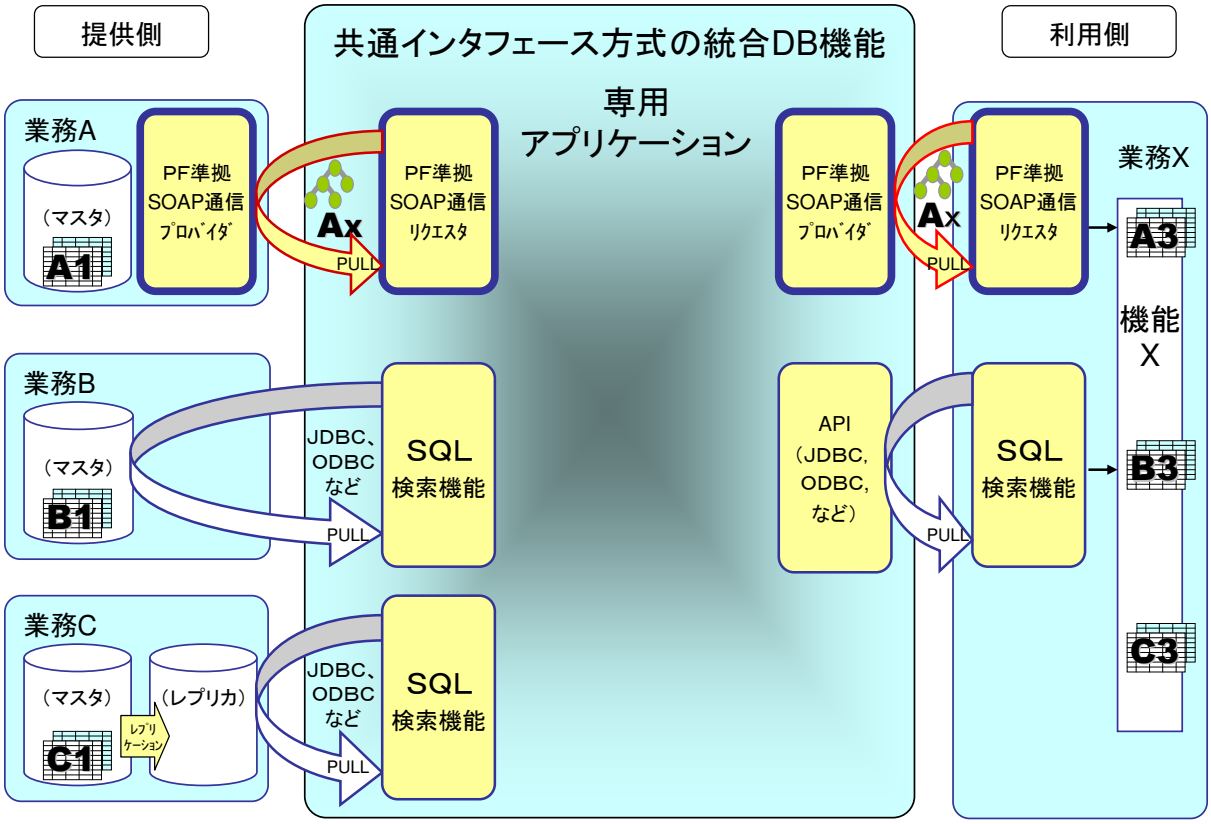


図３．２．２２ 共通インタフェース方式の統合 DB 実装パターン(3)

利用 I/F を含めて、地域情報 PF の統合 DB 機能専用のアプリケーションとして実装するパターンであり、特徴や留意点はパターン(2)と同様になる。

現時点では新規開発になるが、地域情報 PF の普及に伴って、統合 DB 機能専用のアプリケーションが開発されることにより現実的になるパターンである。

３．２．４　統合 DB 機能の導入

統合 DB 機能の導入に際して考慮すべき事項として、統合 DB 機能側の設定、既存の業務ユニットに対する統合 DB 機能の導入および、統合 DB 機能の移行（入れ替え）について解説する。

３．２．４．１　統合 DB 機能側の設定

最初に統合 DB 機能側に必要な設定及び留意事項について表３．２．２にまとめる。

これらは、新規に統合 DB 機能を導入するケースは勿論、既存の業務ユニットに対する統合 DB 機能の導入、統合 DB 機能の移行（入れ替え）、業務ユニットの入れ替えに際して、導入する統合 DB 機能側または、変更する業務ユニットとの統合 DB 側 I/F に必要な共通事項である。

表３．２．２ 統合 DB 機能側の設定

	公開用DB方式		共通ｲﾝﾀﾌｪｰｽ方式	
	SOAP	SQL	SOAP	SQL
	□対象業務ユニットをサポートしていること □特に統合DB機能のデータ構造が十分か確認→不足項目はDBMSの表を拡張する。		□対象業務ユニットをサポートしていること	
提供IF	□PUSH型のデータ登録サービスプロバイダ機能を設定	□JDBC,ODBCなどによるPUSH型データ提供機能の設定（DBMSの更新権を提供側業務ユニットに公開）	□PF標準のSOAP通信リクエスタを設定（提供側業務ユニットの参照権を設定）	□提供側DBMSをサポートしていること □提供側のスキーマなどを設定 □提供側業務ユニットの参照権を設定
利用IF	□PF標準のSOAP通信プロバイダ機能を設定	□JDBC,ODBCなどによるPULL型データ検索機能の設定（DBMSの参照権を利用側業務ユニットに公開）	□PF標準のSOAP通信プロバイダ機能を設定	□利用側に必要なスキーマなどを設定 □JDBC,ODBCなどによるPULL型データ検索機能の設定（統合DB機能の参照権を利用側業務ユニットに公開）

例えば、共通インタフェース方式の統合 DB 機能を導入する際に、提供 I/F として SQL を採用するケースでは、統合 DB 機能が対象業務ユニット及び、提供側業務ユニットの DBMS をサポートしている事を確認する。次に、提供側業務ユニットから提供されるデータのスキーマ及び提供側業務ユニットへのアクセス権を統合 DB 機能に設定する必要がある。

なお、これらの事項は代表的な統合 DB の実装についてまとめており、具体的には統合 DB 機能の実装毎に定められた設定を行うことになる。

３．２．４．２　既存の業務ユニットに対する統合 DB 機能の導入

既存の業務ユニットが統合 DB 機能に対応するとき（統合 DB 機能の導入時）に必要となる事項を表３．２．３に示す。

表３．２．３ 既存業務ユニットの対応

	公開用DB方式	共通ｲﾝﾀﾌｪｰｽ方式
提供側	PUSH型のデータ提供機能 （統合DB機能のデータ更新） ①提供I/F=SOAP ②提供I/F=SQL	PULL型のデータ提供機能 （統合DB機能へデータ公開） ③提供I/F=SOAP ④提供I/F=SQL
利用側	統合DB機能の利用 ⑤利用I/F=SOAP ⑥利用I/F=SQL	

（１）提供側業務ユニットの対応

提供側業務ユニットの対応は、統合 DB 機能の方式により必要な対応が異なる。

公開用 DB 方式の統合 DB 機能を採用する場合は、PUSH 型で統合 DB 機能にデータを提供する機能（統合 DB 機能のデータを更新する機能）を提供側業務ユニットに実装する必要がある。このとき使用する統合 DB 機能の提供 I/F は、統合 DB 機能がサポートする①SOAP または②SQL となり、その仕様は一般的に統合 DB 機能の実装に依存する。

共通インタフェース方式の統合 DB 機能を採用する場合は、PULL 型で統合 DB 機能にデータを公開する必要がある。提供側業務ユニットが地域情報 PF 対応のものについては、提供 I/F として③SOAP を採用することにより、提供側業務ユニットがサポートしている標準の SOAP インタフェースをそのまま使用できる。

一方、提供側業務ユニットが地域情報 PF に対応していないものについては、提供 I/F として④SQL を採用することにより、提供側業務ユニットで採用されている DBMS へのアクセス手段を統合 DB 機能に公開することにより対応できるので、導入が容易になる。

（２）利用側業務ユニットの対応

利用側業務ユニットは、他の業務ユニットのデータを活用する際に統合 DB 機能を介して必要なデータを取得するようにすべきである。この際に使用する統合 DB 機能の利用 I/F は、地域情報 PF で標準化す

る⑤SOAP によるインタフェースを採用することにより統合 DB 機能の実装に関わらず、高い差し替え性を確保できる。

また、利用 I/F として統合 DB 機能がサポートする⑥SQL インタフェースを使用することも可能である。この場合、インタフェースは統合 DB 機能に依存するが、性能などを追求することができる。

（３）段階的な導入

統合 DB 機能により、既存の業務システムへの地域情報 PF 導入を段階的に進めることができる。

①統合 DB を先行的に導入する

②多くの業務ユニットにデータを提供する業務ユニットから順に、統合 DB 機能の提供側業務ユニットに対応し、管理しているデータを提供する環境を整える。

③統合 DB 機能に対応したデータから順次、利用側業務ユニットからの統合 DB 利用を開始する。

３．２．４．３　統合 DB 機能の移行（差し替え）

既に導入されている統合 DB 機能を差し替える際に必要となる業務ユニットの対応について、提供側業務ユニット、利用側業務ユニットに分けて解説する。なお、ここでは既存の業務ユニットは地域情報 PF 対応済みであること前提に説明している。

（１）　提供側業務ユニットの対応

旧統合 DB 機能を新統合 DB 機能に差し替える際に必要となる提供側業務ユニットの対応について表３．２．４で整理する。

表３．２．４ 統合 DB 機能の差し替えにおける提供側業務ユニットの対応

新 / 旧		公開用DB方式		共通ｲﾝﾀﾌｪｰｽ方式	
		⑤SOAP	⑥SQL	⑦SOAP	⑧SQL
公開用DB方式	①SOAP	SOAPのPUSH型提供機能①を新しい統合DB機能のSOAP⑤に変更	SOAPのPUSH型提供機能①を新しい統合DB機能のSQL⑥に変更	既存の提供機能①を停止	既存の提供機能①を停止して、提供側DBMSのSQLインタフェース⑧を公開
	②SQL	SQLのPUSH型提供機能②を新しい統合DB機能のSOAP⑤に変更	SQLのPUSH型提供機能②を新しい統合DB機能のSQL⑥に変更	既存の提供機能②を停止	既存の提供機能②を停止して、提供側DBMSのSQLインタフェース⑧を公開
共通IF方式	③SOAP	新しい統合DB機能のSOAPによるPUSH型提供機能⑤を追加	新しい統合DB機能のSQLによるPUSH型提供機能⑥を追加	○対応不要	提供側DBMSのSQLインタフェース⑧を公開
	④SQL	新しい統合DB機能のSOAPによるPUSH型提供機能⑤を追加	新しい統合DB機能のSQLによるPUSH型提供機能⑥を追加	○対応不要	○対応不要

表３．２．４は、行方向（旧）が入れ替え前の統合 DB 機能の方式と提供側で採用されているインタフェースを示し、列方向（新）が差し替え後の統合 DB 機能の方式と提供側で新たに採用するインタフェースを示し、枠内はそれぞれの組み合わせにより必要となる提供側業務ユニットの作業を示している。共通インタフェース方式の統合 DB 機能は、比較的差し替えに強いことが解る。

（２）利用側業務ユニットの対応

旧統合 DB 機能を新統合 DB 機能に差し替える際に必要となる利用側業務ユニットの対応について表３．２．５で整理する。

表３．２．５は、行方向（旧）が入れ替え前の統合 DB 機能の方式と利用側で採用されているインタフェースを示し、列方向（新）が差し替え後の統合 DB 機能の方式と利用側で新たに採用するインタフェースを示し、枠内はそれぞれの組み合わせにより必要となる利用側業務ユニットの作業を示している。

SOAP インタフェースを新旧共に利用 I/F として採用している場合は、統合 DB 機能の差し替えは利用側業務ユニットには影響しない。
但し、利用 I/F として新旧いずれかに SQL を採用している場合には、利用側業務ユニットにも修正が必要となる。

表３．２．５ 統合 DB 機能の差し替えにおける利用側業務ユニットの対応

新 / 旧		公開用DB方式		共通ｲﾝﾀﾌｪｰｽ方式	
		⑤SOAP	⑥SQL	⑦SOAP	⑧SQL
公開用DB方式	①SOAP	○対応不要	標準SOAPによる利用I/Fを新しい統合DB機能のSQL⑥に変更	○対応不要	標準SOAPによる利用I/Fを新しい統合DB機能のSQL⑧に変更
	②SQL	旧SQLによる利用I/F②を標準のSOAPに変更	旧SQLによる利用I/F②を新しい統合DB機能のSQL⑥に変更	旧SQLによる利用I/F②を標準のSOAPに変更	旧SQLによる利用I/F②を新しい統合DB機能のSQL⑧に変更
共通IF方式	③SOAP	○対応不要	標準SOAPによる利用I/Fを新しい統合DB機能のSQL⑥に変更	○対応不要	標準SOAPによる利用I/Fを新しい統合DB機能のSQL⑧に変更
	④SQL	旧SQLによる利用I/F④を標準のSOAPに変更	旧SQLによる利用I/F④を新しい統合DB機能のSQL⑥に変更	旧SQLによる利用I/F④を標準のSOAPに変更	旧SQLによる利用I/F④を新しい統合DB機能のSQL⑧に変更

３．２．５　統合 DB 機能のデータモデル

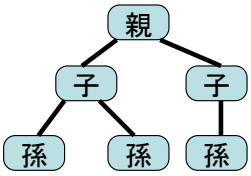
地域情報 PF 標準で規定しているデータ連携のインタフェース（ツリー構造）と、統合 DB 機能の実装として採用する表構造間のマッピングについて考え方および留意点を解説する。

「自治体業務アプリケーションユニット標準仕様」で規定している業務ユニット間のデータ連携のインタフェースは SOAP(XML)で規定しており、データモデルとして「ツリー構造」を採用している。
一方、業務ユニットや統合 DB 機能の具体的な実装は、データの管理に RDB（リレーショナルデータベース）を採用するのが一般的であり、データモデルとして「表構造」を採用する必要がある。

コラム３．２．５　「ツリー構造と表構造」

ツリー構造と表構造の一般的な性質について、表３．２．６にまとめる。

表３．２．６ データモデル

ツリー構造	表構造
□SOAPで使用するXMLはツリー構造 □1つの親ノードが0～N個の子ノードを持つ階層構造	□RDBMSやSQLは表構造を使用する □列方向にフラットな項目が並び、行方向にデータ(値)が並ぶ2次元構造
	項目１ 項目２ 項目３
□ツリーは先頭ノードで識別する □ノードの名前は任意(重複を許す) □ノードの性質は位置で決まる □存在しないノードを許す。 □ノードをたどって検索する	□表は名前(表名)で識別する □項目名は表内でユニーク □項目の性質は列で決まる □値が無い項目も存在する(NULL値) □キー項目の値で検索する
◇ツリー構造は表構造を表現可能	◇表構造で表現できないツリー構造がある。
☆ノードの繰り返し構造 → ☆複数の表に分割することで表現可能 ・同じ位置に同じノード構造が複数存在する構造	
★ノードの再帰構造 → ★再帰構造は表現できない ・1つのノード構造の中に自分のノード構造が存在する。	

地域情報 PF 標準では表構造について規定していないが、地域情報 PF で標準化しているツリー構造は表構造への変換を次の点で考慮しているので、相互に変換が可能である。
◇再帰的なツリー構造はない（ノードの繰り返しは存在する）

表構造とツリー構造の特徴を十分理解した上で、統合 DB 機能のデータモデル設計および変換機能の実装を行う必要がある。以下に変換規則の例を示す。

（１）原則的な変換（別表への分割）

全ての場合に変換可能であることを保障するためには、ツリー構造の繰り返し部分は、別の表に分割する。（図３．２．２３）

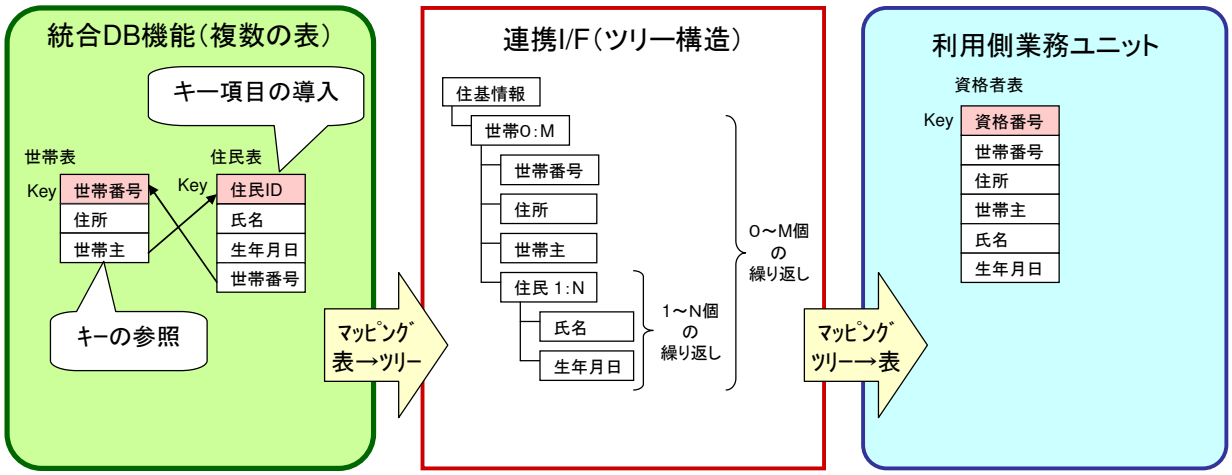


図３．２．２３ データモデルの変換（データ構造例）

【分割ルール】

①ツリー構造の同じ位置に繰り返し現れる可能性のあるノードは、別の表に分割する。

②別の表に分割したノードには、ユニークなキー項目を追加して一意性を保障する。

③分割したノード位置には、②で追加した表への参照を置く。

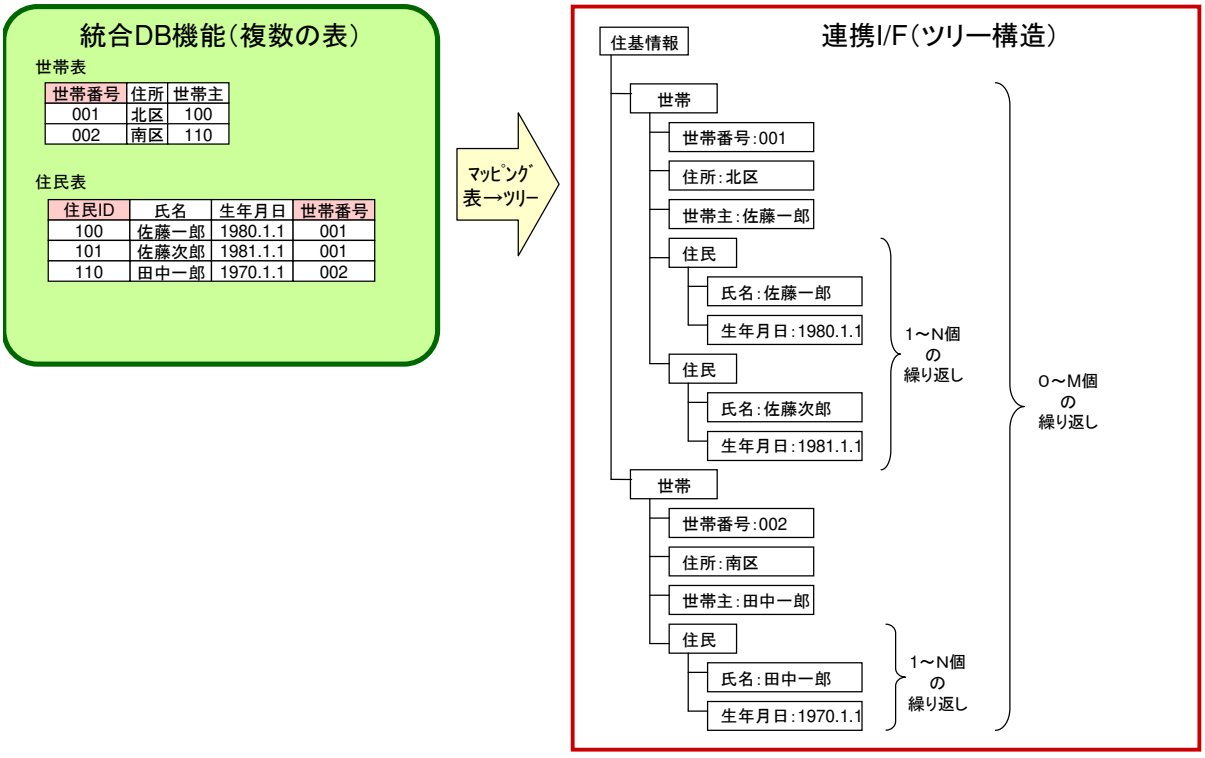


図３．２．２４ データモデルの変換（データ例）

（２）簡易的な変換（同一表への展開）

小さい表がたくさんできることを抑えるために、次の要件を満足する繰り返しノードは、別表に分割しないで、簡易的な変換とすることもできる。（図３．２．２５）

【簡易的な変換の採用条件】

◇繰り返し数の最大値が有限個であること。

◇繰り返し数の最大値が将来増えないこと。

◇繰り返し数の最大値が少ないこと。

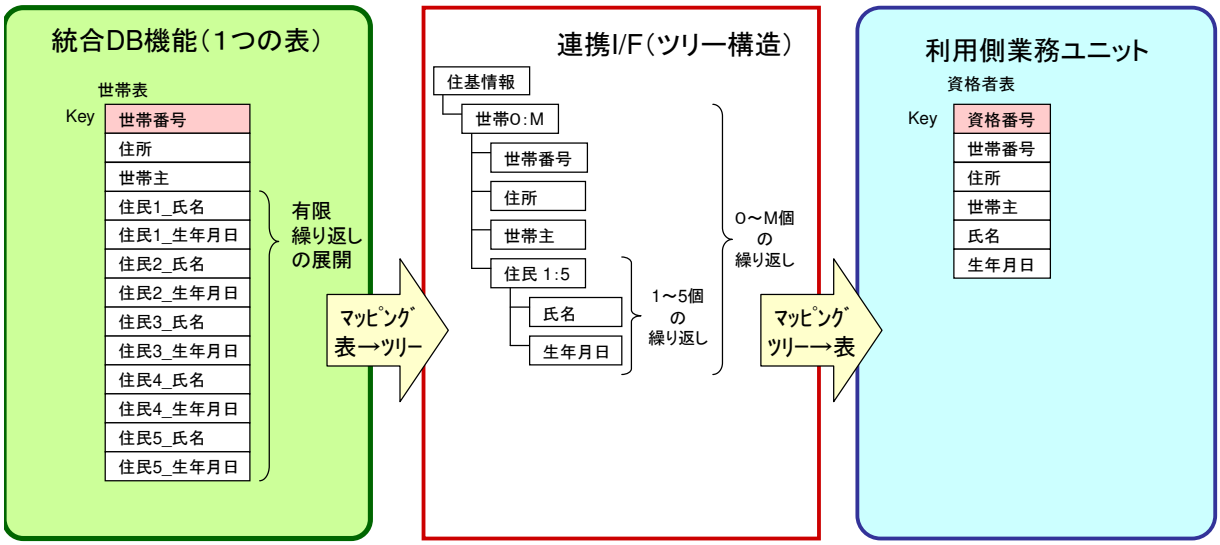


図３．２．２５ 簡易的なデータモデルの変換（データ構造例）

【簡易的な変換の適用ルール】

①ツリー構造の同じ位置に繰り返し現れるノードを、最大の繰り返し数だけ、別の名前の項目として展開する。（表構造で表現できる先頭ノードの繰り返しはそのままにする）

②該当ノードの子ノードが存在する場合は、該当ノードと併せて１つの項目にする。

③該当ノードの子孫ノードは別途変換規則（原則的な変換または簡易的な変換）を再帰的に適用する。

【簡易的な変換の規則例】

①繰り返し項目の表展開

有限回数の繰り返し項目について、統合 DB 機能は RDB 表として最大繰り返し数分の項目を準備する。（但し先頭ノードの繰返しは表のタプルとして表現されるので変換の対象外）

②繰り返し項目の項目名変更

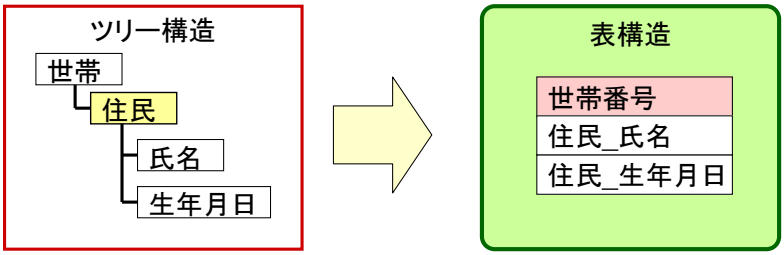
繰り返し項目は、表構造にしたときに同じ項目名にならないようにユニークな名前に置き換える。例えば、項目名の末尾に１から始まる追番を半角数字で付与する。

世帯番号	住民1	住民2	住民3	住民4	住民5

③子ノードの展開

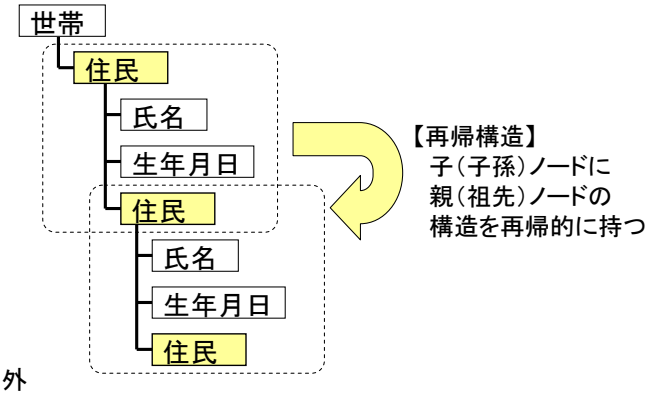
子ノードを表に展開する場合は、子ノードの名前と併せてユニークな項目名とする。

例えば、ツリー構造のノード名の各階層を_(半角アンダーバー)で繋いだものを表構造の項目名とする。



④再帰的な構造の禁止

再帰的なツリー構造は表構造で表現しにくいので使用しない(地域情報 PF では使用しない)。



⑤再帰的な構造の例外

有限回数の再帰構造は次の手段により表構造に展開できるが、複雑になるので推奨しない。

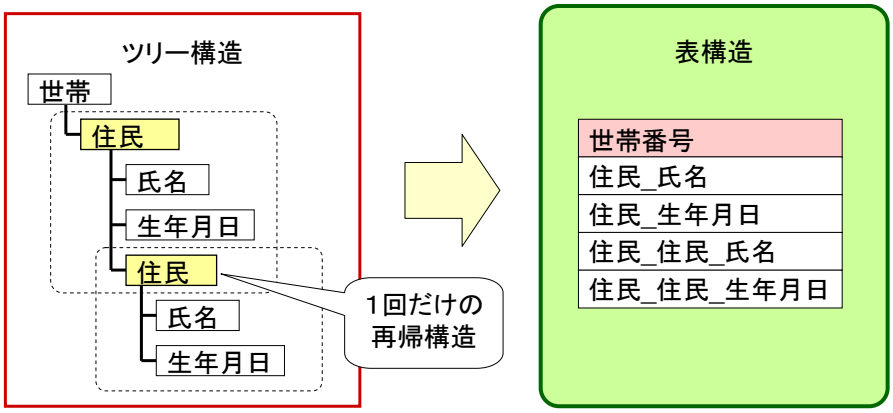


図３．２．２６ 簡易的なデータモデル変換の規則例

（３）留意事項

表構造は地域情報 PF で標準化していないが、一度実装されると変更することが難しくなるので、将来の拡張を考慮して慎重に設計すべきである。

①原則的な変換（別表への展開）の際は、性能や管理面を考慮して必要以上に小さな表にしない。

②簡易的な変換（同一表への展開）の際は、繰り返し項目の最大回数を現実的な値とする。

◇多すぎると性能劣化やデータ量増加の原因となる。

◇少ないと機能への制約になるばかりでなく、将来設計変更が必要となる。

③表名や項目名には外字や文字コード系に依存する文字を使わない。

３．２．６　統合 DB 機能へのアクセス方式

利用側業務ユニットから統合 DB 機能を利用する場合の SOAP、SQL それぞれのアクセス方式について解説する。

（１）　標準化した SOAP インタフェース

「自治体業務アプリケーション標準仕様」で規定している業務ユニット間のデータ連携のインタフェースを使用して、統合 DB 機能を利用する方法である。標準化しているインタフェースの採用により、統合 DB 機能の方式や実装に依存することなく、利用側業務ユニットは統合 DB 機能を利用して必要なデータを取得することができる。（図３．２．２７）

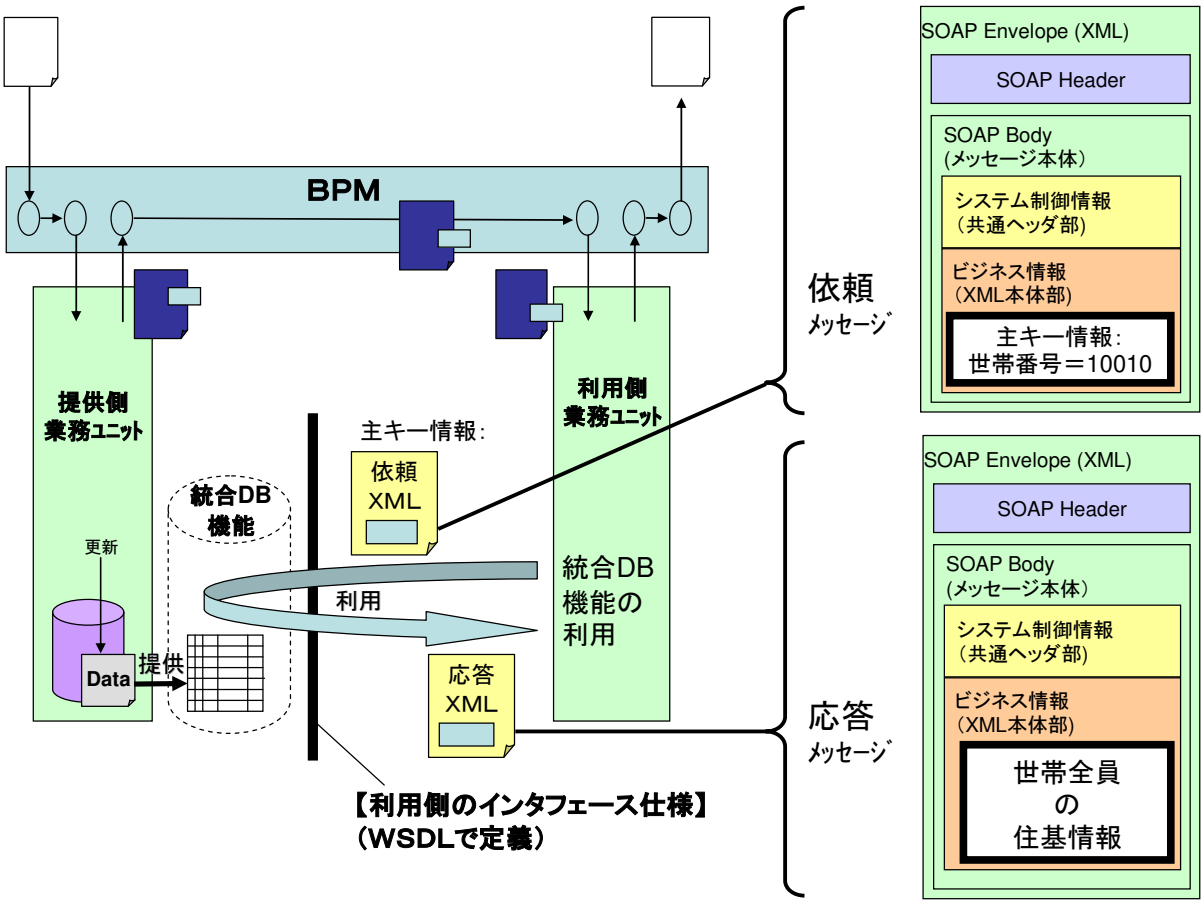


図３．２．２７ 統合 DB 機能の利用 I/F（概念）

具体的には次の仕様として規定している。

◇自治体業務アプリケーション標準仕様【インタフェース一覧】資料№.「業務 1-9」

この中で、提供側業務ユニット毎にサービス仕様(WDSL)およびデータ構造（XML スキーマ）を定義しており、統合 DB 機能に対するアクセスは、このインタフェース(SOAP)でアクセスすることになる。

例えば、図３．２．２８に示す「住民基本台帳」業務ユニットは、「個人情報メッセージ」を返す「1-1」と、「世帯情報メッセージ」を返す「1-2」の 2 つのサービスインタフェースを定義している。

【インタフェース定義】

インタフェース一覧			業務ユニット名 ： 住民基本台帳	
インタフェース番号	入出力		メッセージ定義	WSDL定義
1-1	入力	識別番号	識別番号メッセージ	lgxml01s-0150.xsd lgxml01s-0150.wsdl
	出力	個人情報	個人情報メッセージ	
1-2	入力	世帯番号	世帯番号メッセージ	lgxml01s-0150.xsd lgxml01s-0150.wsdl
	出力	世帯情報	世帯情報メッセージ	

【依頼メッセージ】

メッセージ定義				メッセージ定義名 ： 世帯番号メッセージ								
NO	データ項目名			データ型	桁数	コード		出現回数		外字使用	サンプル値	項目説明
						CD	コード名	最小	最大			
1	世帯番号			X	15			1	1		123456789	世帯情報を識別する番号

【応答メッセージ】

メッセージ定義					メッセージ定義名 ： 世帯情報メッセージ								
NO	データ項目名				データ型	桁数	コード		出現回数		外字使用	サンプル値	項目説明
							CD	コード名	最小	最大			
1	世帯情報								1	1			世帯を構成する全員の住基情報
2		世帯構成員情報							1	N			
3			識別番号		X	15			1	1		123456789	個人情報を識別する番号
4			世帯番号		X	15			1	1		123456789	世帯情報を識別する番号
5			住民種別		X	1	○	住民種別	1	1		1	住民・外国人・住登外等の種別を表す
6			住民状態		X	1	○	住民状態	1	1		1	住民・未登録・転出・死亡等住民の状態を表す
7			住民票コード		X	11			0	1		12345678901	住民基本台帳ネットワークの住民票コード
8			氏名		氏名情報				1	1	○		
9			性別		X	1	○	性別	1	1		1	男女別
10			生年月日		生年月日情報				1	1			
11			続柄		続柄情報				1	1		01	世帯主との続柄を表す
12			住民となった情報						1	1			
13				異動年月日	日付情報				1	1			住民基本台帳上の住民となった日
14				届出年月日	日付情報				1	1			住民基本台帳上の住民となった届出日
15				増異動事由	X	2	○	住民基本台帳異動事由	1	1		01	転入・出生などを表すコード
16			戸籍情報						0	1			
17				本籍地	住所情報				0	1	○		戸籍上の本籍を表す
18				筆頭者	氏名情報				0	1	○		戸籍上の筆頭者を表す
19			住民でなくなった情報						1	1			
20				異動年月日	日付情報				1	1			住民基本台帳上から除票となった日
21				届出年月日	日付情報				1	1			住民基本台帳上から除票の届出を行った日
22				減異動事由	X	2	○	住民基本台帳異動事由	1	1		04	転出・死亡などを表すコード
23			前住所情報		住所情報				1	1	○		転入、転居等の異動前住所を表す
24			転出先情報						1	1			
25				転出先住所	住所情報				1	1	○		転出先の住所を表す
26				住所区分	X	1	○	住所区分	1	1		2	予定・確定住所区分
27			世帯主氏名情報		氏名情報				1	1	○		
28			現住所情報		住所情報				1	1	○		
29			住所を定めた情報						1	1			
30				異動年月日	日付情報				1	1			現住所地に住所を定めた日
31				届出年月日	日付情報				1	1			現住所地に住所を定めた届出日
32				異動事由	X	2	○	住民基本台帳異動事由	1	1		01	転入・出生・転居などのを表すコード
33			独自領域		X	50			1	1			自治体個別利用領域
34			異動中区分		X	1	○	異動中区分	1	1		1	異動中・異動中でないを示す
35			異動事由		X	2	○	住民基本台帳異動事由	1	1		01	転出・死亡などを表すコード
36			異動年月日		日付時間情報				1	1			登録更新した日付時間

図３．２．２８ 自治体業務アプリケーション標準仕様【インタフェース一覧】（抜粋）

即ち、個人情報を参照する「1-1」のサービスを利用する場合は、利用側業務ユニットが住民個人を識別する「識別番号メッセージ」を統合 DB 機能に送信し、統合 DB 機能は指定された識別番号を持つ個人（1 名分）の「個人情報メッセージ」を応答として返す。

また、世帯情報を参照する「1-2」のサービスを利用する場合は、利用側業務ユニットが世帯を識別する「世帯番号メッセージ」を統合 DB 機能に送信し、統合 DB 機能は指定された世帯番号を持つ家族（1 名～50 名分）の「世帯情報メッセージ」を応答として返す。

このように、標準化した SOAP インタフェースは、規定した項目（キー項目）に対する検索条件（値）を指す依頼メッセージに対して、規定した検索結果を格納した応答メッセージを返すものであり、データ構造は XML で表現されたツリーを採用している。

利用側業務ユニットが表構造でデータを扱っている場合には、利用側業務ユニット内のデータモデルとの間で、マッピング（データ構造変換）が必要となる。

（２）SQL インタフェース

SQL による統合 DB 機能の利用 I/F は標準化していないため、その利用方法は統合 DB 機能の実装に依存するが、性能や通信情報の削減によるネットワーク負荷軽減などの目的で採用することができる。

SQL インタフェースを利用 I/F として採用する場合の留意点について解説する。

①標準的なインタフェースの採用

SQL による統合 DB 機能の利用において使用するインタフェースは統合 DB 機能から提供されるものを利用することになるので、厳密には統合 DB 機能の実装に依存するが、依存部分を最小化するために ODBC、JDBC などの標準化されたインタフェースを優先して採用するべきである。

②データモデルへの配慮

SQL による統合 DB 機能の利用では、データモデルとして統合 DB 機能側で規定される表モデルを採用し、表に対する条件検索として利用することになる。この表モデルは標準化していないため、統合 DB 機能により利用方法が異なるので留意する必要がある。この時、「3.2.5 統合 DB 機能のデータモデル」で解説した事項へ配慮することで、非互換を最小化するのに役立つ。

③データ表現への配慮

統合 DB 機能の SQL インタフェースは統合 DB 機能の実装に依存して扱うデータの表現が異なる場合がある。例えば、JDBC インタフェースを利用する際の数値の表現（バイナリ表現）は、統合 DB 機能の実装（採用している DBMS など）により異なる場合がある。

④利用する SQL 機能への配慮

統合 DB 機能の SQL インタフェースにおける SQL 機能は、その文法も含めて厳密には統合 DB 機能の実装（採用している DBMS など）により異なる場合がある。対策として、利用する SQL 機能は統合 DB の実装に依存しない範囲の必要最小限の機能だけにすることにより、非互換を最小化することができる。推奨する SQL の利用範囲は「汎用検索インタフェース」の問い合わせ言語欄を参照のこと。(3.2.7.3(3)を参照)

３．２．７　統合 DB 機能の応用

統合 DB 機能の応用として、「共通データの公開」「コード辞書の統合と公開」「汎用的な検索」「複数の情報源の統合」について解説する。これらは標準化していないが、有用な活用方法である。

３．２．７．１　共通データ公開

複数の業務ユニットで共通的に利用する共通データのマスタを共通データ管理機能により集中管理する機能である。

この機能は標準化していないので自治体の実情に合った仕組みで導入されるが、統合 DB 機能が貢献できるポイントとして共通データの公開がある。

共通データのマスタは、専用のユニット（共通データ管理機能）で管理するべきである。また、複数の部署に跨るデータの管理になるので、共通データのメンテナンス体制を確立することも重要である。

独立して運用されている共通データ管理機能を１つの提供側業務ユニットとして扱うことにより、統合 DB 機能を利用して共通データを各業務ユニットから利用できるようにすると便利である（図３．２．２９）。

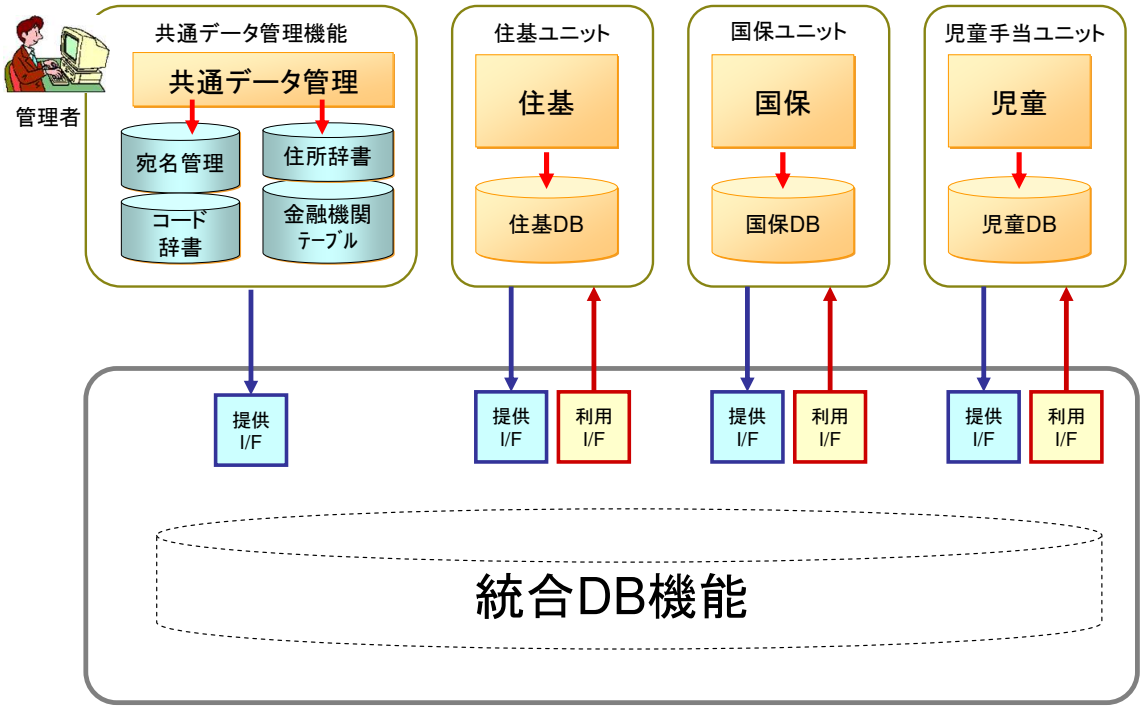


図３．２．２９ 統合 DB 機能による共通データの公開

また、共通データを物理的に集中管理するのではなく、複数の業務ユニットに分散して管理されている共通データを統合 DB 機能により統合して公開する方法もある。この場合、複数の業務ユニットに跨るデータになるので、「3.2.7.4 複数の情報源の複雑な統合」も参考になる。

３．２．７．２　コード辞書の統合と公開

コード辞書は、次の業務標準に記述しており、自治体内で統一することを推奨している。
◇自治体業務アプリケーション標準仕様【コード辞書】資料№.「業務 1-13」

統合 DB 機能でコード辞書の統合を行い、公開することにより次のメリットが期待できる。
①自治体内でコード辞書の利用を一本化することにより、各種コードが統一され、業務ユニット間のコード辞書非互換が是正される。
②コード辞書の分散管理が可能になる。
③将来の自治体間または民間との連携の際、コード辞書による読み替えや、コード辞書の変換を行う基盤となる。

コード辞書は次の２種類について業務標準で記述している。
・業務ユニットに共通のコードである「コード辞書（共通)」
・業務ユニット毎に定義されている「コード辞書（業務ユニット名)」

統合 DB 機能はマスタデータを管理する機能を持たないので、具体的なシステム構成として次の 2 種類がある。

（１）共通データとしてコード辞書のマスタを集中管理する方法
共通データとして全てのコード辞書の集中管理を行い、統合 DB 機能を介して各業務ユニットから利用（参照）できるように構成する方法である。コード辞書の管理を一本化できるメリットがある。

（２)「コード辞書（業務ユニット名)」の管理を業務ユニットに分散する方法
共通データとしては、業務ユニットに依存しない「コード辞書（共通)」のマスタ管理を行い、業務ユニットに依存する「コード辞書（業務ユニット名)」のマスタ管理は各業務ユニットに委ねて分散管理する方法である。

各業務ユニットで管理されているコード辞書は、統合 DB 機能で統合され、各業務ユニットに公開されるため、各業務ユニットは(1)と同様に統合されたコード辞書を活用することができる。

分散方式(2)のメリットは、コード辞書のメンテナンスを各業務ユニットに閉じて実施できる点と、業務ユニットの差し替えにより業務ユニットに依存するコード辞書に変更が生じた際の対応が容易になる点である。
また、体系の異なるコード辞書を使っている業務ユニットとのデータ交換では、交換時に業務コードの相互変換が必要になる場合があり、統合 DB 機能を介して双方のコード辞書を利用することにより変換機能の実装及び運用が容易になる効果も期待できる。

３．２．７．３　汎用的な検索インタフェース

自治体業務アプリケーション標準仕様で標準化している統合 DB 機能の利用 I/F（SOAP：3.2.6(1)参照）は、規定したキーによる統合 DB 機能の利用(検索)であり、定型業務では有効である。一方、非定型業務など自由な条件による検索を利用したいケースもあるので、統合 DB 機能には、標準化した SOAP インタフェース以外に、汎用的な検索インタフェースを持つと便利である。

ここでは、汎用的な統合 DB 検索インタフェースの仕様例について解説する。

（１）SQL インタフェース

SQL インタフェース（3.2.6(2)参照）は、キー項目以外の項目も検索条件として指定することができるので汎用的な統合 DB 検索インタフェースである。

（２）SOAP インタフェース

自治体業務アプリケーション標準仕様で標準化している統合 DB 機能の利用 I/F について、次の 2 件の拡張を行うことにより汎用的な条件検索を実現することができる。（汎用検索）

◇依頼メッセージの検索条件として、汎用的な検索条件（SQL など）を記述できるようにする。
◇応答メッセージの検索結果は、依頼メッセージで指定された検索対象の構造を持つ検索結果となるように拡張する。

図３．２．３０に SOAP による汎用検索インタフェースの例を示す。

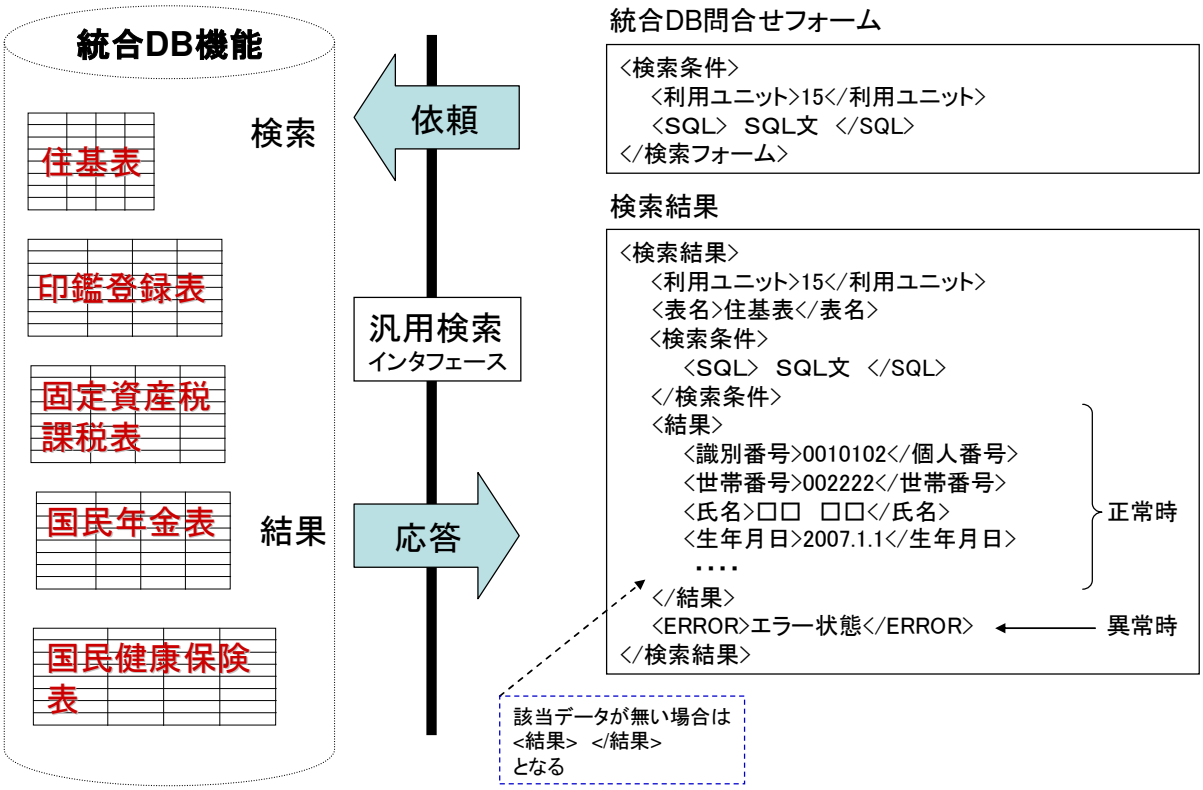


図３．２．３０ 汎用検索インタフェースの例(SOAP)

①利用側業務ユニットは問い合わせフォームの<SQL>タグに検索条件を SQL で記述して統合 DB 機能に検索依頼メッセージを送信する。

②統合 DB 機能は、受信した検索依頼メッセージの<SQL>タグを参照し、SQL 文で指定された表の検索を実行してその結果を<結果>タグに格納して利用側業務ユニットに応答する。

この時、結果が 0 件であれば、<結果>タグの値は空になる。
また、統合 DB 機能側で何らかのエラーが発生した場合には、<ERROR>タグでエラー内容を返す。
<結果>タグの内容として次の方法がある。
①検索対象表の項目名を<COLUMN>タグの属性値とするフラットな XML タグセットにして値を格納する。
②CSV フォーマットで検索結果を格納する。

◇「利用業務ユニット番号」は自治体業務アプリケーション標準仕様【業務ユニット番号一覧】資料 No.「業務 1-3」を参照

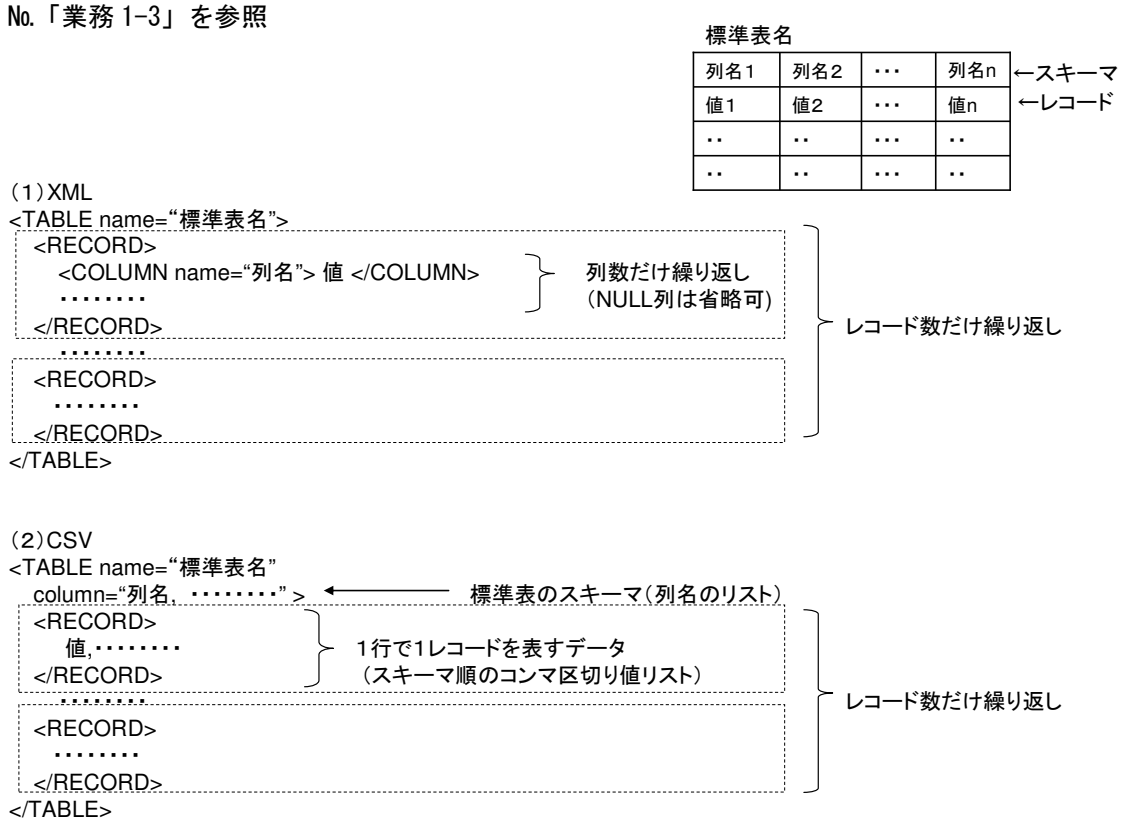


図３．２．３１ 汎用検索インタフェースの結果例

（３）必要最低限の検索言語(SQL)

汎用検索で使用する SQL タグの値として指定できる検索要求（SQL 文）は次に示す SQL の範囲で使用することが望ましい。

また、利用 I/F、提供 I/F として SQL インタフェースを採用する場合にも、実装(RDBMS)への依存をなくすため、必要最小限の SQL 機能を利用するとよい。

理由 1：実装依存の防止
SQL 文には実装に依存する仕様があるので、必要最小限の共通的な仕様に限定することで、非互換を最小限に抑える効果がある。

理由 2：実装を容易にする
SQL 文を単純にすることで、統合 DB 機能や SOAP サービス機能の実装が容易になる。
特に DBMS の実装を前提としない共通インタフェース方式の統合 DB 機能では、複雑な SQL を実装す

るコストが高い。

◇SQL 仕様

SELECT　選択リスト　FROM　標準表名　WHERE　検索条件

選択リスト ::= * | 選択項目[, 選択項目]...　　（*は全項目の選択）

検索条件 ::= 検索条件 AND 検索条件　| 検索条件 OR 検索条件　|　(検索条件) | 述部

述部 ::= 項目名 比較演算子 値　|
項目名　LIKE 検索パターン　|　項目名　IN　(値 [,値])

比較演算子 ::= < | > | <= | >= | = | <>

値 ::= 数字列　|　‘ 文字列 ’

・LIKE の検索パターンには曖昧検索条件値として、次のメタ文字を使える。
　　%: 任意の数の文字（ゼロ文字を含む）と一致する。
　　_: 厳密に 1 つの文字と一致する。(マルチバイト文字も 1 文字として扱う)

・FROM 句に記述できるのは、1 つの標準表名のみ
・選択項目は、対象表の項目名
・項目名は、対象表の列名

（留意点）
①複数の対象表間における関係演算はしない（項目同士の比較演算も不要）
　→　同じ結果となるビュー表で対応できる。

②XML の値に記述できない記号はエンティティー参照を使用する。
　例)　<SQL>SELECT * FROM 住民台帳
WHERE 氏名='田中'　AND 年齢 > 20 <SQL>

置換対象文字	エンティティー文字
<	&lt;
>	&gt;
‘	&apos;
“	&quot;

３．２．７．４　複数の情報源の複雑な統合

自治体業務アプリケーション標準仕様で標準化している統合 DB 機能の利用 I/F は、提供側業務ユニット毎に閉じたデータ構造になっており、複数の提供側業務ユニットに跨るデータの取得（検索）機能はない。より便利な統合 DB 機能の使い方として複数の情報源（提供側業務ユニット）に跨るデータの統合機能について解説する。

統合対象の一次情報（マスタデータ）は、分散管理されていても良いが、何れかの業務ユニットで管理され、提供側として統合 DB 機能に設定することが前提となる。また、複数の情報源に跨るデータの検索には、統合 DB 機能がデータ統合機能を持つ必要がある。

提供側業務ユニットへの影響がなく、利用側業務ユニットは目的とするデータ構造で統合 DB 機能を利用できることが望ましい。統合 DB 機能の方式毎に実現方法を以下に示す。（図３．２．３２）

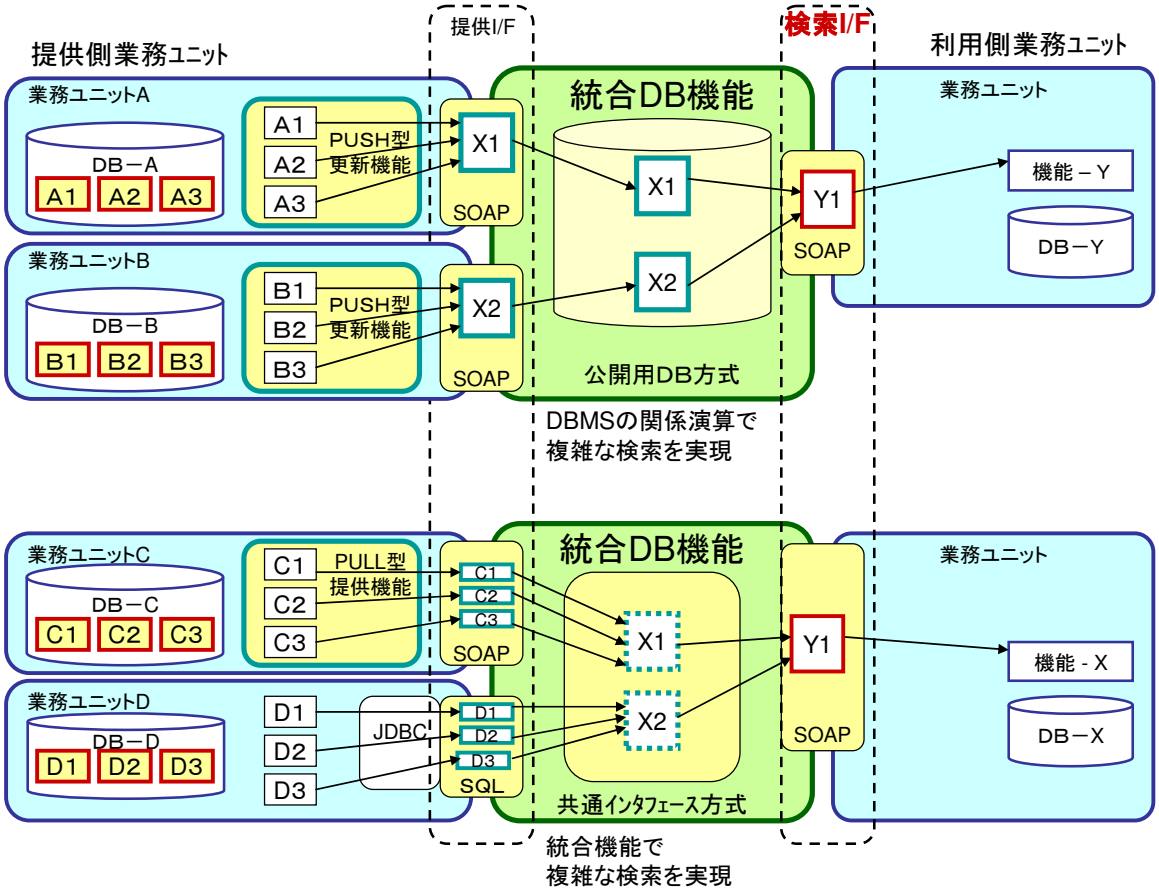


図３．２．３２ 複雑な検索の実現例

（１）公開用 DB 方式

公開用 DB 方式の統合 DB 機能は RDBMS により実現されるものであり、RDBMS のビュー機能を活用することで複数の情報源に跨るデータの検索（データ統合）を実現できる。

（２）共通インタフェース方式

共通インタフェース方式の統合 DB 機能は、RDBMS を必要とせず、統合 DB 機能が情報源（提供側業務ユニット）のデータモデルを意識して動的に仮想統合するものであり、複数の情報源のデータモデル（ス

キーマ定義）を持ち、情報源を仮想統合するための仮想ビューを利用側業務ユニットからアクセス可能にする機能により複数の情報源に跨るデータの検索（データ統合）を実現できる。

３．２．８　統合 DB 機能のセキュリティ

統合 DB 機能は、多くの業務ユニットのデータを公開する機能なので、セキュリティに対する考慮が必要である。

以下に統合 DB 機能のセキュリティに対する考え方および留意事項について記載する。

（１）自治体外部からの統合 DB 利用制限

統合 DB 機能は自治体内の業務ユニット間におけるデータ交換を担うものであり、自治体外部（サイト外）から直接利用されることは想定していない。（サイト間におけるデータ交換はスコープ外である。）従って、自治体間や官民連携など、サイトを跨るデータ交換は、外部データ連係専用の機能（外部ｻｰﾋﾞｽ連携機能）を介して実施する。（図３．２．３３）

この外部サービス連携機能とは、外部からの統合 DB 利用依頼に対して、認証や権限の確認を行い、外部に提供可能なデータであることを確認した上で利用側業務ユニットに代わって統合 DB 機能へのアクセスを行い、外部向けのデータ表現に補正して外部にデータを提供する機能である。

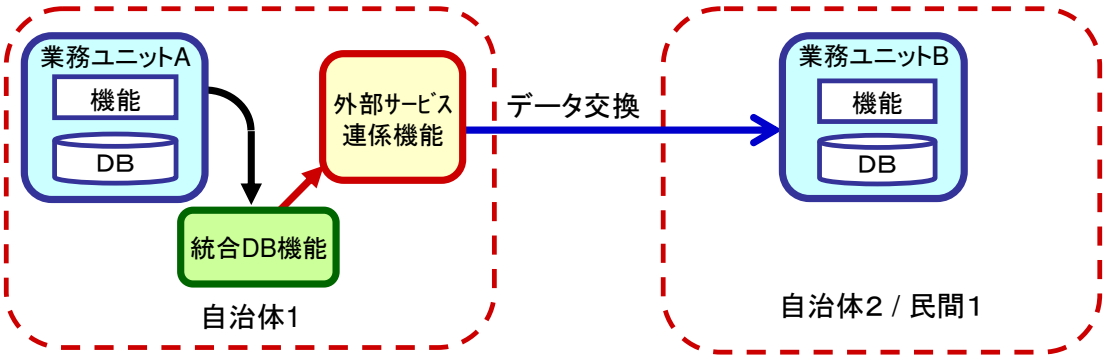


図３．２．３３ 自治体外部とのデータ交換

（２）統合 DB 機能のアクセス権制御

統合 DB 機能で公開されるデータに対するアクセス権制御は、そのデータ毎に、利用側業務ユニット単位で制御されるべきである。

個人に対する認証やアクセス権の制御は利用側業務ユニット側で実施されるべきである。

（３）データに対するアクセス権

統合 DB 機能で公開されるデータには、データ項目毎に公開または非公開とする業務を規定する方法がある。

表３．２．７ アクセス権の設定方法

アクセス権の設定方法	指定業務からアクセス	その他の業務からアクセス
公開する業務を指定	○	×
非公開の業務を指定	×	○
何も指定しない	○	○

データ項目単位にどちらの指定方法で規定するかを選択する。（混在禁止）

（４）統合 DB 機能は利用記録として次の項目のアクセスログを取得するべきである。
この利用記録を追跡する事により、統合 DB 機能の利用状況管理に役立つ。

表３．２．８ 取得すべきアクセスログ

記 録 項 目	解　　説
アクセス日時	統合 DB 機能から情報を取得した日時
アクセス業務	利用業務ユニット番号(※1)およびメッセージ ID
アクセス条件	統合 DB 機能に対する検索条件
アクセス結果件数	検索結果データ件数（※2）

（※1）利用業務ユニット番号は自治体業務アプリケーション標準仕様
【業務ユニット番号一覧】資料№.「業務 1-3」を参照。
（※2）アクセスされたデータそのもの（データ実体）をログとして記録すると、ログを通して情報漏えいが発生するリスクがあるので、データ実体は記録しないことを推奨する。

（５）提供側業務ユニットへのアクセス権制御
統合 DB 機能は全ての提供側業務ユニットとデータ交換を行うため、そのアクセス権は統合 DB 管理以外の目的で利用されないように適切に管理すべきである。（例：アクセス権定義ファイルの暗号化）

３．２．９　統合 DB 機能におけるエラーの扱いについて

アーキテクチャ標準仕様の「4.5.4.9 統合 DB 機能のエラー処理」および「4.5.4.10 統合 DB 機能における共通ヘッダの扱い」に規定する統合 DB 機能におけるエラーの扱いの考え方について解説する。
統合ＤＢ機能を適用する範囲（主に自治体内）においてエラーの扱い方について統一したルールが必要になると考えられる。従って、そうしたルールを策定する際に以降に示す解説が一助となれば幸いである。

図 3.2.34、図 3.2.35 に統合 DB 機能の方式毎に想定するエラーの発生箇所を「障№」で示す。表 3.2.9 ではエラー毎に「エラー状態」「統合 DB 機能における扱い」「共通ヘッダにおける扱い」を示す。

「障 1」～「障 6」は、統合 DB 機能の利用 I/F で想定するエラーを示し、統合 DB 機能の方式によらず共通である。表 3.2.A における「障 1」～「障 6」行の「9.業務結果」「10.結果」「11.システムエラー」列は、アーキテクチャ標準仕様の「4.5.4.10 統合 DB 機能における共通ヘッダの扱い」で規定する利用側応答メッセージ②における共通ヘッダ№9～11 の値を示す。

「障 10」～「障 15」は、共通インタフェース方式の統合 DB 機能における提供 I/F で想定するエラーを示す。提供 I/F として SQL を採用する場合も対応するエラー状態を検出して利用側にエラーを通知することを推奨する。表 3.2.A における「障 10」～「障 15」行の「9.業務結果」「10.結果」「11.システムエラー」列は、アーキテクチャ標準仕様の「4.5.4.10 統合 DB 機能における共通ヘッダの扱い」で規定する提供側応答メッセージ④における共通ヘッダ№9～11 の値を示す。

「障 16」～「障 18」は、公開用 DB 方式の統合 DB 機能における提供 I/F で想定するエラーを示す。公開用 DB 方式の統合 DB 機能における提供 I/F では、共通ヘッダを使用しないので、表 3.2.9 の「9.業務結果」「10.結果」「11.システムエラー」列は規定しない。

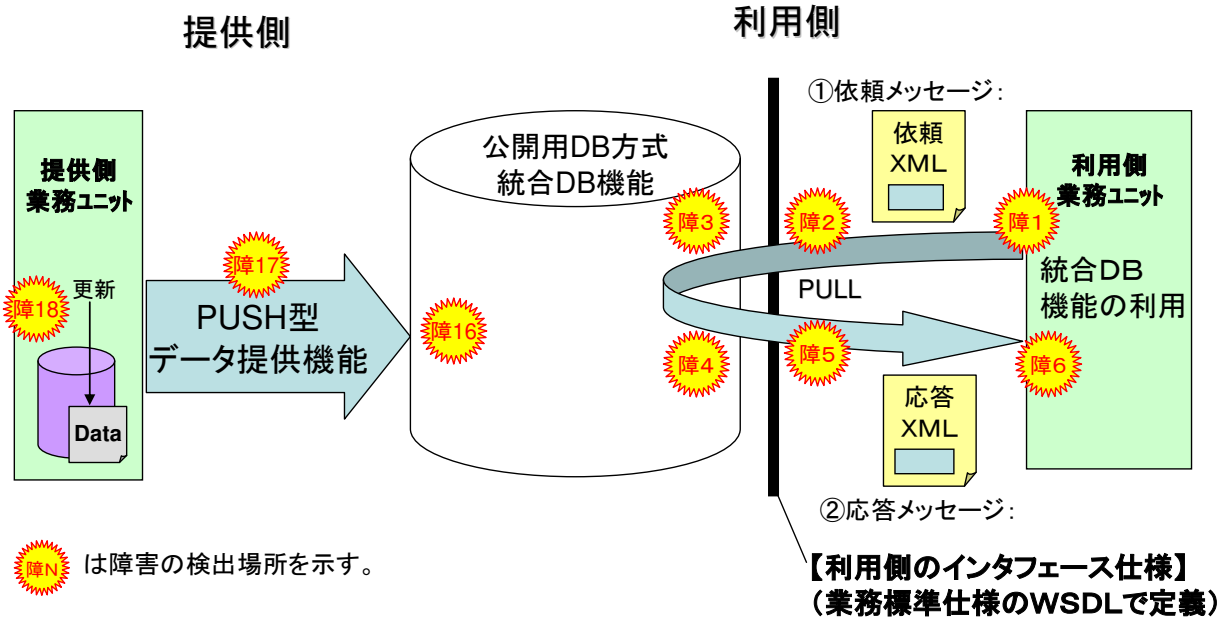


図３．２．３４　公開用 DB 方式の統合 DB 機能におけるエラーの発生箇所

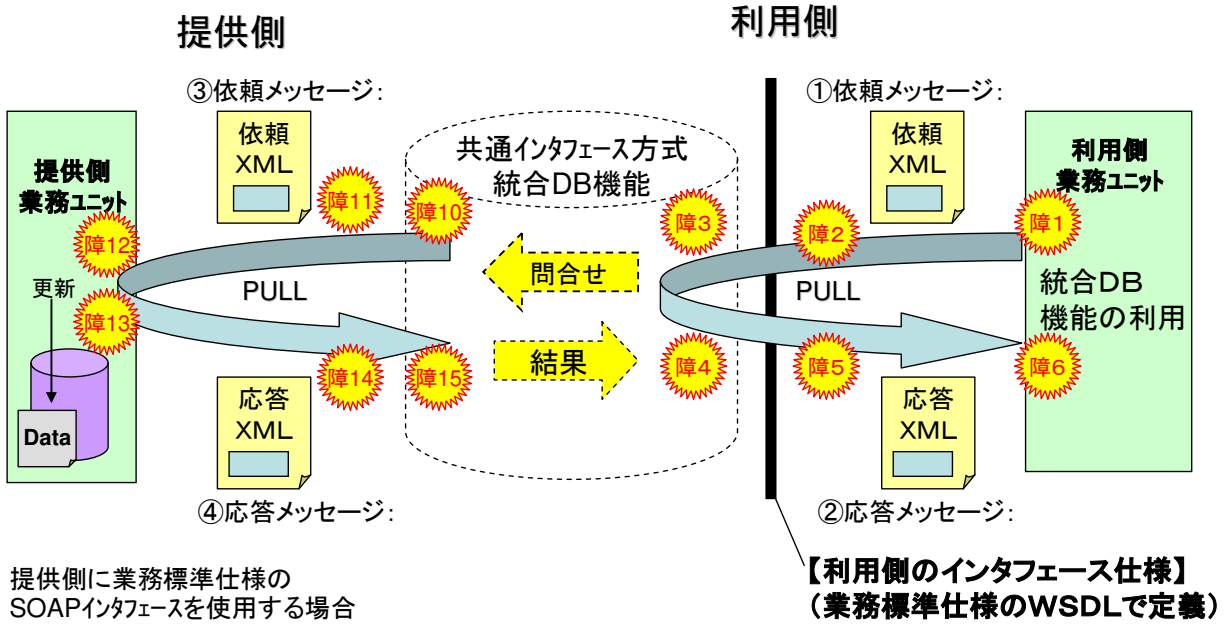


図３．２．３５　共通インタフェース方式の統合 DB 機能におけるエラーの発生箇所

表３．２．９　統合 DB 機能のエラー状態と共通ヘッダにおける扱い（詳細）

No.	利用側	提供側	エラー状態	統合ＤＢ機能の動作	9:業務結果	10:結果	11:ｼｽﾃﾑｴﾗｰ
障1	○		利用側業務ユニットの送信ｴﾗｰ	関与しない（利用側業務ﾕﾆｯﾄで処理）			
障2	○		依頼ﾒｯｾｰｼﾞの通信ｴﾗｰ	関与しない（利用側業務ﾕﾆｯﾄにSOAP-Fault）			
障3	○		依頼ﾒｯｾｰｼﾞの受付時ｴﾗｰ	◇⑤統合DB異常 ◇依頼ﾒｯｾｰｼﾞ異常 ①検索対象が存在しない(ｱｸｾｽ権無を含む) ②該当項目が存在しない(ｱｸｾｽ権無を含む) ③検索条件異常	なし ↓	1 ↓	5 - 1 2 3
障4	○		応答ﾒｯｾｰｼﾞの生成時ｴﾗｰ	◇結果が無い ◇⑤統合DB異常 ⑥情報量超過 ⑦タイムアウト ◇④提供側異常	1 なし ↓	0 1 ↓	なし 5 6 7 4
障5	○		応答ﾒｯｾｰｼﾞの通信ｴﾗｰ	統合DBにSOAP-Fault(ﾘﾄﾗｲ後にｴﾗｰ処理)	統合DB機能の処理結果		
障6	○		利用側業務ユニットの受信ｴﾗｰ	関与しない（利用側業務ﾕﾆｯﾄで処理）	統合DB機能の処理結果		
次の６項目は共通インタフェース方式のみ（提供側にＳＱＬを使用する場合も該当するエラー状態を検出して利用側にエラー報告する）							
障10		○	統合DBの依頼ﾒｯｾｰｼﾞ送信ｴﾗｰ	利用側にエラー報告（⑤）			
障11		○	依頼ﾒｯｾｰｼﾞの通信ｴﾗｰ	統合DBにSOAP-Fault(利用側にｴﾗｰ報告④)			
障12		○	依頼ﾒｯｾｰｼﾞの受付時ｴﾗｰ	利用側にエラー報告（④）	なし	1	N
障13		○	応答ﾒｯｾｰｼﾞの生成時ｴﾗｰ	利用側にエラー報告（④）	なし	1	N
障14		○	応答ﾒｯｾｰｼﾞの通信ｴﾗｰ	利用側にエラー報告（⑦）タイムアウト	提供側業務ﾕﾆｯﾄの結果		
障15		○	統合DBの応答ﾒｯｾｰｼﾞ受信ｴﾗｰ	利用側にエラー報告（④）	提供側業務ﾕﾆｯﾄの結果		
次の３項目は公開用DB方式のみ							
障16		○	提供側業務ユニットエラー	利用側にエラー報告（④）			
障17		○	PUSH型データ提供機能エラー	利用側にエラー報告（④）			
障18		○	統合DB側PUSH更新エラー	利用側にエラー報告（④）			

以下では、表 3.2.9 についてエラー状態毎に解説する。

【利用 I/F におけるエラー状態（統合 DB 機能の方式に共通）】

「障 1」：　利用側業務ユニットにおいて、利用側依頼メッセージ①を統合 DB 機能に対して送信するまでに発生したエラーであり、利用側業務ユニット内で処理される。統合 DB 機能には依頼メッセージが届かないので統合 DB 機能は関与できず、共通ヘッダを有する応答メッセージを返すこともできない。

「障 2」：　利用側業務ユニットから正常に送出された利用側依頼メッセージ①が、統合 DB 機能に到着するまでの間に発生する通信エラーであり、利用側業務ユニットに対して SOAP-Falt が返されるので、利用側業務ユニットがエラーへの対処を行う。統合 DB 機能には依頼メッセージが届かないので統合 DB 機能は関与できず、共通ヘッダを有する応答メッセージを返すこともできない。

「障 3」：　到着した利用側依頼メッセージ①を統合 DB 機能が受信する際に発生するエラーである。統合 DB 機能は発生したエラー毎にアーキテクチャ標準「4.5.4.9　統合 DB 機能のエラー処理」で規定するエラー状態を利用側応答メッセージ②の共通ヘッダに設定して利用側業務ユニットに送出す

る。ここで、「統合 DB 機能の動作」における丸付数字は、共通ヘッダNo.11 の値を示す。例えば、利用側依頼メッセージ①で指定された検索対象が存在しない（無効である）場合や、検索対象に対するアクセス権が無い場合には、利用側応答メッセージ②の共通ヘッダNo.10 の値として「１」を設定し、No.11 の値として「１」を設定し、No.9 は存在しない。

「障 4」: 統合 DB 機能が利用側依頼メッセージ①の受付を正常に完了した時点から、利用側応答メッセージ②を生成して、利用側業務ユニットに送出するまでの間に、統合 DB 機能内において発生するエラーである。統合 DB 機能は発生したエラー毎にアーキテクチャ標準「4.5.4.9　統合 DB 機能のエラー処理」で規定するエラー状態を利用側応答メッセージ②の共通ヘッダに設定して利用側業務ユニットに送出する。ここで、「統合 DB 機能の動作」における丸付数字は、共通ヘッダNo.11 の値を示す。例えば、提供側業務ユニットの異常によって、正常な結果を返すことができない場合は、「④提供側異常」として、利用側応答メッセージ②の共通ヘッダNo.10 の値として「１」を設定し、No.11 の値として「４」を設定し、No.9 は存在しない。

「障 5」: 統合 DB 機能から正常に送出された利用側応答メッセージ②が、利用側業務ユニットに到着するまでの間に発生する通信エラーであり、統合 DB 機能に対して SOAP-Falt が返される。統合 DB 機能は、リトライなど、可能な復旧策を講じ、異常が解消しない場合は送信をあきらめる他無い。この場合、利用側業務ユニットには利用側応答メッセージ②は届かないので、利用側業務ユニットはタイムアウトなどにより異常を検出して対処することになる。

「障 6」: 到着した利用側応答メッセージ②を利用側業務ユニットが受信する際に発生する利用側業務ユニット内のエラーであり、利用側業務ユニット内で対処する。統合 DB 機能の役割は終えているので、統合 DB 機能は関与しない。この利用側応答メッセージ②に含まれる共通ヘッダには、統合 DB 機能の処理結果により決まる処理状態が格納されている。

【提供 I/F におけるエラー状態（共通インタフェース方式の統合 DB 機能)】

「障 10」: 統合 DB 機能において、提供側依頼メッセージ③を提供側業務ユニットに対して送信するまでに発生したエラーである。統合 DB 機能は、「障 4」の「⑤統合 DB 異常」として処理する。提供側業務ユニットは提供側依頼メッセージ③が届かないので、関与できず、提供側応答メッセージ④も返されない。

「障 11」: 統合 DB 機能から正常に送出された提供側依頼メッセージ③が、提供側業務ユニットに到着するまでの間に発生する通信エラーであり、統合 DB 機能に対して SOAP-Falt が返されるので、統合 DB 機能は、「障 4」の「④提供側異常」として処理する。提供側業務ユニットは提供側依頼メッセージ③が届かないので、関与できず、提供側応答メッセージ④も返されない。

「障 12」: 到着した提供側依頼メッセージ③を提供側業務ユニットが受信する際に発生するエラーである。提供側業務ユニットは、共通ヘッダNo.10 の値として「１」を設定し、No.11 の値として「提供側業務ユニットで規定されたエラー値 N」を設定し、No.9 は存在しない状態の提供側応答メッセージ④を統合 DB 機能に送出する。この提供側応答メッセージ④を受信した統合 DB 機能は、「障 4」の「④提供側異常」として処理する。

「障 13」: 提供側業務ユニットが提供側依頼メッセージ③の受付を正常に完了した時点から、提供側応答メッセージ④を生成して、統合 DB 機能に送出するまでの間に、提供側業務ユニット内において

発生するエラーである。提供側業務ユニットは、共通ヘッダNo.10の値として「１」を設定し、No.11の値として「提供側業務ユニットで規定されたエラー値N」を設定し、No.9は存在しない状態の提供側応答メッセージ④を統合DB機能に送出する。この提供側応答メッセージ④を受信した統合DB機能は、「障4」の「④提供側異常」として処理する。

「障14」： 提供側業務ユニットから正常に送出された提供側応答メッセージ④が、統合DB機能に到着するまでの間に発生する通信エラーであり、提供側業務ユニットに対してSOAP-Faltが返される。統合DB機能には提供側応答メッセージ④が届かないので、統合DB機能はタイムアウトとして異常を検出し、「障4」の「⑦タイムアウト」として処理する。

「障15」： 到着した提供側応答メッセージ④を統合DB機能が受信する際に発生する統合DB機能内のエラーであり、統合DB機能は、「障4」の「④提供側異常」として処理する。この提供側応答メッセージ④の共通ヘッダには、提供側業務ユニットの処理結果により決まる処理状態が格納されている。

【提供I/Fにおけるエラー状態（公開用DB方式の統合DB機能)】

「障16」： 統合DB機能において、提供側業務ユニットのエラーを検出して、正常な値を返すことができない場合には、「障4」の「④提供側異常」として処理する。

「障17」： 統合DB機能において、利用側業務ユニットからのPUSH型データ提供機能のエラーを検出して、正常な値を返すことができない場合には、「障4」の「④提供側異常」として処理する。

「障18」： 統合DB機能において、利用側業務ユニットからのPUSH型データ提供機能に対する処理でエラーを検出し、正常な値を返すことができない場合には、「障4」の「④提供側異常」として処理する。

３．２．１０　統合DB機能の運用（バックアップ、リカバリ）

統合DB機能の運用に関する留意点について解説する。
特に統合DB機能の運用が停止すると、地域情報PF全体の業務が継続できなくなるので運用への配慮は重要である。

運用に関する事項は特に採用する製品が持つ機能に依存する部分が多いので、製品の利点や留意点を考慮して検討すべきである。ここでは、統合DB機能の方式における一般的な特徴と留意点について解説する。

（１）バックアップ
メタ定義や環境設定類は、構成変更毎に確実なバックアップを取得して復旧可能な状況にすること。

①公開用DB方式の統合DB機能
公開用DB方式の統合DB機能は、提供された全ての二次データを物理的に格納して運用されるため、二次データのバックアップについて工夫する必要がある。特に膨大な情報量となる二次データのバックアップと復元の仕組みは、異常時の停止時間を最小化するために重要であり、次の手法がある。

・完全バックアップ　＋　差分バックアップ
・完全なレプリケーション

また、ディザスタリカバリ（緊急時の復旧対策）として遠隔地にバックアップセンターを設置して、オンラインバックアップを行う手法もある。

②共通インタフェース方式の統合 DB 機能

共通インタフェース方式の統合 DB 機能は､原則として二次データを蓄積管理しないので､バックアップおよび復旧は、メタ定義や環境ファイルなど少ないデータ量で済む。

（２）負荷分散

各業務ユニットのデータ交換を仲介する統合 DB 機能の性能は、直接全業務ユニットの性能に影響するので、運用時の負荷見積や検証を確実に行い、柔軟な負荷分散によりスケーラブル性を確保すべきである。

①公開用 DB 方式の統合 DB 機能

公開用 DB 方式の統合 DB 機能を単純に負荷分散すると、提供側業務ユニットは分散運用している全ての統合 DB 機能に対してデータの提供を行う必要が生じるため、提供性能への負荷が増大するので工夫が必要である。

対策として次の手法がある。

・担当するデータ（提供側業務ユニット）を区切って複数の統合 DB 機能に分散する。
　→この場合利用側は統合 DB 機能を対象データ毎に使い分ける必要があるので､支援機能が必要。
・高性能 DB サーバの採用
・DBMS 自体のレプリケーションや仮想化（クラスタリング）

②共通インタフェース方式の統合 DB 機能

共通インタフェース方式の統合 DB 機能は、原則として二次データを蓄積管理しないので、全く同じ動作環境で複数の統合 DB 機能を分散運用することが可能である。

即ち、利用側業務ユニットからの要求に応じて必要な情報源（提供側業務ユニット）にアクセスして動的にデータを取得するので、共通の情報源にアクセスする統合 DB 機能を複数並列に運用し、利用側（または中継機能）がアクセスを振り分けることにより負荷分散を実現する。

（３）冗長運転

可用性の確保のため、冗長運転構成は必須である。

なお、統合 DB 機能が運用停止した場合の回避策として、統合 DB 機能を経由しないで直接提供側業務ユニットにアクセスする手段もある。

①公開用 DB 方式の統合 DB 機能

公開用 DB 方式の統合 DB 機能の冗長運転は、膨大な二次データの管理を行うので、採用している DBMS の冗長構成により対応することが多い。

②共通インタフェース方式の統合 DB 機能

負荷分散の②で述べたように、共通インタフェース方式の統合 DB 機能は複数の統合 DB サーバを並行運転することが容易であり、この構成は冗長構成としても使用できる。即ち、アクセス中の統合 DB 機能がエラーを起こした場合は、利用側業務ユニット（または中継する仕組み）が他の稼働中の統合 DB 機能に切り替えることで運用を継続できる。

用語集：

本ガイドラインで使用する統合 DB 機能に関する用語を次表にまとめる。

	用語	意味
1	統合 DB 機能	業務ユニット間で、データを利用（参照）する仕組みであり、方式として【公開 DB 方式】と【共通インタフェース方式】がある。
2	共通データ	PF 全体で共通に使用するデータ（コード辞書など）
3	業務ユニット	独立して特定の業務を処理するユニットで、ユニット毎に業務固有のマスタ DB を持つ
4	提供側業務ユニット（提供側）	統合 DB 機能に対して、自己で管理するデータを提供する役割を持つ業務ユニット
5	利用側業務ユニット（利用側）	統合 DB 機能で公開されているデータを利用する業務ユニット
6	提供 I/F	提供側業務ユニットから統合 DB 機能にデータを提供する機能を持つインタフェース
7	利用 I/F	統合 DB 機能を利用側業務ユニットからアクセス機能を持つインタフェース
8	二次データ	業務ユニットで管理されているマスタデータ（一次データ）に対して、マスタデータの複製またはマスタデータから変換された二次的なデータを二次データと呼ぶ。公開 DB 方式の統合 DB 機能は、提供側業務ユニットから提供された二次データを統合 DB 機能内に保持して利用側に公開する。
9	公開 DB 方式	物理的な DBMS に公開対象の二次データ（提供側から提供されたデータ）を保持し、この二次データを介してデータの受け渡しを行う仕組みを持つ統合 DB 機能の方式
10	共通インタフェース方式	統合 DB 機能内における二次データの保持を必須とせず、利用側からの要求に応じて、必要なデータを提供側から収集して利用側に回答することにより、利用側から提供側を仮想的に参照する仕組みを持つ統合 DB 機能の方式
11	SQL	Structured Query Language(構造化問合せ言語)の略称。 リレーショナル型データベース用のアクセス言語として、ANSI、ISO、JIS で規格化されている。
12	JDBC	Java Database Connectivity の略称。 Java アプリケーションからデータベースにアクセスする機能を持つ API(アプリケーションインタフェース)。
13	ODBC	Open DataBase Connectivity の略称。 Microsoft 社により提唱された、アプリケーションからデータベースにアクセスするための API。
14	EII	Enterprise Information Integration の略称。 仮想的な統合を行うことでデータのフォーマットや、保管場所を変える必要がない分散データ管理を実現するデータ管理ソフトウェアの総称であり、フェデレーションと呼ぶこともある。

３．３　BPM 機能

本章では、BPM(Business Process Management)機能に関する概要、それを用いたシステムの構築、留意事項について概説する。

３．３．１　BPM 機能とは？

（１）　地域情報プラットフォームにおけるワンストップサービスと BPM の関係

ビジネスプロセスマネージメント(BPM)に関する業界団体の１つである､日本 BPM 協会の定義によれば、ビジネスプロセスマネージメントとは、経営目標を実現させる業務（ビジネスプロセス）を迅速に設計し、その改善サイクルを継続的に進める新しい経営手法と定義され、業務の可視化、デザイン、業務プロセス構築、業務モニタリング、業務実績評価という一連の業務改善サイクルを、ICT(Information and Communication Technology)を利用し継続的に実施することと定義される。

地域情報プラットフォームでは、このビジネスプロセスマネージメントの考え方を広義概念として取り込み、自治体等を含めた住民サービス業務のあり方について改善するべく、一度の手続きで必要とする作業を全て完了でき､真に利便性に優れた住民サービスを提供できるワンストップサービス､並びに､このワンストップサービスを実現するプラットフォームの標準の定義を試み、その普及促進を図っている。これにより、地域情報プラットフォームでは広義のビジネスプロセスマネージメントによる種々の住民サービスの向上を指向すると伴に、副次的効果として公共領域における業務内容の可視化、開かれたガバナンス対応、各種コンプライアンス等も期待できることになる。

上記に基づけば、地域情報プラットフォームで「BPM 機能」という言葉を定義する場合、広義のビジネスプロセスマネージメントの定義よりも、むしろ、より限定された狭義の意味で用いられる。それは、前述ビジネスプロセスマネージメントの定義要素である着眼、モデリング、再設計、実装・配置、業務適用、運用最適化、新たな着眼という一連のビジネスプロセス改善のライフサイクルのうち、再設計されたビジネスプロセスを具現化、自動実行するためのソフトウェア基盤の意味である。つまり地域情報プラットフォームでは、BPM 機能と称するものはビジネスプロセス管理機能、すなわち自治体内の内部向けサービス、および、自治体間連携サービス、官民間連携サービスの実行を制御する機能であり、業務サービスインタフェースを組み合わせてビジネスプロセスを組上げ、ワンストップサービスを構築することができるソフトウェア基盤の意味で定義される。

（２）　地域情報プラットフォームにおける BPM 機能に関する要件事項

地域情報プラットフォームの BPM 機能に関連した分野は、ワークフロー(Workflow)、企業アプリケーション統合 (EAI: Enterprise Application Integration)、企業間ではビジネスプロセス統合(BPI: Business Process Integration)と発展して来た、ICT 領域の中で比較的長い歴史を持つ分野である。これに従い、一口に BPM 機能と言っても、複数の標準化団体による関連標準案が複数提案されており、複数の実装方法が存在する。例えば､ワークフローに関する国際的な団体である WfMC(Workflow Management Coalition)が定義する XPDL(XML Process Definition Language)は、従来のワークフローの標準化の流れを受けて定義されているものであり､BPM 機能の一部をも提供し得る。また､国連と OASIS(Organization for the Advancement of Structured Information Standards)が共同で取り組んだ ebXML(Electronic Business using eXtensible Markup Language)等でも似た機能として BPSS (ebXML Business Process Specification Schema)が定義されている。

地域情報プラットフォームでは、各業務サービスインタフェースを組み合わせてビジネスプロセスを組上げ、より高付加価値で利便性の高いワンストップサービスの実現を、その狙い、要件としている。このことから、地域情報プラットフォームでは、必要時に必要な形で接続が可能となる疎結合特性を持つ Web サービスの技術を全面的に、そのアーキテクチャに採用している。このため、BPM 機能、並びにプロセス定義のためのスクリプト等は、地域情報プラットフォームのプラットフォーム通信標準仕様の他

の部分と十二分に親和性がある必要がある。そのため、プラットフォーム通信標準仕様のアーキテクチャでは、SOAP(Simple Object Access Protocol)通信等と親和性が高く、最新版である WS-BPEL(Web Services Business Process Execution Language 2.0)を全面的に採用している。

（３） BPM 機能の導入効果

BPM 機能を導入して得られる主要な期待効果は、高付加価値で利便性の高いワンストップサービスの導入ができる点にある。その意味で、BPM 機能だけではなく、それを応用した利便性の高いサービスが提供されて始めて狙いとする主要な期待効果が実現されることになる。

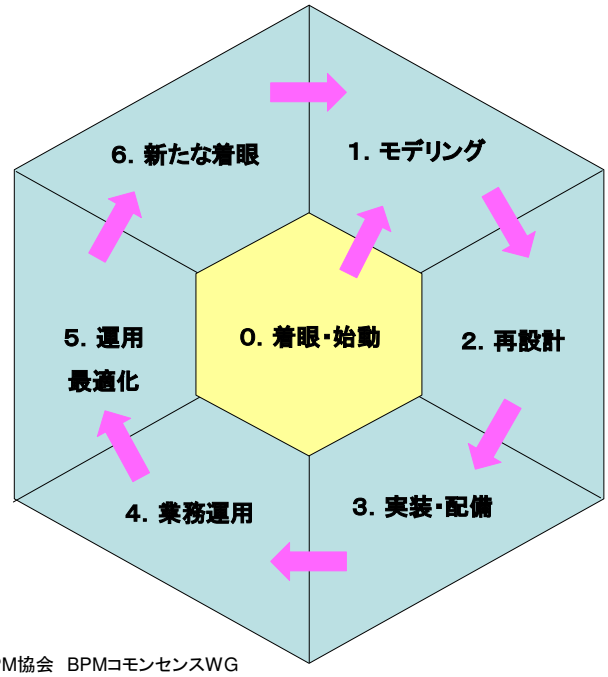
しかし ICT の論点で眺めてみると、BPM 機能は通常、XML(Extensible Markup Language)を利用した仕様言語、スクリプト言語により実行される。このため、柔軟性が高く、種々の評価指標に応じてビジネスプロセスを発展させるために、これらを変更することは容易にできる。このため、例えば業務ユニット間のデータ連携のため、単純・繰り返しに行なわれる操作プロセスの自動化、不必要な手続きの効率化・省力化、さらに、１つの自治体内部に於けるビジネスプロセスの改善だけではなく、民間等へのアウトソーシング、BPO(Business Process Outsourcing)の実施等を、自治体内のプロセスと同等のインタフェースを維持しながら実施できる点で、BPM 機能は大きな副次的導入効果を持つと言える。

３．３．２ BPM 機能の構築手順

（１） 地域情報プラットフォームでのビジネスプロセスマネージメントの実施手順と特徴的事項

一般的な民間企業において BPM 機能を導入する手順については、幾つかの文献や、各ベンダが提供するホワイトペーパー等が存在する。例えば、日本 BPM 協会の定義している図３．３．１の様な BPM 推進フレームワークが該当する。この中では、モデリング(分析)～再設計等の一連のビジネスプロセスマネージメントに関連した方法論も詳細に定義されており、全体の課題解決を網羅的に解決するフレームワークとしては概ね妥当なものと言える。

地域情報プラットフォームの展開においても、概ね同等の技術的な考え方を継承しているが、高付加価値・利便性の高いワンストップサービスを個々自治体等の組織体に限らず、全国の自治体、関係団体で共通的に定義するため、標準的な業務標準、業務インタフェースを定義するという点において、一般の民間企業におけるビジネスプロセスマネージメントの実施手順とは質的に異なる面がある。その差分について図解したものを図３．３．２に記す。



出典：日本BPM協会　BPMコモンセンスWG

図３．３．１　日本 BPM 協会の提唱する BPM 推進フレームワーク

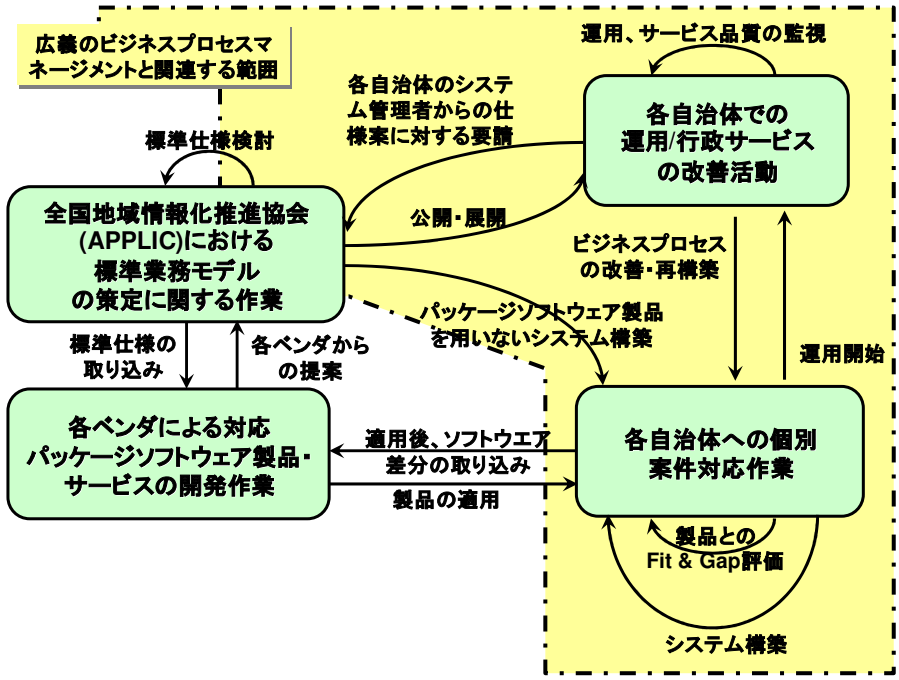


図３．３．２　地域情報プラットフォームにおけるビジネスプロセスマネージメント

図３．３．２において、一点鎖線で囲まれた部分は、各自治体、関連団体により実施されるモデリング（分析）～再設計等の一連のビジネスプロセスマネージメントに関連した作業に該当する。例えば、「各自治体での運用/行政サービスの改善活動」においては、先の図３．３．１の BPM 推進フレームワークのうち、「０．着眼・始動」「１．モデリング」「４．業務運用」「５．運用最適化」「６．新たな着想」等が

相当すると考えられる。これに対して、「各自治体での個別案件対応作業」では、「２．再設計」「３．実装・配備」等が相当すると考えられる。

この図３．３．２で明らかな様に、地域情報プラットフォームでは一般的なビジネスプロセスマネージメントのフレームワークとは大きく２つの点で違いが存在する。一つは「全国地域情報化推進協会(APPLIC)における標準業務モデルの策定に関する作業」であり、他方は「各ベンダによる対応パッケージソフトウェア製品・サービスの開発作業」である。

通常の BPM 推進フレームワークでは個々の自治体、企業内、もしくは特定の関係自治体群、企業群との関係に閉じて実施されることを指向しているのに対して、地域情報プラットフォームでは自治体固有の問題を除外した上で汎用的な標準業務モデルの構築を目指しており、これが従来のフレームワークに基づくアプローチ方法とは異なる点の一つである。

さらに汎用的な標準業務モデルに応じる形で、各ベンダがパッケージソフトウェア製品や、標準的なネットワーク上のサービスを提供する。そしてその製品や提供形態についてもさまざまであり、従来のフレームワークの前提とは異なる点が想像される。

加えてこれらの２つの差異に応じる形で、幾つかの付随的な問題も発生してくる。例えば、従来のフレームワークで重点的には討議されていない標準業務モデルの表現形式、モデルライブラリの管理の仕方、モデルライブラリの流通形式、そして各ベンダが提供するパッケージソフトウェア製品や、標準的なネットワーク上のサービスに対する準拠性に関する課題は、利用者が享受する利益を考慮した場合、従来のフレームワークに関する討議と同程度に大きな課題となる。つまり、以上に列挙した事項は、BPM 推進フレームワークが限定的に利用される場合においては課題とはならないが、地域情報プラットフォームを浸透させるに当ってはインフラストラクチャ化の様相を示すことになる。

以上を踏まえて、次節ではガイドラインに含まれるワンストップサービス連携に関する定義策定手順に基づき作成される生成物のフローの関係を示し、ここで対象としている BPM 機能を導入するための段取り等について概説する。

（２） BPM 機能導入に向けた生成物のフロー

前述の様に、地域情報プラットフォーム における BPM 機能とは、着眼、モデリング、再設計、実装・配置、業務適用、運用最適化、新たな着眼という一連のビジネスプロセス改善のライフサイクルを意味する広義のビジネスプロセスマネージメントではなく、狭義の意味で、再設計されたビジネスプロセスを自動実行するためのソフトウェア基盤を意味する。

図３．３．３は、別に定義されるワンストップサービス連携定義策定手順に基づき作成される生成物のフローの概要を記したものである。地域情報プラットフォーム において、BPM 機能を導入する場合、必要になる定義類はワンストップサービス連携の定義手順の最終工程で作成される生成物となる。そして、図３．３．３上で網掛けされた WS-BPEL 定義(BPM 機能への実装用) 部分が、これに相当する。ここでは、前工程から BPMN(Business Process Modeling Notation)の記述や、WS-BPEL(Abstract) の記述が、その入力となる。そして WSDL 定義 (BPEL 用) や、XML Schema による定義が、必要に応じて参照されることになる。特に、これらは各案件の詳細設計～実装工程に相当する成果物として提供され、通常は各社の MDA(Model Driven Architecture)ツールの出力として期待されるべきものである。

これに対して、図３．３．３のユニット関連ブロック図や、ワンストップ処理フローは、ワンストップサービス連携の定義手順では前工程に相当し、広義のビジネスプロセスマネージメントに相当する部分とも言える。この前工程の作業の主要部分を実施するに当り、それが図３．３．２の「全国地域情報化推進協会(APPLIC)における標準業務モデルの策定に関する作業」の中で実施するか、各自治体への個別案件対応作業の中で実施されるかは、策定するワンストップサービスの内容に依存する。

但し、どの様な業務標準モデルが定義されたとしても、各自治体への個別案件対応作業の中では、必ず一致と差異の特定を行なう「Fit&Gap 評価」を実施する必要がある。この為、広範囲に定義・利用されるワンストップサービスの様なもので、全国地域情報化推進協会(APPLIC)における標準業務モデルの

策定に関する作業で標準的なものとして策定されたとしても、各自治体への個別案件対応作業の中では必ず、検証が必要になってくる。

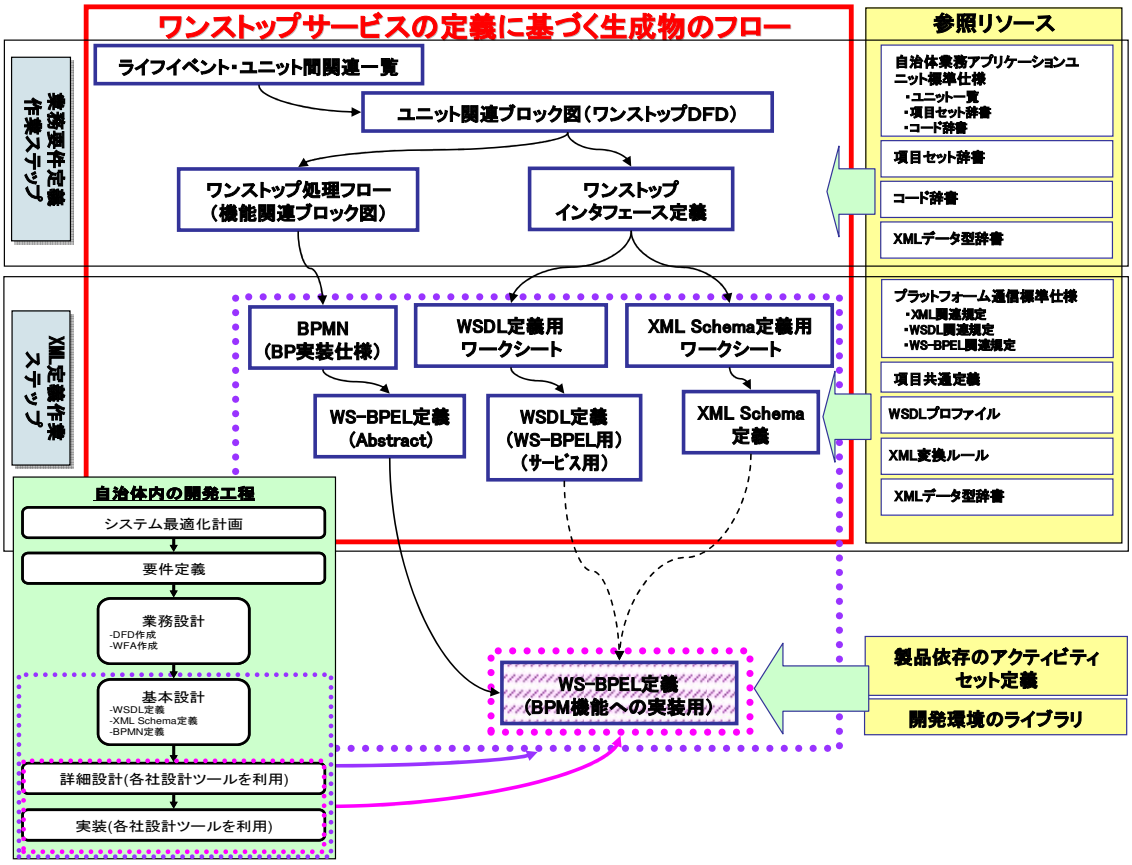


図３．３．３　ワンストップサービスの定義に基づく生成物のフロー概要

３．３．３　BPM 機能の製品選択時の留意点

BPM 機能の製品選定する場合の留意事項は次のとおりである。

（１）　プラットフォーム通信標準仕様の要件を満足するもの

プラットフォーム通信標準仕様では BPM 機能として WS-BPEL の V2.0 を採用している。従って、これを満足する製品でなければならない。

WS-BPEL は、OASIS により国際標準として勧告されたビジネスプロセスを記述するための言語ではあるが、製品間のポータビリティについて、完全に保証している訳ではない。そのため、地域情報プラットフォームの機能範囲として、WS-BPEL の V2.0 の定義機能のなかで個々の製品が概ね実装・サポートしているアクティビティのセットに限定するように選定している。これは、各種機能のカセッタブルを実現するためであり、結果として、その相互運用性についても概ね保証される。なお、製品選定の際は、WS-BPEL の V2.0 をサポートする製品の中でも、その実装範囲をマニュアル等でよく確認する必要がある。

（２）　セキュリティ要件に対して準拠、対応できるもの

地域情報プラットフォームを使い、各種のワンストップサービスを実現する場合、WS-BPEL の処理系の側面のみを注目して製品選定をすることは望ましいとは言えない。プラットフォーム通信標準仕様では、セキュリティに関する規定事項が定められており、選定するべき製品はこれらの事項に対しても準拠する、もしくは柔軟に対応できる必要がある。

（３）　MDA 等の開発環境と親和性が高く、開発・実行環境が統合化されたもの

前節で説明した様に、地域情報プラットフォーム における BPM 機能を導入する作業は、実際にはワンストップサービス連携の定義手順概要における最終工程に位置付けられる WS-BPEL 定義(BPM 機能への実装用）の生成作業に関連し、広義のビジネスプロセスマネージメントの一連の作業フローの末端に位置付けられる。そして、その前工程では、例えば BPMN 等の計算機可読のモデリング言語・方法論を用いて、モデリング作業が実施される。従って、製品を選択する際には、MDA(Model Driven Architecture)による開発環境、開発ツール等が揃ったもの、もしくはそれに準じて MDA と親和性の高いものを選択するべきである。通常の製品は概ね、対象とするものをグラフィカルに描画し、そのプロパティ等で実際の起動コードを追加する様になっている。そのため、通常の製品では、モデルによる援用の点で効果的なプログラミングが可能であるが、それに加えて完全に MDA 指向のものはビジュアルプログラミングが行なえるため、より効果的なモデリング～プログラミング作業を行なうことができる。

また、上記の様なモデリング言語・方法論については総じて、発展途上の段階にあり、必ずしもマルチベンダ環境間でのポータビリティの保証が取れている訳ではない。従って開発環境、開発ツールを選定する場合、ベンダが変更になったとしても組織体として統一した開発環境および開発ツールの選定が必要であるとともに、その製品の相互接続性を考慮することも必要である。これは開発環境、開発ツール間だけではなく、開発ツールと WS-BPEL の処理系・実行環境でも同様である。

（４）　サポートが安定して実績のあるもの

SOA(Service Oriented Architecture)、BPM(Business Process Management)の領域は、技術革新のスピードが速く、さまざまなベンダがそれぞれ特徴を持った製品を供給している。これに対して、実際のワンストップサービスの定義～再設計作業は比較的長い時間的スパンを要して実施され、発展する。その為、製品・技術とも安定して供給されていないと、継続的な実現することが出来なくなるリスクも存在する。製品を選定する上では、安定してサービス供給がされるものが必要となる。

（５）　システム運用ツールと連携の取れるもの

BPM 機能の実行系で障害等を検知した場合についての対処方法を、プラットフォーム通信標準仕様で規定している。プラットフォーム通信標準では、必要最小限の対処方法を規定しているため、より迅速かつ詳細な対応を図る場合には、運用管理ツールを導入し、これと連携した障害状況のロギング等により異常検知の強化をおこなうことが望ましい。

３．３．４　留意事項

製品選択時の留意事項以外の BPM 機能に関する留意事項、その他の種々の技術的なポイントを下記に簡単に記す。

（１）　トランザクション、補償動作に関する事項

後続の３．８節、メッセージ交換パターンと異常系処理に関するガイドラインで言及する様に、現在のプラットフォーム通信標準仕様では、トランザクションの処理機能等は特段規定されていない。これは、複数仕様の並立、並びにミドルウェアの普及の点からである。このため、障害が検出された場合の対処を、特別に定めている。BPM 機能を利用する場合、製品によってはトランザクションの処理機能を既に実装しているものも存在する。しかし、ワンストップサービスが複数の組織体に跨って実行される場合、必ずしもパートナーとなる組織体が採用する BPM 機能が、同等機能を有するとは限らない。従って、メッセージ交換パターン、異常系処理ではプラットフォーム通信標準仕様で定める方式に従う必要が出てくる。

（２）　利用できるアクティビティのセットの限定に関する事項

上記（１）と同様な事項として、地域情報プラットフォームの現在のプラットフォーム通信標準仕様では、WS-BPEL V2.0 の内、利用できるアクティビティのセットを限定している。これは、前述の通り、カセッタブルを指向して製品間のポータビリティを保証するためである。しかし、同時に地域情報プラットフォームの現在のプラットフォーム通信標準仕様では、各製品が独自に提供しているアクティビティセット・機能を利用することは禁じていない。これは、実際の BPM 機能の実装においては、独自のアクティビティセットを含んだコーディングが為されることも十二分に想定されることによるものである。自治体内においては、複数ベンダから BPM 機能を持つ製品を導入する場合、独自のアクティビティセットが WS-BPEL 記述を移植する上で問題となることがある。この様なことから、製品の導入にあたっては独自アクティビティセットの利用を含めた拡張機能をどこまで利用するか、そのポリシーを統一的に定めておく必要がある。

（３）　地域情報プラットフォームの特有な事項

地域情報プラットフォームでは、ビジネスプロセス実施の依頼元に、そのビジネスプロセスのフロー制御について、規定事項が存在しており、プラットフォーム通信標準仕様の７章である「プラットフォーム通信仕様におけるメッセージ共通ヘッダ仕様」にて定義されるメッセージの共通ヘッダ部分にセットされる値に応じて、処理内容を動的に変更する等の規定が存在する。この為、WS-BPEL の記述が、複雑化する傾向がある。

複雑化に対する対処として、通常のプログラミング言語ではモジュール定義、サブルーチン定義による構造化という手段を用いるが、WS-BPEL V2.0 では、この様な概念を直接持たず、サービスの連鎖的な呼出と言う方法を用いて同等のことを実現する。従って、複雑な構造を持つものは、共通ルーチンとしてローカルに配置されるサービス化を行なうことで対処する必要がある。但し、このローカルサービス化により、処理性能の悪化の可能性も否定出来ない。従って、ローカルサービス化にあたっては、処理性能とのトレードオフを考慮して実装することが必要になる。

（４）　モニタリング機能との連携

地域情報プラットフォームでは、モニタリング機能はオプションである。しかしこの機能は、障害発生時の対応等において有効な機能でもある。なお、この機能は、将来、ミドルウェアなどのパッケージソフトウェア製品として調達されることも考えられるが、相当機能をアプリケーションとして作り込むことも可能である。（※その場合は、メッセージ Audit、プロセス Audit と呼ぶものを生成する必要がある。）BPM 機能において、モニタリング機能と連携するためのインタフェースをサポートし得ることは留意するべき事項である。

（５）　「リクエスト・レスポンス型同期型受領 Ack＋非同期型レスポンス」時の対応関係

地域情報プラットフォームで選択可能なメッセージ交換パターンの一つである「リクエスト・レスポンス型同期型受領 Ack＋非同期型レスポンス」時のリクエストの処理とレスポンスの処理を連携させる方式は、業務ユニットの機能の一部として実装しなければならない。この場合、2 つの対処方法が存在する。

- ［方式１］：非同期型レスポンスの応答メッセージの共通ヘッダ内の「RelatedTo」は、リクエストの要求メッセージの共通ヘッダ内の「MsgID」に相当する。従って、業務ユニット側で対応関係を確認する。
- ［方式２］：非同期型レスポンスの応答メッセージの共通ヘッダ内の「受付番号」は、リクエストの要求メッセージの共通ヘッダ内の「受付番号」に一致する。従って、業務ユニット側で対応関係を確認する。

図３．３．４は、要求側の業務ユニットに係わるメッセージ処理を模式化したもので、上記の方式１を記したものである。この場合、点線矢印で対応付けられている様に非同期型レスポンスの応答メッセージの共通ヘッダ内の「RelatedTo」は、リクエストの要求メッセージの共通ヘッダ内の「MsgID」に相当する。従って、要求側業務ユニットでは、これらの対応関係を取る必要がある。

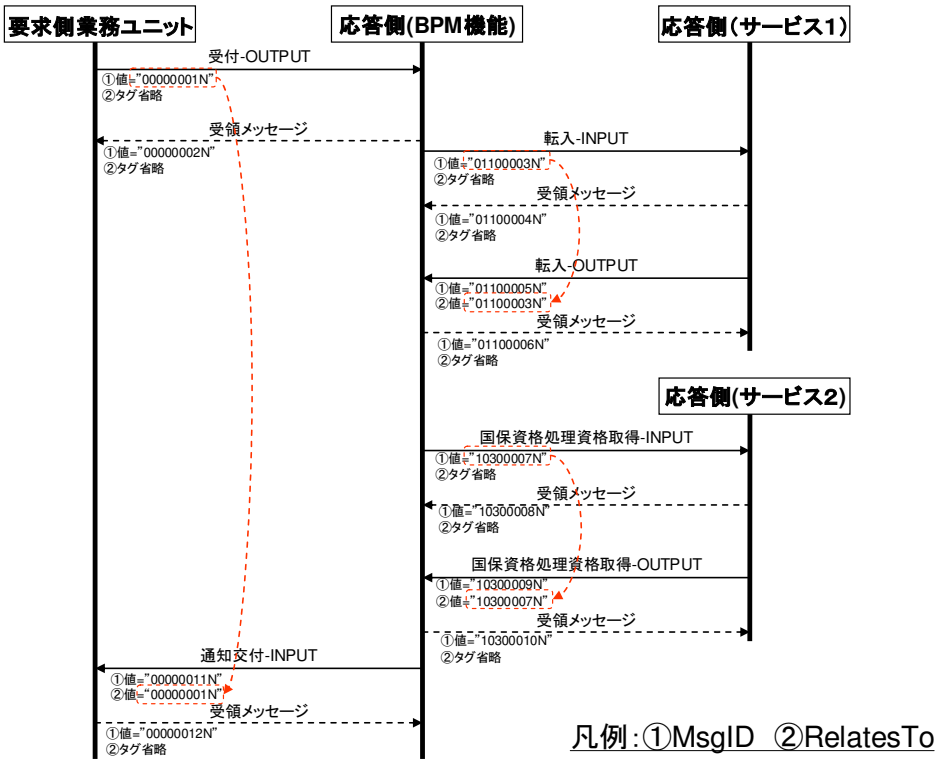


図３．３．４　リクエスト処理とレスポンス処理の関係付け

（６）　WS-BPEL が受信側となった場合の非同期型の受信タイムアウトの実装方式

プラットフォーム通信標準仕様の４．５節において、WS-BPEL が受信側となった場合、非同期型の受信タイムアウトに関して、WS-BPEL で実装することは規定外である。そのため、呼出元の業務ユニットでタイムアウト監視を行なうことを推奨している。

しかし、利用できるアクティビティセット以外のものを用いれば、WS-BPEL で実装することも可能である。必要に応じて検討されたい。

- WS-BPEL 処理系が pick アクティビティに対応できるものであれは、受信に WS-BPEL の receive ではなく pick と onMessage, onAlarm を組み合わせて使用することで、非同期メッセージのタイムアウトを WS-BPEL で記述することができる。
- WS-BPEL 処理系によっては、独自拡張機能によるタイムアウト検知や、ネイティブコードによる受信処理記述が可能なものも存在する。

ただし、これらの方法は地域情報プラットフォームの仕様範囲外であるため、相互接続性は保障されない。このことを考慮して、採用の有無を検討する必要がある。

（７）　排他制御に関する留意事項

WS-BPEL から、データの登録/更新/削除の操作を伴うサービスを利用するため、ある業務ユニットを呼び出す場合、その結果に応じて DBMS の同じ表に更新を実施する場合もある。例えば、図３．３．５には、ある WS-BPEL 内部で flow することで、処理を分割した結果、同じ DBMS 上の同じ表に対してサービス呼出を介して更新操作をする事例を記す。WS-BPEL 処理系側では、通常、この様な操作に対して排他制御は業務ユニット内部の責任で実施されることを想定している。その為、WS-BPEL 処理系側はロック等の排他操作を意識せずにサービスの呼出を行なう。

一般には、業務ユニット内部での排他制御の実施を基本とするが、それが困難である場合、WS-BPEL 処理系側でも特別な対処も考慮しなければならい。その対処方法の一つとして、業務ユニットの起動をシングルトン化し、直列化することも考えられる。但し、この場合、シングルトン起動を保障するため、同じサービスの重複呼出時に、後発のサービス呼出をエラーとして扱われ、障害として検出される可能性も存在する。

より確実化させるには、図３．３．５の例の様に flow を用いる場合、そこから呼び出される Activity が呼び出すサービスで、更新を伴う場合、同じ操作を呼び出してはならない、もしくは flow についての利用に制限を掛ける等も必要となる。

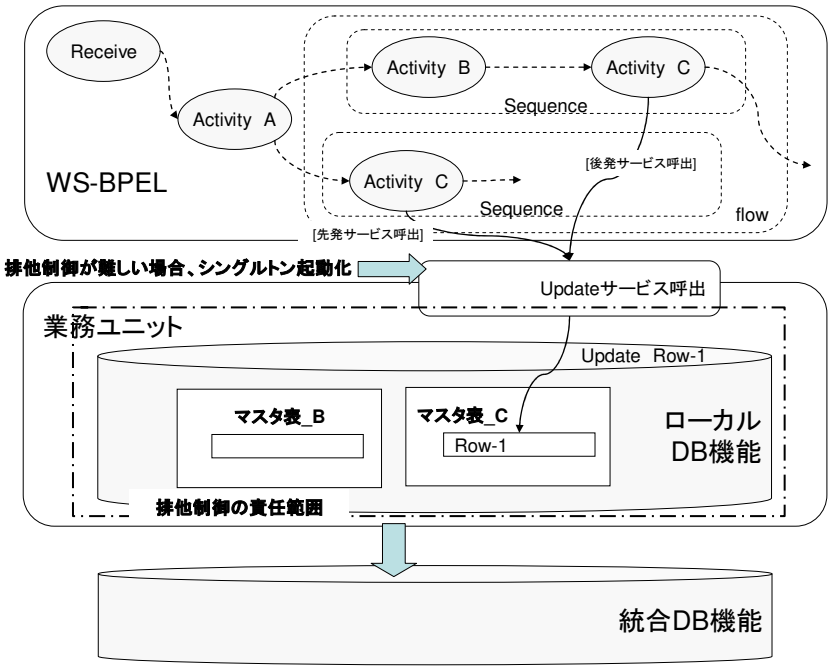


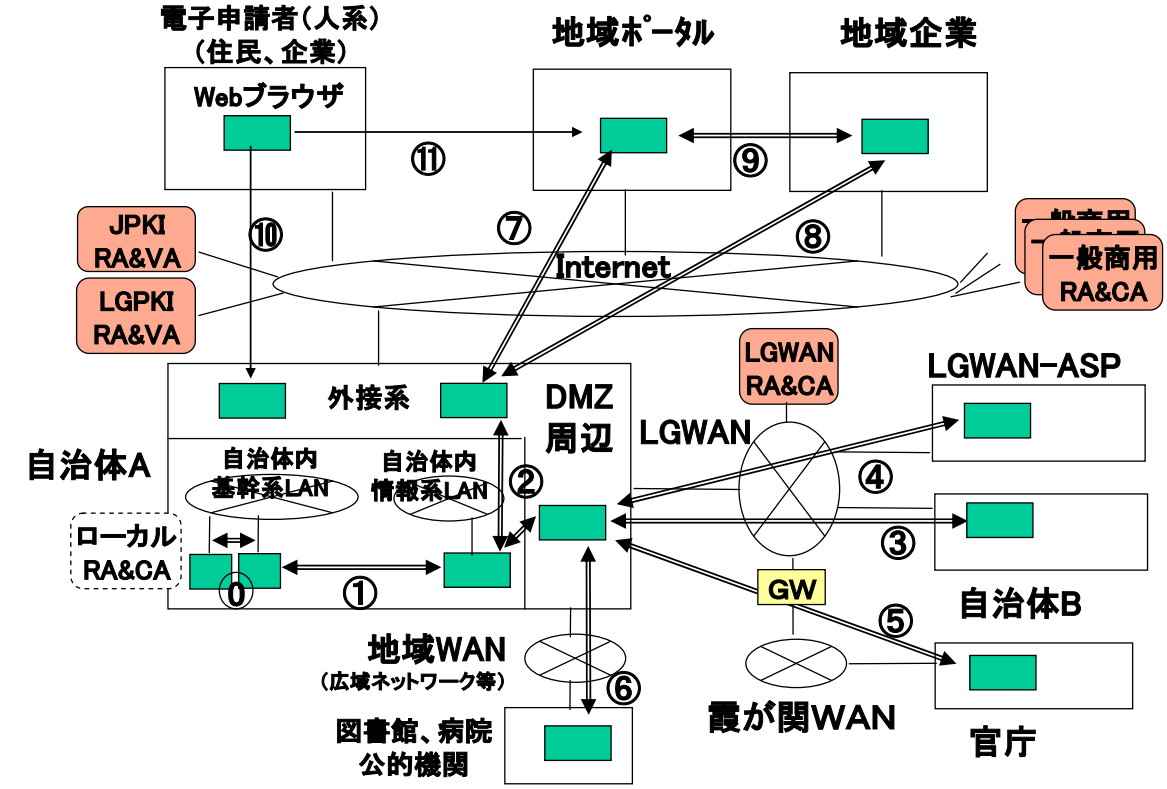
図３．３．５　排他制御に関する留意事項

３．４　セキュリティ

本節では、地域情報 PF に関連するネットワークと関連するセキュリティについて記載する。なお、PF 通信機能に関する認証・認可・セキュリティの仕様に関してはガイドライン３．５節を参照のこと。

３．４．１　地域情報 PF に関連するネットワーク

地域情報 PF に関連するネットワークと想定される接続パターンを図３．４．１に示す。



LGPKI：地方公共団体組織認証基盤
JPKI：公的個人認証サービス

LGWAN（総合行政ネットワーク）の解説：http://www.lasdec.nippon-net.ne.jp/lgw/lgwan.htm

図３．４．１　地域情報 PF に関連するネットワークと接続パターン

以下の表３．４．１にて、図３．４．１のネットワークと接続パターンの解説を行う。

表３．４．１　地域情報 PF に関連するネットワークと想定される接続パターンの整理

番号	接続の当事者	説明
(0)	同一 LAN 上の業務ユニット同士	自治体内の同一 LAN（基幹系 LAN,　内部情報系 LAN）に接続する業務ユニット同士のデータ交換やサービス連携する場合。
①	基幹系 LAN 上の業務ユニットと内部情報系 LAN 上の業務ユニット	基幹系 LAN と内部情報系 LAN 間の業務ユニット同士のデータ交換やサービス連携の場合。自治体では、住民情報の漏洩の防止策として、基幹系ネットワークと情報系ネットワークを分離している場合がある。自治体のポリシーにより、基幹系 LAN と内部情報系 LAN の接続可否が異なる。
②	自治体内業務ユニットと外部接続向けサーバ	自治体がインターネット側の外部接続を行うサーバと自治体 LAN 側の内部の業務ユニットとの連携の場合。自治体がインターネット側の外部接続を行うサーバは、ファイアウォールで仕切られた干渉エリア（DMZ）周辺に配置され、自治体 LAN 側の内部の業務ユニットとの連携を行う。自治体のポリシーにより、インターネット側と自治体 LAN 側の接続可否が異なる。
③	自治体同士の外部接続サーバ同士	LGWAN を介して自治体の LGWAN 外部接続サーバ同士が接続する場合、個々の自治体で外部接続サーバを LGWAN－ASP としての登録が必要。また、LGWAN 側の外部接続サーバは、DMZ 周辺に配置され、自治体 LAN 側の業務ユニットとの連携を行う。自治体のポリシーにより、LGWAN 側と自治体 LAN 側の接続可否が異なる。
④	自治体の外部接続サーバと LGWAN 上の ASP のサーバ	LGWAN 上で提供される ASP サービスと、LWGAN 側の外部接続サーバとの間でサービス連携する場合。ASP サービスと外部接続サーバ間の接続と、クライアント―ASP サービス間接続があり、現在は、自治体の LGWAN 端末から LGWAN-ASP サービスを使うものが多い。
⑤	自治体の外部接続サーバと官公庁の外部接続サーバ	自治体の外部接続サーバと官公庁の外部接続サーバが、LGWAN と霞が関 WAN を経由して接続する。現在は、GW サーバなどでネットワーク的に分断されていると想定される。
⑥	自治体の外部接続サーバと地域の公共団体のサーバ	自治体の地域 WAN 側の外部接続サーバと地域の公共団体のサーバがサービス連携する場合。
⑦	地域ポータルサーバと自治体サーバ	地域ポータルサーバと自治体のインターネット側の外部接続サーバがサービス連携する場合。
⑧	地域企業のサーバと自治体サーバ	地域企業のインターネット側の外部接続サーバと自治体のインターネット側の外部接続サーバがサービス連携する場合。
⑨	地域ポータルサーバと地域企業のサーバ	地域ポータルサーバと地域企業のインターネット側の外部接続サーバがサービス連携をする場合。

⑩	地域住民の PC と自治体のサービスサーバ	地域住民の PC と自治体のインターネット側の外部接続サーバが接続する場合。自治体の電子申請や情報提供サービス等を受ける。
⑪	地域住民の PC と地域ポータルのサービスサーバ	地域住民の PC と地域ポータルのインターネット側の外部接続サーバが接続する場合。自治体や民間の電子申請や情報提供サービスを受ける。

３．４．２　地域情報 PF におけるセキュリティの脅威と対策技術

地域情報 PF に関連するセキュリティ上の課題と対象技術について下記に記述する。

（１）なりすまし （他人に成りすましてのサービスの利用、文書の偽り）
（２）否認 （送信の否認、受信の否認、等）
（３）通信上の盗聴 （送信データの盗聴）
（４）通信上の改ざん （送信データの改ざん、破壊）
（５）データへの不正アクセス （データの改ざん、漏洩、削除、追加）
（６）システムへの不正侵入 （システム侵入、不正なサービス利用）
（７）ウィルス侵入 （感染したビジネス文書やウィルスそのものが侵入）
（８）DDoS 攻撃 （多量の送信データを送信してシステム性能劣化）
（９）プライバシ情報の不正利用、漏えい

DDoS(Distributed Denial Of Service)攻撃解説 :
http://www.ipa.go.jp/security/fy12/contents/crack/soho/soho/chap1/dos.html
（上記は、「情報セキュリティにおける脅威と対応技術の動向」ECOM 資料を参考とした）

自治体等の１つの組織がセキュリティポリシーを定め、管理対象全体のセキュリティを確保するための技術は、既存の PKI 技術や、ファイアウォール等のインターネットの技術として確立されている。

異なるサイト間（自治体間、自治体・民間間等）におけるセキュリティ上の課題を図３．４．２に示す。

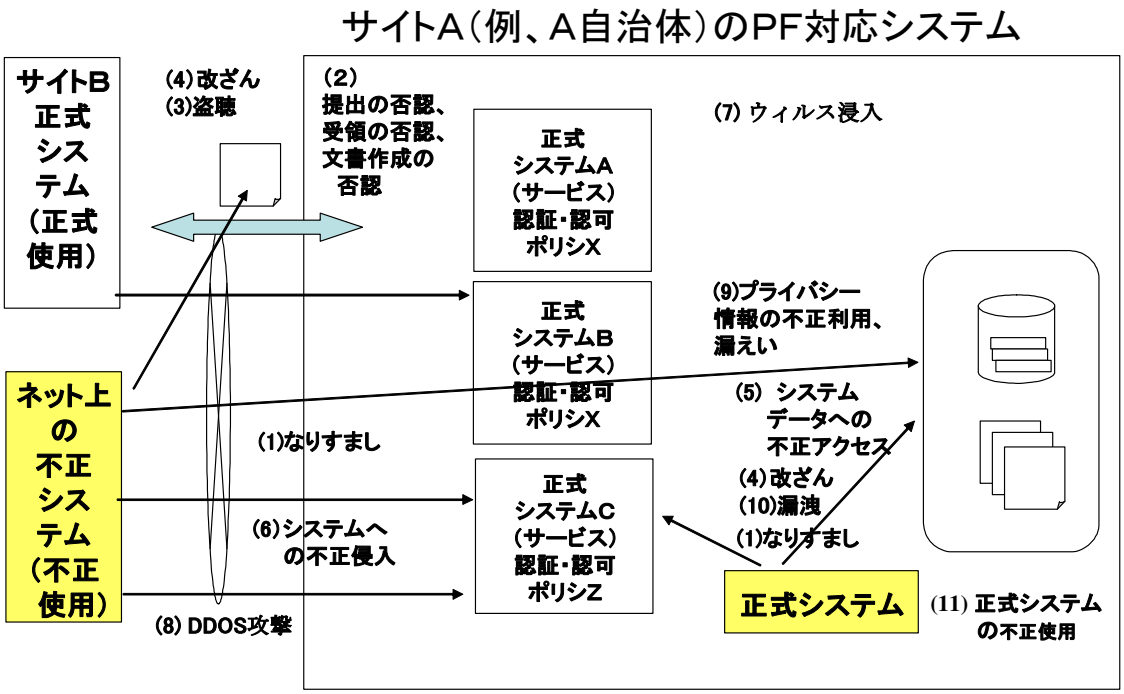


図３．４．２　異なるサイト間（自治体間、自治体・民間間等）におけるセキュリティ上の課題

異なるサイト間におけるセキュリティ上の課題に対する対策技術を下記に示す。

（a）認証技術
- ・認証（Authentication）技術とは、本人であることを判定し証明する技術である。
  （ここでいう本人とは、「クライアントのユーザ」「マシン」「サービス」の抽象的表現）
- ・対策対象：　(1)なりすまし、(6)不正浸入

（b）認可技術
- ・認可(Authorization)技術とは、認証されたユーザに対し、システムまたはアプリケーションの使用許可を判定することである。
- ・対策対象：　(3)盗聴、(10)漏洩

（c）秘匿性確保技術
- ・秘匿性確保技術とは、通信の内容を第 3 者が参照できなくする技術である。
- ・対策対象：　(3)盗聴、(10)漏洩

（d）電子署名と検証技術
- ・電子署名と検証技術とは、情報に対し電子的な署名を行い、情報の改ざん検知や署名者の確認を可能にする技術である。
- ・対策対象：　(1)なりすまし、(4)改ざん

（e）サービス認証・サービス認可連携技術
- ・サービス認証・サービス認可連携技術とは、ワンストップ用のサービス認証・サービス認可情報の管理/伝播方法およびそのモデルに関する技術で、シングルサインオンなどが実現できる。
- ・対策対象：　(1)なりすまし、(5)データの不正アクセス、(6)不正浸入

（f）プライバシ情報公開技術
- ・プライバシ情報公開技術とは、認証・認可の際やサービス処理におけるプライバシ情報の不正利用防止方法・交換許諾方法の技術である。
- ・対策対象：　(9)プライバシ情報の不正利用、(10)漏えい

(g) 監査証跡技術
・正式なシステムを不正に使用していないかの情報を統一的に収集し監査したい場合に、使用する技術である。
・対策対象： (11)正式システムの不正使用
(h)ウィルス対策技術
・感染したビジネス文書やウィルスそのもの浸入の検知や感染したデータの駆除等の技術である。
・対策対象： (7)ウィルス浸入
(i) DDoS 攻撃対策技術
・複数のサイトから特定サイトへの集中アクセスによるシステムダウンやサービス停止の攻撃への対策技術である。負荷集中の監視やネットワーク機器の IP フィルタリング設定等の技術である。
・対策対象： (8)DDOS 攻撃

３．４．３　地域情報 PF に関連する接続パターン別セキュリティ要件

地域情報 PF に関連する接続パターン別セキュリティ要件を下表で整理する。整理の観点は、下記。
・漏洩防止（通信上の盗聴）
・認証による対策（なりすまし対策、データの不正アクセス対策、システムの不正侵入）
・署名による対策（文書の改ざん防止、通信上の改ざん防止、否認防止）
（なお、ウィルス侵入や、DDoS 攻撃は、地域情報 PF 独自の課題でなく、一般的な課題であり、その対策は、インターネット対策技術として現在使用されているもであるため、整理の観点からはずした）
要件に対応した PF 通信機能に関する認証・認可・セキュリティの仕様に関しては、３．５節を参照。

表３．４．２　地域情報 PF に関連するネットワーク上の接続パターンとセキュリティ要件

番号	接続の当事者	セキュリティ要件
(0)	同一 LAN 上の業務ユニット同士	・漏洩：ローカル LAN なのでシステム全体で考慮 ・認証：認証技術
①	基幹系 LAN 上の業務ユニットと内部情報系 LAN 上の業務ユニット	・漏洩：秘匿性確保技術（SSL(TLS)(以降 TLS 省略)） ・認証：認証・認可技術（ベーシック認証、SSL クライアント認証、SSL サーバ認証）
②	自治体内業務ユニットと外部接続向けサーバ	・漏洩：秘匿性確保技術（SS） ・認証：認証・認可技術（ベーシック認証、SSL クライアント認証、SSL サーバ認証） ・署名：電子署名と検証技術（自治体からの外部公文書には署名(LGPKI)）
③	自治体同士の外部接続サーバ同士	・漏洩：秘匿性確保技術（SSL（LGPKI）） ・認証：認証・認可技術（ベーシック認証、SSL クライアント認証（LGPKI）） ・署名：電子署名と検証技術（自治体からの外部公文書には署名(LGPKI)）
④	自治体の外部接続サーバと LGWAN 上の ASP のサーバ	・漏洩：秘匿性確保技術（SSL（LGPKI）） ・認証：認証・認可技術（ベーシック認証、SSL クライアント認証（LGPKI)、SSL サーバ認証） ・署名：電子署名と検証技術（自治体から外部公文

		書には署名(LGPKI))
⑤	自治体の外部接続サーバと官公庁の外部接続サーバ	・漏洩：秘匿性確保技術（SS（LGPKI)） ・認証:認証・認可技術(SSL クライアント認証(LGPKI)) ・署名：電子署名と検証技術（自治体から外部公文書には署名(LGPKI))
⑥	自治体の外部接続サーバと地域の公共団体のサーバ	・漏洩：秘匿性確保技術（SSL（LGPKI)） ・認証：認証・認可技術 （SSL クライアント認証(LGPKI)、SSL サーバ認証) ・署名：電子署名と検証技術（自治体から外部公文書には署名(LGPKI))
⑦	地域ポータルサーバと自治体サーバ	・漏洩：秘匿性確保技術（SSL（一般商用 CA)) ・認証：認証・認可技術 （SSL クライアント認証（GPKI、SSL サーバ認証,一般商用 CA)) ・署名：電子署名と検証技術（自治体から外部公文書には署名(LGPKI)、住民からの電子申請には署名（JPKI))
⑧	地域企業のサーバと自治体サーバ	・漏洩：秘匿性確保技術（SSL（一般商用 CA)) ・認証：認証・認可技術 （SSL クライアント認証、SSL サーバ認証,（一般商用 CA)) ・署名：電子署名と検証技術（自治体から外部公文書には署名(LGPKI）、住民からの電子申請には署名（JPKI))
⑨	地域ポータルサーバと地域企業のサーバ	・漏洩：秘匿性確保技術（SSL（一般商用 CA)) ・認証：認証・認可技術 （SSL クライアント認証、SSL サーバ認証,（一般商用 CA) ・署名：電子署名と検証技術（自治体から外部公文書には署名(LGPKI)、住民からの電子申請には署名（JPKI))
⑩	地域住民の PC と自治体のサービスサーバ	・漏洩：秘匿性確保技術（SSL（一般商用 CA)) ・認証：認証・認可技術（自治体の独自の ID/PW 管理 or SSL クライアント認証、SSL サーバ認証（一般商用 CA)) ・署名：電子署名と検証技術（自治体から外部公文書には署名(LGPKI)、住民からの電子申請には署名（JPKI))
⑪	地域住民の PC と地域ポータルのサービスサーバ	・漏洩：秘匿性確保技術（SSL（一般商用 CA)) ・認証：認証・認可技術（地域ポータイル独自の ID/PW 管理、or SSL クライアント認証、SSL サーバ認証（一般商用 CA)) ・署名：電子署名と検証技術（自治体から外部公文書には署名(LGPKI)、住民からの電子申請には署名（JPKI))

３．４．４　地域情報 PF を実現する上での自治体ネットワークインフラの課題

本節では、自治体内やサイト間のサービス連携を実現する上で、自治体ネットワークに関する課題とその対策を記載する。

平成 18 年度 APPLIC で実施の自治体アンケート（母数 108 自治体）によると、自治体のネットワーク設置状況は下記であった。

(a)54%の自治体が、基幹系業務ネットワークおよび内部系業務ネットワークを物理的に分離している。
(b)(a)と答えた自治体の内、ほとんどの自治体が、基幹系業務ネットワークおよび内部系業務ネットワークは分離しているが、ファイアウォール、VLAN で接続可能としている。
(c)25%の自治体が、ネットワーク分離の条例などを制定している。
(d)97%の自治体が、セキュリティポリシーを策定している。

上記のアンケートを元にした、自治体内やサイト間のサービス連携を実現する上での課題とその対策案を以下に記載する。自治体への PF 導入の際はネットワークポリシを参照し、システム連携の方針を策定する必要がある。

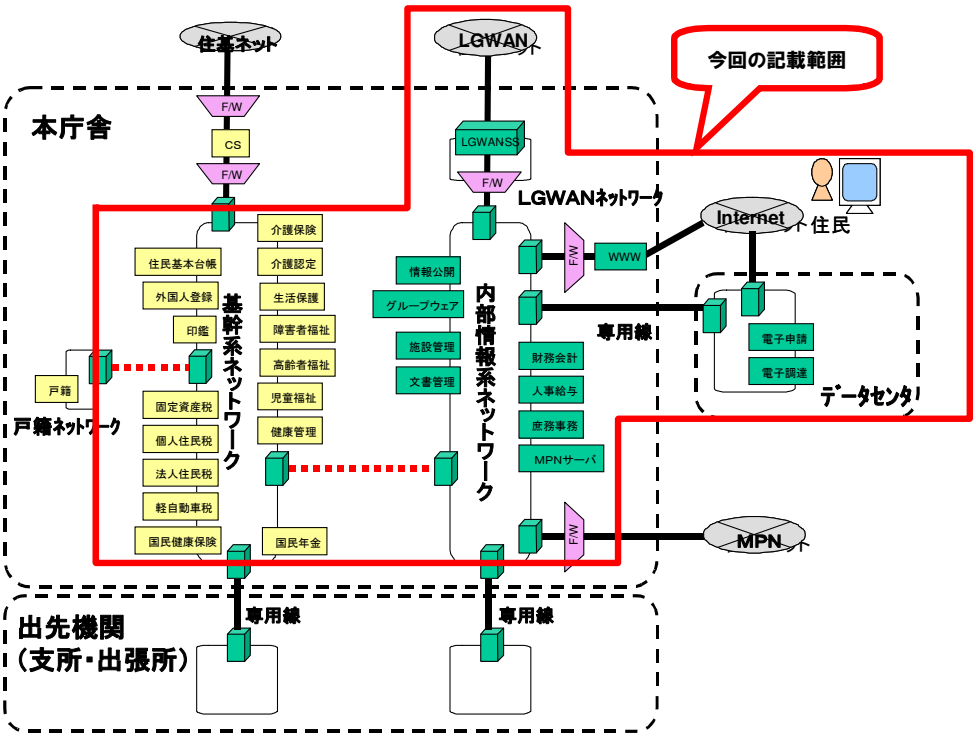


図３．４．３　現在の自治体のネットワークの構成のイメージ図

（１）LAN が異なる業務ユニット間のデータ交換（業務サービス連携）での課題と対策案

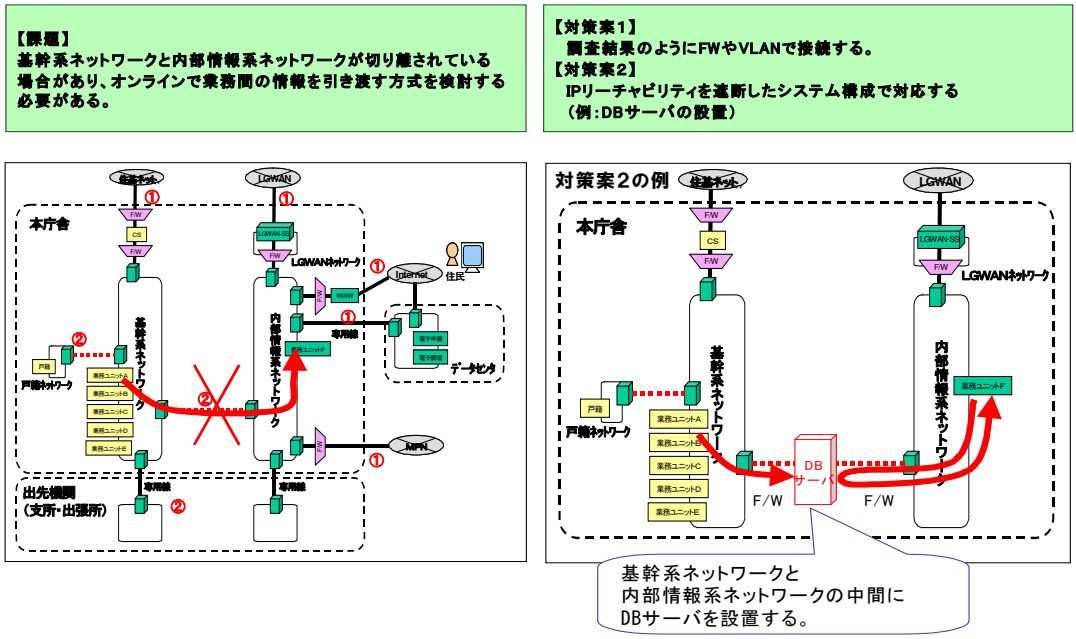
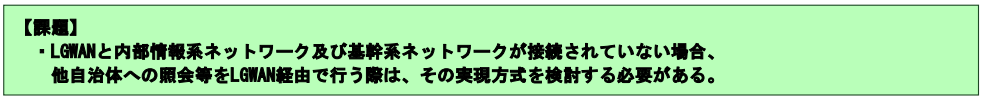


図３．４．４　LAN が異なる業務ユニット間のデータ交換（業務サービス連携）での課題と対策案

（２）自治体間連携での課題

現状の自治体業務を前提にした場合、システムが自動的に連携処理を行うようなニーズは少ないと思われるが、今後の新たな自治体間サービス連携として、職員が介在しないで自動的にデータを取り出してくるようなサービス連携も必要になる可能性がある。このようなサービス連携を実現する際は、ネットワークインフラの検討が必要となる。



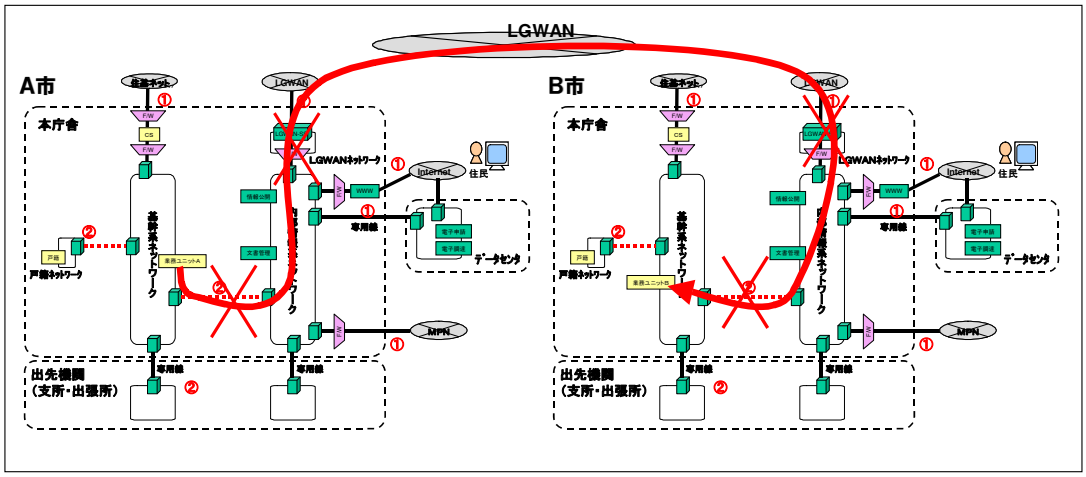


図３．４．５　自治体間連携での課題

３．４．５　地域情報 PF における自治体の外部接続の実現と検討ポイント

本節では、自治体の外部接続の実現イメージを記載する。自治体の外部接続のネットワークとシステム構成の例を図３．４．６に示す。

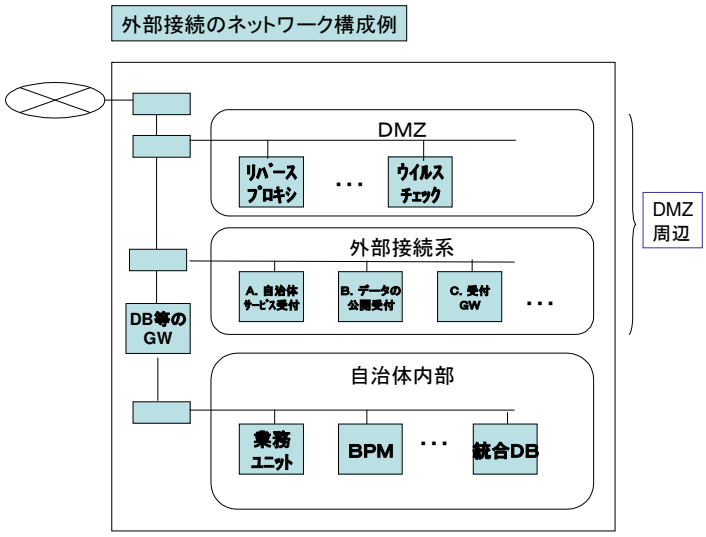


図３．４．６　自治体の外部接続の実現イメージ（ネットワークとシステム構成）

上図の実現例では、　ネットワークのセグメントとして、下記の 3 エリアを想定している。

（１）DMZ（緩衝エリア、外部と内部とが両方アクセスできるエリア）
・ファイアウォール、NAT（アドレス変換）、リバースプロキシ、プロキシ等を設置
（２）外部接続エリア（　外部接続サーバを配置）
・外部に公開されるサービスを実施するサーバを配置し、自治体内部とデータ交換する。
（３）自治体内部エリア
・自治体内部のサーバ群（情報系ネットワークと、基幹系ネットワークで分断されている場合がある）

【対策案】
・自治体内部のサーバと、外部接続サーバとデータ交換するサーバは、IP リーチャビリティを遮断するため、DB や、共有サーバを中継して、外部接続サーバとデータ交換する。また、同様に、自治体内部で、情報系ネットと、基幹系ネットで分断されている場合、DB や、共有サーバを中継してデータ交換を行なう。

汎用受付等システムの構築・運用に関する共通事項(平成 15 年 6 月 6 日改定、共通システム専門部会了承)を踏まえた、汎用受付システム構築の参考資料(調達編・共同方式の場合　最新 V1.2 版)が、下記に公開されているので参照してください。

（※１）~~URL：http://www.lasdec.nippon-net.ne.jp/rdd/olg15/02_kouchikuchoutatsu.pdf~~
現在は削除されています

次に、外部接続サーバと外部、自治体内部とのサービス呼び出し関連図を、図３．４．７に示す。なお、この図は、論理的なサービス呼び出し図となっており、物理的にネットワークが分断されている場合は、データ交換や、サービス呼び出しの中継を行う GW 機能を、両方のネットワークの中間に設置する必要がある。

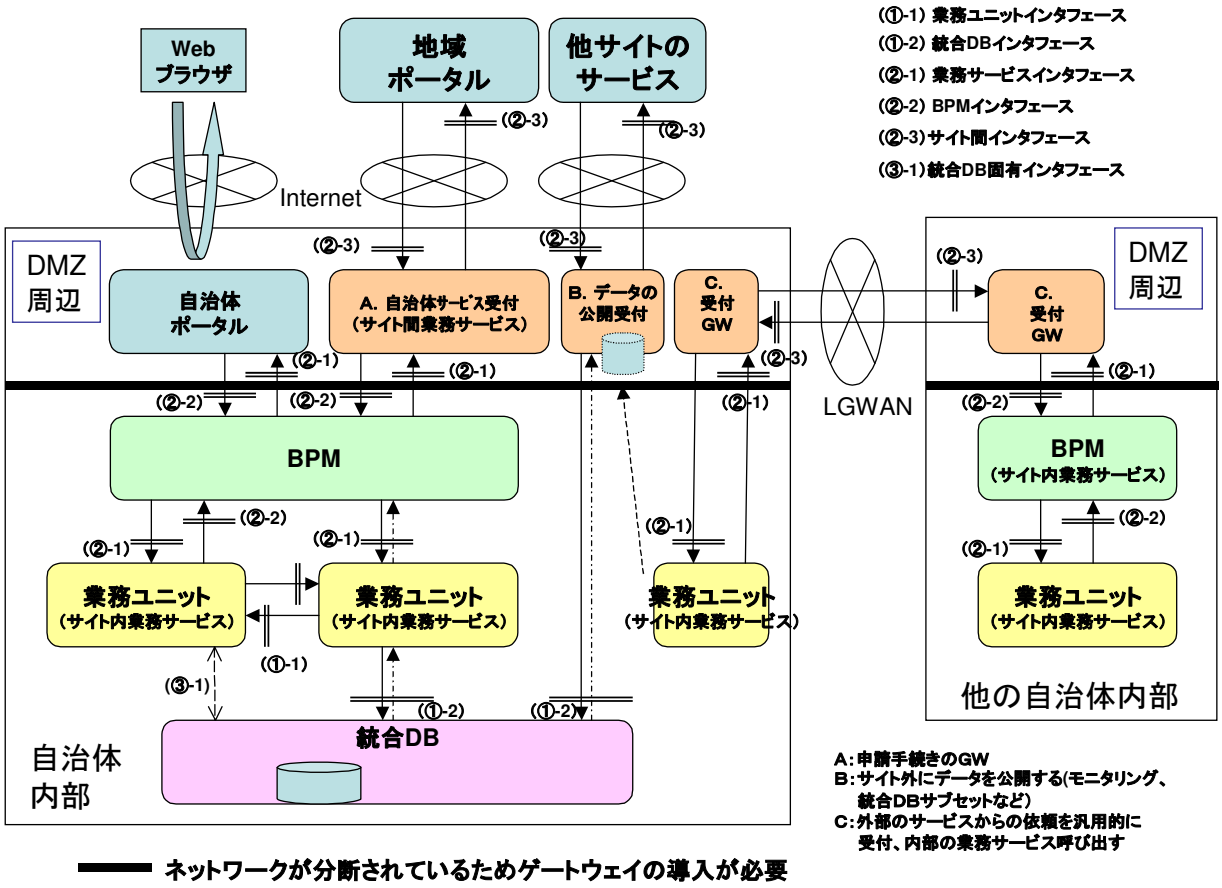


図３．４．７　自治体の外部接続の実現イメージ

３．５　PF 共通機能　認証・認可・セキュリティ

本節では、異なるサイト間（自治体間、自治体・民間間等）における認証・認可・セキュリティに関して、ユースケースを想定し説明する。

なお、自治体等の１つの組織がセキュリティポリシーを定め、管理対象全体のセキュリティを確保するための技術は、既存の PKI 技術や、ファイアウォール等のインターネットの技術として確立されている。自治体内の、電子申請における申請者の認証、自治体内システムの職員端末とサーバマシン間のセキュリティに関しては、既に自治体内で採用されている仕様があり、PF 仕様としては、特に規定していない。

３．５．１　サイト間システム連携におけるユースケースとセキュリティ要件

本節にて、PF に基づくサイト間のシステム連携に影響を与えるセキュリティ要件について、ユースケースを考慮しながら記載する。

（１）地域情報 PF 全体システムにおける電子申請ユースケースとセキュリティ要件

図３．５．１は、地域情報 PF で想定する、申請者、地域ポータル、自治体、連携する自治体、民間のシステム連携のユースケースである。本ケースでは、地域ポータル、自治体、連携する自治体、民間を、サイトとみなした。

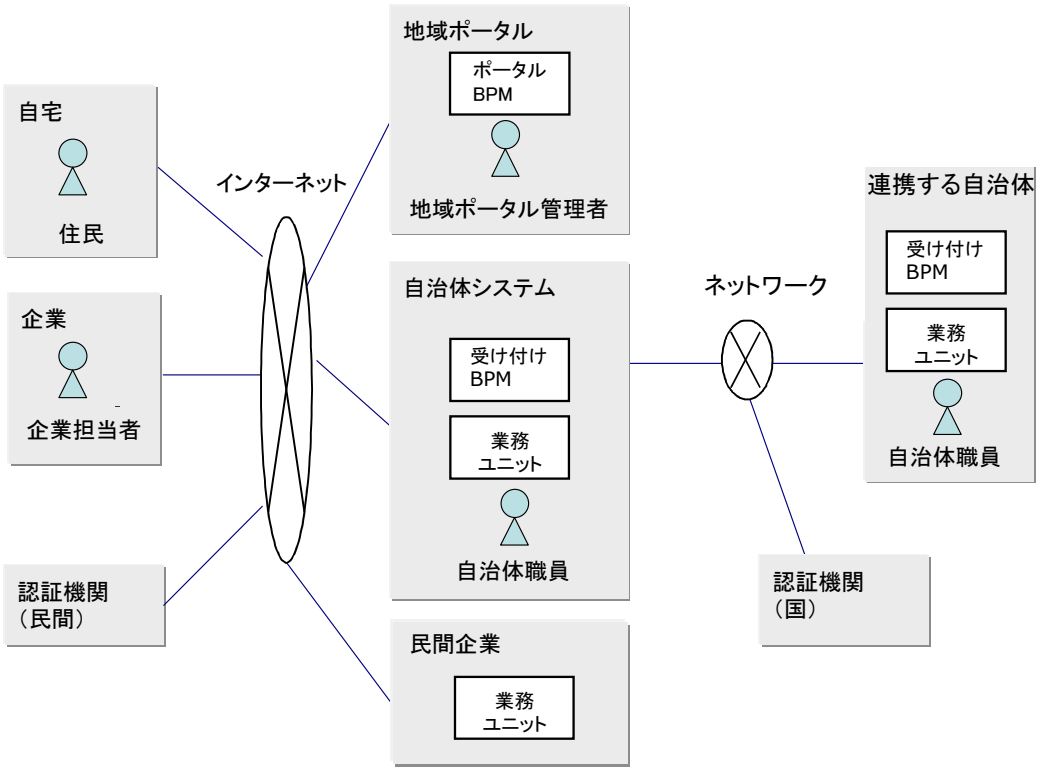


図３．５．１　地域情報 PF で想定するシステムのユースケース例

図３．５．１のユースケースにおいて、主に次の４つの業務処理を想定した。

１）申請者による申請書の送信
２）申請者による進捗状況の問合せ
３）自治体から他自治体への照会

４）自治体による公文書の配布

各業務処理のシーケンス図を図３．５．２に示す。

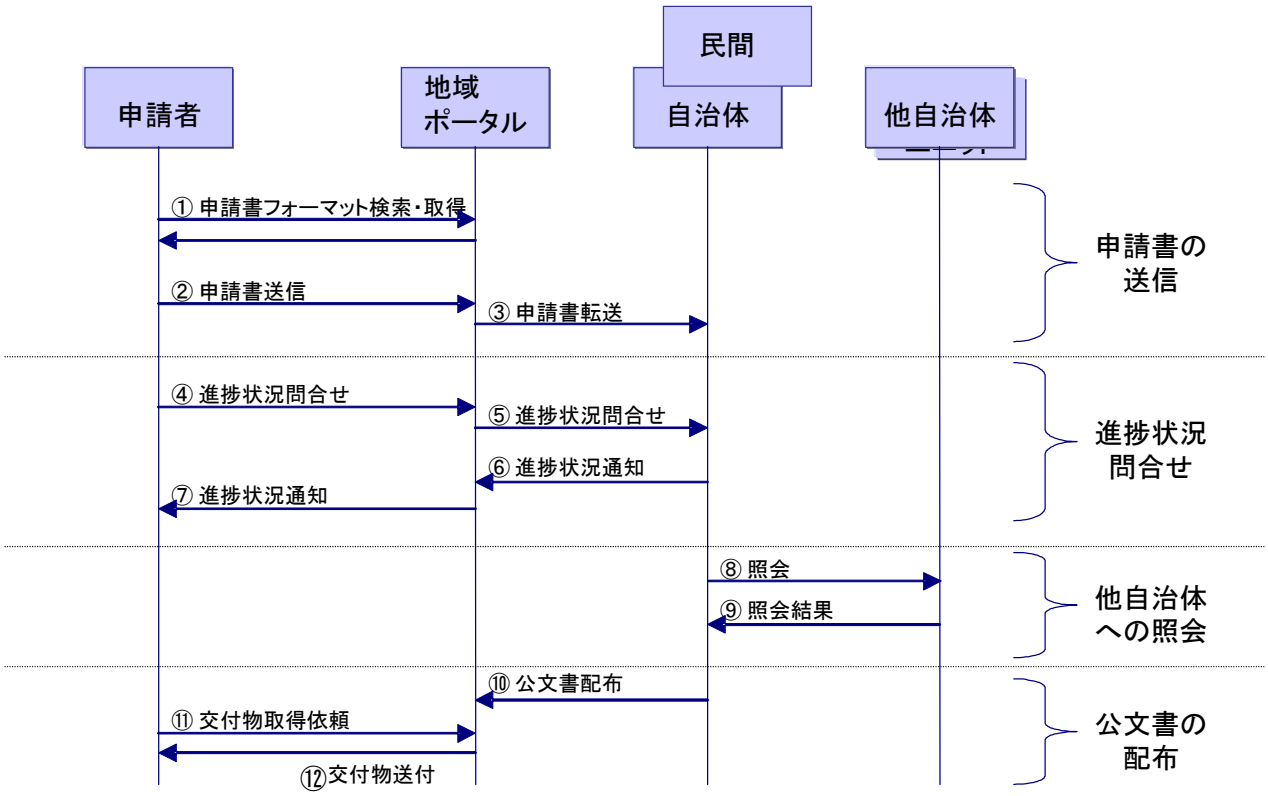


図３．５．２　地域情報 PF のシステム連携で想定する処理の流れ

上記図で想定するセキュリティ要件について、以下、記載する。

（a-1）データは他の人に読み取れられないようにすること　：　秘匿性確保
（a-2）データの改ざんが検出できること　：　改ざん防止、なりすまし防止
（a-3）他のサイトからの処理依頼において依頼元を認証できること　：　なりすまし防止
（a-4）正式システムの不正使用を監査できること　：　不正使用
（a-5）複数サイトとの間の連携処理に認証や認可処理の一元化が必要な場合、
　　認証認可の連携ができること　：　改ざん防止、なりすまし防止
（a-6）複数サイトとの間の連携処理に認証や認可処理の一元化が必要な場合、
　　認証のための他サイトへのプライバシ情報公開制御ができること：
　　プライベート情報の不正利用や漏洩、なりすまし、不正浸入

次に、ユースケースの各場面別のセキュリティ要件と対応技術要件を示す。

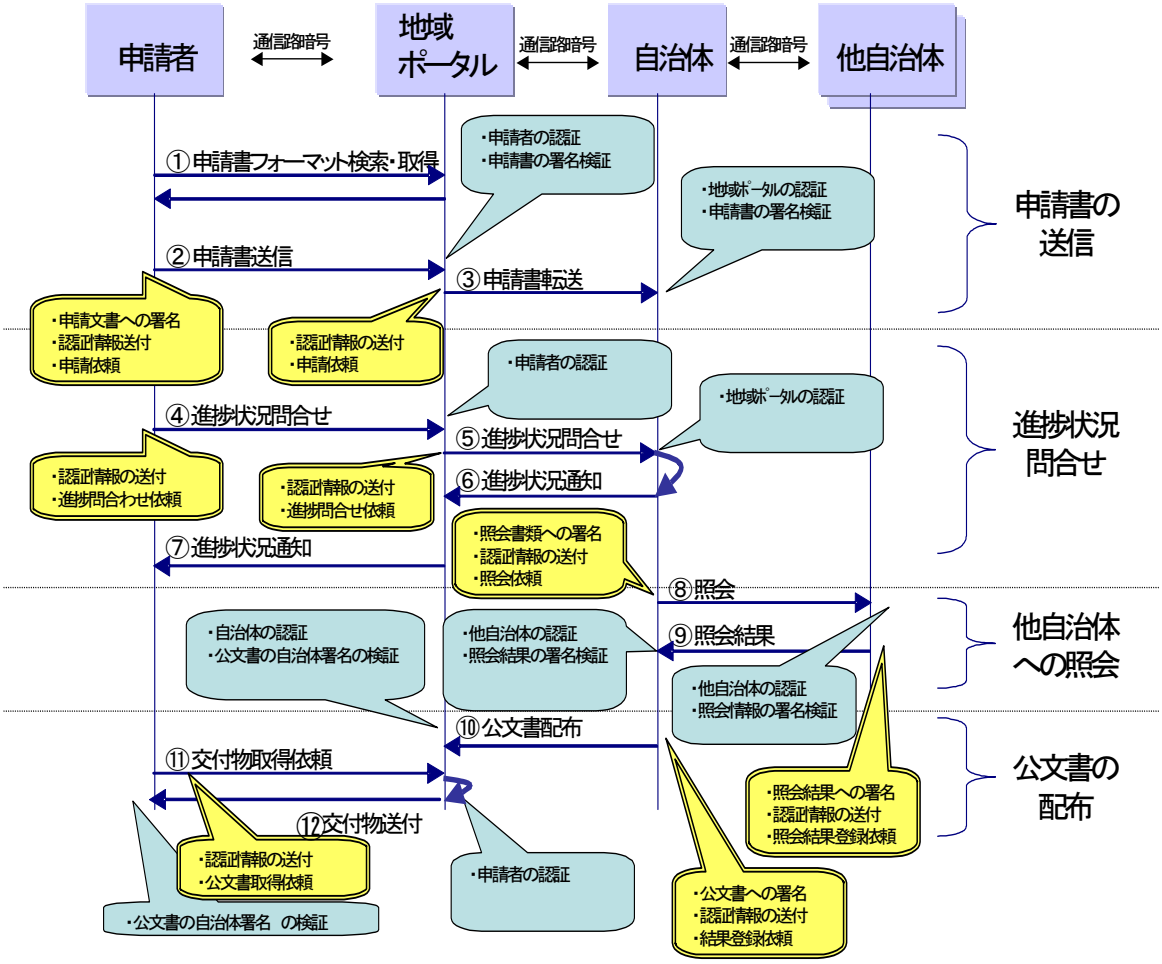


図３．５．３　地域情報 PF におけるセキュリティ要件と技術要件

図３．５．３における、各場面別のセキュリティ要件と対策上使用する認証・認可・セキュリティ仕様について表３．５．１にまとめる。

表３．５．１　地域情報 PF におけるセキュリティ要件と技術要件

	ユースケース	セキュリティ要件（A⇒B B の側）	採用する PF 仕様
①	申請書フォーマットの検索と取得	（PF 標準規定の範囲外）	（PF 標準規定の範囲外）
②	申請者⇒地域ポータルへの申請書送信	申請者の認証	（PF 標準規定の範囲外）
		申請書の改ざん防止	PF 本人電子署名・検証仕様
		申請書の伝送路上の秘匿性確保	PF 秘匿性確保仕様
③	地域ポータル⇒自治体（民間）への申請書転送	地域ポータルサイトの認証	PF サイト認証仕様
		申請書による申請者の認証	PF 本人電子署名・検証仕様
		申請書の改ざん防止	PF 本人電子署名・検証仕様
		申請書の伝送路上の秘匿性確保	PF 秘匿性確保仕様

④	申請者⇒ 地域ポータルへの 進捗問合せ（同期）	申請者の認証	（PF 標準規定の範囲外）
		問合せの伝送路上の秘匿性確保	PF 秘匿性確保仕様
⑤	地域ポータル⇒ 自治体（民間）への進捗問合せ（同期）	地域ポータルサイトの認証	PF サイト認証仕様
		問合せの伝送路上の秘匿性確保	PF 秘匿性確保仕様
⑥	自治体（民間）⇒ 地域ポータルへの 進捗状況通知（同期）	問合せ結果の秘匿性確保	PF 秘匿性確保仕様
⑦	地域ポータル⇒ 申請者への 進捗状況通知（同期）	問合せ結果の伝送路上の 秘匿性確保	PF 秘匿性確保仕様
⑧	自治体⇒他の自治体への照会	依頼元自治体の認証	PF サイト認証仕様
		照会情報の認証	PF 自治体組織電子署名・検証仕様
		照会情報の改ざん防止	PF 自治体組織電子署名・検証仕様
		照会情報の伝送路上の秘匿性確保	PF 秘匿性確保仕様
⑨	他の自治体⇒自治体への照会結果	依頼先自治体の認証	PF サイト認証仕様
		照会結果の認証	PF 自治体組織電子署名・検証仕様
		照会結果の改ざん防止	PF 自治体組織電子署名・検証仕様
		照会結果の伝送路上の秘匿性確保	PF 秘匿性確保仕様
⑩	自治体⇒ 地域ポータルへの 公文書配布	公文書配布元の自治体の認証	PF サイト認証仕様
		公文書の認証	PF 自治体組織電子署名・検証仕様
		公文書の改ざん防止	PF 自治体組織電子署名・検証仕様
		公文書の伝送路上の秘匿性確保	PF 秘匿性確保仕様
⑪	申請者⇒ 地域ポータルへの 公文書取得依頼(同期)	申請者の認証	（PF 標準規定の範囲外）
		公文書の伝送路上の秘匿性確保	PF 秘匿性確保仕様
⑫	地域ポータル⇒ 申請者への 公文書送付(同期)	公文書の認証	PF 自治体組織電子署名・検証仕様
		公文書の改ざん防止	PF 自治体組織電子署名・検証仕様
		公文書の伝送路上の秘匿性確保	PF 秘匿性確保仕様

（※）「(PF 標準規定の範囲外)」：要件に対応するセキュリティ仕様を採用するが、PF の通信に影響しないため、サイト独自に決定して良いことを示す。但し、サービスサイトの申請者や職員の認証の技術自体は、PF 標準の対象外で、既存の認証技術で実施してかまわないが、認証された後、そのサイトから他のサービスサイトへ PF 通信を行なう際、認証された情報を引継ぎ、サービスサイトでの認証に使用することがある。このため、サービスサイトの申請者や職員の認証が、正しく、且つ、安全に行なわれる必要がある点に注意が必要である。

３．５．２　自治体職員の認証・認可方式と地域情報 PF との関係

本節では、特に、自治体職員の認証・認可方式と地域情報 PF との関係を記載する。

（１）業務ユニットの職員認証（PF 標準規定の範囲外）

本節では、自治体職員の認証・認可方式と地域情報 PF との関係について記載する。　大別すると下記の 2 種類の認証方式が採用されている。

(a)　個別業務システム毎のログイン（認証と認可）

(b)　Web アプリケーションや統合ソフトウェアによるシングルサインオン（SSO）

現在、認証方式も個別のベンダーソリューションを採用している場合が多い。

地域情報 PF の標準仕様では、特に、自治体内部の職員の認証や SSO に関する規定はない。

（２）サイト間の職責認証の処理方式（提言）

地域情報 PF の標準仕様では、特に、自治体内部の職員の認証や SSO に関する規定はない。今後、自治体間で職員による問合せ型の文書照会型のサービス（自治体間での処理依頼と結果通知）が、職員の職責で実施される場合が想定される。

自治体間問合せは、依頼元自治体内の業務ユニットにて、職員を認証し、起案、承認後、この職員が依頼文書に対し LGPKI で職責署名（PF 自治体組織電子署名・検証仕様）し、他自治体へ依頼文書を送付する。受けた自治体は LGPKI の証明書検証により、依頼元の職責を確認（PF 自治体組織電子署名・検証仕様）し、依頼された照会業務を職員が業務ユニットで実施し、回答文書を作成し返す。

この場合のシステム処理フローについて図３．５．４に示す。

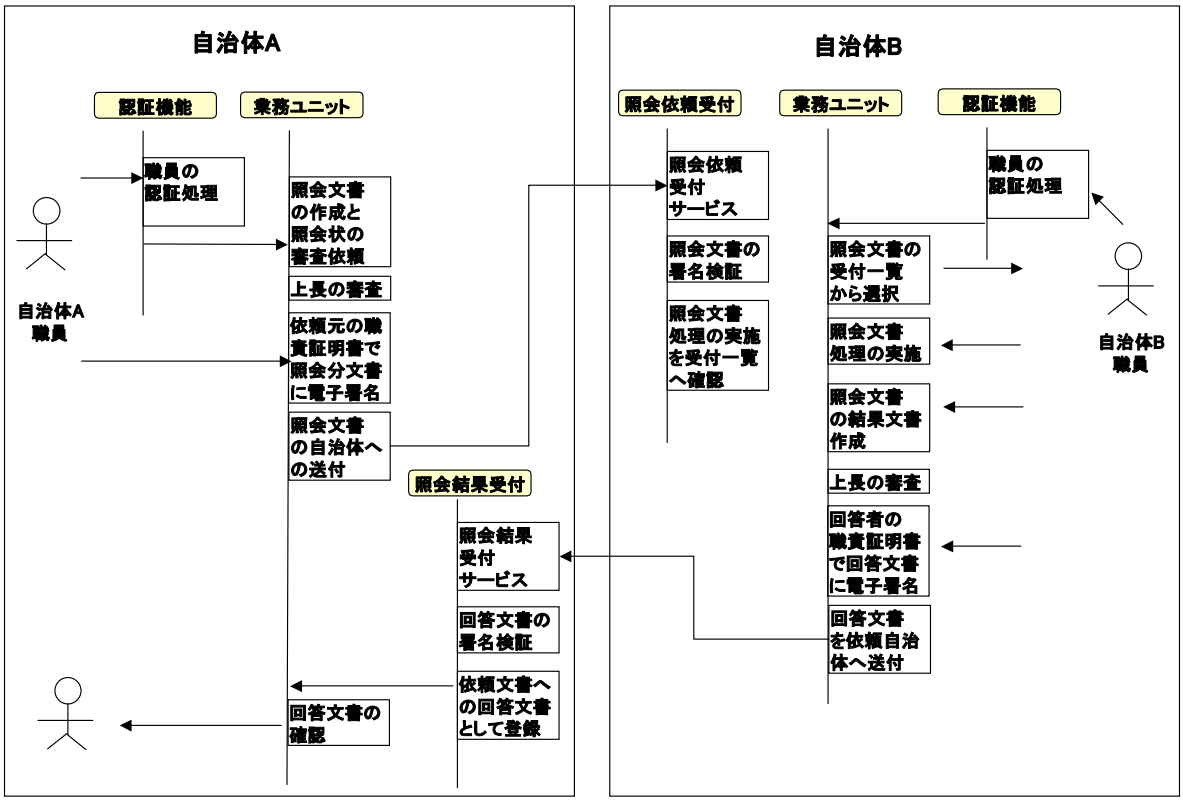


図３．５．４　サイト間の自治体職員の職責認証の処理方式

３．５．３　住民の認証・認可方式と地域情報 PF との関係

本節では、公開されている Web サイトへのログイン時に行われる、住民に対する認証・認可の考え方、及びその延長で PF 通信を使用したサービス使用における PF 通信の認証・認可の考え方について記載する。

（１）住民が単一 Web サイトを利用する場合の認証と認可（PF 標準規定の範囲外）

住民がインターネットを使って単一サイト内のサービスをうける場合は、例えば、図３．５．５に示す Web アプリケーションとしての認証・認可の方式で対応する。

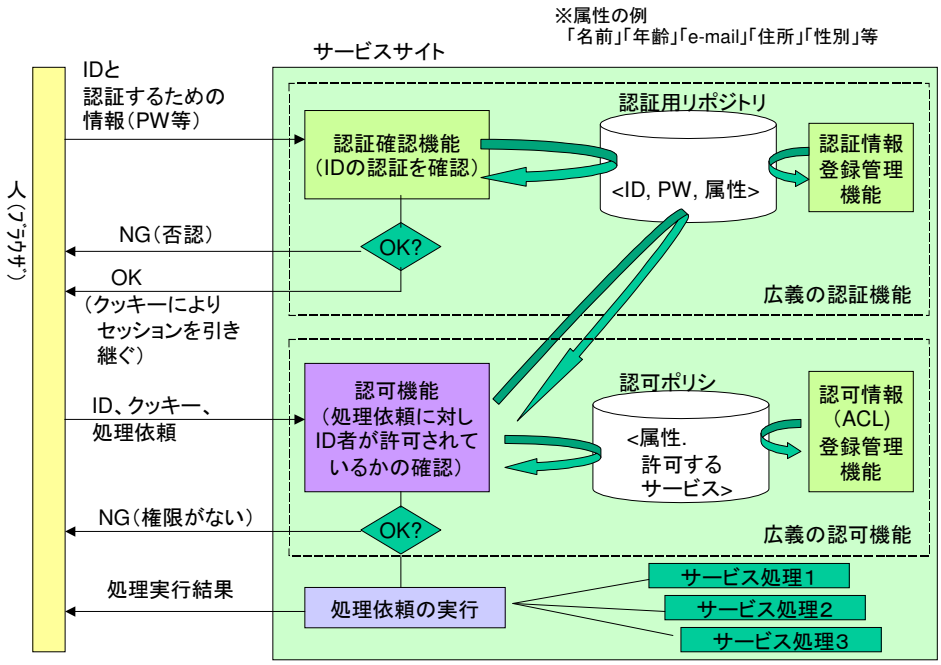


図３．５．５　Web アプリケーションにおける認証・認可の処理方式

（２）住民がポータルを経由して複数 Web サイトを利用する場合の認証と認可

地域情報 PF では、住民が地域ポータルで認証を受け、自治体や民間の地域サービスを使う場合が想定される。

実現方式の１つを下図に示す。この方式では、地域ポータルに、PF 通信で接続する自治体等のサービスの認証処理のための ID と認証情報（パスワード等）を地域ポータルに登録する必要がある。この実現方式には、いくつか課題があり、対策については、３．５．７節に示す。

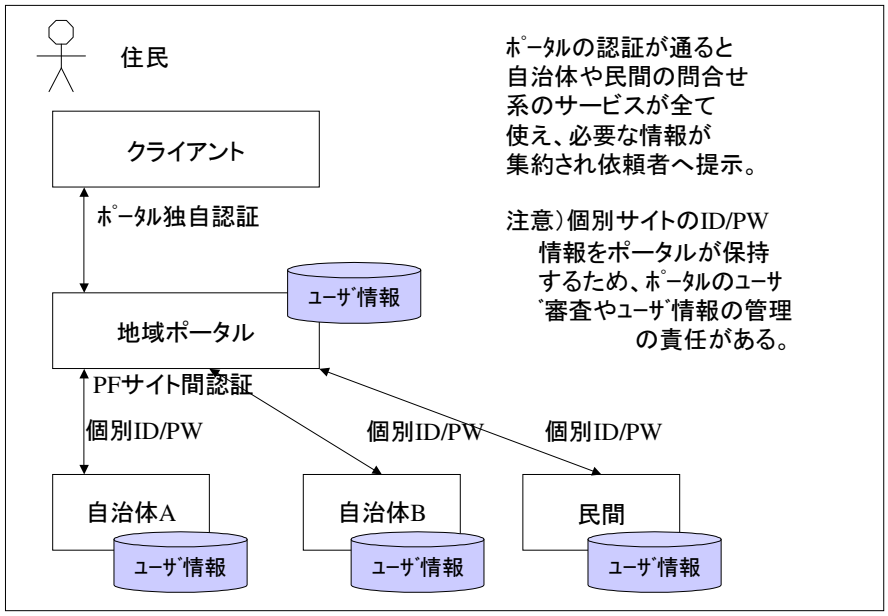


図３．５．６　住民がポータルを経由して複数 Web サイトを利用する場合の認証・認可実現方式例

図３．５．６の地域ポータルと自治体の間は、PF 通信による自治体・民間が公開するサービスの呼び出しとなる。自治体の外接系サーバのサービスを呼び出しする場合の認証・認可処理の例を、図３．５．７に示す。

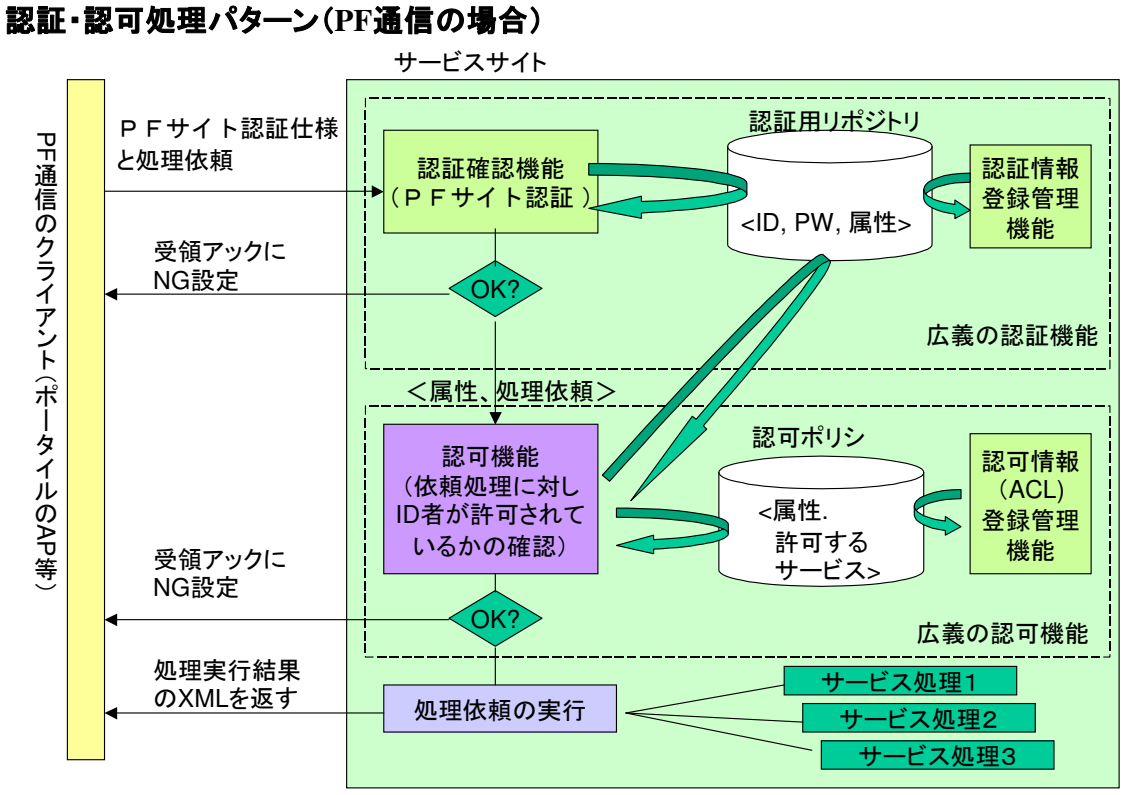
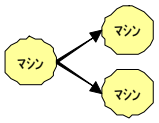
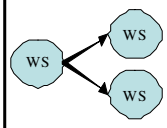
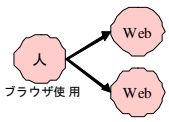
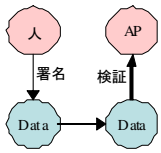


図３．５．７　PF 通信における認証・認可の処理方式

３．５．４　地域情報ＰＦにおける「認証」機能の導入

図３．５．９に、認証の要件を記載する。地域情報 PF で想定される認証の対象として、「人（Web ブラウザ等の AP 画面を操作する人）（PF 標準規定の範囲外）」「サービス（Web サービス）」「文書」「マシン」が想定される。下図に認証に関する要件の定義例を示す。

次に、上記の認証のユースケースに対する、実現技術を図３．５．８に示す。

区分	技術要件	ユースケース	自治体内	民間－自治体 自治体－自治体	一般用語	証明書･運用
マシン系 L1	･マシーンからの接続要求の認証 ⇒ **マシン認証**	 ･L1	**標準化済み:** ･SSL(TLS)認証 ･HTTPベーシック認証	**標準化済み:** ･ SSL(TLS)認証 ･HTTPベーシック認証	･サーバ認証 ･クライアント認証	･自治体内: 独自 ･自治体間: LGPKI ･民間と自治体: 民間
サービス系 L2	･依頼元のサービス認証 ⇒ **サービス認証**	 ･L2−1 ･L2−2	**標準化対象:** 申請を各ユニットへ: ･WSS (ID/PW) ･独自	**標準化対象:** 地域ポータルから自治体へ ･WSS(ID/PW) ･WSS+SAML(★)	･サービス認証	･自治体内: 独自 ･自治体間: LGPKI ･民間と自治体: 民間
Web系 L3	･人に割付けたID/PW or証明書でブラウザからのアクセスを認証 ⇒**Webユーザ認証（PF標準規定の範囲外）**	 ･L3	**標準化対象外** 職員ポータル: ･クッキー(独自)	**標準化対象外** 住民が官と民へ: ･SAML (★) ･クッキー(独自)	･SSO (ｼﾝｸﾞﾙｻｲﾝｵﾝ)	･自治体内: 独自 ･自治体間: 未整備 ･民間と自治体: 未整備
文書系 L4	･データ作成者の認証 ⇒ **電文認証**	 ･L4−1 ･L4−2 ･L4−3	**標準化済み:** ･電子申請形式	**標準化済み:** 自治体間: ･電子申請形式 民間－自治体: ×標準化未だ	･署名検証 ･証明書検証	･自治体へ申請:JPKI ･自治体間: JPKI ･民間と自治体: （JPKIは使えない、新しい枠組みが必要）

（※）　用語の説明：

人　：申請者や自治体職員等の特定の個人を示し、人がブラウザや電子申請ソフトウェアを使って操作をすることを意味する。

WS　：Web サービスのサービス要求者や Ｗｅｂサービス提供者のアプリケーションを示す

Web：Web アプリケーションを示し、人が Web ブラウザで操作する対象アプリケーションを示す。

AP　：Data の署名を検証し、正しい署名のとき、その申請者の権限に基づき、処理を行なうアプリケーションを示す。

（★）異なる運用組織間での SSO の実現では、IDP（ID プロバイダ）の運営母体の不在が課題

図３．５．８　認証のユースケースと実現技術の関係

	#	誰(何)を (IDの保持者)	IDを何によって 認証し	IDに何の属性を 付与するか？
L1	マシン系	・サーバ （レスポンダ側） ・クライアント （リクエスタ側）	・SSLサーバ証明書 ・SSLクライアント証明書 ・HTTPベーシック認証	マシーンの分類: ・接続許可
L2	サービス系	・PF管理主体を同じくするサービス(AP) ・PF管理が異なるサービス(AP) ・異なるPFのサービス(AP)	認証連携: ・Webサービスセキュリティ ・XML署名、SAML　等	同一PF内: ・認証グループ 異なるPF間: ・管理団体
L3	Web系	・一般の人 ・職員(事務系) ・職員(システム系) ・民間サービスの人(事務系) ・民間サービスの人(システム系)	個人認証: ・電子証明書(PKCS＃12)認証 ・パスワード認証 ・ICカード認証や生体認証 シングルサインオン: ・シングルサインオン(SSO)技術	ロール: ・自治体内の住民 ・システム管理者(自治体) ・業務実施者(自治体) ・システム管理者(民間) ・業務実施者(民間)
L4	文書系	・申請文書 ・起案文書 ・交付文書	個人認証: ・電子証明書(PKCS＃12)認証 ・XML署名	ロール: ・自治体内の住民 ・システム管理者(自治体) ・業務実施者(自治体)

図３．５．９　認証の要件定義の例

（１） L1：マシン系（マシンの２点間の通信）で採用する認証技術（PF サイト認証仕様）

PF サイト認証仕様には、サイト間通信において、サービスのリクエスト側がサービス側の通信先マシン（サイト）を認証するサーバ認証と、サービス側がアクセスしてくるリクエスト側の PF 通信の通信元マシン（サイト）を認証する SSL クライアント認証とがある。PF 標準書では、PF サイト認証仕様として、下記の国際標準仕様を採用している。

異なるサイト間での PF 通信では、PF サイト認証仕様は必須であり、サーバ間での通信においては SSL(TLS)サーバ認証および SSL(TLS)クライアント認証を必須となっている。また、リクエスト側がサーバではなく PC などの場合には SSL(TLS)クライアント認証の代用として HTTP Basic 認証も可能とすしている。

○SSL サーバ認証

SSL(TLS)で暗号通信を実施する場合、必須でサポートされている機能である。

○HTTP Basic 認証

HTTP プロトコルで規定されているベーシック認証。ユーザ毎に、ID とパスワードをサーバ側で設定し、クライアントのアクセス時に指定し認証する方法。

メリット　：ほとんどの製品で対応している

デメリット：パスワードによる認証のため、セキュリティ強度が低い。また、パスワード管理のコストがかかる。

○SSL クライアント認証

SSL(TLS)クライアント認証により、Web サービスクライアントを認証する。SSL(TLS)クライアント証明書は Web サービスクライアントごとに発行する。

メリット　：　ほとんどの製品で対応している。パスワードに比べセキュリティ強度が高い。
デメリット：　クライアント証明書の配布、管理の方法を検討する必要がある。

（２）L2：サービス通信系で採用するサービス認証技術
〇サービス通信系の認証技術として、Web サービスセキュリティ（略称：WSS）技術等があるが、必要に応じて、L1,L4 等の技術で代用することが可能。

（３）L3：Web アプリケーション系の SSO で採用する認証技術　（PF 標準規定の範囲外）
〇クッキーによる SSO 技術
HTTP プロトコルの技術あるクッキー機能を使い、ID や認証情報の引継ぎを行う技術
同一ドメイン内の Web サイトで情報の引継ぎが可能。認証そのものには、ID/PW 等が使われる。
〇SAML による SSO 技術
国際標準規格である、SAML V2.0 を使用する。普及状況をみて採用の可否を検討する。

（４）L4:データ（文書）の作成者を認証する技術
〇XML Signature：　XML Signature を用いてデータに署名をする。
メリット　：　データを SOAP 以外のプロトコルで送信する場合でも適用可能。
デメリット：　署名方法の自由度が高く、相互接続性に懸念がある
（各種アルゴリズム、鍵の参照方法、署名タイプなど）。
クライアントに、XML 署名に対応したソフトウェアをインストールする必要がある。

３．５．５　「認可」の要件、ユースケース、実現技術（PF 標準規定の範囲外）

本節では、認可についての要件とユースケースを記述する。図３．５．１０が、認可の要件定義の例である。

許可の要件

IDに対する属性	認可の条件	許可の対象	認可の範囲
申請人	サービス時間内	・申請(CRUD)	・同じ自治体内
システム管理者(自治体)	何時でも	・システムモニタ(CRUD)	・同じ自治体内の全システム
業務実施者(自治体)	サービス時間内	・ビジネスプロセス状態(R)	・同じ自治体内のBP状態のみ
システム管理者(民間)	サービス時間内	・自治体受付システムの状態(R)	・管理対象自治体の受付システムのみ
業務実施者(民間)	サービス時間内	・自治体のXX文書(R)	・自治体内で正式承認されたもののみ

※C(作成)R(参照)U(更新)D(削除)の権限を表す。

図３．５．１０　認可の要件

次に、認可のユースケースを図３．５．１１に示す。

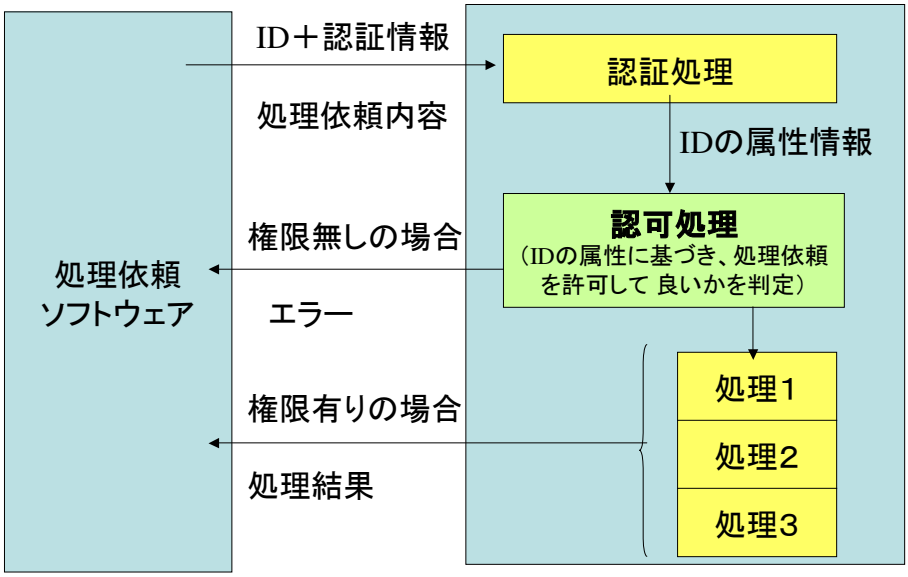


図３．５．１１　認可のユースケース

上記の認可のユースケースに対し、実現技術との関係を図３．５．１２に示す。認可は個別の技術でなく処理の考え方である。

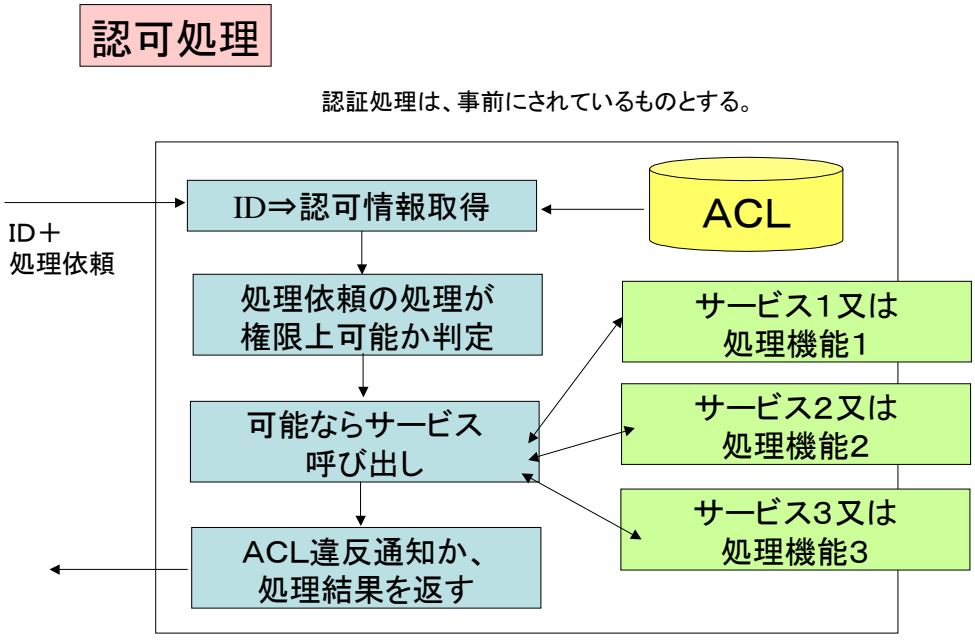


図３．５．１２　認可処理の流れ

次に、認可処理に類似した情報フィルタモデルと認可処理モデルとの相違を示す。

（1）認可技術とは
・認可（Authorization）とは、認証されたユーザに対し、システム又はアプリケーションの使用許可を判定することである。
・認可技術とは、認可を実現するコンピュータシステム、及び、ソフトウェアの実現技術である。
（2）情報フィルタ技術とは、
・ユーザの権限により、情報へのアクセスを抑制したり、情報の内容を抑制する技術

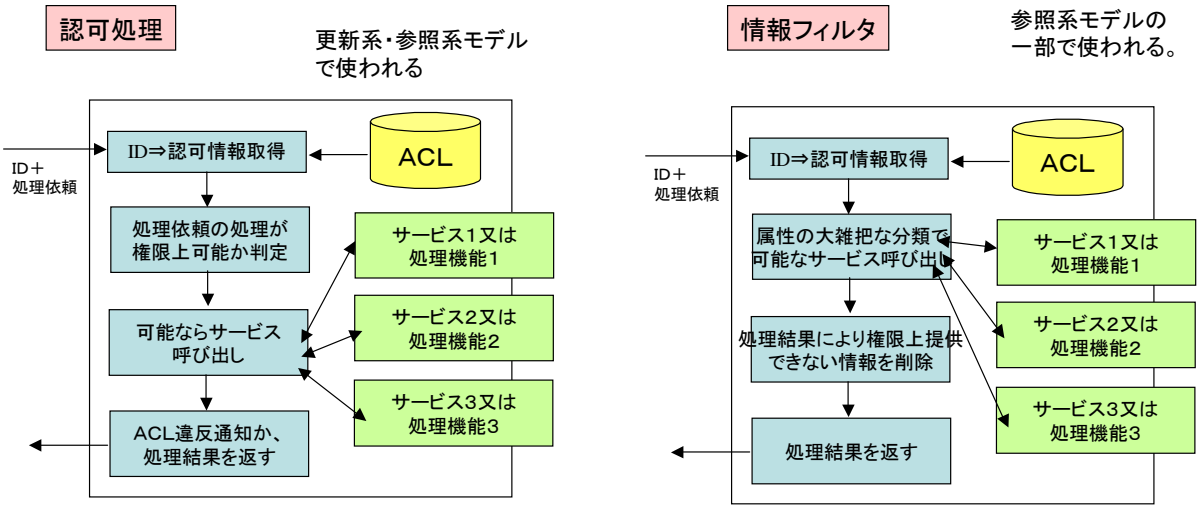


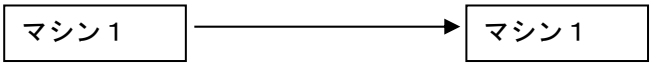
図３．５．１３　認可処理と情報フィルタ処理の違い

３．５．５　秘匿性確保の要件、ユースケース、実現技術

PF 秘匿性確保仕様とは、通信路上のデータの内容を第 3 者から参照させないための仕様である。秘匿性確保のためには、通信路上の 2 点間の秘匿性確保と、End-to-End での秘匿性の確保が必要となる。PF 秘匿性確保仕様ではサイト間における秘匿性確保を対象としていることから、通信路上の 2 点間の秘匿性確保仕様を異なるサイト間では必須としている。

（１）　通信路上の秘匿性の確保

ポータル、自治体、民間の間で、電子申請や公文書等を伝送される場合、データの漏洩を防止するため、SSL 及び TLS 通信仕様を、通信路暗号として使用する。



伝送路上のデータ漏洩のための暗号化

図３．５．１４　通信路上の秘匿性の確保

（２）　通信路によりつながった複数中継点を経由した End-To-End での秘匿性の確保

申請データや公文書等の内容に、個人情報や、秘匿性が高い情報がある場合、電子申請や公文書等の全体、または、一部に暗号化を実施し、権限がある確認者のみ内容が見えることが必要である。実現技

術としては、XML 暗号仕様や SMIME 仕様がある。
（課題：本要件の実現には、暗号鍵の管理運用と密接であるため、具体的なユースケースを分析し、鍵の運用を含め慎重に導入を検討する必要がある）

図３．５．１５　End-To-End での秘匿性の確保

（３）　地域情報 PF では、秘匿性確保の技術として、通信路上の秘匿性の確保技術である以下の技術を採用している。(XML 暗号等の技術は、必要要件がでた場合に検討する。但し、使用においては、鍵運用の課題の解決が必要である）

○SSL(TLS)：　SSL(TLS)を使い TCP/IP 層での通信路暗号を行う技術。
メリット：　通信路暗号として最もよく使われている技術。
デメリット：　通信路のみの暗号のため、中継を含む End-to-End の秘匿性確保には向かない。中継者にて情報が見えてしまう。

３．５．６　電子署名と検証の要件、ユースケース、実現技術

地域情報 PF における、電子署名と検証の要件、ユースケースについて図３．５．１６に示す。
下図は、ワンストップ申請処理における申請書類形式と署名との関係を示した例である。
なお、電子署名はオプションであり、民間の業務サービスの申請への電子署名については、該当サービスの要件に応じて適用有無を決定する。

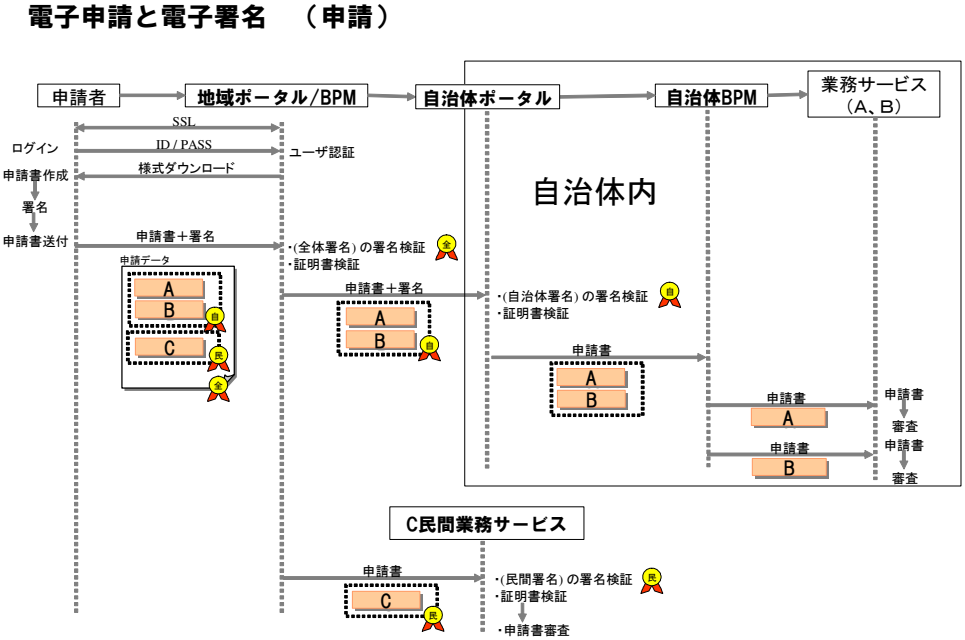


図３．５．１６　ワンストップ申請処理における申請書類形式と署名との関係

次に、ワンストップ申請処理における申請結果の公文書形式と署名との関係を図３．５．１７に示す。

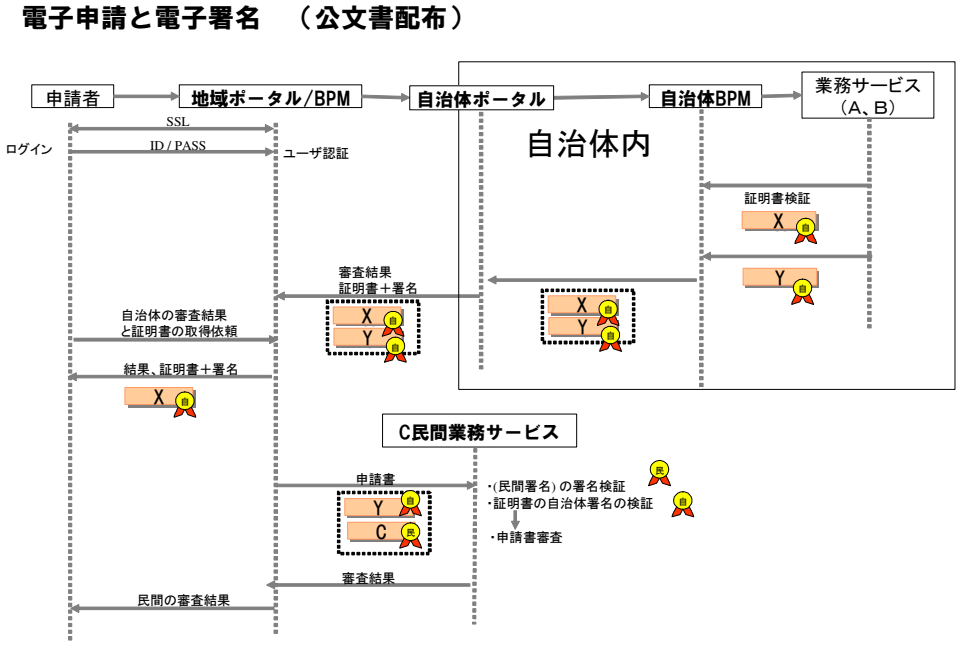


図３．５．１７　ワンストップ申請処理における申請結果の公文書形式と署名との関係

（１）　PF 本人電子署名・検証仕様

PF 本人電子署名・検証仕様は、通信路の仕様ではないが、電子申請者が申請する申請書を異なるサイト間で交換する際に使用し、情報の改ざん検知や署名者の検証を可能にする仕様である。電子申請等、本人が作成したこと、及び、改ざん防止が必要な場合に使用する。本仕様は、オプションである。

電子申請では、なりすましによる申請、データ改ざん、送信否認などのセキュリティ脅威が考えられる。これらの脅威を防ぐに場合には、申請書に電子署名を付与することが有効であり、下記の仕様を選択することを PF 標準で規定している。

・実現する技術として、XML Signature を使用する。
　電子申請する本人の端末で、申請データに XML Signature を行ない。XML Signature のある電子申請を、オンライン申請する。 申請をうけたデータを受信した申請先は、その申請書の XML Signature 検定を行う。
・住民が自治体用で申請する書類の電子署名は、JPKI を用いた電子署名技術を採用すると規定している。JPKI に関しては、公的個人認証サービスポータルサイト（http://www.jpki.go.jp/）を参照。

（２）　PF 自治体組織電子署名・検証仕様

PF 自治体組織電子署名・検証仕様は、通信路の仕様ではないが、自治体が発行する照会書類や公文書書類を、異なるサイト間で交換する際に使用し、情報の改ざん検知や署名者の検証を可能にする仕様である。本仕様は、オプションである。

なお、異なるサイト間で照会書類や公文書を交換する場合、に応じて、下記の仕様を選択することを PF 標準で規定している。

・書類がXMLの場合、XML Signatureを使用する。
・書類がファイルの場合、PKCS#7、または、XML Signatureで署名する。
PKCS#7の署名の場合は、ファイルと署名データを添付ファイルとして送信する。
PF自治体組織電子署名・検証仕様として、自治体間の組織認証基盤である「LGPKI」仕様を採用すると規定している。LGPKIに関しては、Webサイト（http://www.lgpki.jp/）を参照。

３．５．７　PFサービス認証の実現技術

PFサービス認証技術は、複数サイトとの間の連携処理にサービス認証処理の一元化が必要な場合に使用する（オプション）。本節では、ワンストップサービス用のサービス認証情報の管理/伝播方法およびそのモデルに関する仕様を説明する。

(1) 対象領域

Webサービスなどを利用したワンストップサービスを実現する場合、一般的なWebアプリケーションとは異なり、必ずしもエンドユーザがサービスの利用者とはならないために、呼び出された側のWebサービスではエンドユーザの認証を必ずしも行うことができないという問題がある。

例えば、エンドユーザがWebブラウザを用いてポータルにログインを行った後、ポータルが自治体AのWebサービスを呼び出す場合を考える。この場合、呼び出された自治体AのWebサービスで、エンドユーザの認証を行うには、ポータルと自治体Aで同一のユーザIDを共有してこれを使いまわすといった手段が考えられる。しかし、これを行うには、ポータルとポータルが呼び出す全てのサービスでのユーザIDの共有が必要となり、これを、自治体・民間間のように異なるID体系で連携を行う必要がある場合には適用し難いことが分かる。

(2) ユースケース

以下の図３．５．１８に、自治体と民間におけるサービス認証処理を連携させる場合のユースケースを示す。

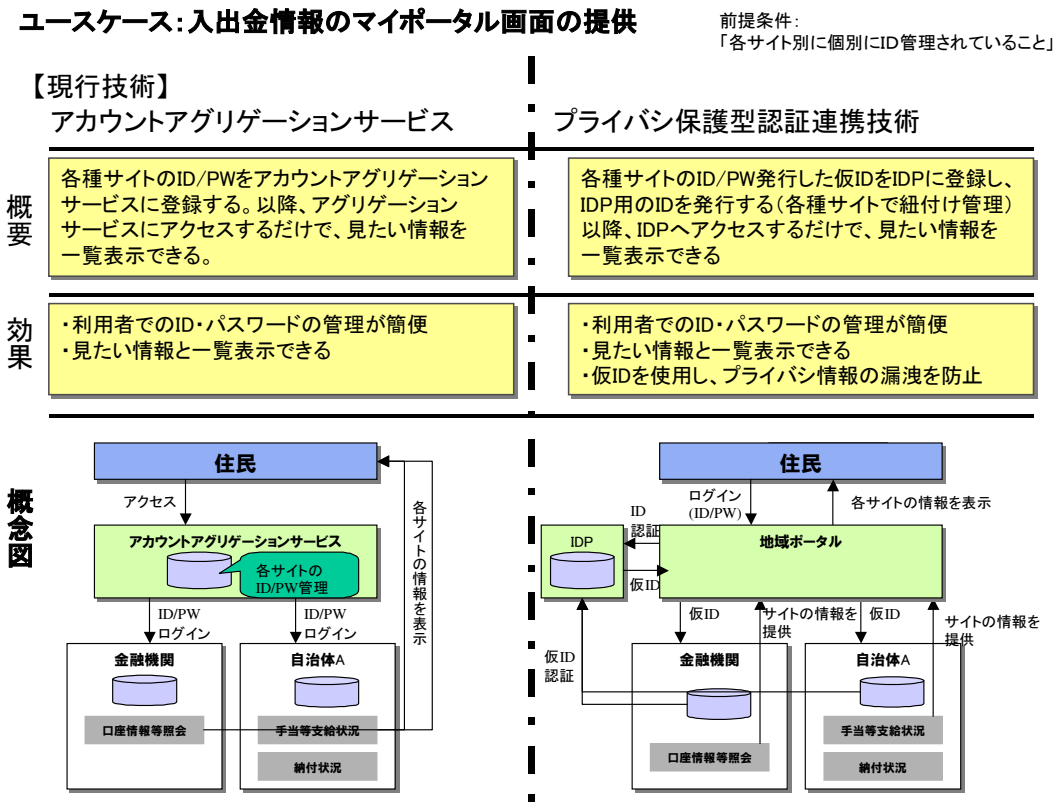


図３．５．１８　プライバシ保護型認証連携技術のユースケース図

①アカウントアグリゲーションサービスでのユースケース

（ｉ）住民がアカウントアグリゲーションサービスへ金融機関､自治体Ａの ID/PW を登録する。

（ii）住民は、アカウントアグリゲーションサービスへログインを行う。

（iii）アカウントアグリゲーションサービスは､登録されている住民の各サイトの ID/PW を使用して、口座情報の照会や児童手当の支給状況を取得、画面へ表示する。

②プライバシ保護型認証連携技術でのユースケース

（ｉ）金融機関、自治体Ａの ID/PW を住民があらかじめ入手する。

（ii）住民が金融機関、自治体Ａ、地域ポータルそれぞれにログインし IDP と連携を行うことで仮 ID が IDP から発行される。

（iii）住民は、地域ポータルにアクセスする際に、IDP から認証を受ける。

（iv）地域ポータルから、各サイトへアクセス時は、各サイトが IDP から仮 ID を入手することで、住民の識別と、サービスの提供を行う。

(3) 機能概要

以下では、サービス認証連携として、アグリゲーション型認証連携とプライバシ保護型認証連携の２つの方法を記載する。

①サービス認証（アグリゲーション型認証連携）

アグリゲーション型認証連携は、ポータル等のサービスを統合するサイトが、サイトの ID と個別サイトが提供する認証情報を紐付け保持し、サイトの認証後、バックの個々サイトのサービスを呼び出すとき、個別の ID と認証依頼情報でサービスを呼び出す方法である。

図３．５．１９に、アグリゲーション型認証連携のモデルを示す。

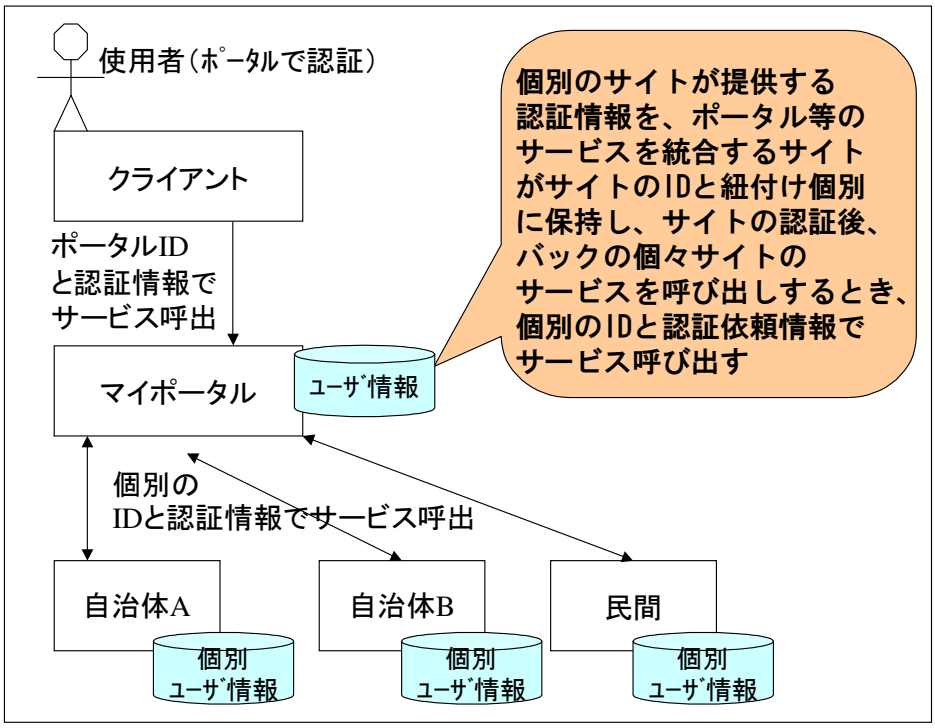


図３．５．１９　　アグリゲーション型認証連携

②サービス認証（プライバシ保護型認証連携）

プライバシ保護型認証連携は、ID 体系の異なるサイト間のサービス認証において、サービス間で認証情報の交換を可能とする機能（認証連携機能）と、認証情報を統合した場合に、複数サービスで使用されているプライバシ情報の名寄せを防止する機能（名寄せ防止機能）から成る。本機能を実現する標準仕様として、SAML V2.0 を採用する。

(a)認証連携機能

ID 体系の異なる様々なサービスが連携する際に、サービス間でサービス認証情報を交換することによってシングルサインオンを実現する機能である。

(b)名寄せ防止機能

名寄せを防止するには、他のサービスで提供される認証情報の伝播を防ぐ必要があり、例えば認証用の ID とは別の ID を使用するなどで情報の連鎖を切ることが可能になる。サービス利用者に認証サーバ内で認証に使用する ID とは異なる仮 ID を発行し、各サービスに割り当てる。認証情報を交換する際は、認証サーバから連携サービスに割り当てられた仮 ID を受け取り、これを使用する。

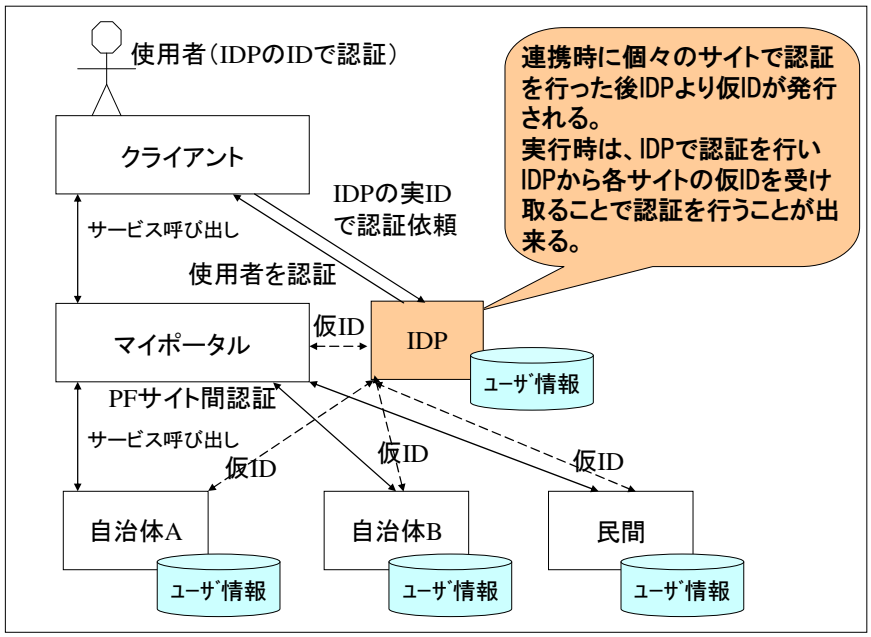


図３．５．２０　　プライバシ保護型認証連携技術

PF サービス認証（プライバシ保護型認証連携）では、IDP にて認証を行った使用者がサービス呼び出しを行った場合、IDP が使用者の仮 ID を呼び出し先のサービスに応じて提供する。これにより、サービス側では認証連携が可能となる。

ただし、職員が他自治体の住民情報を照会するなどのケースでは、職員がサービス呼び出しを行った際に、住民の仮 ID を照会先の自治体に応じて変換する仕組みが必要となる。この仕組みを実現する方法の１つとして、ID-WSF が考えられる。ID-WSF については、具体的な業務ユースケースへの適合性、標準化状況、普及の状況を鑑みて、今後採用候補として検討する。（付録 5 参照）

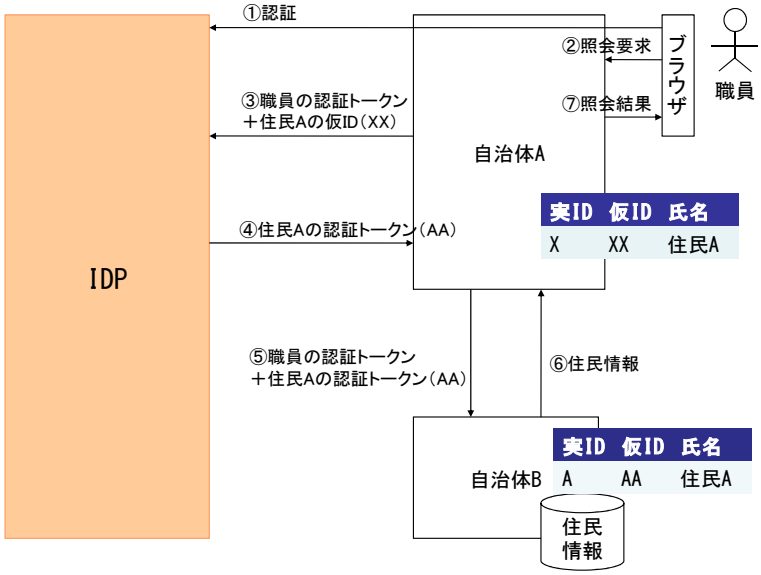


図３．５．２１　職員による住民情報の照会（ID-WSF による仮 ID 変換イメージ）

３．５．８　PF サービス認可の実現技術

PF サービス認可連携仕様とは、複数サイトとの間の連携処理にサービス認可処理の一元化が必要な場合に使用する（オプション）。

本節では、ワンストップサービス用のサービス認可情報の管理/伝播方法およびそのモデル、権限管理基盤技術に関する仕様を説明する。

(1) 対象領域

従来のシステムでは、アクセス制御機能を実装し、サービス提供者によってサービス利用者のアクセス制御情報（権限情報）の管理は行われている。しかし、その多くは、アプリケーション毎にアクセス制御機能が実装され、アクセス制御情報（権限情報）も個別管理されている。また、アクセス制御に利用するルールもアプリケーション毎に固有となっている。

このため、サイト内の場合、サービス提供者は、利用者 ID 情報の登録・削除の操作とともに、ID 情報と個別管理された権限情報のマッピングを手作業で行っている状況にあり、複数サイトでのサービス連携が実現した場合でも同様の問題は発生する。

これには、アクセス制御の管理作業に多大なコストと時間を要するため、サービス提供者にとって大きな問題となる。

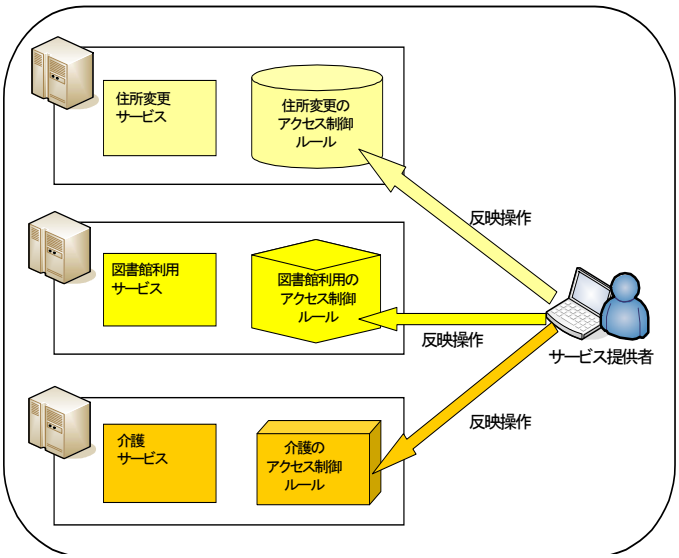


図３．５．２２　サービス提供者の問題点

(2) 全体概要とユースケース

ⅰ. 全体概要

権限管理基盤技術は、異なるサイト間の認可において、複数のアプリケーションで使用されるサービス利用者の権限情報を一元的に管理し、サイトをまたがったサービス利用者に対しても権限情報の整合性を維持する機能である。

ⅱ. ユースケース

図３．５．２３に、自治体間における情報照会処理における認証・認可連携のユースケースを示す。

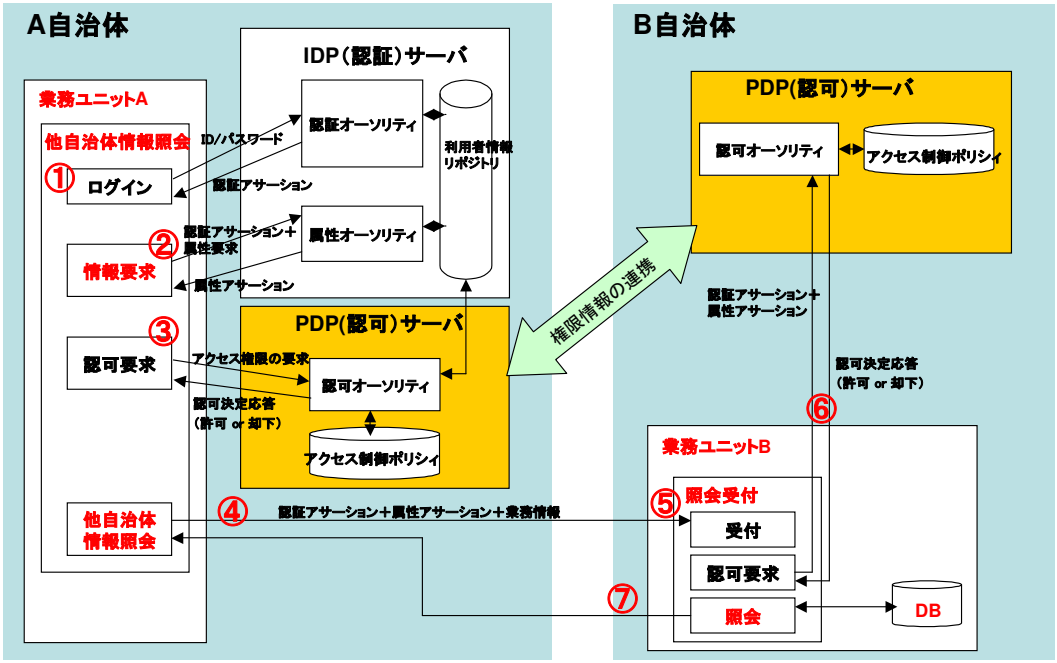


図３．５．２３　権限管理基盤技術のユースケース

① Ａ自治体の業務ユニットＡがある申請者のＢ自治体での情報を照会したい場合に、他自治体情報照会機能にログインする。
② IDP による認証を受け、認証アサーションと属性アサーションを受け取る。
③ 他自治体情報照会機能は、ログオンした職員がＢ自治体にアクセスする権限があるかを PDP に対し問い合わせる
④ 職員に権限がある場合は、情報照会に進み、Ｂ自治体の業務ユニットＢに対し認証アサーション、属性アサーション、業務情報（対象の申請者情報等）を送付し、照会依頼をする
⑤ Ｂ自治体の業務ユニットＢでは、情報照会受付機能が④の認証アサーション、属性アサーション、業務情報を受け取る。
⑥ 情報照会受付機能では、Ａ自治体から受け取った照会依頼している職員に情報照会の権限があるかを PDP に問い合わせる
⑦ 職員に権限がある場合は、対象申請者の情報をＡ自治体に送信する。

将来的には、自治体間の所得照会等の業務において、上記のようなユースケースの適用の検討も想定される。

(3) 機能概要

権限管理基盤ではサービス認可の機能を提供するが、その機能は権限情報提供機能と、権限情報同期機能から構成される。

(a) 権限情報提供機能

権限情報提供機能では、アクセス制御ルールを共通化し一元管理する。また、権限管理基盤を提供することにより、共通化したアクセス制御ルールによる判定を可能にする。

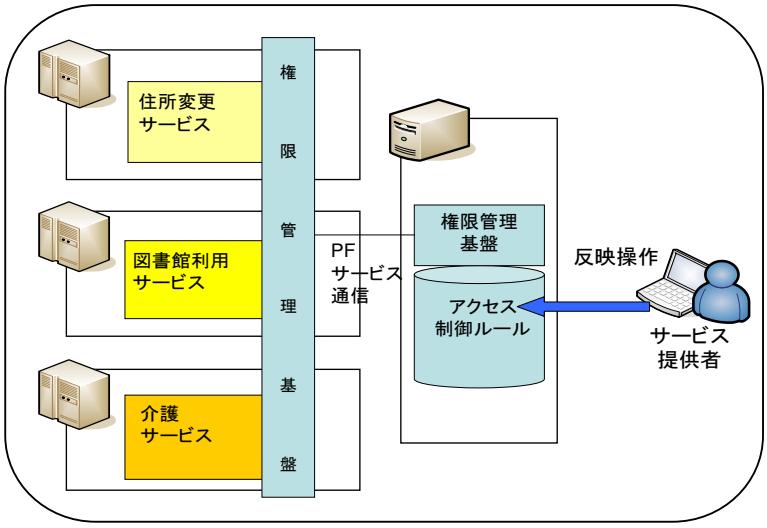


図３．５．２４　権限情報提供技術のイメージ図

権限情報提供機能に関する規定は、プラットフォーム通信標準仕様に記載している。なお、自治体内での権限管理情報の一元管理では、LDAP,RDB 等が使われる。

(b) 権限情報同期機能

権限情報同期機能では、「権限情報の連携」および「権限情報の整合性の維持」によりサイト間のアクセス制御ルールの管理および判定を実現する。

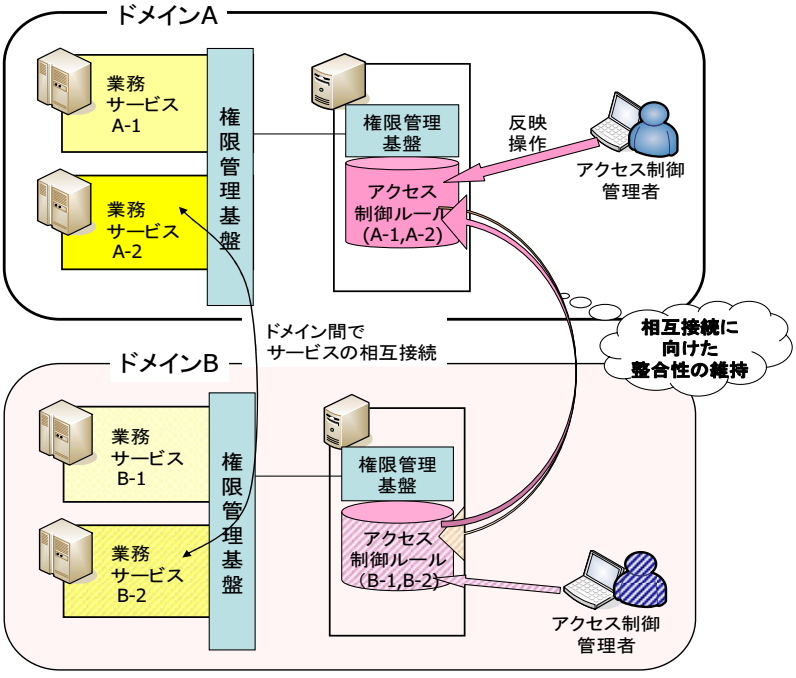


図３．５．２５　権限情報同期技術のイメージ図

権限情報同期機能に関する規定は、プラットフォーム通信標準仕様に記載している。

３．５．９　PF プライバシ情報公開仕様

PF プライバシ情報公開仕様とは、複数サイトとの間の連携に際して認証や認可処理の一元化が必要な場合に使用するプライバシ情報の漏洩防止方法・交換許諾方法に関する仕様である。PF プライバシ情報公開仕様は、オプションである。

（１）ワンストップサービスにおいて属性のようなプライバシ情報の交換を行う場合の課題

ワンストップサービスにおいて属性のようなプライバシ情報の交換を行う場合に次のような問題が発生する。まず、個人情報保護法遵守の観点からプライバシ情報の交換の際には、エンドユーザへその旨を掲示し、許諾を得る必要があるが、ワンストップサービスにおいて Web サービスを跨る度に許諾の作業を行う必要があるという効率の問題がある。さらに、認証時と同様の問題として、呼び出された側の Web サービスでどのようにエンドユーザから属性交換の許諾を得るかという問題もある。

（２）全体概要とユースケース

① 全体概要

ｉ．プライバシ情報公開調整技術

サービスの提供、およびサービスの連携の局面において、サービスの提供者におけるプライバシ情報の利用目的と、サービスの利用者のプライバシ情報の利用許諾意志との間の調整を行い、サービスの利用者の意図したプライバシ情報の公開のみを保証する技術である。

② ユースケース

ｉ．プライバシ情報公開調整技術

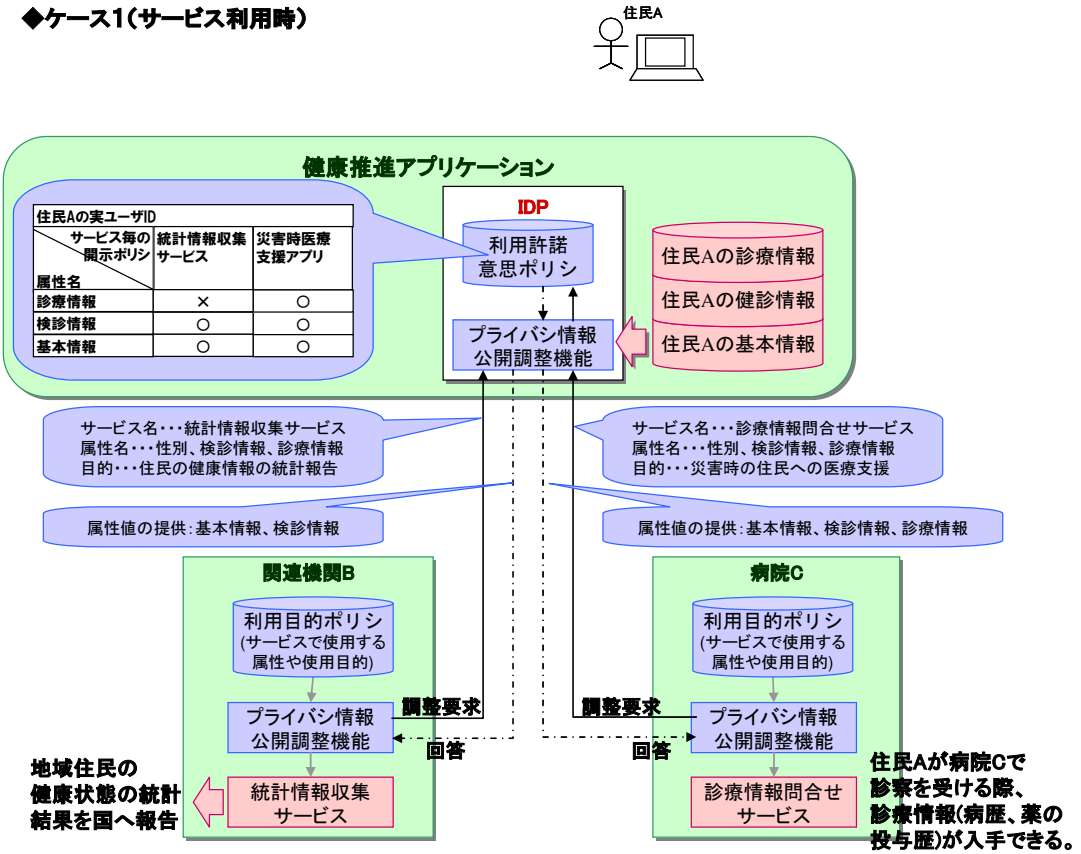


図３．５．２６　プライバシ情報公開調整技術のユースケース図（サービス利用時）

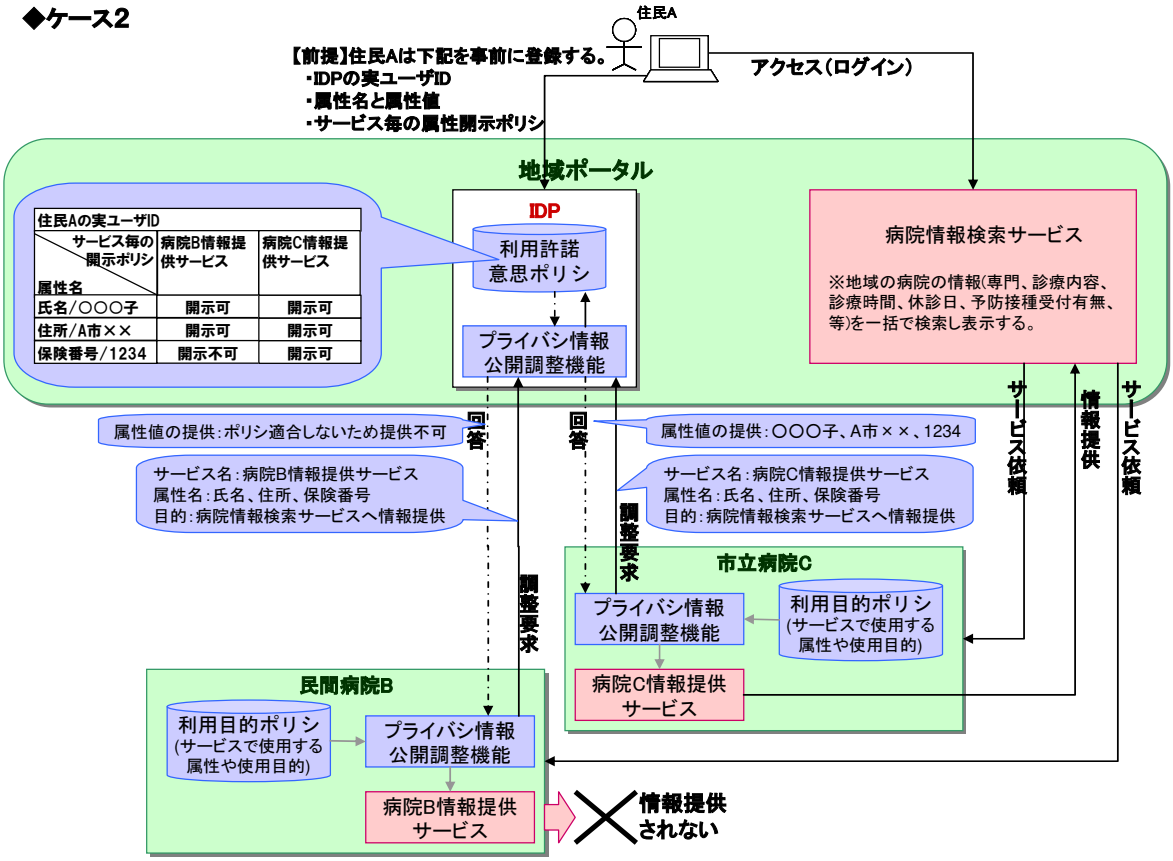


図３．５．２７　プライバシ情報公開調整技術のユースケース図

（３）機能概要

(a) 利用目的ポリシーの作成機能

提供するサービスの内容に応じて、使用するプライバシ情報の内容、およびその利用目的を記述した、サービスの提供者におけるプライバシ情報の利用目的ポリシー(P3P ポリシーと、P3P ポリシー参照で記述)を作成する機能である。

(b) 利用許諾意志ポリシーの作成機能

利用するサービスの内容に応じて、利用されるプライバシ情報の内容、およびその利用目的に応じて、許諾の意思を記述した、サービスの利用者におけるプライバシ情報の利用許諾意志ポリシ(P3P プリファリンスの記述)を作成する機能である。

(c) 利用目的ポリシーと利用許諾意志ポリシーの調整機能

プライバシ情報を提供するサイトが、プライバシ情報を利用するサイトから送付された利用目的ポリシーと、利用許諾意志ポリシー間の調整を行う機能。利用許諾意志ポリシーは、予めプライバシ情報を提供するサイトがエンドユーザから取得することもできるが、プライバシ情報を利用するサイトがエンドユーザから取得して送付することもできる。ポリシーが適合すればプライバシ情報をサービス提供者に提供し、ポリシーが適合しない場合にはプライバシ情報を提供しない。

本機能に対する技術仕様としては、ID-WSF が考えられる。ID-WSF については具体的な業務ユースケースへの適合性、標準化状況、普及の状況を鑑みて、今後採用候補として検討する。(付録 5 参照)

（４）規定事項

・プライバシ情報を扱う際のポリシーの記述仕様として、P3P V1.0 の採用をプラットフォーム通信標準仕様で規定している。

３．５．１０　PF 監査証跡仕様

PF 監査証跡仕様は、個々に点在するセキュリティ監査情報や複数サイト間に跨るセキュリティ監査情報を統一的に管理し、監査する際の基本的な仕様（考え方）について、PF 標準書で規定したものである（オプション）。なお、自治体内等の単一サイト内における監査証跡について本仕様にて規定するものではないとしている。

（１）対象領域

今日、多くの自治体や民間企業で、システムへの不正アクセス等による情報漏えいなどの事件・事故が頻発し、システムのセキュリティ強化とともに、内部・外部監査や法令遵守を前提とした、監査証跡としてシステム上の各種ログなどの証拠保全が必要とされている。

今後、複数サイトでのサービス連携が実現した場合でも同様で、サイト間で連携を図りつつ、情報漏えい等の箇所や原因の特定を行う必要がある。

しかし、現状では、同一サイト内でも、アクセスや操作の記録が保存されてなかったり、保存されも監査証跡として十分な形でなかったり、アプリケーション毎に個別管理されている、などの状況にある。サイト間では、サイト毎にログフォーマットやセキュリティポリシーなどの違いがあり、効率的な監査や情報漏えい等の箇所や原因の特定が困難になると考えられる。

監査証跡取得の目的としては、システム監査とアクセス記録が存在する。

(a) システムに対する不正アクセスや不正利用等を検知するためにシステム管理者が実施する監査（システム監査）

(b) 自治体が保有する住民情報に対する不正アクセスや不正利用等を検知するために住民本人が実施する監査（アクセス記録）

尚、アクセス記録については、以下の要件が存在する。

・自治体間連携におけるアクセス記録を出力できること
・自治体間連携で用いる ID により、特定の住民に対するアクセス記録を収集できること
・住民が参照可能なのは、自己情報に対するアクセス記録のみとなるよう制御できること

（２）全体概要とユースケース

① 全体概要

様々なサービスの連携において、サイト内のアプリケーション毎の収集管理を行い、サイト毎に異なるアクセス記録を、サイト間での合意に基づき相互流通させることで監査の効率化を行う。そのための監査証跡収集項目や、監査証跡収集インタフェース（業務アプリケーション向けインタフェース）、収集サービス機能を提供する。

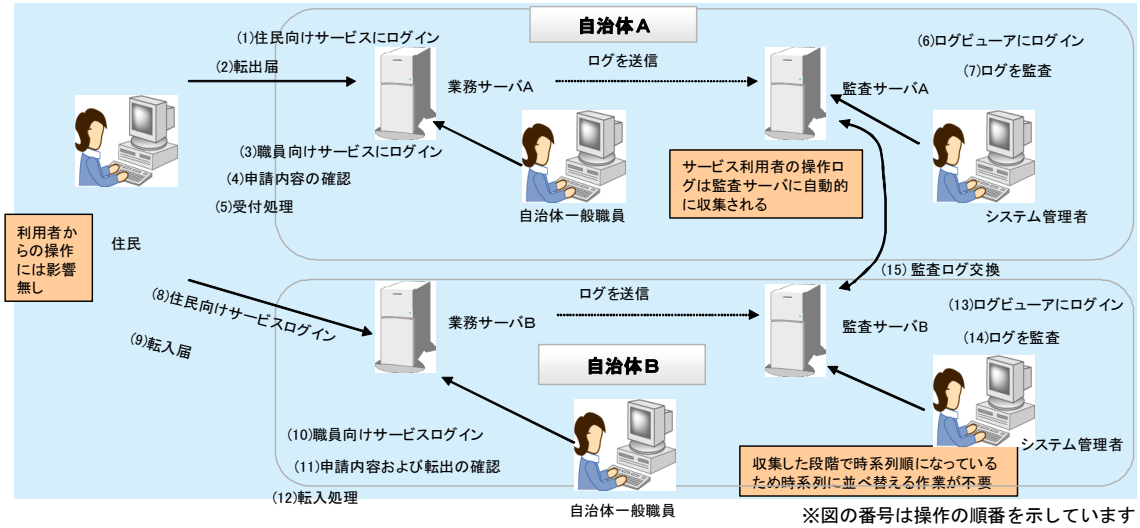


図３．５．２８　監査証跡取得機能の全体概要図

② ユースケース

以下に、他自治体情報照会での自治体間におけるユースケースを示す。

（ア） 自治体 A において自治体 B への情報照会処理を実施後、監査の実施または情報漏えい等の問題が発生した場合、自治体 B も含めた一覧の処理における監査ログの収集と解析が必要となる。

（イ） 自治体 A、自治体 B の合意のもと、自治体 A は自治体 B より関連する監査ログを収集する。

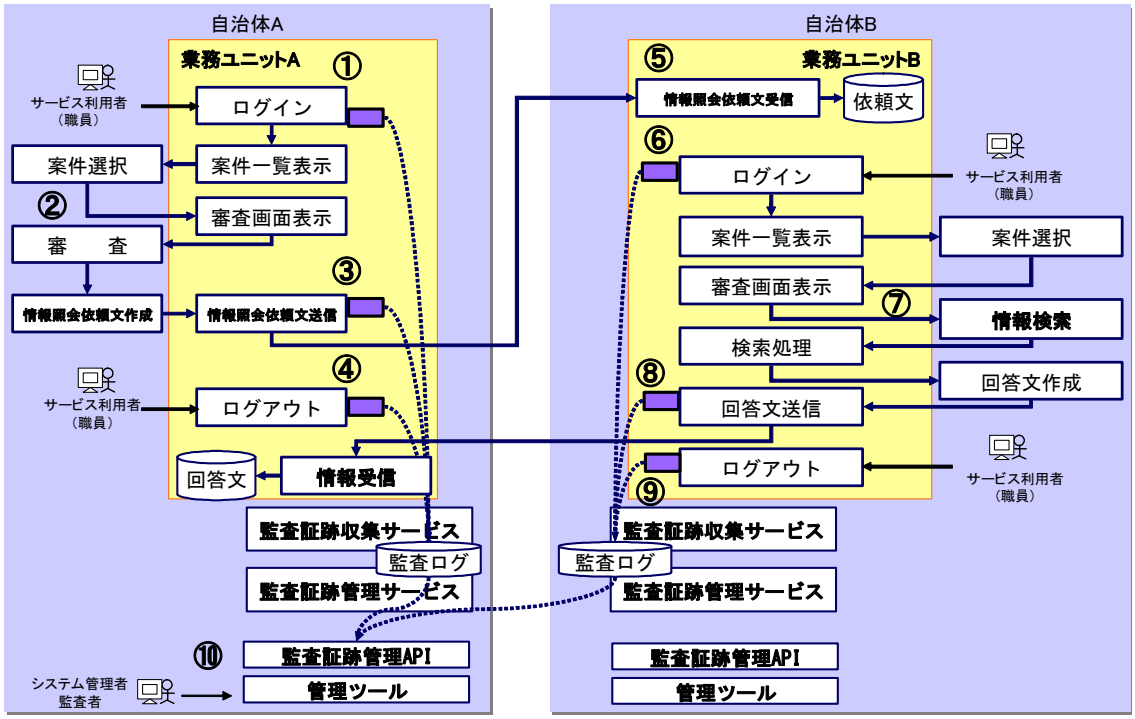


図３．５．２９　監査証跡取得機能でのユースケース図

以下に、詳細な処理フローを示す。

－自治体 A－

① 職員（サービス利用者）は、業務ユニット A へログインする。この時点で監査ログ（ログイン実行）が生成され、これらを監査証跡収集サービスが収集し、監査ログファイルに蓄積する。

② 職員は、審査等の処理を行う。

③ 職員は情報照会のための処理を行う。この時点で監査ログ（アプリケーション機能開始）が生成され、これらを監査証跡収集サービスが収集し監査ログファイルに蓄積する。

④ 職員は、システムログアウトする。この時点で監査ログ（ログアウト実行）が生成され、これらを監査証跡収集サービスが収集し、監査ログファイルに蓄積する。

－自治体Ｂ－

⑤ 業務ユニット B は、依頼文を受信する。

⑥ 職員は、業務ユニット B へログインする。この時点で監査ログ（ログイン実行）が生成され、これらを監査証跡収集サービスが収集し監査ログファイルに蓄積する。

⑦ 職員は、情報照会に対する審査、情報照会等の処理を行う。

⑧ 職員は、回答のための処理を行う。この時点で監査ログ（アプリケーション機能開始）が生成され、これらを監査証跡収集サービスが収集し、監査ログファイルに蓄積する。

⑨ 職員は、システムログアウトする。この時点で監査ログ（ログアウト実行）が生成され、これらを監査証跡収集サービスが収集し監査ログファイルに蓄積する。

－自治体Ａ・自治体Ｂ－

⑩ 自治体Ａで問題が発生した場合、自治体Ａのシステム管理者（または監査者）は、管理ツールから監査ログの収集処理の起動を行う。

これにより、自治体Ａの監査証跡 API が自治体 A と自治体 B の監査証跡管理サービスから監査ログ

を統一形式で収集する。

（３）機能概要

PF 監査証跡仕様は、個々に点在するセキュリティの監査証跡情報や複数サイト間に跨るセキュリティ監査証跡情報を統一的に管理し、監査する際の基本的な仕様（考え方）について規定したものである。

PF 監査証跡仕様に基づき、サイト内の監査情報を外部のサイトに公開するサービスを監査証跡情報提供サービスと呼ぶ。

図３．５．３０に、監査証跡取得機能におけるシステム構成例を示す。

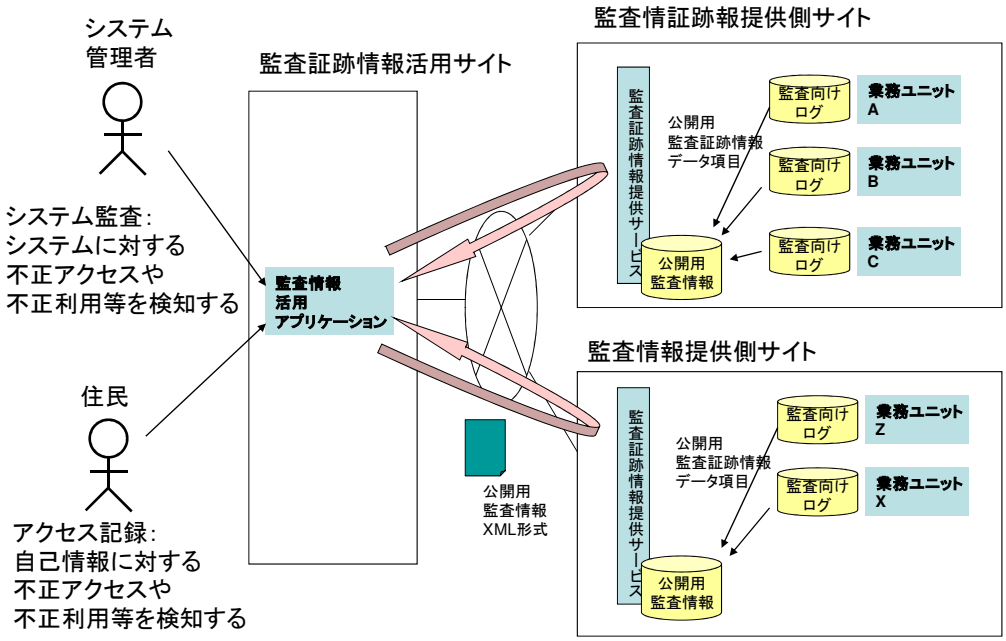


図３．５．３０　監査証跡取得機能のシステム構成例

また監査証跡情報提供サービスに関連するアプリケーションについて表３．５．２に示す。

表３．５．２　監査証跡情報提供サービスに関連するアプリケーション

アプリケーション名	機能概要
業務ユニット	監査対照の監査ログを生成する業務アプリケーション。
監査証跡提供サービス	監査証跡取得機能を実現するサービス。サイト間で公開する公開用監査情報を自治体内のログから取得蓄積管理し、PF サービス通信での取得依頼に基づき、情報を提供するサービス。
監査情報活用アプリケーション	監査ログの収集サービスにより収集された監査ログを、監査を目的に閲覧や分析するためのアプリケーション

ヘルスケア分野における関連情報としては、監査ログ生成の契機として、RFC3881 で定義されたイベントカテゴリが、また、ヘルスケア分野における監査証跡のメッセージ標準規約が以下の URL にある。

http://www.jahis.jp/standard/seitei/st06-002/st06-002.htm

PF としてのサイト間で共有する共通監査情報項目については、今後、サイト間で実施する監査要件を明確化する検討を実施し、その結果に基づき、情報項目を規定していく。

用語辞書

項番	用語	意味
1	認証技術	本人であることを判定し証明する技術 対策対象：　なりすまし、不正浸入
2	秘匿性確保技術	データの内容を第 3 者が参照できなくする技術 対策対象：　盗聴、漏洩
3	電子署名と検証技術	電子署名と検証技術とは、情報に対し電子的な署名を行い、情報の改ざん検知や署名者の確認を可能にする技術 対策対象：　なりすまし、不正浸入、データの不正アクセス
4	プライバシ情報公開技術	プライバシ情報公開技術とは、認証・認可におけるプライバシ情報の漏洩防止方法・交換許諾方法の技術 対策対象：　プライベート情報の不正利用や漏洩
5	監査証跡技術	正式なシステムを不正に使用していないかの情報を統一的に収集し監査したい場合に、使用する技術 対策対象：　正式システムの不正使用

略語

略語	正式名称
DDoS	Distributed Denial of Service
JPKI	Japanese public key infrastructure
LGPKI	Local Government Public Key Infrastructure
LGWAN	Local Government WAN
OASIS	Organization for the Advancement of Structured Information Standards
PKI	Public Key Infrastructure
P3P	Platform for Privacy Preferences
SAML	Security Assertion Markup Language
SSL	Secure Socket Layer
SSO	Single Sign-On
TLS	Transport Layer Security
WS-I	Web Services Interoperability Organization
WS-Security	Web Services Security

３．６　モニタリング機能

本節では、モニタリング機能に関して下記事項について概説する。

(1)モニタリング機能の実装意義と、Audit 収集の価値　(3.6.1 節)

(2)モニタリング機能における実装例、その留意事項　(3.6.2 節)

(3)Audit 群とモニタリング問い合わせインタフェースとの関係　(3.6.3 節)

上記(1)では、モニタリング機能の実装意義と、Audit を収集することの価値について説明する。ここで Audit とは、ログの概念を一般化したものである。従来、サービスやシステムリソースのある一時点の状態を示すデータ、スナップショット像、あるいはそれらの集合は、領域毎に呼称が異なり、統一化されない状況であった。例えばシステムリソースの分野では、“ログ”が一般的であるのに対して、ビジネスプロセスでは“イベントログ”、“Audit　Trail”等の語彙が使われる。モニタリング機能はビジネスプロセス、サービスという複合的な領域を対象とするため、シノニムとして扱われるこれらに対して統一的な「呼称」「概念」を与える必要がある。そこで、地域情報プラットフォームの仕様では、“ログ”等を一般化した概念として“Audit”という語彙を導入している。

ビジネスプロセスの改善では、モニタリング機能は大きな役割を持ち得るが、その為には、この Audit の内容に依存する。そこで、Audit の定義、ならびにその品質が高い程、モニタリング機能の価値が高まることについて概説する。

続く(2)では、モニタリング機能の実装例とそのガイドラインについて概説する。モニタリング機能は概して新たな要件に応える機能である。その為、種々の国際標準が出揃わない部分的な機能も、その構成要素に取り込んでおり、仕様事項が抽象化しやすい事情がある。ここでは、実装例を明示することで、より実際の機能イメージについて言及する。

続く(3)では、アーキテクチャ仕様書で定義されるモニタリング問い合わせ通信プロトコルと、奨励する Audit 群との関係について説明する。

最後に関連する付録として以下の 2 つの推奨事項を記載すると伴に、マルチベンダ下でモニタリング機能を実装した場合の例も記す。

[付録１]Audit、インタフェースに関する詳細定義

推奨する Audit の形式等の詳細事項について記載する。なお、このガイドラインで定義する Audit は推奨案に留まっている。本 Audit と等価で別な形式のものがある場合は、本方式以外の実装も許容する。

[付録２]実装コンポーネント構成に関する推奨モデル

実装コンポーネント構成に関する推奨モデルを定義する。モニタリング機能の構成、アーキテクチャについては、アーキテクチャ標準仕様に定義があるものの、実装に対する推奨事項はない。ここでは推奨モデルとして記載する。

[付録４]モニタリング機能の実装例

マルチベンダ下でモニタリング機能を実装した場合の例を記す。

３．６．１　モニタリング機能の意義と、Audit 収集の価値

（１）　モニタリング機能の意義

運用のポリシー、アーキテクチャが異なる複数組織間連携の基盤として SOA (Service Oriented Architecture)が注目を浴びているが、これを要約すると、各組織が利用・保持する機能を外部から利用しやすい形に切り出して、サービスとして定義・提供し、さらには複数のサービスを組み合わせることでシステム間連携を実現するアーキテクチャ、と言うことができる。SOA 化が進展することで、組織をまたがるシステム間の連携を大規模に実現することができ、この結果としてビジネスプロセスと呼ばれる

大きなサービス実施、ワンストップサービスの実現が可能になる。

高い潜在力を持つ SOA ではあるが、さらに、より高い価値を提供するためには、その上で進められるビジネスプロセスの実施状況・品質をモニタリングでき、サービス選択の幅が広いことも重要になる。ビジネスプロセスの実施状況・品質をモニタリングすることができれば、その品質を低下させているサービスや、それを実施しているシステムリソースを特定でき、対策を講じることができる。

以上の様なビジネスプロセスを改善するために実施状況・品質を正しく把握しようとする取り組みを「可視化」とも呼ぶ。この「可視化」のためには、組織間に散在し、実施結果を意味する各種情報を Audit として収集する仕組みが必要になる。モニタリング機能は、散在する情報を Audit として収集する仕組みを提供し、実施状況の把握、進捗管理を通して「可視化」の基盤を提供するものである。

（２）　Audit の定義とその収集・管理の価値

以上の様な意義を持つモニタリング機能ではあるが、その直接的な効果は、複数組織を跨るビジネスプロセスが実行された場合の、実施状況の把握、進捗管理にある。このため、モニタリング機能は、アーキテクチャ標準仕様で定義されるように収集された Audit の利用を基礎としている。地域情報プラットフォームの仕様では、この Audit に、BPM(Business Process Management) Audit、WS アプリケーション Audit、メッセージ Audit と言うものを定義している。

この Audit は、事象が発生した段階で発生するものであるから、これを収集し、解析することでサービス、ビジネスプロセスの種々の特性、KPI (Key Performance Indicator)を把握することが可能である。そして、これが、本来の目的であるビジネスプロセスの改善に関連し得る。

以上から、Audit 収集の密度が高く、その品質が高いほど、モニタリング機能をより高次な段階まで利用することが出来、結果としてビジネスプロセスの改善を促すことができる様になる。現在の Audit に関する仕様は、斯様な要求を見通して、その内容を詳細に定義している。

３．６．２　モニタリング機能の実装例とその留意事項

（１）　モニタリング機能の実装例

自治体、民間会社、地域ポータル等の横断的なサービスを事例とした場合のモニタリング機能の実装例について模式化したものを図３．６．１に記す。図３．６．１は、地域情報プラットフォームのアーキテクチャ標準仕様に記されたものをより、実装に近い形で記した図である。各組織間のセキュリティポリシー等の違いのため、ファイアウォール装置、ルータ装置、LGWAN(総合行政ネットワーク)等を含めて明記するべきであるが、図３．６．１はモニタリング機能の実装・配置に力点を置いているため、省略している。

モニタリング機能は、各要素が分散的に配置されるが、その中心となる監視アプリケーションに関しては、大きくは四つ配置されている。1 つはポータルサイト、もしくは集中管理センターに配置されるべき監視アプリケーション、監視クライアントである。これは、当該モニタリング機能の通常利用を想定している。これに対して、他の 3 つは、地域ポータル、自治体 A、民間企業等の各々に配置される Audit Proxy や監視アプリケーションである。これは、当該モニタリング機能の拡張利用時を想定しており、各々の組織体内、もしくは組織体からのプロセス監視、サービス監視を行なう。

地域ポータル、自治体 A、自治体 B、民間企業の各々に配置される BPM 機能、業務ユニット等はマルチベンダ条件下で調達されるため、各 Audit 提供機能、その変換機能はそれぞれ、Audit のガイドラインに適合しつつも調達機器に適合する形で実装される。例えば、BPM Audit 提供は、図３．６．１の自治体 A の様に BPM 機能のサーバに組み込まれる場合もあるが、専用のアダプタ装置が置かれる場合もある。また、メッセージ Audit を取得するに当たっては専用の収集モジュールを配した装置が置かれる場合がある。また、これらの Audit 機能に関する国際標準化も、今後、検討が進むと予想されるため、このガイ

ドラインでは、Audit 提供の配置について実際の運用条件に柔軟に対応するため、緩やかな構成を推奨する。

また、このガイドラインでは Audit の形式についても推奨案を定義しているが、これも実際の運用条件に柔軟に対応するため、緩やかな定義を前提とする。このため、付録４に見られる様に XML によるメッセージング、ソケット通信等のものを許容している。これらの対応策として、以上の様な柔軟性を確保する為に、そのインタフェースを運用の段階で必要に応じて公開する等の処置が必要となる。

監視アプリケーションと Audit Proxy 間の通信プロトコルについては、アーキテクチャ標準仕様に基づき、実装され、その問い合わせは受付番号等を用いて為される。アーキテクチャ標準仕様においては、十分な柔軟性を許容している。アーキテクチャ標準仕様では、監視アプリケーションと Audit Proxy 間の通信プロトコルについて、プラットフォーム通信標準仕様である SOAP(Simple Object Access Protocol)を許容しているが、限定的に利用可能な場合、プラットフォーム通信標準仕様との相互連携を前提の上で他方式も利用可能とする。例えば、付録４の場合、WSDM(Web Services Distributed Management V1.1)を利用している。

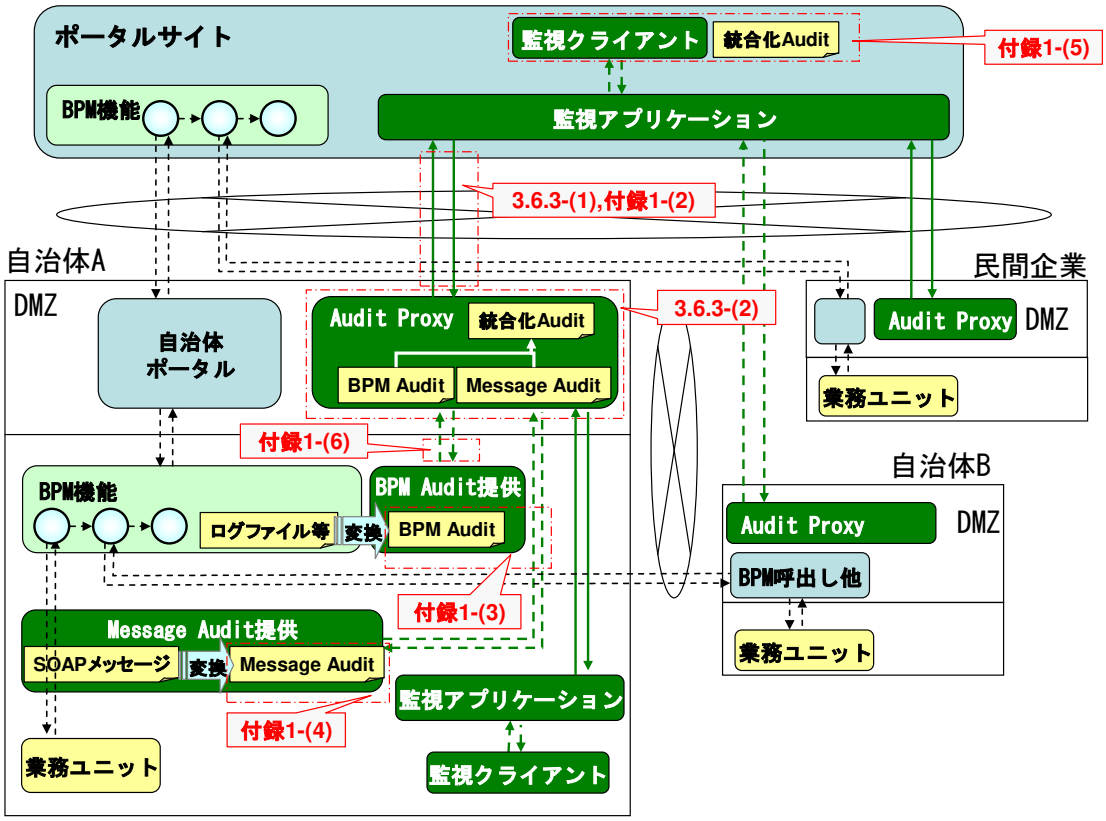


図３．６．１ モニタリング機能の実装模式図

モニタリング機能では、アーキテクチャ標準仕様で定義されるように、モニタリング問い合わせインタフェース、監視クライアントインタフェース、BPM・WS アプリケーション Audit 提供インタフェース、メッセージ Audit 提供インタフェースの４つが存在する。この図３．６．１では、監視クライアントインタフェース以外の３つについて、網掛け・番号付けて定義している。例えば、モニタリング問い合わせインタフェースについては、3.6.3 節(1)、並びに付録１－(2)に、その詳細な説明を記す。なお、マルチベンダ下でモニタリング機能を実装した場合の事例を付録４に記す。

（２）　　モニタリング機能実装における留意事項

本節では、モニタリング機能を実装する上での留意事項を記す。

- このガイドラインが執筆された段階で、モニタリング対象に関する情報モデル等で、国際的な標準類が策定されていない状況にある。しかし、いずれはこの分野で国際標準化が進むのは確実と考えられる。その為、各 Audit 提供等ではインタフェースの構成を柔軟にし、変換モジュール等の組込みが柔軟にできる設計が求められる。付録４の実装例でもその様な方式を採用している。
- モニタリング機能のアーキテクチャ標準仕様に基づき、監視アプリケーション等のアーキテクチャについては規定していない。その為、監視アプリケーションの配置位置を柔軟に選択・構成できるが、より広域にモニタリングを実現する上で、Audit Proxy 等との密結合な実装は推奨しない。
- 監視クライアントと監視アプリケーション間のインタフェースについても、仕様事項としては規定しない。

３．６．３　Audit 群とモニタリング問い合わせインタフェースとの関係

（１）　　モニタリング問い合わせインタフェースと Audit 取得モデルの関係

Audit 取得モデルとは、Audit Proxy が各種 Audit 提供から Audit を入手する方式についてモデル化したものである。図３．６．２は、モニタリング機能が採用する Audit 取得モデルについて、概要と位置付けを記したものである。この図ではアーキテクチャ標準仕様書に規定されているモニタリング問い合わせインタフェースとの関係も記される。

モニタリング問い合わせインタフェースと Audit 取得モデルは、基本的に階層を成し、一体化している訳ではない。これに従い、実行タイミングも異なっている。モニタリング問い合わせインタフェースによるクエリが、システム運用管理者からの要求がきっかけで、送信される可能性が高いのに対して、Audit 取得モデルは、それに追随するとは限らない。詳細は、次節にて説明する。

Audit Proxy が、必要とする Audit (BPM Audit ないし WS アプリケーション Audit)を取得する方法としては、大きく 2 つの方法が考えられる。一方は、Audit 提供から Audit Proxy へ一方的に送付するプッシュ方式であり、他方は Audit Proxy が必要とする Audit を Audit 提供に要求し、取得するクエリーレスポンス方式である。適時性の点ではプッシュ方式のほうが有利であるが、ネットワークに必要以上に負荷を掛けない様にすることを配慮して、クエリーレスポンス方式を奨励する。

このため、BPM Audit 提供から BPM Audit を取得するためのインタフェースとしてクエリ形式が用意されている。これに基づき実装されている場合は、BPM Audit 提供に対し、取得したい BPM Audit を特定するクエリを送信することで、該当する BPM Audit を収集できることになる。同様に WS アプリケーション Audit 提供から WS アプリケーション Audit を取得するためのインタフェースとしてもクエリ形式が用意されている。これに基づき実装されている場合は、WS アプリケーション Audit 提供に対し、取得したい WS アプリケーション Audit を特定するクエリを送信することで、該当する WS アプリケーション Audit を収集できることになる。

BPM Audit 提供および WS アプリケーション Audit 提供は、BPM 機能を実装した WS-BPEL(Web Services Business Process Execution Language 2.0)処理系、および WS アプリケーションのそれぞれ複数のものから実行状態に関する情報を取得し、Audit を生成する。そこで各 Audit 提供は、それが担当する WS-BPEL 処理系、WS アプリケーションから取得した Audit のインスタンスにシーケンス番号(sequenceNo)を付与し、一時的に管理する。このシーケンス番号は各 WS-BPEL 処理系、WS アプリケーション毎に独立して採番する値で、例えば Audit の発生時刻基づき昇順に値を採番しても良い。奨励するクエリ形式が具体的に定義されており、BPM Audit 提供から BPM Audit を取得するクエリ形式、WS アプリケーション Audit 提供から WS アプリケーション Audit を取得するクエリ形式は、同じ様な形式を用いる。付録１にて推奨する形式を記す。

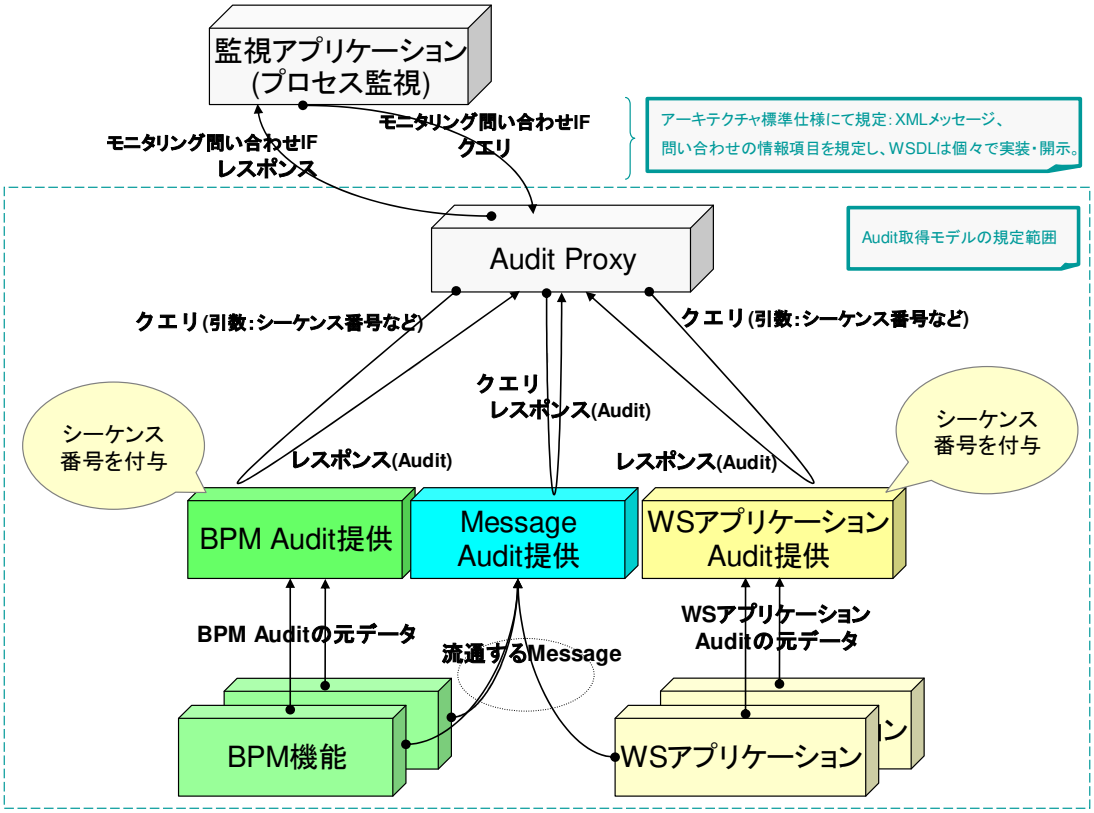


図３．６．２ Audit 取得モデルの構成とモニタリング問い合わせインタフェースとの関係

（２）　　モニタリング問い合わせインタフェースと Audit 取得モデルの論理的な関係

図３．６．３は、Audit 取得モデルにて実行されるクエリーレスポンスの XML メッセージと、モニタリング問い合わせインタフェースで流通する XML メッセージの関係を記した図である。前述の様にモニタリング問い合わせインタフェースと Audit 取得モデルは階層を成し、一体化している訳ではない為、実行されるタイミングも異なる。ここでは、Audit Proxy の実装方式までは言及しないが、この実行タイミングのずれから、Audit Proxy には、適宜起動する問い合わせモジュール相当のもの、Audit 取得モジュール相当のものが必要となる。

Audit 取得モジュールは、Audit 取得モデルを実現するものであり、定期的もしくは、それに準じた条件で関連する Audit データを全ての Audit を収集し、Audit Proxy 内に保持する。これに対して問い合わせモジュールは、モニタリング問い合わせインタフェースに応じて、クエリを処理し、収集済みの全ての Audit から該当するものを選定し、モニタリング問い合わせインタフェースのレスポンス形式に変換する。

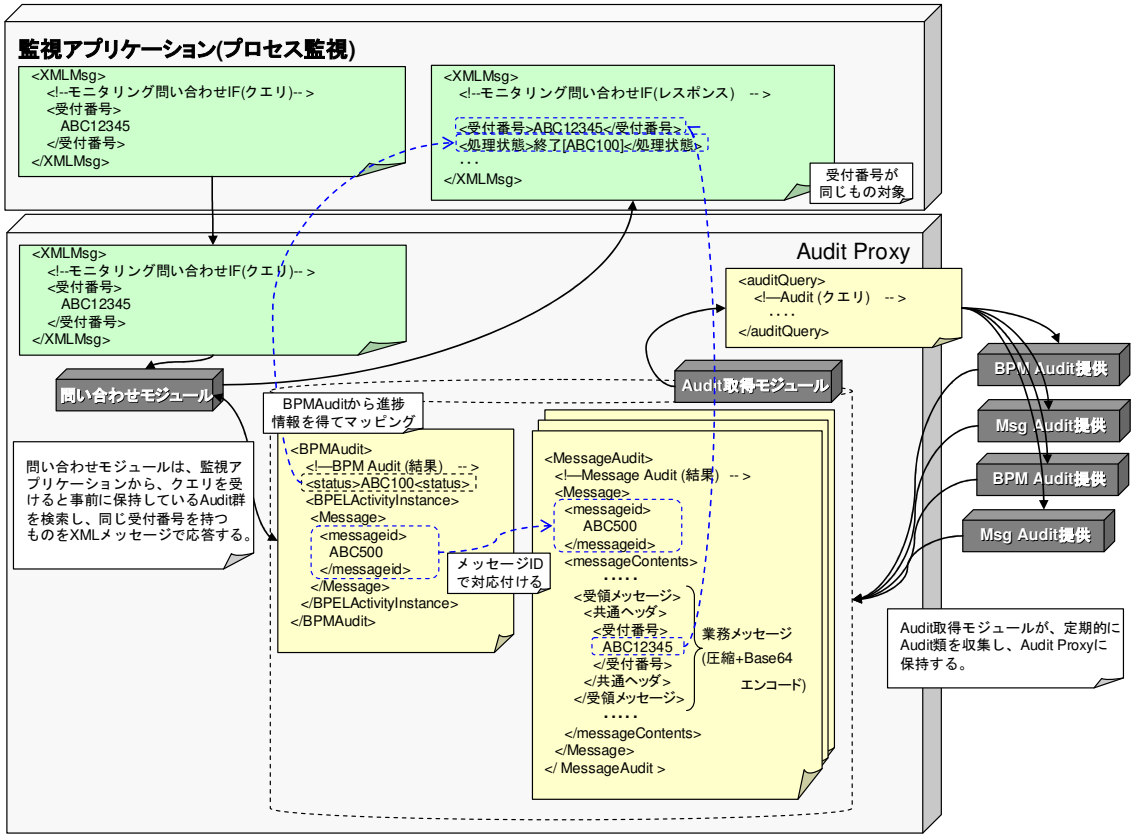


図３．６．３ Audit とモニタリング問い合わせインタフェースの論理的な関係

続いて、処理の概要について説明する。モニタリング問い合わせインタフェースの WSDL(Web Services Description Language)については事前に公開されているが、Audit Proxy 毎に異なることも想定している。そこで、監視アプリケーションは、問い合わせ対象の Audit Proxy 毎に受付番号に関する指定値を記載した XML メッセージを準備する。その後、監視アプリケーションから当該 XML メッセージを各 Audit Proxy に送付する。

任意の Audit Proxy が XML メッセージを受理すると、この XML メッセージに記載される受付番号に関する指定値を取り出す。その後、この値を用いて該当する Audit を検索する。図３．６．３の様に、Message Audit が存在する場合、受付番号は、Message Audit に含まれる圧縮、Base64 エンコーディングされた業務メッセージ内のシステム制御情報のヘッダの一部として含まれている。従って、流通する業務メッセージ全てに対応する Message Audit を対象として、該当するものを検索する。検索の結果、ある一連の Message Audit 群を入手した場合は、今度は BPM Audit を検索する。その際、Message Audit 並びに BPM Audit で記載される messageid を連携のキーにすることで、該当する BPM Audit を全て見つけることができる。Message Audit が存在しない場合、受付番号を含むデータを BPM Audit や WS アプリケーション Audit に変数(Variable)として含め、Variable から受付番号を取り出すことも可能である。

検索された該当 BPM Audit には、BPM 機能の実行状態を意味する status 情報が記載されている。そこで、この status 情報をビジネスプロセス上の状態に対応させることで、モニタリング問い合わせインタフェースのレスポンスを作成することができる。前述の様に、モニタリング問い合わせインタフェースの WSDL については事前に公開されているが、Audit Proxy 毎に異なることも許容している。そして、事前公開される WSDL には、前述 status 情報とビジネスプロセス上の状態対応が事前に定義されることが求められる。このため、状態の意味に関する調整は監視アプリケーション側で事前に行なわれていることが必要である。このため、各 Audit Proxy に定義している XML メッセージ形式で、前述レスポンス相

当の XML メッセージを作成することが可能となる。

（３）　ビジネスプロセスの進捗管理手順

最後に、モニタリング機能が実現するビジネスプロセスの進捗管理を実施する手順の一例について、ステップを追って概説する。ここでは、オプションである Message Audit(メッセージ Audit)が存在する場合を考える。従って、Message Audit(メッセージ Audit)が存在しない場合は、それに相当する手順で実装することも可能である。

ステップは大きく 3 つの部分から構成される。[ステップ 01] [ステップ 02]は、Audit が発生する度に、適宜実施される。これに対して、[ステップ 10]以後は、システム運用管理者からの明示的な起動要求があって、初めて実施される。これは、前述の通り、モニタリング問い合わせインタフェースと Audit 取得モデルが一体化している訳ではなく、さらに監視アプリケーションでは、システム運用管理者からの明示的な起動要求が無ければ、特に動作は発生しないためである。

[ステップ 01]：全ての Audit Proxy において、Message Audit(メッセージ Audit)、BPM　Audit を適宜受信する。モニタリング機能を実装する限りにおいて、BPM　Audit は必ず生成し、Message Audit(メッセージ Audit)は状況に応じては、生成されない場合も存在する。

[ステップ 02]：受信された Audit は、後の利活用のために、適切な形で管理される。特に、Message Audit(メッセージ Audit)が生成される場合、後に指定「受付番号」を持つ Message Audit(メッセージ Audit)が検索される。この「受付番号」は Message Audit(メッセージ Audit)に含まれる圧縮、Base64 エンコーディングされている業務メッセージ内のシステム制御情報のヘッダに一部として含まれているので、Message Audit(メッセージ Audit)を受信し、管理する段階で、Base64 デコーディング、解凍等を事前に実施しておき、検索時に利用する「受付番号」の文字列と伴に、Message Audit(メッセージ Audit)を管理する等の事前処置を行なう。(但し、奨励)

[ステップ 10]：監視アプリケーションを起動する場合、システム運用管理者は監視クライアントを介して、進捗管理の対象となるビジネスプロセスが持つ「受付番号」を指定する。

[ステップ 11]：続けて、監視アプリケーションは、問い合わせ対象の Audit Proxy 毎に定義されるモニタリング問い合わせインタフェースの WSDL に応じて、問い合わせの XML メッセージを準備する。

[ステップ 12]：監視アプリケーションから当該 XML メッセージを各 Audit Proxy に順次送付する。ここで、監視アプリケーションは対象とする Audit Proxy 全てに、この処理を実施するため、上記[ステップ 11]と本[ステップ 12]を繰り返し実施する。

[ステップ 13]：ある Audit Proxy が XML メッセージを受理すると、この XML メッセージに記載される「受付番号」に関する指定値を取り出す。

[ステップ 14]：続けて、この Audit Proxy が内部に管理し、流通する業務メッセージ全てに対応する Message Audit(メッセージ Audit)群を対象として、先の「受付番号」を持つ Message Audit(メッセージ Audit)を検索する。受付番号は Message Audit(メッセージ Audit)に含まれる圧縮、Base64 エンコーディングされている業務メッセージ内のシステム制御情報のヘッダに一部として含まれている。従って、検索の段階で Base64 デコーディング、解凍等を行なう、もしくは、事前に取り出した「受付番号」を用いて検索を行なう。MessageAudit(メッセージ Audit)自身は、オプションとなっているため、何等かの理由により Audit が生成されない場合もあり、全てが取り扱われるとは限らないことを考慮する必要がある。

[ステップ15]:該当するMessage Audit(メッセージ Audit)を検索した後、Audit Proxyでは、BPM Auditを検索する。その際、Message Audit(メッセージ Audit)並びにBPM Auditで記載されるmessageidを連携のキーにすることで、該当するBPM Auditが見つける。なお、BPM Audit自身は、指定必須となっているが、messageidが指定選択となっているため、該当するBPM Auditの全て検索出来ない場合も考慮する必要がある。

[ステップ16]:検索されたBPM Audit上では、BPM機能の実行状態を意味するstatus情報が記載されている。そのため、このstatus情報をビジネスプロセス上の状態に対応させることで、モニタリング問い合わせインタフェースのレスポンスであるXMLメッセージが作成される。

[ステップ17]:その後、モニタリング問い合わせインタフェースのレスポンスであるXMLメッセージは、要求元の監視アプリケーションに送信される。

[ステップ18]:送信元の監視アプリケーションは、当該XMLメッセージを受信する。モニタリング問い合わせインタフェースのWSDLについては事前に公開されおり、Audit Proxy毎に異なることも許容している。そこで、監視アプリケーションは、当該XMLメッセージをチェックし、当該XMLメッセージで符号化されている前述status情報を、監視アプリケーション自身で定義するビジネスプロセス上の状態に対応付ける。

[ステップ19]:送信元の監視アプリケーションが、全てのモニタリング問い合わせインタフェースのレスポンスであるXMLメッセージを自身すると、監視クライアントを介して進捗状況について、システム運用管理者に表示できる様に、指定された画面作成処理を行う。

３．７　PF 共通機能（ユーティリティ機能）

本節では、PF 共通機能における、「ユーティリティ機能」について、その位置付けや機能概要についての技術解説を示す。PF 共通機能におけるユーティリティ機能とは、データ交換、サービス連携、プロセス連携などを実現する地域情報 PF の開発、実行、運用に関連し、横断的に使用される下記の機能を集めたものであり、アーキテクチャ標準仕様４．５．３プラットフォーム共通機能に規定されている。

(1) 時刻同期機能
(2) サービスレジストリ機能
(3) リポジトリ機能
(4) 統合レジストリ機能
(5) ビジネスメッセージルーティング-ゲートウェイ（BMR-GW）機能

３．７．１　時刻同期機能

地域情報 PF 標準を採用し動作するサーバ等のマシンの「時刻同期機能」を実施する機能である。

（１）時刻同期機能を必要とする条件

a. マシン間の時刻が一致していることを前提にしている機能がある。
　モニタリング機能、監査証跡機能、等
b. 問題発生時のトラブルシューティング時に各マシン間の時刻を踏まえたログなどから責任切り分けなどを実施する必要があるため。

（２）時刻同期機能とシステム例

時刻同期の方法と運用上の時刻同期処理に関しては、NTP サーバと NTP クライアントの設置と設定に関する定義が、アーキテクチャ標準仕様の４．５．３．３　ユーティリティ機能に記載されているので参照ください。図３．７．１に、階層的に配置した NTP サーバと NTP クライアントのシステム例を示す。サイト内の業務ユニットや BPM 機能や統合 DB 機能等のサーバマシンが、NTP クライアントとなる。NTP クライアントは、マシンが起動された段階で、サイト内の NTP サーバにアクセスし、時刻同期を行う。

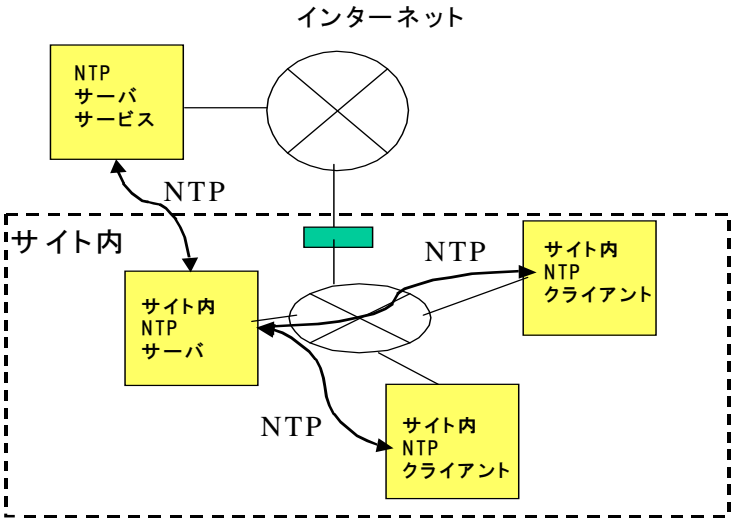


図３．７．１　時刻同期の実装例

３．７．２　サービスレジストリ機能

サービスレジストリ機能とは、サービス情報を管理する蓄積庫である。サービス情報のライフサイクル（登録、更新、削除）を管理し、これら登録されたサービス情報の検索できるサービスを提供する機能である。各種 Web サービスの登録・検索・公開するケース、もしくは対象者の情報を持つサイトのサービスを検索するケースがある。また前者のケースには動的にサービス先を探す UDDI 等を使用する場合と、サービス情報を Web サイト等で公開する静的な方法がある。詳細な定義は、アーキテクチャ標準仕様の４．５．３．３　ユーティリティ機能に規定されている。

地域情報 PF によるサービスレジストリ機能のユースケースを下記に示す。

ア）ユースケース１：　PF 標準仕様に基づく各種 Web サービスの登録・検索・公開
管理者：各種サービスを提供するコミュニティの主体者（地域ポータル運営者、など）
使用者：登録された各種 Web サービスを利用し、地域サービスなどを提供する主体者
要件　：管理者が管理・登録する Web サービス（自治体や民間等）のカタログ情報を使用者が検索、ダウンロードできる。

イ）ユースケース２：　対象者の情報を持つサイトのサービスを検索する場合
管理者：自治体の Web サービスと ID 情報の紐づけ情報の管理者
使用者：住民の情報を取得する自治体職員もしくは住民
要件　：住民（ID 情報）と紐づいた Web サービス（自治体や民間等）を使用者が検索できる。
自治体の Web サービスのライフサイクルを管理（登録、変更、削除）できる。
自治体の Web サービスと住民（ID 情報）の紐付け情報のライフサイクルを管理（登録、変更、削除）できる。

本ユースケースに対する技術仕様として、ID-WSF が考えられる。ID-WSF については具体的な業務ユースケースへの適合性、標準化状況、普及の状況を鑑みて、今後採用候補として検討する。（付録 5 参照）

３．７．３　リポジトリ機能

リポジトリ機能とは、標準仕様関連書類、システム開発仕様、プログラム、用語定義、項目辞書等を、蓄積・管理するデータベース等の蓄積機能である。詳細な定義は、アーキテクチャ標準仕様の４．５．３．３　ユーティリティ機能に規定されている。

地域情報 PF によるリポジトリ機能のユースケースを下記に示す。

ア）ユースケース１：　PF 標準仕様のバージョン別の各種ドキュメントの蓄積と公開
管理者：PF 標準仕様の管理者
使用者：公開する各種ドキュメントを参照する人（自治体、製品ベンダ他　など）
要件　：管理者は、地域情報 PF 標準仕様運用規則に基づき、下記の PF 標準仕様の管理と使用者への公開ができること。

a)ドキュメント類の例
・自治体業務アプリケーションユニット標準仕様
・アーキテクチャ標準仕様、プラットフォーム通信標準仕様
・地域情報 PF 準拠及び相互接続仕様
・地域情報プラットフォームガイドライン
・地域情報 PF 基本説明書

b)XML 定義類の例

・自治体業務アプリケーションユニット標準仕様で規定される XML 定義類
-業務ユニットインタフェースに関する標準規定の XML 定義類(WSDL/XSD)
-ワンストップサービスに関するサンプル XML 定義類(BPEL/WSDL/XSD 等)

イ）ユースケース２： PF 標準仕様を活用した自治体の設計共通リポジトリ（文書管理）
管理者：地域情報 PF に対応した自治体システムを調達～運用・保守する自治体の管理者
使用者：地域情報 PF に対応した自治体システムの開発を実施する担当者
要件 ：管理者は、使用者が作成した下記の自治体システムに関する設計情報の管理ができる。
・PF 標準仕様や、自治体個別要件に基づきカスタマイズした自治体内標準仕様書
・工程管理資料、設計ドキュメント、UML、ソースコード 等

３．７．４ 統合レジストリ機能

「統合レジストリ機能」について記載する。本機能はオプションである。

（１）統合レジストリ機能の目的

高付加価値サービスをサービス利用者に対して迅速に安定して提供するためには、サービス提供者またはサービス管理者がサービスの運用状況や、サービス利用者との合意に基づいた目標品質を管理・把握し、これら「サービス情報」をサービスの品質にフィードバックすることが可能なサービス管理基盤が必要である。

そこで多様化するサービス利用者とサービス提供者間の関係（サービス利用形態）を二者間の「合意」として策定し、UDDI(Universal Description, Discovery and Integration)、等の従来のレジストリが管理する情報とあわせて「サービス情報」として管理・監視する統合レジストリ機能を用い、高付加価値サービスの効率的な提供を実現する。

（２）統合レジストリ機能の利用シーンについて

住民が地域ポータルを経由して転居などの各種申請を行う際、地域ポータルの裏では自治体の提供する各種サービスや民間企業の提供する住所変更サービスとの連携が必要となる。このように連携した形でサービス（高付加価値サービス）を提供する際、複数サイトを跨ってのセキュリティ（暗号化）の確保や、必要に応じたログの提供、取得を行うことが必要である。

これらのニーズに対し、あらかじめサービス利用者とサービス提供者との間でセキュリティ（暗号化）や、目標品質（サービスレスポンスタイム）の管理について合意交渉を行うことで、住民に対し必要な情報を提供すると同時に、サービス管理者の運用状況把握も容易にすることで、安定した高付加価値サービスの提供をすることが可能となる。また、サービス間の連携や、その際の品質・運用状況の管理を行う場合、既存のサービスに対し接続や管理に関する機能追加や運用フローの拡充などが必要となるが、それら機能追加をエージェント機能が提供することで連携する地域ポータルのサービス管理者、サービス提供者の双方にとって負担が軽減され、効率的なサービス提供を実現することができる。

図３．７．２に統合レジストリ機能の利用シーンを示す。

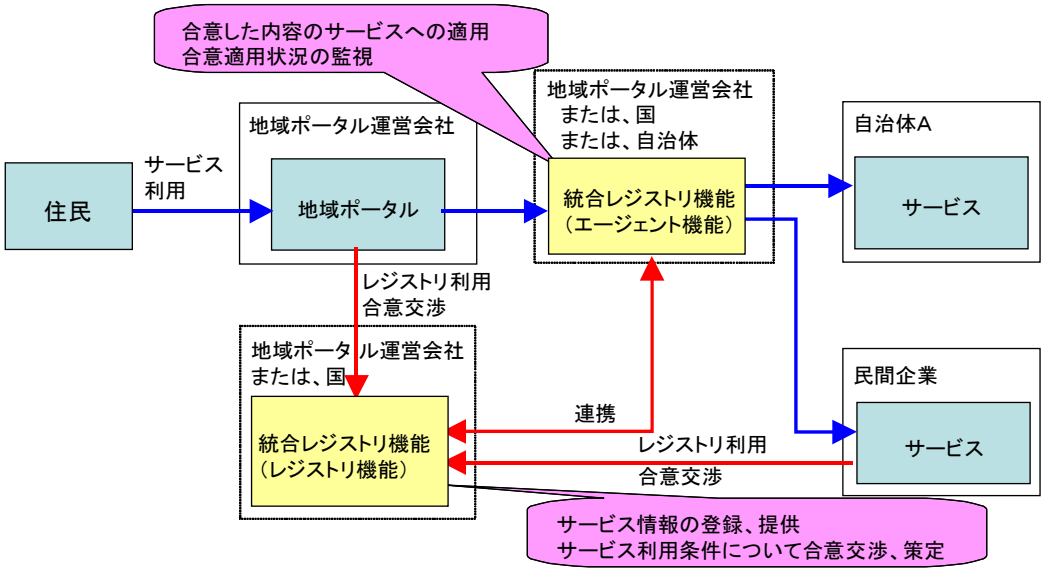


図３．７．２　統合レジストリ機能の利用シーン

（３）統合レジストリ機能の処理概要

アーキテクチャ標準仕様４．５．３．３（４）②統合レジストリの機能要件に記載されている機能要件を満たした統合レジストリ機能を利用することにより、実現できる処理フローと、処理概要を示す。

ア）統合レジストリ機能の処理フロー

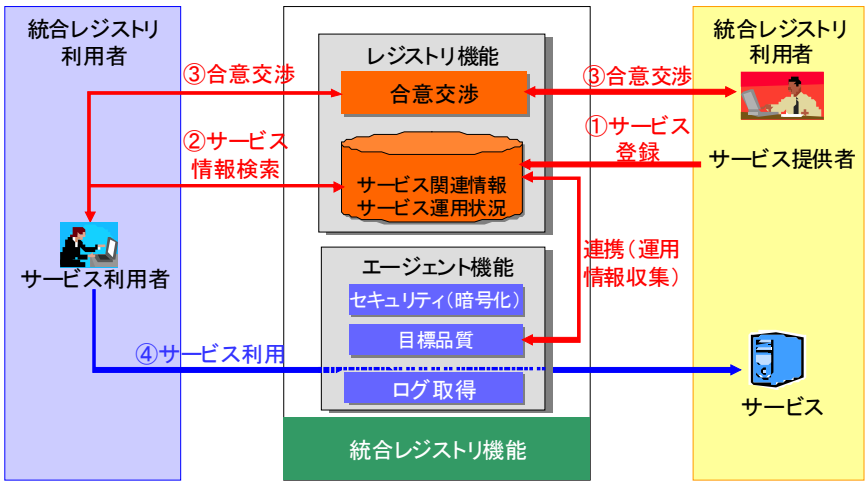


図３．７．３　統合レジストリ機能の処理フローと処理概要

イ）統合レジストリ機能の処理機能

a)、b)のエージェント機能を提供することができる。

a)暗号化処理

サイトを跨ってサービス利用者とサービス提供者間でのメッセージを交換する際、サービス

提供者側がメッセージ暗号化を行えない場合、エージェントがメッセージ暗号化を代理することによる通信路上のセキュリティの確保

b)目標品質（サービスレスポンスタイム）の管理

サイトを跨ってサービス利用者とサービス提供者間がサービス連携をする際、予めサービス利用者とサービス提供者の間で取り交わしたレスポンスタイムなどに基づき、サービス提供者側のリクエスト、レスポンスタイムの計測による目標品質の管理ができる

（4）統合レジストリ機能の実装モデル

ア)エージェント機能の Web サービス化

実装モデルとして、アーキテクチャ標準仕様４．５．３．３（４）③の図を図３．７．４に示す。

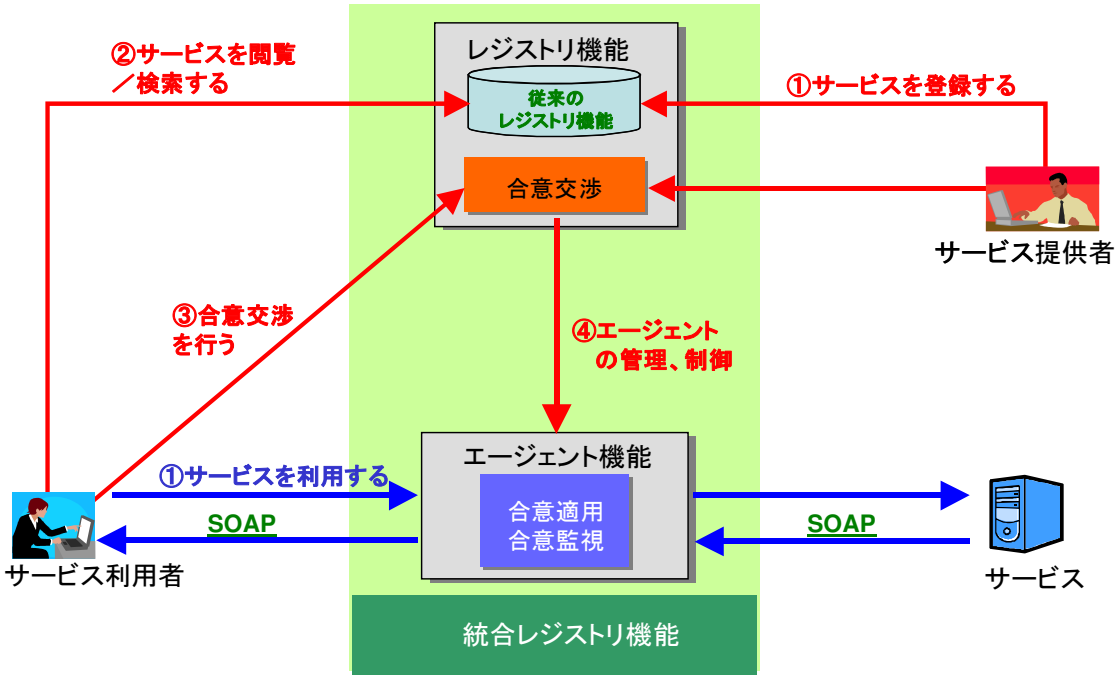


図３．７．４　統合レジストリ機能の実装モデル

３．７．５　ビジネスメッセージルーティングゲートウェイ機能（BMR-GW）

BMR-GW 機能は、業務メッセージの送信先を、受信した申請書類の内容や業務プロセスの処理結果に応じて動的に変更する要求がある。例えば、ポータルからの情報照会の申請書類から照会先団体を取り出し、その団体名に基づいて照会依頼を送信するような利用シーンが考えられる（図３．７．５）。あるいは、ある団体からの業務メッセージの処理結果を後日返信する場合、業務メッセージの送信元に応じた返信先の動的変更が必要となるような利用シーンが考えられる(図３．７．６)。

地域情報 PF では、この動的な業務メッセージ送信先の制御機能を「ビジネスメッセージルーティングゲートウェイ機能（略称：BMR-GW 機能)」と呼ぶ。BMR-GW 機能は、オプションであるが、複数個所から呼び出される非同期呼び出しがある場合は必須となっている。

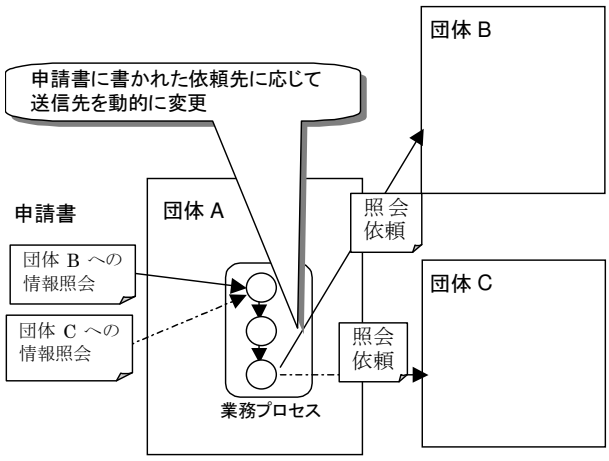


図３．７．５ 動的な送信先変更

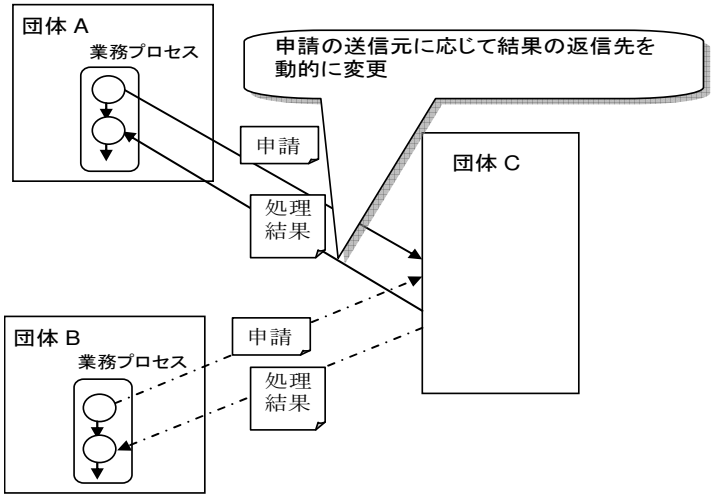


図３．７．６　送信元に応じた動的な返信先の変更

WS-BPEL で記述されたビジネスプロセスで送信先の動的変更を実現する場合、WS-BPEL 仕様に従って EndpointReference を使用することが考えられる。しかし、ビジネスプロセス配備時に送信先を静的にバインドする WS-BPEL 処理系の場合、こうした機能が使用できない可能性がある。また、一般的な業務ユニットにおいても、複数の呼び出し元から非同期で呼び出される場合や、業務内容により呼び出し先の業務ユニットを動的に変える必要がある。このため、動的に送信先を変更するゲートウェイ（BMR-GW）機能をプラットフォームに用意する。

３．８　メッセージ交換パターンと異常系処理

本章では、メッセージパターンと異常系処理の下記事項について概説する。

(1) メッセージ交換パターンの種類と選定理由：

想定されるメッセージ交換パターンについて説明した上で、プラットフォーム通信標準仕様として定めたメッセージ交換パターンの選定理由について説明する。

(2)応答側処理におけるメッセージ交換パターンの制約：

データ交換システムパターンに関連し、組み合せて利用されるメッセージ交換パターンのうち、特に応答側処理における制約、奨励事項について説明する。

(3)異常系処理における問題解決、障害復旧対応の概説：

問題を複雑化させないように、地域情報プラットフォームでは、異常系処理については、プラットフォーム通信標準仕様に関係する事項のみに限定している。ここでは、地域情報プラットフォームにおける問題解決のための手順概要を説明する。

(4)異常系処理における留意するべき事項　：

異常時の WS-BPEL(Web Services Business Process Execution Language 2.0)プロセスをどのように扱うかなどをはじめとして、障害発生時に留意すべき事項について概説する。

３．８．１　メッセージ交換パターンの種類と選定理由

（１）考えられ得るメッセージ交換パターンの定義

（１）　メッセージ交換パターン類型

メッセージ交換パターンの検討にあたっては、レスポンスの有無により「リクエスト型」と「リクエスト・レスポンス型」の 2 種類に分類し、その上で受領 Ack(Acknowledgement)の有無、受領 Ack もしくはレスポンス受信方法の同期、非同期等の観点で細分化した。この結果、下記７パターンが定義される。

・リクエスト型受領 Ack なし
・リクエスト型受領 Ack あり
・リクエスト型非同期型受領 Ack
・リクエスト・レスポンス型同期型レスポンス
・リクエスト・レスポンス型非同期型レスポンス
・リクエスト・レスポンス型同期型受領 Ack＋非同期型レスポンス
・リクエスト・レスポンス型非同期型受領 Ack＋非同期型レスポンス

これらのメッセージ交換パターンについて以下に説明する。なお、以降の説明において使用する用語および表記法については、プラットフォーム通信標準仕様の６．１ 概念、用語の定義に準じる。

（２）　リクエスト型

リクエスト型のメッセージ交換パターンとは、要求メッセージのみで構成され、業務処理結果を含んだ応答メッセージが存在しないパターンである。リクエスト型の例としては、業務ユニットからの帳票出力指示、バッチプログラムや他の業務ユニットへの通知などが想定される。リクエスト型のメッセージ交換では、受領 Ack の有無および受信の仕方により以下の３パターンが存在する。

a.リクエスト型受領 Ack なし

開始側から処理要求等の要求メッセージを送信し、応答側では要求メッセージに対する受領 Ack を含めて応答を返さないパターンである。

【特徴】
・送信のみであるため実装が容易である。
・受領 Ack も応答メッセージも相手から通知されないため、SOAP(Simple Object Access Protocol)通信レベルよりも上位の送達確認を行うことができない。
・障害検知も SOAP レベルから下位となるため、他の方法による上位層の障害検知が必要となる。

【シーケンス】
・開始側：要求メッセージ送信
・応答側：要求メッセージ受信 ⇒ 内容に基づく処理

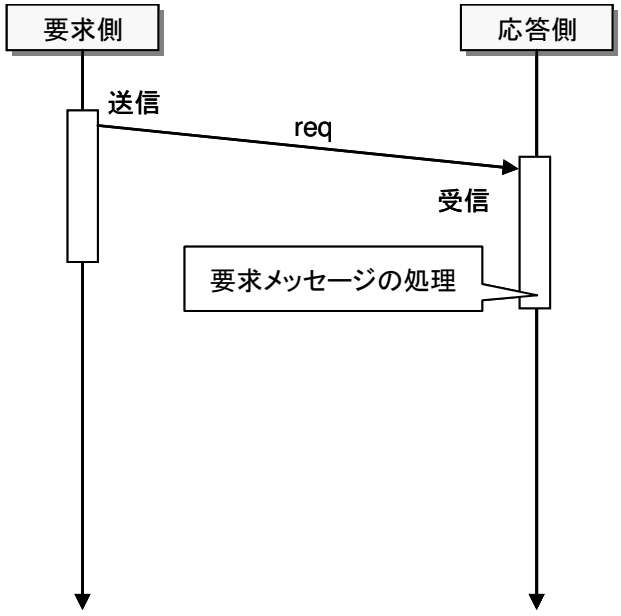


図３．８．１　リクエスト型受領 Ack なし

b.リクエスト型受領 Ack あり
同期型のメッセージ交換パターンである。開始側は、処理要求等の要求メッセージを送信するとともに、同一セッションで応答側からの受領 Ack の受信も行う。一方、応答側では、要求メッセージに対する応答メッセージは返さないが、受領 Ack を同一セッションにて送信するパターンである。

【特徴】
・比較的簡単なプロセス定義、サービスプログラミングにより、応答側での要求メッセージの受信の成功もしくは失敗を検知できる。
・開始側にて送達確認を行うことができる。
・応答側での処理結果が開始側に返されないため、要求メッセージに基づく処理が成功したかどうかを開始側で確認できない。

【シーケンス】
・開始側：要求メッセージ送信 ⇒ 受領 Ack 受信 ⇒ 受領 Ack チェック

・応答側：(要求メッセージ受信)⇒ 受領 Ack の生成 ⇒ 受領 Ack 返信 ⇒ 要求に基づく処理

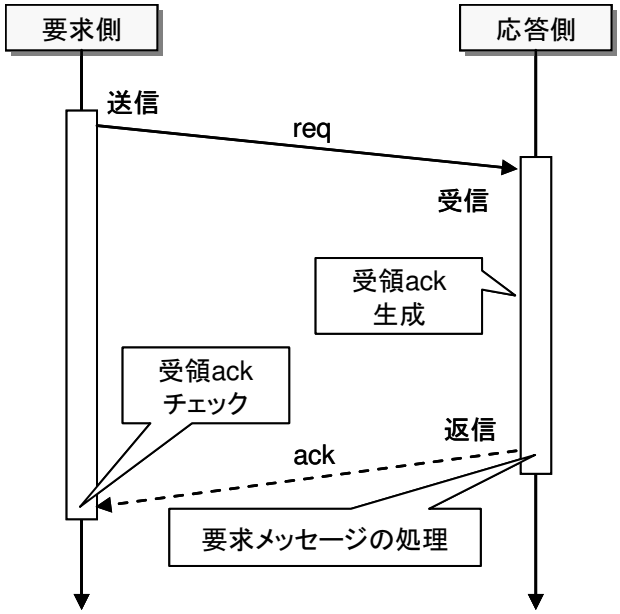


図３．８．２　リクエスト型受領 Ack あり

c. リクエスト型非同期型受領 Ack

要求メッセージに同期して同一のセッションで受領 Ack を返すのではなく、非同期で受領 Ack を返すパターンである。開始側は、処理要求等の要求メッセージを送信してセッションを終了させ、応答側からの受領 Ack を、別セッションにて受信する。同様に、応答側では、要求メッセージを受信したならセッションを終了させ、別セッションにて受領 Ack を開始側に送信するパターンである。

【特徴】

・開始側における障害検知能力としては、前述(b)の同期型受領 Ack と同等である。

・前述の「b. リクエスト型受領 Ack あり」と比較して処理方法が複雑になる可能性がある。

・開始側のセッションタイムアウト時間よりも、応答側の受領 Ack 生成に時間がかかる場合に有効である。

【シーケンス】

・開始側：要求メッセージ送信 ⇒ 受領 Ack 受信（非同期）⇒ 受領 Ack チェック

・応答側：要求メッセージ受信 ⇒ 受領 Ack 生成 ⇒ 受領 Ack 送信（非同期）⇒ 要求に基づく処理

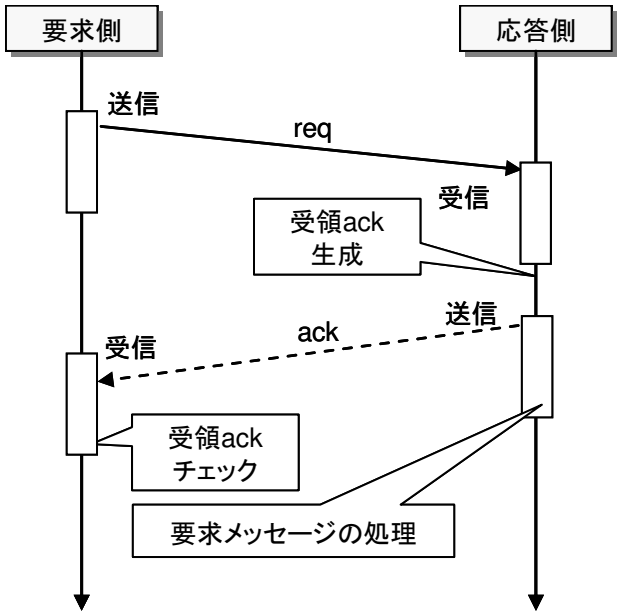


図３．８．３　リクエスト型非同期型受領 Ack

（３）　リクエスト・レスポンス型

このパターンはリクエストとそれに対する業務データを含むレスポンスの組で構成される。受領 Ack の有無、および、受領 Ack とレスポンスの送受信の仕方で以下の４パターンが存在する。

a. リクエスト・レスポンス型同期型レスポンス

要求メッセージの送信と同一のセッションで、応答側からの業務処理結果情報の応答メッセージを返す同期型のパターンであり、受領 Ack の通知は行わない。開始側は、要求メッセージの送信を行った後に同一のセッションで応答側からの応答メッセージを受信する。一方、応答側は要求メッセージを受信後、受信確認用の受領 Ack を返すことなく、要求された業務処理を実施し、その後、その結果情報を開始側に応答メッセージとして返す。

【特徴】

- 開始側にて処理結果の確認を行うことができる。
- 照会処理や更新処理など幅広い処理に対応できる。
- 同一セッションにて処理結果を受信する方式のため、後述のメッセージ交換パターンと比較して実装が容易である。
- 開始側のセッションタイムアウト時間内に応答側の処理が終了して応答メッセージを返せるような、比較的軽い処理に有効である。
- 開始側のセッションのタイムアウト時間内に応答側の処理が終了しなかった場合、開始側ではすべてタイムアウトとして検知されるので、開始側では通信障害なのか、業務処理の遅延によるタイムアウトなのか判断できない。

【シーケンス】

・開始側：要求メッセージ送信 ⇒応答メッセージ受信 ⇒受信後の処理
・応答側：要求メッセージ受信 ⇒ 要求に基づく処理、応答メッセージ生成 ⇒返信

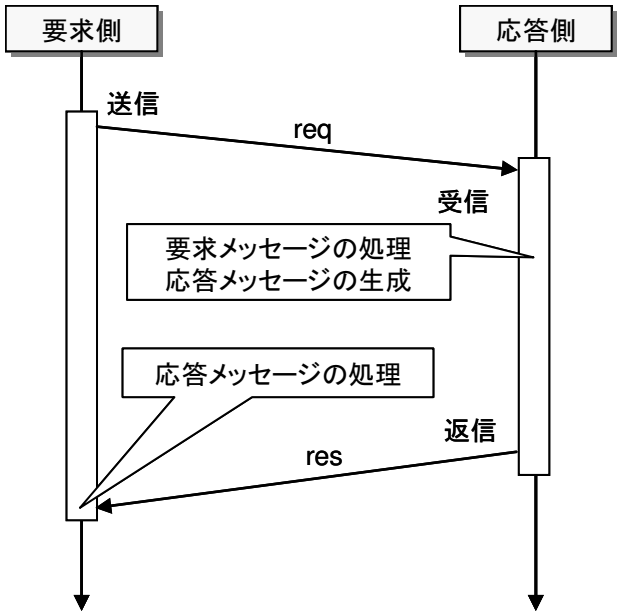


図３．８．４　リクエスト・レスポンス型同期型レスポンス

b.リクエスト・レスポンス型非同期型レスポンス

要求メッセージの送信とは別セッションで応答側からの業務処理結果情報の応答メッセージを返す非同期型のパターンであり、受領 Ack の通知は行わない。開始側は、要求メッセージの送信を行ったセッションとは別セッションで応答側から応答メッセージを受信する。一方、応答側は要求メッセージを受信してセッションを切断後、受信確認用の受領 Ack を返すことなく、要求された業務処理を実施し、その後に新たなセッションを開設して結果情報を開始側に応答メッセージとして返す。

【特徴】

開始側にて処理結果の確認を行うことができる。

照会処理や更新処理など幅広い処理に対応できる。

開始側から要求された処理が応答側にて比較的時間のかかる、あるいは、人手が介するような処理に有効である。

応答側で要求メッセージを受信したにも関わらず、後続の処理に失敗したなどの場合、開始側での応答メッセージの受信あるいは受信タイムアウトまで障害検知が行えない。

正常系における応答メッセージの送信に数時間～数日かかる処理において、要求メッセージの受信で検出される通信障害が発生した場合、迅速な異常系処理の対応が困難となる。(受領 Ack 等を返すパターンであれば即座に通信障害を検知できる)

【シーケンス】

・開始側：要求メッセージ送信 ⇒ 応答メッセージ受信 ⇒ 受信後の処理

・応答側：要求メッセージ受信 ⇒ 要求に基づく処理、応答メッセージ生成 ⇒ 返信

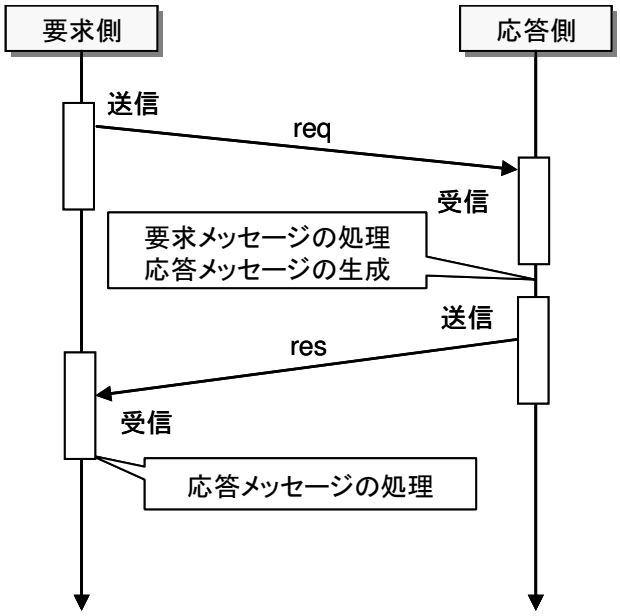


図３．８．５　リクエスト・レスポンス型非同期型レスポンス

c.リクエスト・レスポンス型同期型受領 Ack＋非同期型レスポンス

リクエスト・レスポンス型非同期型レスポンスと同様、要求メッセージの送信とは別セッションにて応答側から業務処理結果情報の応答メッセージを返す非同期型ではあるが、本パターンでは要求メッセージおよび応答メッセージのそれぞれに対して相手側から受領 Ack を受信するものである。なお、応答側（レスポンス送信側）にて応答メッセージに対する受領 Ack を無視することも許容する。

【特徴】

照会処理や更新処理など幅広い処理に対応できる。

開始側から要求された処理が応答側にて比較的時間のかかる、あるいは、人手が介するような処理に有効である。

同期型として同一セッションにて受領 Ack を受信するため、後述 d.の非同期型と比較して簡潔に双方向の受領 Ack を含むメッセージ処理が記述可能である。

要求メッセージの送信に失敗した場合、受領 Ack のタイムアウト検出により異常系処理の対応が迅速に行える。

【シーケンス】

・開始側：要求メッセージ送信 ⇒ 受領 Ack 受信 ⇒ 受領 Ack チェック ⇒ 応答メッセージ受信 ⇒ 受領 Ack 生成 ⇒ 受領 Ack 返信 ⇒ 受信後の処理

・応答側：要求メッセージ受信 ⇒ 受領 Ack 生成 ⇒ 受領 Ack 返信 ⇒ 要求に基づく処理 ⇒ 応答メッセージ生成 ⇒ 返信 ⇒ 受領 Ack 受信 ⇒ 受領 Ack チェック

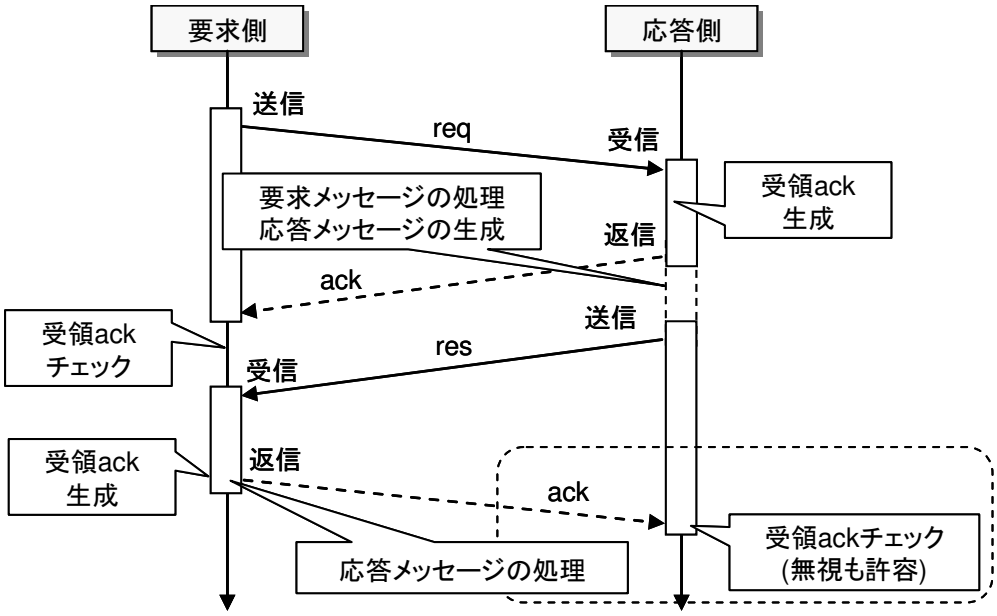


図３．８．６　リクエスト・レスポンス型同期型受領 Ack+非同期型レスポンス

d.リクエスト・レスポンス型非同期型受領 Ack＋非同期型レスポンス

これは要求メッセージに対して、同期の返信で応答メッセージを返さず、非同期で受領 Ack、応答メッセージをそれぞれ返すもので、要求メッセージ、応答メッセージとそれぞれの受領 Ack 全てを別のセッション（非同期）で送受信する。このパターンは、一組のリクエスト・レスポンス交換に 4 セッションを使用することになり、開始側、応答側ともに送受信処理の記述が非常に複雑になる。

【特徴】

・照会処理や更新処理など幅広い処理に対応できる。
・開始側から要求された処理が応答側にて比較的時間のかかる、あるいは、人手が介するような処理に有効である。
・要求メッセージの送信に失敗した場合、受領 Ack のタイムアウト検出により異常系処理の対応が迅速に行える。
・非同期型として別セッションにて受領 Ack を受信するため、複雑な処理になる。

【シーケンス】

・開始側：要求メッセージ送信 ⇒ 受領 Ack 受信 ⇒ 受領 Ack チェック ⇒ 応答メッセージ受信 ⇒ 受領 Ack 生成 ⇒ 受領 Ack 返信 ⇒ 受信後の処理
・応答側：要求メッセージ受信 ⇒ 受領 Ack 生成 ⇒ 受領 Ack 返信⇒ 要求に基づく処理 ⇒ 応答メッセージ生成 ⇒ 返信 ⇒ 受領 Ack 受信 ⇒ 受領 Ack チェック

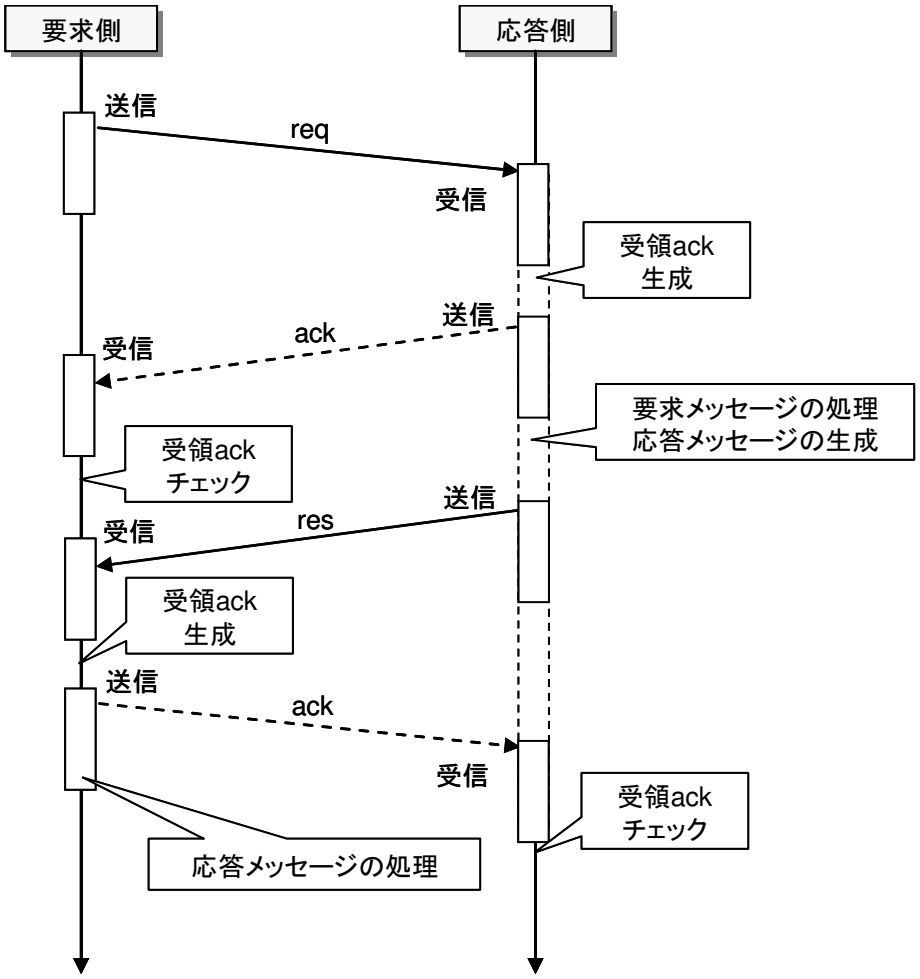


図３．８．７　リクエスト・レスポンス型非同期型受領 Ack+非同期型レスポンス

（２）選定理由

地域情報プラットフォームとしては、

・実装が容易であること。
・障害検知が容易であること。
・障害検知の仕組みが明確に定義できること。
・業務の要件に応じた最適な方式が選択可能であること。
・SOAP 1.1 等の Req-Res パターン合致すること。

という観点に基づいて、これまで述べたメッセージ交換パターンの評価を行い、次の３種類のメッセージ交換パターンを標準仕様として採用した。なお、利用に際しては業務処理時間やタイムアウト時間などを含めた業務要件に応じてメッセージ交換パターンを選択する必要がある。

【標準仕様】

・リクエスト型受領 Ack あり
・リクエスト・レスポンス型同期型レスポンス
・リクエスト・レスポンス型同期型受領 Ack＋非同期型レスポンス

３．８．２　応答側処理におけるメッセージ交換パターンの制約

（1）概説

プラットフォーム通信標準仕様では、メッセージ交換パターンを要求側と応答側間の一般的なメッセージのやり取りとして定義している。このため、地域情報プラットフォームの全ての点で適用可能である。しかし、応答側は実際には応答側 BPM(Business Process Management)機能、並びに応答側業務ユニットから構成され、この中でも要求側と応答側間のメッセージ交換パターンと同様のメッセージ交換パターンを利用する。そして、一部にはメッセージ交換パターンの組み合せに関する制約事項も存在する。この様な制約事項は、概ね、データ交換システムパターンとメッセージ交換パターンの関係の中で定義されているが、ここではメッセージ交換パターンの組み合せ、特に発生する制約事項に関してより具体的に概説する。

制約を大別すると２つに分類される。一つは、応答側が一つの BPM 機能と一つの業務ユニットから成る基本的な構成における制約である。もう一方は、応答側が一つの BPM 機能に対して、複数業務ユニットから成る一般的な構成における制約である。そこで以下、奨励する基本的構成、一般的構成、そして奨励しない構成について説明する。

（2）奨励する基本的な構成

図３．８．８～図３．８．１０は、応答側が応答側 BPM 機能、並びに一つの応答側業務ユニットで構成されている場合に、組み合わされて実施されるメッセージ交換パターンの手順について記したものである。この場合、要求側は、要求側 BPM 機能、要求側業務ユニットのいずれも可能である。

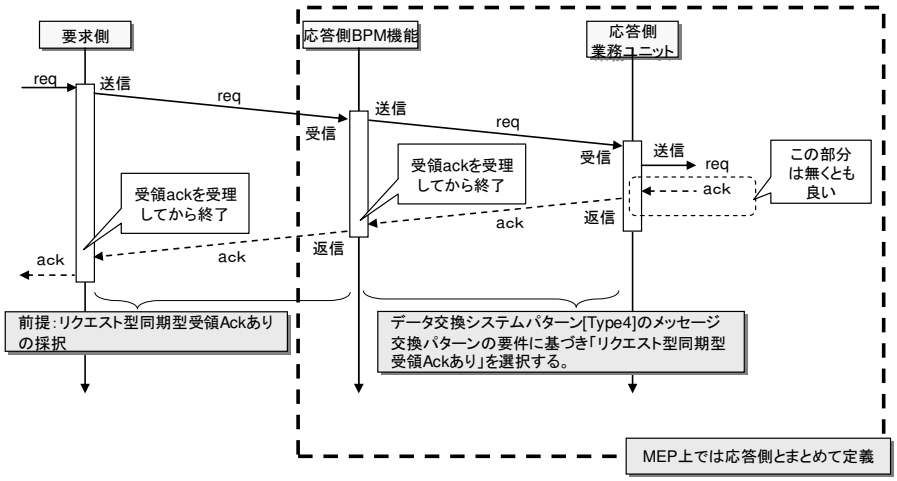


図３．８．８　リクエスト型受領 Ack ありの組み合せ制約(基本的な構成)

図３．８．８は、要求側～応答側間のメッセージ交換パターンが、「リクエスト型受領 Ack あり」となった場合の一連の処理フローである。この場合、応答側の内部である応答側 BPM 機能～応答側業務

ユニット間で為されるメッセージ交換パターンにおいても、「リクエスト型受領 Ack あり」が選択される必要がある。これは、データ交換システムパターン[Type4]のメッセージ交換パターンの要件に基づく。応答側業務ユニットにおいては、業務ユニット内部で受領 Ack を明示的に戻す必要はなく、業務ユニット部分の SOAP 処理系が受領 Ack を戻すことでもよい。

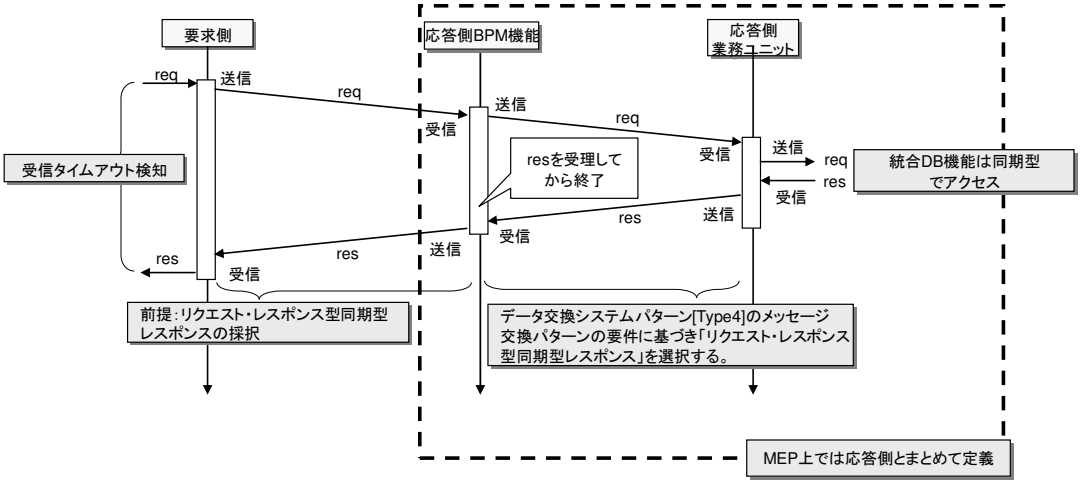


図３．８．９　リクエスト・レスポンス型同期型レスポンスの組み合せ制約(基本的な構成)

図３．８．９は、要求側～応答側間のメッセージ交換パターンが、「リクエスト・レスポンス型同期型レスポンス」となった場合の一連の処理フローである。この場合、応答側の内部である応答側 BPM 機能～応答側業務ユニット間で為されるメッセージ交換パターンにおいても、「リクエスト・レスポンス型同期型レスポンス」が選択される。これは、データ交換システムパターン[Type4]のメッセージ交換パターンの要件に基づく。また、応答側業務ユニットが、さらに統合 DB 機能に対してアクセスする場合、データ交換システムパターン[Type5]のメッセージ交換パターンの要件に基づき、「リクエスト・レスポンス型同期型レスポンス」が利用される。また、要求側の業務ユニットでは受信タイムアウトについての検知が行なわれる。

なお、図３．８．９の方法は、リクエスト・レスポンス型同期型レスポンスの組み合わせであるため、比較的短時間に要求側～応答側間のメッセージ交換パターン全体が終了することが期待される。従って、応答側業務ユニットにおいて、自治体職員等の操作者が介在して処理を進める場合、若干のレスポンスの遅れに対しても、要求側の業務ユニットで受信タイムアウトとなるリスクも存在する。このため、応答側業務ユニットにおいて自治体職員等の操作者が介在して処理を進める場合は、これは奨励しない。

これに対して、図３．８．１０は要求側～応答側間のメッセージ交換パターンが、「リクエスト・レスポンス型同期型受領 Ack+非同期型レスポンス」となった場合の一連の処理フローである。この場合、

応答側の内部である応答側 BPM 機能～応答側業務ユニット間で為されるメッセージ交換パターンにおいても、「リクエスト・レスポンス型同期型受領 Ack+非同期型レスポンス」が選択される。この場合も要求側の業務ユニットでは、非同期に受信する応答メッセージに対して受信タイムアウトについての検知が行なわれるが、非同期型レスポンスであるため、十二分に長い時間が想定される。このため、応答側業務ユニットにおいて自治体職員等の操作者が介在して処理を進める場合は、この方法を奨励する。

なお、応答側業務ユニットが、さらに統合 DB 機能に対してアクセスする場合、データ交換システムパターン[Type5]のメッセージ交換パターンの要件に基づき、「リクエスト・レスポンス型同期型レスポンス」が利用されることになる。

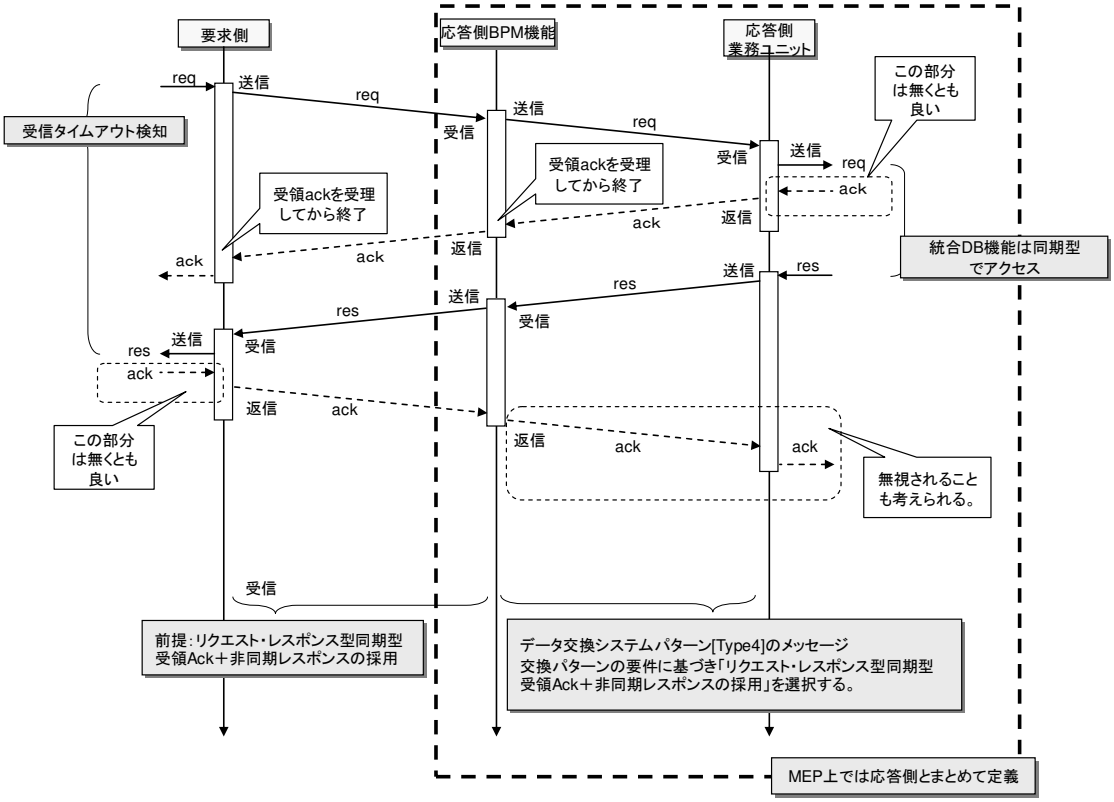


図３．８．１０　リクエスト・レスポンス型同期型受領 Ack+非同期型レスポンスの組み合せ制約(基本的な構成)

(3)奨励する一般的な構成

図３．８．１２～図３．８．１４は、応答側が一つの応答側 BPM 機能、並びに複数の応答側業務ユニットで構成されている場合に、組み合わされて実施されるメッセージ交換パターンの手順について記したもので、一般化された構成を記したものある。この場合、要求側は、要求側 BPM 機能、要求側業務ユニットのいずれも可能である。

図３．８．１２の上段は、要求側～応答側間のメッセージ交換パターンが、「リクエスト型受領 Ack あり」となった場合の一連の処理フローであり、応答側業務ユニット呼び出しを直列化して実施する場合である。WS-BPEL では、構造化アクティビティとして sequence を用いる。

これに対して下段は、要求側～応答側間のメッセージ交換パターンは、上段と同様に「リクエスト型受領 Ack あり」となった場合の一連の処理フローであるが、応答側業務ユニット呼び出しを並列化

して呼び出す場合である。WS-BPEL では、構造化アクティビティとして並列化する部分に対して flow を用い、並列化された各々は sequence を指定する。

図３．８．１１の上下段に関係なく、データ交換システムパターン[Type4]のメッセージ交換パターンの要件、並びに図３．８．８に記した基本的な構成に基づき、応答側の内部である応答側 BPM 機能～応答側業務ユニット間で為されるメッセージ交換パターンにおいても､「リクエスト型受領 Ack あり」が選択される。各々の応答側業務ユニットにおいては、業務ユニット内部で受領 Ack を明示的に戻す必要はなく、各々の応答側業務ユニットの SOAP 処理系が受領 Ack を戻すことでもよい。

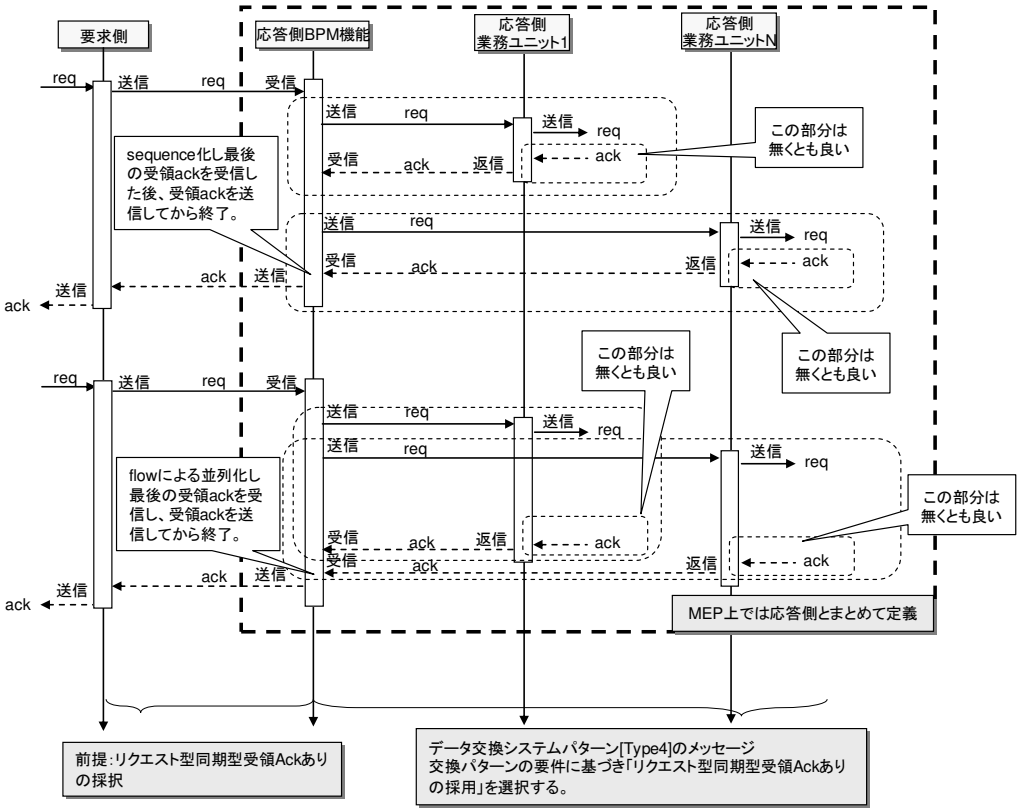


図３．８．１１　リクエスト型受領 Ack ありの組み合せ制約(一般的な構成)

図３．８．１２は、要求側～応答側間のメッセージ交換パターンが、「リクエスト・レスポンス型同期型レスポンス」となった場合の一連の処理フローである。この場合、データ交換システムパターン[Type4]のメッセージ交換パターンの要件、並びに図３．８．９に記した基本的な構成に基づき、応答側の内部である応答側 BPM 機能～応答側業務ユニット間で為されるメッセージ交換パターンにおいても、「リクエスト・レスポンス型同期型レスポンス」が選択される。

複数の応答側業務ユニットを呼び出す場合、直列化して実施する、並列化して実施するどちらでも可能である。この場合、WS-BPEL では、構造化アクティビティとして並列化する部分に対して flow を用い、並列化された各々は sequence を指定する。但し、「リクエスト・レスポンス型同期型レスポンス」であるため、受信タイムアウトは比較的短い時間が設定されることも想定されるため、並列化して実施することを奨励する。

また、応答側業務ユニットにおいて自治体職員等の操作者が介在して処理を進める場合、若干のレスポンスの遅れに対しても、要求側の業務ユニットで受信タイムアウトとなるリスクが存在する。このため、応答側業務ユニットにおいて自治体職員等の操作者が介在して処理を進める場合は、これは奨励しない。

各々の応答側業務ユニットが、さらに統合 DB 機能に対してアクセスする場合、データ交換システムパターン[Type5]のメッセージ交換パターンの要件に基づき、「リクエスト・レスポンス型同期型レスポンス」が利用される。また、要求側の業務ユニットでは受信タイムアウトについての検知が行なわれる。これらは、基本的な構成と同様である。

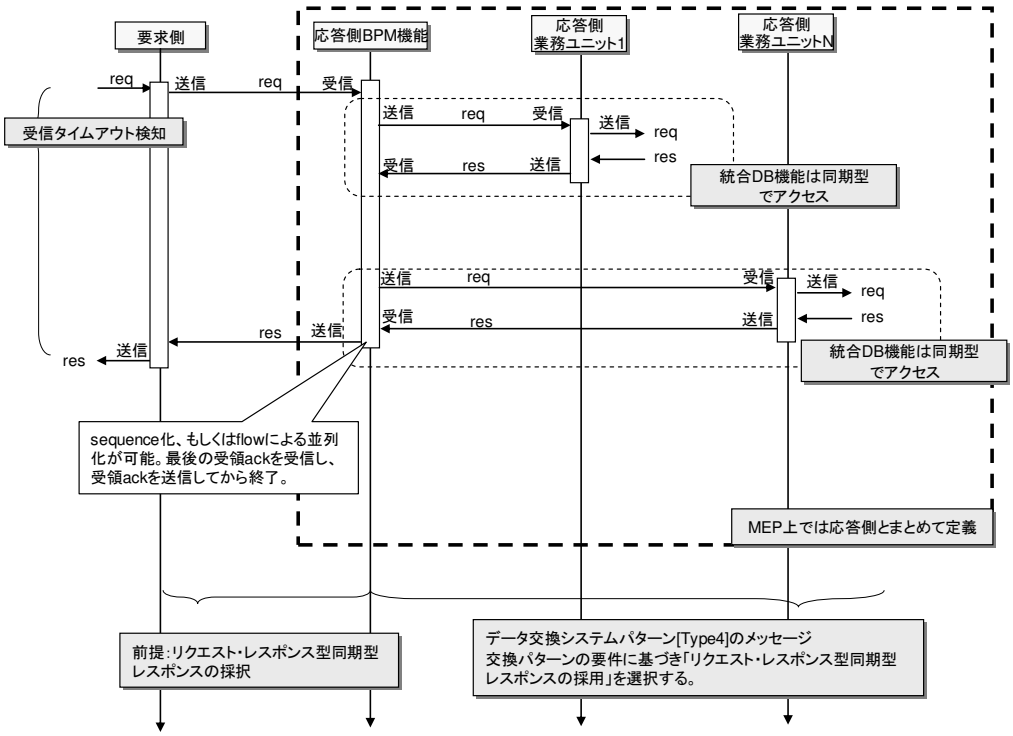


図３．８．１２　リクエスト・レスポンス型同期型レスポンスの組み合せ制約(一般的な構成)

図３．８．１３は、要求側～応答側間のメッセージ交換パターンが「リクエスト・レスポンス型同期型受領 Ack+非同期型レスポンス」となった場合の一連の処理フローである。この場合、要求側～応答側間のメッセージ交換パターンには「リクエスト・レスポンス型同期型受領 Ack+非同期型レスポンス」を用い、応答側の内部である応答側 BPM 機能～応答側業務ユニット間で為されるメッセージ交換パターンにおいても、「リクエスト・レスポンス型同期型受領 Ack+非同期型レスポンス」が選択される。そして、応答側 BPM 機能は一連の動作を全て制御する。

この場合も要求側の業務ユニットでは、非同期に受信する応答メッセージに対して受信タイムアウトについての検知が行なわれるが、非同期型レスポンスであるため、十二分に長い時間が想定される。このため、応答側業務ユニットにおいて自治体職員等の操作者が介在して処理を進める場合は、この方法を奨励する。

なお、留意事項として応答側 BPM 機能では、リクエスト(req)とレスポンス(res)を分割して記述してはならない。これに対して、要求側の業務ユニットでは、リクエスト(req)とレスポンス(res)を一体化して処理してはならない。これらは、次節の奨励しない構成の中で解説する。

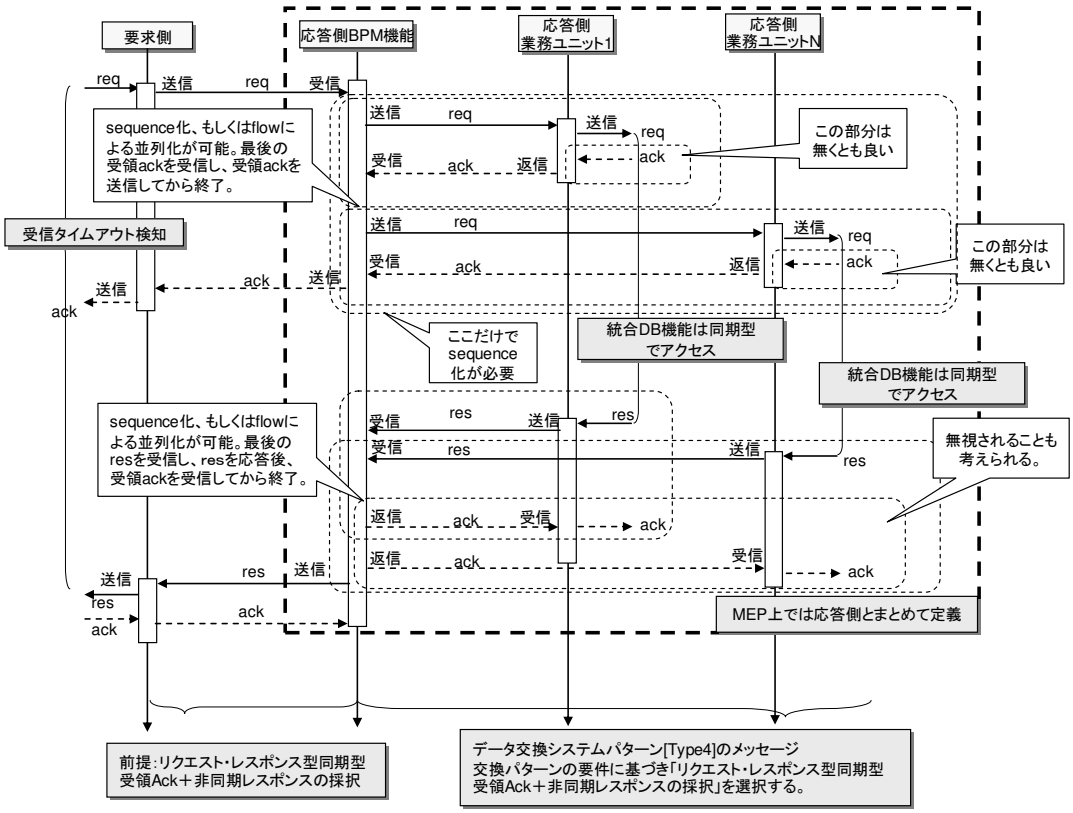


図３．８．１３　リクエスト・レスポンス型同期型受領 Ack+非同期型レスポンスの組み合せ制約(一般的な構成)

図３．８．１３においても、図３．８．１２と同様に、複数の応答側業務ユニットを呼び出す場合、直列化して実施する、並列化して実施するどちらでも可能である。直列化、並列化に関しては通常、障害（異常）の対処方法に応じて決める必要がある。このため、業務内容に依存して適切なものを選択しなければならない。応答側業務ユニット呼び出しを直列化して実施する場合、WS-BPEL では構造化アクティビティとして sequence を用いる。これに対して、並列化する場合は WS-BPEL では、構造化アクティビティとして並列化する部分に対して flow を用い、並列化された各々は sequence を指定する。また、各々の応答側業務ユニットが､さらに統合 DB 機能に対してアクセスする場合､データ交換システムパターン[Type5]のメッセージ交換パターンの要件に基づき､「リクエスト・レスポンス型同期型レスポンス」が利用される。また、要求側の業務ユニットでは受信タイムアウトについての検知が行なわれる。これらは、図３．８．１３や基本的な構成と同様である。

（４）奨励しない構成

図３．８．１４は、基本的構成に対して奨励しないメッセージ交換パターンである。図３．８．１４では、要求側～応答側間のメッセージ交換パターンが「リクエスト・レスポンス型同期型受領 Ack+非同期型レスポンス」であるのに対して、応答側の内部である応答側 BPM 機能～応答側業務ユニット間で為されるメッセージ交換パターンでは、同期型である「リクエスト・レスポンス型同期型レスポンス」を利用している。この方式は、可能ではあるが、データ交換システムパターン[Type4]のメッセージ交換パターンの要件に基づくと、図３．８．１０の「リクエスト・レスポンス型同期型受領 Ack+非同期型レスポンス」が優先的に利用されるべきであるため、奨励しない。

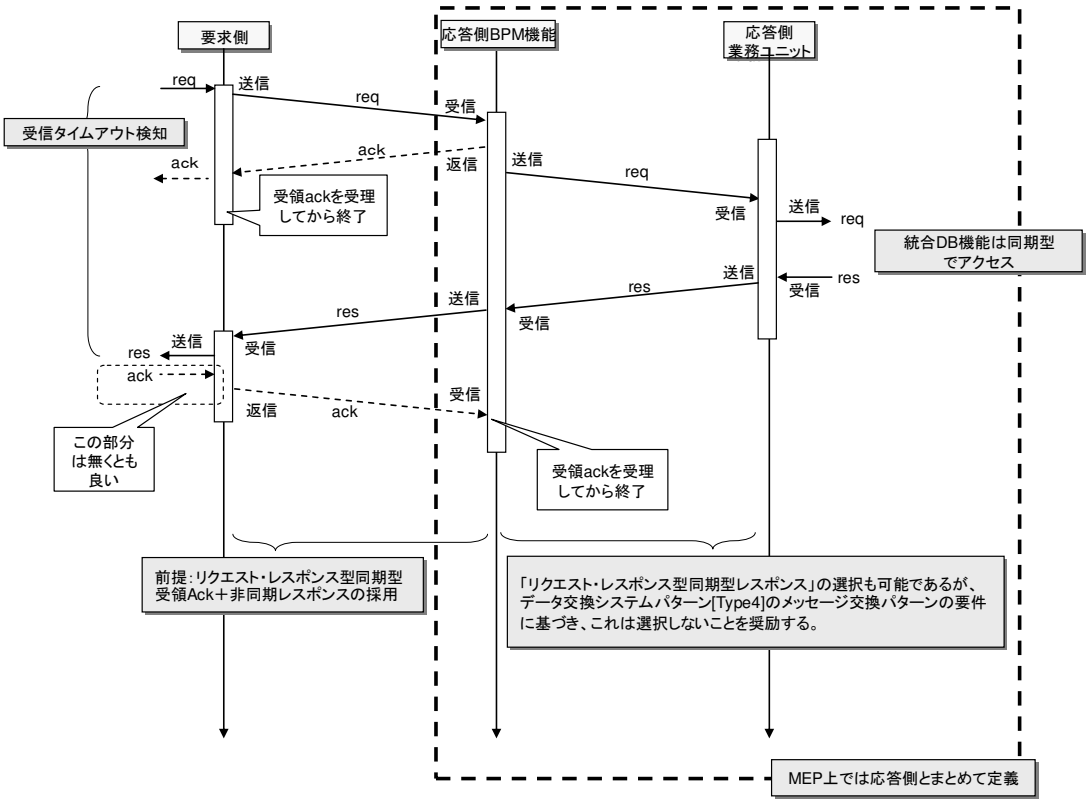


図３．８．１４ リクエスト・レスポンス型同期型受領 Ack+非同期型レスポンスの
組み合せ制約(奨励しない)

図３．８．１５～図３．８．１６は、一般的な構成に対して奨励しないメッセージ交換パターンである。図３．８．１５では、要求側～応答側間のメッセージ交換パターンが、「リクエスト・レスポンス型同期型レスポンス」であるのに対して、応答側の内部である応答側 BPM 機能～応答側業務ユニット間で為されるメッセージ交換パターンでは、「リクエスト・レスポンス型同期型受領 Ack+非同期型レスポンス」を利用している。この場合、応答側の内部である応答側 BPM 機能～応答側業務ユニット間で為されるメッセージ交換パターンが、比較的長時間に渡る処理を前提としているのに対して、要求側～応答側間のメッセージ交換パターンには、「リクエスト・レスポンス型同期型レスポンス」故に短い応答時間を期待されることになるため、要求側の業務ユニットで実施・検知される受信タイムアウトが頻発され易くなる。メッセージ交換パターンを遵守することは、障害（異常）の対処方法を標準化し、通信品質の向上を目指すことであるため、受信タイムアウトが頻発され易くなる危険性を内包するメッセージ交換パターンは、採択するべきではなく、奨励しない。

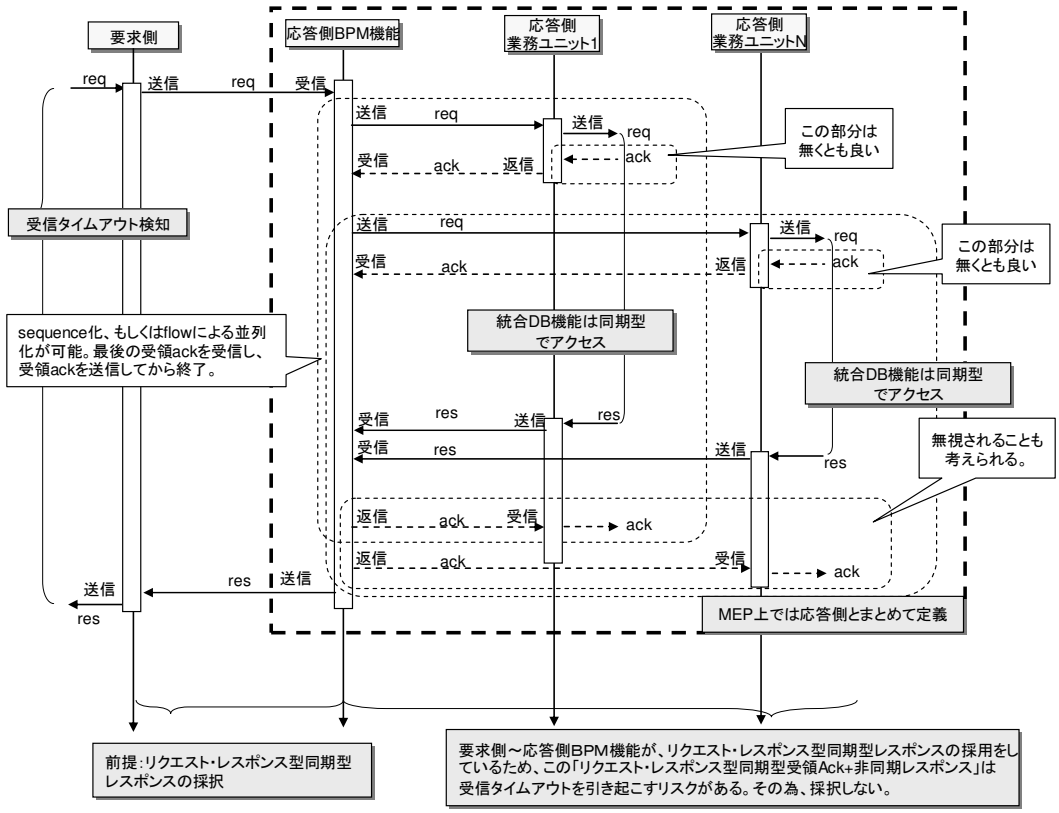


図３．８．１５　リクエスト・レスポンス型同期型受領 Ack+非同期型レスポンスの
組み合せ制約(奨励しない)

図３．８．１６も、奨励しないメッセージ交換パターンである。要求側～応答側間のメッセージ交換パターンが、「リクエスト・レスポンス型同期型受領 Ack+非同期型レスポンス」であるのに対して、応答側の内部である応答側 BPM 機能～応答側業務ユニット間で為されるメッセージ交換パターンでは、「リクエスト・レスポンス型同期型受領 Ack+非同期型レスポンス」を利用している。この場合、応答側 BPM 機能は一連の動作を全て制御せず、要求メッセージに同期して同一のセッションで受領 Ack を受信した後、一度終了する。そしてレスポンスを受信する場合は、別の応答側 BPM 機能のロールを用いる。この場合、WS-BPEL の持つコリレーションセットの機構を用いて要求メッセージに対するレスポンスの対応付け、並びに複数のレスポンスの関係付けを行なうことが困難であるため、デッドロックを発生させる危険性もあり、奨励しない。

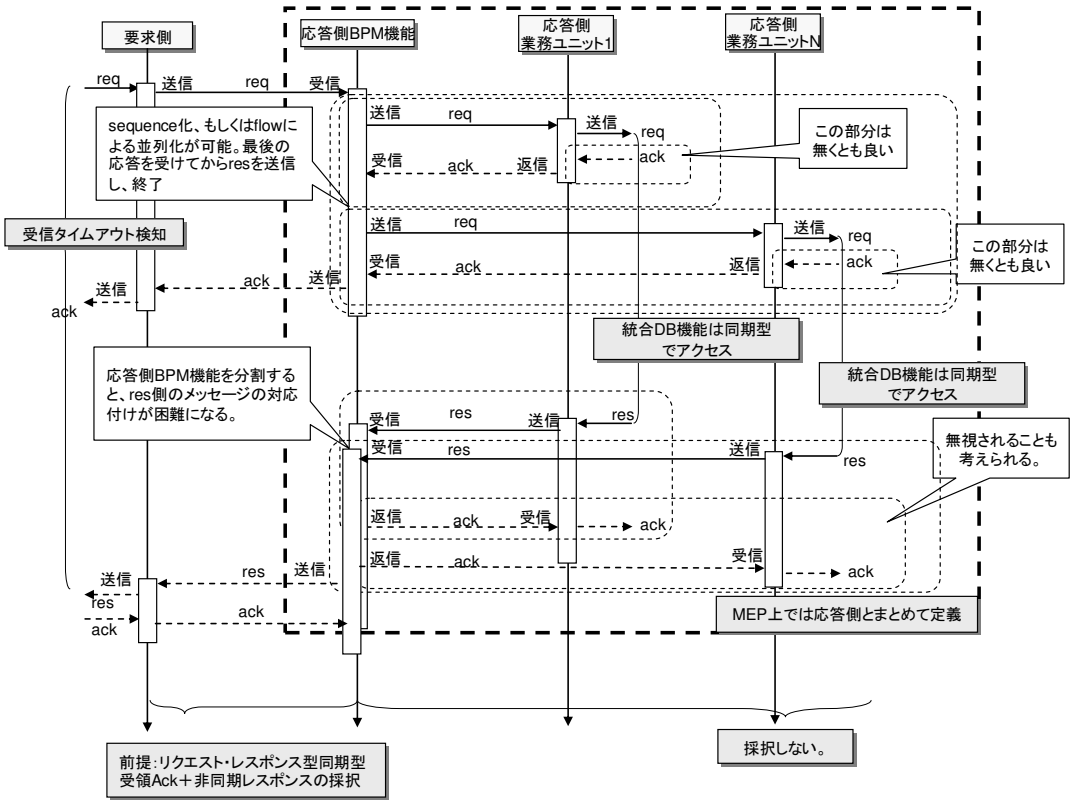


図３．８．１６　リクエスト・レスポンス型同期型受領 Ack+非同期型レスポンスの
組み合せ制約(奨励しない)

３．８．３　異常系処理における問題解決と障害復旧手順

ここでは、地域情報プラットフォームのメッセージ交換パターンにて検知された障害（異常）の対処方法について、その概要を説明する。

（１）　地域情報プラットフォームの異常系処理の定義と位置付け

プラットフォーム通信標準仕様では、通信障害に対して次のような３段階での処理を規定している。

- 障害の検知
- 即時対応処理
- 異常系処理

ここで障害検知とは、異常を捕捉することである。即時対応処理とは、障害検知を受け、それを通知する等の問題解決を伴わない対応処理である。これに対して異常系処理とは障害検知、即時対応処理を受けて、障害を解決するための一連の処理である。

異常系処理は、システム運用管理の一プロセスとして位置づけられ、システム運用設計のアウトプットとしてシステム運用手順書等の形でマニュアルとしてドキュメント化されるべきものである。このドキュメントには異常系処理として、典型的な障害事象と原因の内容定義、その後の対処シナリオおよび再起動のための手順（以降、リランマニュアルと呼ぶ）が個々のシステム運用条件やシステム構成に対応して設計・整備されている必要があり、その詳細は共通的に定義できるものではない。そこで、本節では、地域情報プラットフォームの BPM 機能によるユニット間連携を前提として、メッセージ通信にお

ける異常系処理の手順概要を説明するにとどめることとする。

地域情報プラットフォームにおいて異常が検知された場合、適切な処置を講じずに処理を継続すると、それまでに処理した関連データと以降で処理されるデータとの間でデータの不整合が発生するリスクや、他のビジネスプロセスにおいても同様の不具合が発生するリスクを抱えることになる。従って、副次的な問題の発生、事態の複雑化を避けるために、異常系処理ではビジネスプロセスを停止・中断することを強く推奨する。

以降では、異常系処理の最初のステップであるビジネスプロセスの停止・中断から原因究明、復旧に至るまでのスケルトンを定義している。実システムへの適用に当たっては、このスケルトンを取り込む形で、システム運用設計を行い、システム運用管理者をはじめとする各担当者（部門）の役割を含めた作業手順についてシステム運用手順書等の形で整備されることを推奨する。

（２） 異常系処理の手順例に関する解説

メッセージ通信における異常系処理手順の概要を図３．８．１７に記す。実運用にあたっては、既に述べているように個々の団体の規模、運用組織および運用形態などに応じて処理手順の見直しを行い、例えば、次のような観点での手順の詳細化が必要である。

- 管理対象となるシステムの特性
- 導入済みあるいは導入予定のツールの状況
- 個々の作業をシステム系（機械系）と人間系のいずれで行うかの切り分け
- システムを構成する関連組織と重要度、影響の明確化
- システム運用体制

次に、詳細化した異常系処理手順に基づいて対処シナリオを作成する。また、システム運用手順書の作成に当たっては、それのみで異常系処理の対応が行えるように具体的な作業項目まで落とし込まれていることが望ましい。なお、地域情報プラットフォームが対象とするシステムは、複数のサイト内の部署・部門、サイトを統括する組織体に跨ることが想定されるため、システム運用体制や緊急連絡先を維持・整備しておくことは言うまでもないことである。

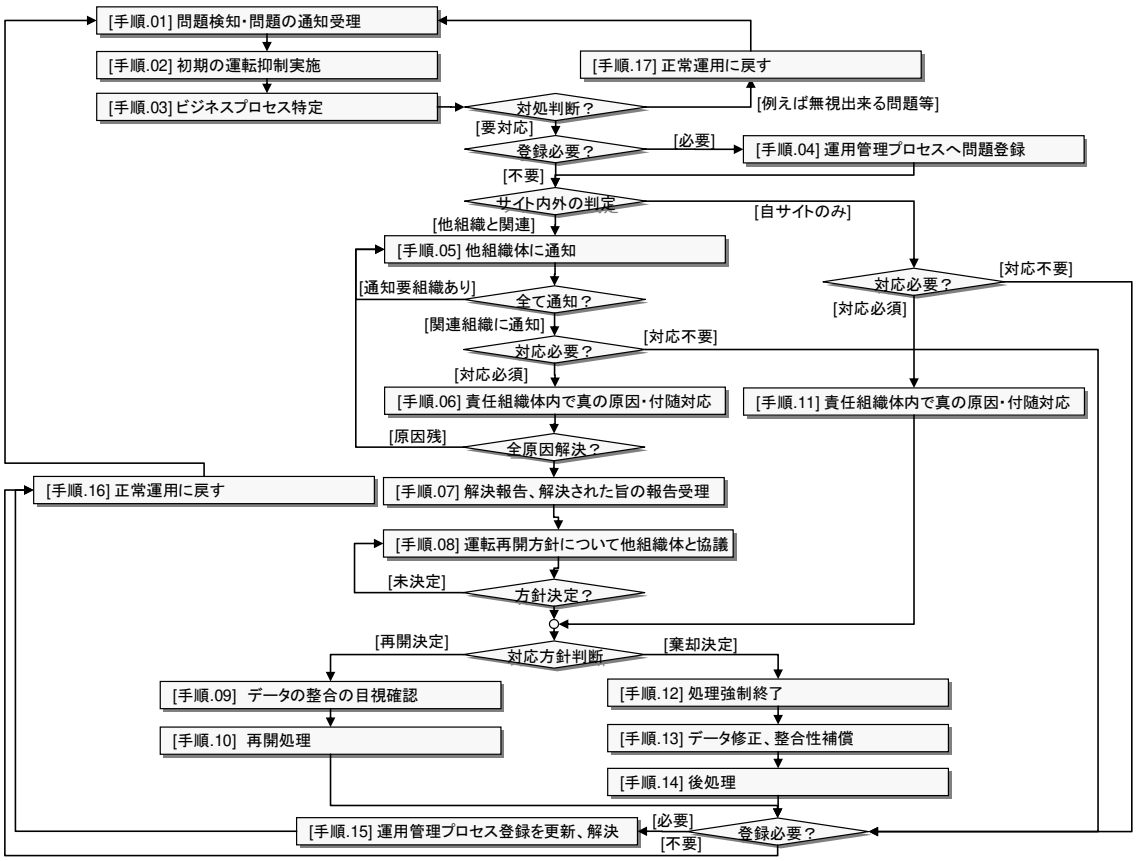


図３．８．１７ 異常系処理の手順例

表３．８．１　異常系処理の手順概要

手順番号	処理内容の概説	処理後に移行する手順番号
[手順.01]	問題検知・問題の通知受理 　プラットフォーム通信標準仕様で規定される「障害の検知」「即時対応処理」を行った段階に相当し、業務ユニットやミドルウェア等のログチェックや異常検出通知、システム監視ツール（管理機能）による自動通報や自治体職員等の操作者からの報告などにより異常として検知される。	[手順.02]
[手順.02]	初期の運転抑制実施 　障害から生じる二次障害を避けるため、必要に応じて運転抑制を行なう。特に必要が無い場合は、この限りではない。	[手順.03]
[手順.03]	ビジネスプロセスの特定 　問題が発生しているビジネスプロセス、対象業務ユニット、対象情報（受付番号など）などを特定する。問題箇所の特定においては、業務ユニットやミドルウェア、OS(Operating System)などのログ情報やガイドラインにて述べているモニタリング機能等の情報を利用することにより行うのが一般的である。	[対処判断]
[手順.04]	運用管理プロセスへの問題登録 　ITIL(Information Technology Infrastructure Library)に基づく	[サイト内外判断]

	運用規定により、標準プロセスとしてインシデント管理や問題管理が定義されている場合は、インシデントとして登録し問題管理に引き継ぐ。	
[手順.05]	他組織体（関係組織体）に通知 問題となっているビジネスプロセスが、他サイトに影響する、あるいは他サイト管轄のものである場合は、当該サイトのシステム運用管理者に通知する。	[通知確認]
[手順.06]	責任組織体内で真の原因・付随対応 問題の検知・通知を受けて、自らの責任範囲で真の原因を特定し、修正する。また、検知・通知された問題に付随して発生した事項についても是正処置を行い、本来の状況に戻す。なお、問題解決の途上で、別の問題であることを発見した場合は、[手順.05]に戻り、再度調整を行なう。	[原因解決確認]
[手順.07]	解決報告、解決された旨の報告受理 自らの責任範囲で、真の原因を特定・解決し、付随した問題も解決した場合は、[手順.04]で問題をインシデントとして管理しているシステム運用管理者に対して、問題が解決した旨を報告する。	[手順.08]
[手順.08]	運転再開方針について他組織体と協議 上記[手順.07]の結果を受けて、問題解決後の対応策について、協議を行なう。その際、データ等の整合性についても確認するべきである。運転再開方針の協議においては、最終的に処理を棄却するか、再開するかを決定する。	[方針決定確認]
[手順.09]	データの整合の目視確認 処理の再開を決定した場合には、関連するデータの整合性について改めて確認する。なお、データ確認は、[手順.08]でも実施されているが、ここでの確認は再開に向けた最終的な確認である。	[手順.10]
[手順.10]	再開処理 システム運用手順書（またはリランマニュアル等）に基づき、処理を再開する。	[登録判断]
[手順.11]	責任組織内で真の原因・付随対応 問題が、自らの責任範囲に留まる場合、問題の検知・通知を受けて、自らの責任範囲で真の原因を特定し、修正する。検知・通知された問題に付随して発生した事項についても是正処置を行い、本来の状況に戻す。	[対処方針判断]
[手順.12]	処理強制終了 強制的に終了することを決定した場合は、関連する処理途上のものを全て終了させる。	[手順.13]
[手順.13]	データ修正、整合性補償 強制的に終了させた場合には、分散環境で処理されたデータについて不整合が発生することが想定される。地域情報プラットフォームでは、トランザクションに関する４要素(原子性,一貫性,隔離性,持続性)を維持する機構は実装（規定）していないため、システム運用手順書等のドキュメントに基づいて、不整合データの修正を行い、整合性を確保する。	[手順.14]
[手順.14]	後処理	[手順.15]

	データ修正、整合性補償の後に、残っている後処理等をシステム運用手順書に基づき実施する。	
[手順.15]	運用管理プロセス登録を更新、解決 ITIL に基づいて運用規定により標準プロセスとして、インシデント管理や問題管理が定義されている場合は、[手順.04]にてインシデントとして登録された事項は、問題解決が完了した段階で更新する。	[手順.16]
[手順.16] [手順.17]	正常運用に戻す 障害から生じる二次障害を避けるため、[手順.02]で運転抑制を行なった場合は、抑制状態を解き、正常な運用に戻す。	[手順.01]

（３） 異常系処理手順における補償・回復処理の例外事項

地域情報プラットフォームのプラットフォーム通信標準仕様では、副次的な問題の発生、事態の複雑化を避けるために、自動化された異常系処理ではビジネスプロセスの停止・中断迄を実施し、その後の一連の回復処理については全て手動で対応することを推奨している。

しかし、次のようなケースにおいては、より効率的な運用管理が可能となることから、例外事項として補償・回復処理の自動化を行うことも許容する。

例外事項１：プラットフォーム通信標準仕様外での補償・回復処理の自動化

プラットフォーム通信標準仕様の範囲外で実施されるビジネスプロセスであり、かつ、異常系処理による不整合を発生させないとの保証がある場合においては、補償・回復処理を自動化させることができる。

例外事項２：同一運用ポリシーによる限定的範囲での補償・回復処理の自動化

自治体内等の組織体で、同一の運用ポリシーに基づいて運用されている同一サイト間での通信であり、かつ、異常系処理による不整合を発生させないとの保証がある場合においては、補償・回復処理を自動化させることができる。

ここで、例外事項として認める補償・回復処理について簡単に説明する。補償・回復処理とは、異　常が発生した場合に、既に実行された処理をキャンセルするなどにより、一連の処理が行われる前の状態に戻す処理である。例えば、複数のサイトに跨って実施されるトランザクションにおいては、従来から 2 フェーズコミットメントが一般的な分散トランザクションの手法である。この 2 フェーズコミットメントでは、サービス要求側の業務ユニットが、他の複数サイト全てに対して必要な更新処理を行い、コミットメントが可能か否かを確認した後、一斉に複数サイトの更新についてコミットメントを実施する方式である。この方式では、通常、データ更新をする複数のサイトは、階層的な呼出、呼び出しのネストがない同等・等価レベルであることが好ましい。

Web サービスの分野においても、この 2 フェーズコミットメントを実施する方法も存在する。しかし WS-BPEL に代表される様に、1 つのサービス呼出が他のサービス呼出を連鎖的に起動し、2 フェーズコミットメントにおけるコミットメントを実施するまでが長時間となる場合が想定される。その場合、更新に対するロック等も実施されるため、分散トランザクションのスループットを指数関数的に悪化させてしまう問題が発生する。そのため、このような長時間に亘って生存するトランザクションの問題を解決するために、ロングトランザクションという考え方を導入した方式も提案されている。ロングトランザクションでは、2 フェーズコミットメントの実行をせず、更新処理等の要求を受けた段階で、各サイトは一度、更新を完了させてしまう。その後、ロングトランザクションの最終段階で、トランザクションを完結させるには問題となる事象が確認された場合、このロングトランザクションの途上で実施した更新処理に関して、キャンセル等の処理を実施し、最終的にはトランザクションを無効（失敗）とする方式である。

３．８．４　異常系処理における留意事項

（１）　MEP 処理系で滞留するビジネスプロセスの処置

プラットフォーム通信標準仕様の「６．３．７節　開始側の MEP 実行系レベル以上の障害検知、即時対応方法」では、MEP 処理系でタイムアウトの検出が出来ない場合、上位の業務ユニットレベルで応答の有無に対するタイムアウト判断を行なうよう推奨している。しかし、MEP 処理系ではタイムアウトが検出できないために、半永久的に応答待ちの状態を維持し続ける。この様な状況が複数のプロセスで発生し、これを処置しないまま運転を継続すると、MEP 処理系を実装する計算機資源を占領し、その結果、応答性能の悪化、他の処理に対する不必要なタイムアウトの誘発等の悪循環を発生させることになる。そこで、この問題に対する対処方法として、以下の方式を提案する。

(a) 独自例外処理による対応

コマンドレベルで操作可能な MEP 処理系であれば、外部から MEP 処理系の不必要なプロセスを終了（Kill）させる様なバッチプログラムを作成し、業務ユニットレベルがタイムアウトを検知した段階で、このプログラムを起動させる。

(b) 人間系による対応

コマンドレベルでの操作が不可能な MEP 処理系については、業務ユニットレベルからのタイムアウト通知をシステム運用管理者が確認し、マニュアル操作（人間系での操作）により不必要なプロセスを終了（Kill）させる。なお、マニュアル操作も難しい場合には、MEP 処理系だけ計画停止を定義し、計画停止段階で関連データ等の削除を強制的に行なう。

（２）　システム実行状況の監視（モニタリング機能および運用管理ツールとの連携）

システムの実行状況の監視において、MEP 実行系製品の管理画面、コンソール画面だけでは、限定的な BPM 処理のみが対象となり、通信相手のビジネスプロセスが処理中で、その結果待ちとなっているのか、あるいはタイムアウトになっているのかが判断出来ない場合がある。その様な対処としては、モニタリング機能による処理のトラッキングなどが必要になってくる。また、システム上の不具合を早期に発見するために運用管理ツールと連携した運用を強く推奨する。

付録１　Audit、インタフェースに関する詳細定義

（１）概要

付録として、Audit に関する各種の推奨形式について記載する。記載するものは、以下の通りである。この奨励形式と等価で別な形式のものがある場合は、本方式以外の実装も許容される。

(a)Audit 取得モデルにおけるクエリの形式
(b)BPM(Business Process Management) Audit および WS アプリケーション Audit の形式
(c)メッセージ Audit の形式
(d)Audit 項目のセマンティクスを定義する情報モデルと構成要素
(e)Audit 取得モデルにおける推奨インタフェース
(f)各メッセージに関する事例

（２）Audit 取得モデルにおけるクエリの形式

Audit 取得モデルに置けるクエリの情報項目を表. 付１. １に記載する。なお、基数制約は UML(Unified Modeling Language)の表記を取り入れており、必ず１つ存在する場合を「1」、オプションとして存在する場合もある、しない場合もある場合は「0..1」、一つ以上任意の場合を「1..*」とする。

表. 付１. １　Audit 取得モデルにおけるクエリに関する推奨事項

連番	区分 （親区分）	区分直下のクエリ項目	基数制約	説明
1	auditQuery (この親区分はトップ)	-	1	クエリのルートエレメント。[指定必要]
2	engineAll (この親区分は auditQuery)	-	0..1	当該 Audit 提供が収集しているエンジンの Audit 全てを対象とする。engines タグと二者択一である。子エレメントとして、少なくとも duration と filter タグのどちらか一つ以上を設定する。duration と filter タグの両方を設定することも可能である。 複数の検索条件を設定した場合、それらの AND 条件とする。 オプショナルの属性 “limit”を指定すると、各エンジンに対応する検索結果の数に上限が設けられる。limit 属性を用いる場合は、エンジン ID ごとに、エンジンで検索された結果の内、時刻の古い Audit から時刻順に最大 limit 個を検索結果とする。limit を設定しない場合は、検索結果に合致した全ての Audit を時刻順に返す。[指定選択]
3	engines (この親区分は auditQuery)	-	0..1	当該 Audit 提供が収集しているエンジンのうち、engine タグで指定するエンジンの Audit のみを対象とする。 engineAll タグと二者択一である。[指定選択]
4	engine (この親区分は engines)	-	1..*	取得対象とするエンジンを指定する。必須の属性 “engineId”で検索対象とするエンジンのエンジン ID を指定する。 オプショナルの属性 “limit”を指定すると、各エンジンに対応する検索結果の数に上限が設けられる。limit 属性を用いる場合は、エンジン ID ごとに、エンジンで検索された結果の内、時刻の古い Audit から時刻順に最大 limit 個を検索結果とする。limit を設定しない場合は、検索結果に合致した全ての

				Audit を時刻順に返す。少なくとも sequenceNo と duration タグのどちらか一つ以上を設定する。sequenceNo と duration タグの両方を設定することも可能である。 複数の検索条件を設定した場合､それらの AND 条件とする。[指定必要]
5	sequenceNo (この親区分は engine)	-	0..1	検索対象とする sequenceNo の範囲を指定する。gt タグと le タグの両方が設定された場合は、それらの AND 条件とする。audit データの sequenceNo が条件を満たすものを検索結果とする。[指定選択]
6		gt	1	この値より大きな sequenceNo を検索対象とする(greater than)。[指定必要]
7		le	0..1	この値以下の sequenceNo を検索対象とする(less than or equal to)。[指定選択]
8	duration (この親区分は engineAll、engine)	-	0..1	当該期間に発生した Audit データを検索する。from タグと to タグ両方の条件を満たすものを検索結果とする。 [指定選択、但し、親区分が engineAll の場合は必要]
9		from	1	この値を含む時刻以降に発生した audit データを検索対象とする。[指定必要]
10		to	1	この値を含む時刻以前に発生した audit データを検索対象とする。[指定必要]
11	filter (この親区分は engineAll、engine)	-	0..1	Audit データの activityId で検索する。複数の type タグを設定した場合は、それらの OR 条件とする。[指定選択]
12		type	1..*	この値を、activityId 文字列の最後の子エレメント名として持つ Audit データを検索対象とする。[指定必要] 例えば activityId が “/process[1]/scope[1]/sequence[1]/invoke[1]” のとき、 type タグの値が “scope” → 該当しない。 type タグの値が “invoke” → 該当する。

（３） BPM Audit および WS アプリケーション Audit に関する情報項目

モニタリング機能はオプションであるが、モニタリング機能を実装する場合、この BPM Audit および WS アプリケーション Audit の生成は必要となる。

BPM Audit および WS アプリケーション Audit に格納する情報項目を表．付１．２に定義する。ここで、図．付１．３の情報モデルに基づき、WS アプリケーション Audit は、BPM Audit の形式と同一とし、WS アプリケーションを擬似的に WS-BPEL(Web Services Business Process Execution Language 2.0)定義で表現できることを前提としているため、表．付１．２は、WS アプリケーション Audit と BPM Audit との間で同じコンテキストとして扱われ、共用されている。表．付１．２で各項目は階層的に表現され、表中の区分と親区分はそれぞれ注目する区分とその親に相当する区分を表す。項目のルートは「トップ」を記してある。またリソース項目が「-」の欄は、その区分自体の基数制約を表す。

表．付１．２を元に、図．付１．１に推奨する BPM Audit、WS アプリケーション Audit の XML Schema 表現を記す。この推奨 BPM Audit 形式定義では､ルートエレメントとして “BPM Audit”を用いる。BPM Audit の子エレメントとして “processInstance” を複数個格納可能である。

また、BPM Audit において、推奨方式である SOAP(Simple Object Access Protocol)を用いず、WSDM(Web Services Distributed Management 1.1)等により実装する場合は、図．付１．１を拡張要素として利用すれば良い。

表．付１．２　BPM Audit、WS アプリケーション Audit データに関する推奨事項

連番	区分 （親区分）	区分直下の監視対象リソース項目	基数制約	説明
1	processInstance (この親区分はトップで、processInstance タグを持つ。)	-	1..*	BPM Audit ないし WS アプリケーション Audit 1 個には、複数の ProcessInstance 情報を格納可能とする。[指定必要]
2		sequenceNo	1	シーケンス番号。Audit データ実装時特有の値。Query-Response で Audit データを取得する際に Audit を区別するために用いる。[指定必要]
3		processInstanceId	0..1	プロセスインスタンス ID。BPM Audit の場合は必須。[指定選択]
4		createdDate	0..1	プロセスインスタンスの生成日時。BPM Audit のときは必須。InnerState の属性を継承。[指定選択]
5		updatedDate	0..1	プロセスインスタンスの更新日時。BPM Audit のときは必須。InnerState の属性を継承。[指定選択]
6		status	1	状態。プロセスの実行中ないし終了を表す。InnerState の属性を継承。[指定必要]
7		processDefId	1	プロセス定義名。ProcessDefinition の属性を使用。[指定必要]
8	BPELActivityInstance (この親区分は processInstance で、activity タグを持つ。)	-	1	BPELActivityInstance と WSApplicationActivityInstance を同一形式で扱う。[指定必要]
9		date	1	アクティビティインスタンスの発生日時。ActivityInstance の属性を継承。[指定必要]
10		activityId	1	BPM Audit の場合は BPELActivityDefinition の activityId 属性、WS アプリケーション Audit の場合は ActivityDefinitionForWS の checkpointId。[指定必要]
11		status	1	アクティビティの実行中(開始)ないし終了を表す。InnerState の属性を使用。[指定必要]
12	message (この親区分は BPELActivityInstance で message タグを持つ。)	-	0..1	メッセージ。指定選択ではあるが、指定可能時は指定する。[指定選択]
13		messageId	0..1	メッセージ ID。BPM Audit でメッセージ生成が伴わない BPM Activities の場合、メッセージ ID が存在しない。[指定選択]
14		messageContents	0..1	メッセージ内容。メッセージを圧縮し、base64encoding したもの。[指定選択]
15		type	0..1	メッセージタイプ。出力メッセージ、入力メッセージのどちらかを指定する。[指定選択]
16		createdDate	0..1	メッセージの生成日時。[指定選択]
17		sentDate	0..1	メッセージが送信された日時。[指定選択]
18		receivedDate	0..1	メッセージが受信された日時。[指定選択]
19		processedDate	0..1	メッセージが処理された日時。[指定選択]
20	ScopeInstance	-	0..1	Scope インスタンス。[指定選択]

21	(この親区分は processInstance で、scope タグを持つ。)	activityId	1	BPELActivityDefinition の属性を使用。スコープを指定するために Process の activityId を指定する。[指定選択]
22	variable (この親区分は ScopeInstance で、variable タグを持つ。)	-	0..*	Variable のインスタンス。[指定選択]
23		value	1	変数の値。[指定選択]
		updatedDate	1	変数の更新日時。[指定選択]
24		variableName	1	変数名。VariableDefinition の属性を利用。[指定選択]
25	correlation (この親区分は ScopeInstance で、correlation タグを持つ。)	-	0..*	Correlation インスタンス。[指定選択]
26		PropertyVariable	1	correlation 値。[指定選択]
27		activityId	1	Correlation を更新したアクティビティのアクティビティ ID。[指定選択]
28		updatedDate	1	更新日時。[指定選択]
29		correlationSetName	1	correlationSet の名称。CorrelationDefinition の属性を使用。[指定選択]
30		propertyName	1	プロパティ名。CorrelationPropertyDefinition の propertyName 属性を使用。[指定選択]
31	BPELEngine (この親区分は processInstance で、engine タグを持つ。)	-	1	BPEL エンジン。[指定必要]
32		engineId	1	エンジン ID。Engine の属性を継承。[指定必要]
33	machine (この親区分は processInstance で、machine タグを持つ。)	-	1..*	実行体。BPM 機能が複数の実行体を用いて構成・実行されるときに、それらの 1 台以上の情報を Audit データに設定する。[指定必要]
34		machineId	1	実行体 ID。[指定必要]

（注）表．付１．２は階層構造を含まず、並列化されて表現されている。

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xsd:element name="BPMAudit" type="BPMauditType"/>
  <xsd:complexType name="BPMauditType">
    <xsd:sequence>
      <xsd:element minOccurs="1" maxOccurs="unbounded" name="processInstance">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element type="xsd:unsignedLong" name="sequenceNo"></xsd:element>
            <xsd:element minOccurs="0" name="processInstanceId">
              <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                  <xsd:maxLength value="128"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
```

```
<xsd:element minOccurs="0" type="xsd:dateTime" name="createdDate"></xsd:element>
<xsd:element minOccurs="0" type="xsd:dateTime" name="updatedDate"></xsd:element>
<xsd:element name="status">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="running"/>
      <xsd:enumeration value="finished"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
<xsd:element name="processDefId">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="128"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
<xsd:element name="activity">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element type="xsd:dateTime" name="date"></xsd:element>
      <xsd:element type="xsd:string" name="activityId"></xsd:element>
      <xsd:element name="status">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="begin"/>
            <xsd:enumeration value="end"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element minOccurs="0" maxOccurs="1" name="message">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element type="xsd:string" name="messageId"></xsd:element>
            <xsd:element type="xsd:base64Binary"
                                name="messageContents"></xsd:element>
            <xsd:element name="type">
              <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                  <xsd:enumeration value="incoming"/>
                  <xsd:enumeration value="outgoing"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
            <xsd:element minOccurs="0" type="xsd:dateTime"
```

```
                                                    name="createdDate"></xsd:element>
                <xsd:element minOccurs="0" type="xsd:dateTime"
                                                    name="sentDate"></xsd:element>
                <xsd:element minOccurs="0" type="xsd:dateTime"
                                                     name="receivedDate"></xsd:element>
                <xsd:element minOccurs="0" type="xsd:dateTime"
                                                    name="processedDate"></xsd:element>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element minOccurs="0" name="scope">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element type="xsd:string" name="activityId"></xsd:element>
          <xsd:element minOccurs="0" maxOccurs="unbounded" name="variable">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element type="xsd:base64Binary" name="value"></xsd:element>
                <xsd:element type="xsd:dateTime" name="updatedDate"></xsd:element>
                <xsd:element type="xsd:string" name="variableName"></xsd:element>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
          <xsd:element minOccurs="0" maxOccurs="unbounded" name="correlation">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element type="xsd:string" name="propertyVariable"></xsd:element>
                <xsd:element type="xsd:string" name="activityId"></xsd:element>
                <xsd:element type="xsd:dateTime" name="updatedDate"></xsd:element>
                <xsd:element type="xsd:string" name="correlationSetName">
                </xsd:element>
                <xsd:element type="xsd:string" name="propertyName"></xsd:element>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="engine">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element type="xsd:string" name="engineId"></xsd:element>
```

```
                </xsd:sequence>
              </xsd:complexType>
            </xsd:element>
            <xsd:element minOccurs="1" maxOccurs="unbounded" name="machine">
              <xsd:complexType>
                <xsd:sequence>
                  <xsd:element type="xsd:integer" name="machineId"></xsd:element>
                </xsd:sequence>
              </xsd:complexType>
            </xsd:element>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

図. 付１. １ BPM Audit および WS アプリケーション Audit の XML Schema

（４）メッセージ Audit に関する情報項目

メッセージ Audit の生成はオプションである。但し、モニタリング機能をより高度に利用する場合、例えば、(i)ワンストップサービス時に発生するエラーについて捕捉する、(ii)運用状況を捕捉する、(iii)各種 KPI(Key Performance Indicator)を設定しサービスを改善する等の場合、サービスの実行状況を直接反映するメッセージ Audit が生成されることを奨励する。

メッセージ Audit に格納する情報項目を表. 付１. ３に示す。項目は階層的に表現され、表中の区分と親区分はそれぞれ注目する区分とその親に相当する区分を表す。項目のルートは「トップ」を記してある。またリソース項目が「-」の欄は、その区分自体の基数制約を表す。

表. 付１. ２を元に、図. 付１. ２に推奨メッセージ Audit の XML Schema 表現を記す。この推奨メッセージ Audit 形式定義では、ルートエレメントとして“messageAudit”を用いる。messageAudit の子エレメントとして“message”を複数個格納可能である。また、メッセージ Audit において、推奨方式である SOAP 形式を用いず、WSDM 等により実装する場合は、図. 付１. １を拡張要素として利用する。

表. 付１. ３　メッセージ Audit データに関する推奨事項

連番	区分 (親区分)	区分直下の監視対象リソース項目	基数 制約	説明
1	message	-	1..*	メッセージ。[指定必要]
2	(この親区分はトップで、message タグを持つ。)	sequenceNo	1	クエリの開始点。デフォルトは 0 が指定される。[指定必要]
3		messageId	1	メッセージ ID。[指定必要]
4		relatesTo	1	関連するメッセージの messageId で、共通ヘッダに指定されたものがセット。存在しない場合は、タグのみ存在し、コンテンツは指定しない。[指定必要]
5		messageContents	1	メッセージ内容。メッセージを圧縮し、Base64 エンコーディングしたもの。[指定必要]
6		type	1	メッセージタイプ。出力メッセージ、入力メッセージのどちらかを指定する。[指定必要]

7		createdDate	0..1	メッセージの生成日時。[指定選択]
8		sentDate	0..1	メッセージが送信された日時。[指定選択]
9		receivedDate	0..1	メッセージが受信された日時。[指定選択]
10		processedDate	0..1	メッセージが処理された日時。[指定選択]
11	endpoint (この親区分は message で、endpoint タグを持つ。)	-	1	エンドポイント情報(URI(Uniform Resource Identifier)に関する)。[指定必要]
12		uri	1	Endpoint の URI。[指定必要] これは明示的な url のタグは存在せず、endpoint のデータ型としてされる。
13	PartnerLink (この親区分は message で、partnerLinkInfo タグを持つ。)	-	0..1	エンドポイント情報(partnerLink に関する)。 [指定選択]
14		partnerLinkName	1	partnerLink の name 属性。[指定選択]
15		portTypeName	1	portType の name 属性。[指定選択]
16		operation	1	operation の name 属性。[指定選択]

（注）表．付１．３は階層構造を含まず、並列化されて表現されている。

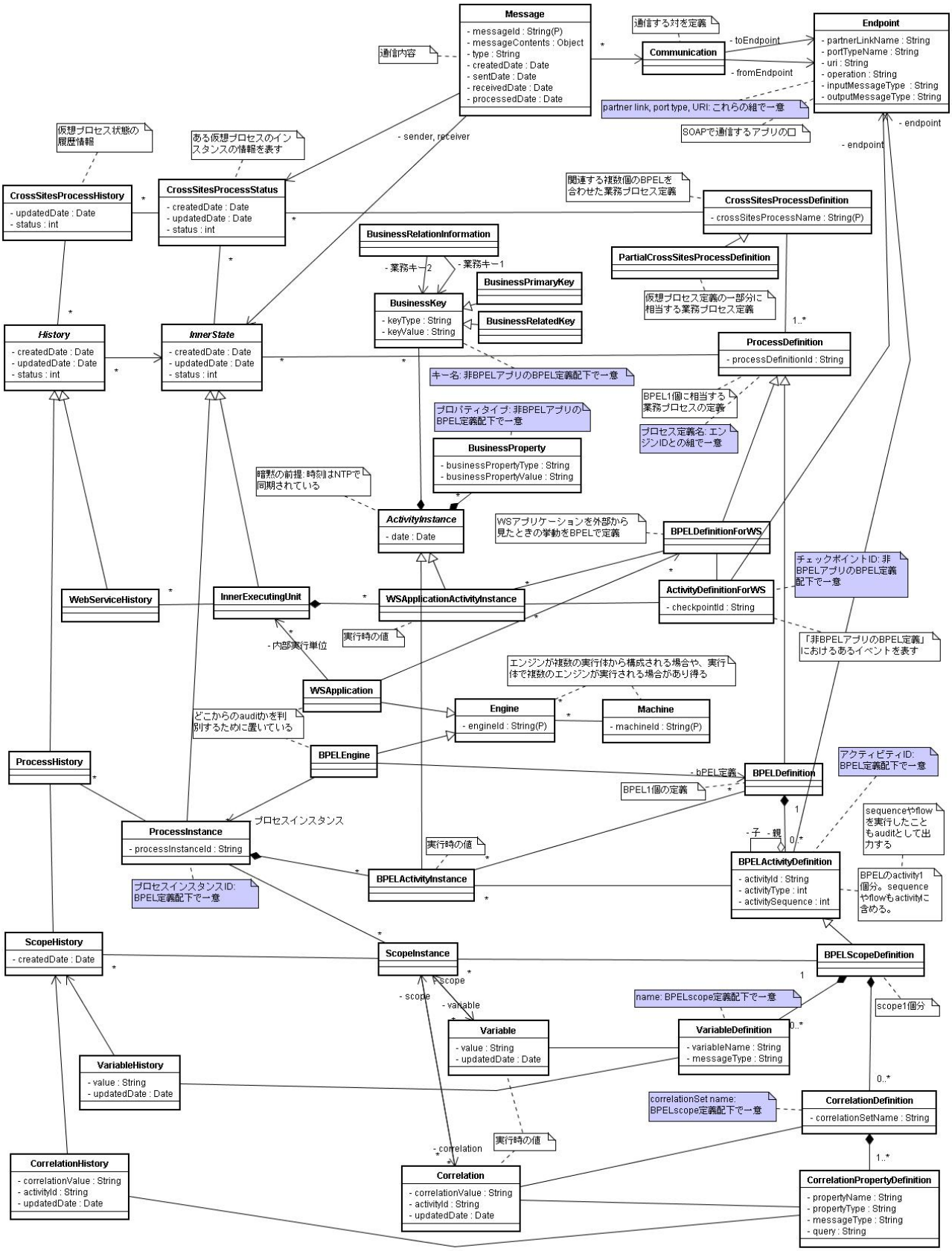
```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xsd:element name="messageAudit" type="messageAuditType"/>
  <xsd:complexType name="messageAuditType">
      <xsd:sequence>
        <xsd:element minOccurs="1" maxOccurs="unbounded" name="message">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element type="xsd:unsignedLong" name="sequenceNo"></xsd:element>
              <xsd:element type="xsd:string" name="messageId"></xsd:element>
              <xsd:element type="xsd:string" name="relatesTo"></xsd:element>
              <xsd:element type="xsd:base64Binary" name="messageContents"></xsd:element>
              <xsd:element name="type">
                <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                    <xsd:enumeration value="incoming"/>
                    <xsd:enumeration value="outgoing"/>
                  </xsd:restriction>
                </xsd:simpleType>
              </xsd:element>
              <xsd:element minOccurs="0" type="xsd:dateTime" name="createdDate"></xsd:element>
              <xsd:element minOccurs="0" type="xsd:dateTime" name="sentDate"></xsd:element>
              <xsd:element minOccurs="0" type="xsd:dateTime" name="receivedDate"></xsd:element>
              <xsd:element minOccurs="0" type="xsd:dateTime" name="processedDate"></xsd:element>
              <xsd:element type="xsd:anyURI" name="endpoint"></xsd:element>
              <xsd:element minOccurs="0" name="partnerLinkInfo">
                <xsd:complexType>
                  <xsd:sequence>
```

```
                        <xsd:element type="xsd:string" name="partnerLinkName"></xsd:element>
                        <xsd:element type="xsd:string" name="portTypeName"></xsd:element>
                        <xsd:element type="xsd:string" name="operation"></xsd:element>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

図. 付１. ２ メッセージ Audit の XML Schema

（５）Audit 項目のセマンティクスを定義する情報モデル

Audit の各項目は、図. 付１. ３に定義されるモニタリングすべきプロセス、またはサービスの情報モデルに基づく。この分野では、適当な既存情報モデルは存在していないため、本仕様独自の情報モデルである。この情報モデルでは定義、実行、実行履歴に関する情報モデルのすべてを含んでおり、実行時に BPM Audit、WS アプリケーション Audit、メッセージ Audit として出力される情報は、概ね、この情報モデルの一部分をインスタンスとしている。



図．付１．３ プロセス、サービスに関する情報モデル

図．付１．３に示した情報モデルにおける構成要素の概要を表．付１．４に示す。

表．付１．４　情報モデルにおける構成要素

連番	クラス名		概要説明
1	ActivityDefinitionForWS		非 WS-BPEL アプリケーションのアクティビティ定義。本来は WS-BPEL で定義されていない WS アプリケーションの動作を擬似的に WS-BPEL で記述したときの、アクティビティに関する定義。
		checkpointId	非 WS-BPEL アプリケーションのアクティビティ定義を WS-BPEL 定義内で一意に識別するための識別子。
2	ActivityInstance		アクティビティインスタンス。WS-BPEL 仕様で定義される basic activities と structured activities のインスタンスに相当する。
		date	アクティビティインスタンスが発生した日時。
3	BPELActivityDefinition		WS-BPEL アクティビティ定義。WS-BPEL 定義におけるアクティビティに関する定義に相当する。
		activityId	WS-BPEL アクティビティ定義を WS-BPEL 定義配下で一意に識別するための識別子。
		activityType	WS-BPEL 仕様で定義されるアクティビティの種類を表す。
		activitySequence	アクティビティ順序。同一の親アクティビティを持つアクティビティ間の順序関係を表す。
4	BPELActivityInstance		WS-BPEL アクティビティインスタンス。WS-BPEL エンジン上で WS-BPEL 定義が実行されるときに生成される、WS-BPEL アクティビティ定義に対応するインスタンス。
5	BPELDefinition		WS-BPEL 定義。WS-BPEL を用いて記述したプロセス定義であり、WS-BPEL 処理系(WS-BPEL エンジンとも呼ぶ)で実行される。
6	BPELDefinitionForWS		非 WS-BPEL アプリケーションの WS-BPEL 定義。WS アプリケーションの動作を擬似的に WS-BPEL で記述したものである。この定義がそのまま WS アプリケーションで実行されるのではない。
7	BPELEngine		WS-BPEL エンジン。WS-BPEL を処理する処理系の実装のこと。
8	BPELScopeDefinition		BPELscope 定義。WS-BPEL アクティビティ定義の内、scope 定義に相当する。
9	BusinessKey		業務キー。業務主キーと業務関連キーの総称。
		keyType	キー名。業務キーの名称。非 WS-BPEL アプリケーションの WS-BPEL 定義配下で一意。
		keyValue	キー値。業務キーの値。
10	BusinessPrimaryKey		業務主キー。業務実行上、業務インスタンスを識別する ID 等キーとして用いる情報。
11	BusinessProperty		プロパティ。業務における、業務主キーと業務関連キー以外の情報を扱う。
		businessPropertyType	プロパティの種別。非 WS-BPEL アプリケーションの WS-BPEL 定義配下で一意。
		businessPropertyValue	プロパティの値。
12	BusinessRelatedKey		業務関連キー。業務実行上、業務主キーと関連した、業務インスタンスを識別する ID 等のキーとして用いられる情報。
13	BusinessRelationInformation		関係づけ情報。業務主キーと業務関連キーを関係づける。

14	Communication		ある通信路に関する、送信側エンドポイントと受信側エンドポイントの対を定義する。
15	Correlation		Correlation 定義で定義されるアクティビティのインスタンス。
		correlationValue	correlation 値。
		activityId	Correlation 値を更新したアクティビティのアクティビティ ID。
		updatedDate	更新日時。
16	CorrelationDefinition		Correlation 定義。WS-BPEL 定義の内、correlation 定義を扱う。
		correlationSetName	correlationSet の名称。WS-BPEL 定義配下で一意。
17	CorrelationHistory		Correlation 履歴。Correlation 情報を保存する。
		correlationValue	correlation 値。
		activityId	アクティビティ ID。
		updatedDate	更新日時。
18	CorrelationPropertyDefinition		CorrelationProperty 定義。WS-BPEL において、correlation で用いる property を定義する。
		propertyName	プロパティ名。WS-BPEL の property の name 属性。
		propertyType	プロパティ型名。WS-BPEL の property の type 属性。
		messageType	correlation で用いるメッセージタイプ。
		query	メッセージタイプで指定されたメッセージから correlation で用いる文字列を抽出するための検索文字列。
19	CrossSitesProcessDefinition		仮想プロセス定義。要素のプロセスを定義するプロセス定義を複数集め、それらを含み、かつ統合する形で導出されるプロセスの等価プロセス定義であり、必ずしもプロセスを形式化された言語で記述したものとは限らない。
		crossSitesProcessName	仮想プロセス名。グローバルで一意。
20	CrossSitesProcessHistory		仮想プロセス履歴。仮想プロセス状態の履歴を管理する。
		updatedDate	更新日時。
		status	状態。実行中ないし終了。
21	CrossSitesProcessStatus		仮想プロセス状態。仮想プロセス定義のインスタンスに相当する。
		createdDate	生成日時。
		updatedDate	更新日時。
		status	状態。実行中ないし終了。
22	Endpoint		エンドポイント。Web サービスのインタフェースが実装された通信上の端点。partnerLinkName、portTypeName、uri の組でグローバルに一意。
		partnerLinkName	WS-BPEL で定義される partnerLink の name 属性。
		portTypeName	WSDL で定義される portType の name 属性。
		uri	binding 情報で定義される、このエンドポイントの URI。
		operation	WSDL で定義される operation の name 属性。
		inputMessageType	このエンドポイントに入力されるメッセージのメッセージタイプ。
		outputMessageType	このエンドポイントが出力するメッセージのメッセージタイプ。
23	Engine		エンジン。処理系実装の総称であり、WS-BPEL を対象とする場合は WS-BPEL エンジン、Web サービスを対象とする場合は Web サービスエンジンと呼ぶ。エンジンは 1 個以上の実行体で構成される。
		engineId	処理系実装を識別するための識別子記述であり、グローバルで一意。

24	History		履歴。業務ユニット等のアプリケーションでの状態を表すインスタンスを履歴として格納する。
		createdDate	履歴の発生日時。
		updatedDate	履歴の更新日時。
		status	状態。実行中か終了かを表す。
25	InnerExecutingUnit		内部実行単位。WS-BPEL におけるプロセスインスタンスに相当し、WS アプリケーションを実行したとき業務のインスタンスを識別する単位。
26	InnerState		内部状態。プロセスインスタンスと内部実行単位の親クラスであり、業務実行時の内部状態を表す。
		createdDate	プロセスインスタンスに相当する内部状態の生成日時。
		updatedDate	業務遂行に伴い内部状態が変化したときの変更日時。
		status	内部状態。実行中ないし終了。
27	Machine		実行体。エンジンを構成するハードウェアのうち、1 つの OS の管理下にある固まりの単位。
		machineId	実行体を識別するための識別子で、グローバルで一意。
28	Message		メッセージ。Web サービスのインタフェースの WSDL 定義に基づき、対をなすエンドポイント間でやりとりされる XML 形式の記述のこと。
		messageId	メッセージ ID。メッセージのインスタンスを識別するための識別子。グローバルで一意。
		messageContents	メッセージ内容。通信路を流れた SOAP メッセージの中身。
		type	メッセージタイプ。
		createdDate	メッセージの生成日時。
		sentDate	メッセージが送信された日時。
		receivedDate	メッセージが受信された日時。
		processedDate	メッセージが処理された日時。
29	PartialCrossSitesProcessDefinition		部分仮想プロセス定義。仮想プロセス定義の一部分に相当する業務プロセス定義。
30	ProcessDefinition		プロセス定義。プロセスを形式化された言語で記述したもの。
		processDefinitionId	プロセス定義名。WS-BPEL で記述されるプロセス定義を識別するための識別子。エンジン ID との組で一意。
31	ProcessHistory		プロセス履歴。プロセスインスタンスの履歴を管理する。
32	ProcessInstance		プロセスインスタンス。プロセスを WS-BPEL エンジンに割り付け、インスタンス化したもの。WS-BPEL 定義の initial start activity 実行時に生成される業務プロセス単位。
		processInstanceId	プロセスインスタンスを識別するための識別子で、WS-BPEL 定義配下でグローバルに一意。
33	ScopeHistory		Scope 履歴。Scope インスタンスの履歴を管理する。
		createdDate	生成日時。
34	ScopeInstance		Scope インスタンス。
35	Variable		WS-BPEL 実行時の variable 値。
		value	変数の値。
		updatedDate	更新日時。
36	VariableDefinition		Variable 定義。WS-BPEL 定義における、variable タグに相当する。
		variableName	variable の name 属性。BPELscope 定義配下で一意。

		messageType	variable の messageType 属性。
37	VariableHistory		Variable 履歴。Variable の履歴を管理する。
		value	値。
		updatedDate	更新日時。
38	WebServiceHistory		WebService 履歴。内部実行状態の履歴を管理する。
39	WSApplication		WS アプリケーション。Web サービスのインタフェースを持つアプリケーションで、WS-BPEL エンジンから呼び出されるもの。
40	WSApplicationActivityInstance		WS アプリアクティビティインスタンス。WS アクティビティ上のインスタンスを表す。

（６）インタフェース

図．付１．４に BPM Audit、WS アプリケーション Audit、メッセージ Audit を取得する場合の推奨インタフェースの WSDL(Web Services Description Language)、図．付１．５に XML Schema を記す。

図．付１．４の推奨インタフェースは、全て同期型のクエリーレスポンス型で実装される。この奨励インタフェース定義以外にも、WSDM 等により同等機能実装することは許容される。WSDM を利用する場合、当該インタフェース相当のインタフェース定義を公開しなければならない。また、当該インタフェースに対して、新たに付加価値を付けたインタフェース、例えば、「メッセージ Audit 一括送信」機能および「メッセージ Audit 送信」機能を Web サービスとして実現される場合も、これらサービスの WSDL は新たに公開されなければならない。

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="audit-wsdl"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" >
        <wsdl:types>
                <xsd:schema>
                        <xsd:include schemaLocation="BPMAudit.xsd" />
                        <xsd:element name="BPMAudit" type="BPMauditType" />
                </xsd:schema>
                <xsd:schema>
                        <xsd:include schemaLocation="MessageAudit.xsd" />
                        <xsd:element name="MessageAudit" type="messageAuditType" />
                </xsd:schema>
                <xsd:schema>
                        <xsd:include schemaLocation="query.xsd" />
                        <xsd:element name="auditQuery" type="auditQueryType" />
                </xsd:schema>
        </wsdl:types>
        <wsdl:message name="BPMAudit-INPUT">
                <wsdl:part name="BPMAudit-INPUT" element="auditQuery" />
        </wsdl:message>
        <wsdl:message name="BPMAudit-OUTPUT">
                <wsdl:part name="BPMAudit-OUTPUT" element="BPMAudit" />
        </wsdl:message>
        <wsdl:message name="MessageAudit-INPUT">
```

```
        <wsdl:part name="MessageAudit-INPUT" element="auditQuery" />
    </wsdl:message>
    <wsdl:message name="MessageAudit-OUTPUT">
        <wsdl:part name="MessageAudit-OUTPUT" element="MessageAudit" />
    </wsdl:message>
    <wsdl:portType name="IF-ADMIN-01PT">
        <wsdl:operation name="IF-ADMIN-01">
            <wsdl:input name="IF-ADMIN-01-INPUT" message="BPMAudit-INPUT" />
            <wsdl:output name="IF-ADMIN-01-OUTPUT" message="BPMAudit-OUTPUT" />
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:portType name="IF-ADMIN-02PT">
        <wsdl:operation name="IF-ADMIN-02">
            <wsdl:input name="IF-ADMIN-02-INPUT" message="MessageAudit-INPUT" />
            <wsdl:output name="IF-ADMIN-02-OUTPUT" message="MessageAudit-OUTPUT" />
        </wsdl:operation>
    </wsdl:portType>
</wsdl:definitions>
```

図．付１．４　インタフェースの WSDL

推奨案ではあるが、モニタリング機能のインタフェース命名規則として、図．付１．４で記される様に全て”IF-ADMIN”と称するプリフィックスを持つ。図．付１．５には、図．付１．４から参照されるクエリの推奨 XML Schema を示す。このタグ名の説明は表．付１．１を参照願いたい。

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            elementFormDefault="qualified" attributeFormDefault="unqualified">

<!-- TOP -->
<xsd:element name="auditQuery" type="AuditQueryType" />

<xsd:complexType name="AuditQueryType">
  <xsd:choice>
    <!-- engineAll か engines のいずれかを選択 -->
    <xsd:element name="engineAll" type="EngineAllType" />
    <xsd:element name="engines" type="EnginesType" />
  </xsd:choice>
</xsd:complexType>

<xsd:complexType name="EngineAllType">
  <xsd:sequence>
    <xsd:element name="duration" type="DurationType" />
    <xsd:element name="filter" type="FilterType" minOccurs="0" maxOccurs="1" />
  </xsd:sequence>
```

```
  <xsd:attribute name="limit" type="xsd:int" />
</xsd:complexType>

<xsd:complexType name="EnginesType">
  <xsd:sequence>
    <!-- 複数の処理系(エンジン)ごとの query を発行可能とする -->
    <xsd:element name="engine" type="EngineType" minOccurs="1" maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="EngineType">
  <xsd:sequence>
    <!-- sequenceNo か duration の少なくとも 1 つを必須とする。←XSD として表現できていない -->
    <xsd:element name="sequenceNo" type="SequenceNoType" minOccurs="0" maxOccurs="1" />
    <xsd:element name="duration" type="DurationType" minOccurs="0" maxOccurs="1" />
    <xsd:element name="filter" type="FilterType" minOccurs="0" maxOccurs="1" />
  </xsd:sequence>
  <xsd:attribute name="limit" type="xsd:int" />
  <xsd:attribute name="engineId" type="xsd:string" use="required" />
</xsd:complexType>

<xsd:complexType name="SequenceNoType">
  <xsd:sequence>
    <xsd:element name="gt" type="xsd:unsignedLong" />
    <xsd:element name="le" type="xsd:unsignedLong" minOccurs="0" maxOccurs="1" />
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="FilterType">
  <xsd:sequence>
    <xsd:element name="type" type="xsd:string" minOccurs="1" maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="DurationType">
  <xsd:sequence>
    <xsd:element name="from" type="xsd:dateTime" />
    <xsd:element name="to" type="xsd:dateTime" />
  </xsd:sequence>
</xsd:complexType>

</xsd:schema>
```

図．付１．５ クエリの XML Schema

（７）各メッセージ、推奨 Audit 形式例

推奨インタフェース定義で示したクエリの形式を用い、BPM Audit を推奨 Audit 形式で取得するときの例を以下に記す。これはクエリーレスポンスで処理されるため、対で定義される。

(a) Audit 取得モデルにおける推奨するクエリ形式のメッセージ例

クエリ対象の BPM Audit 提供から、WS-BPEL 処理系の持つ「エンジン ID」が ″bpel.a.municipal.jp″ の BPM の Audit で、シーケンス番号が 0 より大きな BPM Audit を取得する際のクエリのメッセージ例を図. 付１．６に示す。なお、クエリの結果が最大 123 個までになるよう、limit 属性を設定している。

```
<auditQuery>
  <engines>
    <engine engineId="bpel.a.municipal.jp" limit="123">
      <sequenceNo>
        <gt>0</gt>
      </sequenceNo>
    </engine>
  </engines>
</auditQuery>
```

図. 付１．６ Audit 推奨モデルにおけるクエリのメッセージ例

(b) Audit 取得モデルにおける推奨するレスポンス形式のメッセージ例

上記のクエリに対するレスポンスのメッセージ例を図. 付１．７に示す。BPM Audit 提供から、WS-BPEL 処理系の持つ「エンジン ID」が ″bpel.a.municipal.jp″ で、シーケンス番号が 1 と 2 の BPM Audit が取得されている。なお、value タグのコンテンツは BPEL エンジンの variable 値を圧縮した上、Base64 でエンコーディングしているが、紙面の都合上途中を省略している。メッセージ Audit も同様であるが、この中には、モニタリング問い合わせインタフェースに関連し、業務メッセージ内のシステム制御情報のヘッダの一部として受付番号が含まれている。

```
<BPMAudit>
  <processInstance>
    <sequenceNo>1</sequenceNo>
    <processInstanceId>1</processInstanceId>
    <status>running</status>
    <processDefId>lgAbp</processDefId>
    <activity>
      <date>2007-09-19T05:41:21.203Z</date>
      <status>end</status>
      <activityId>/process[1]/sequence[1]/receive[1]</activityId>
    </activity>
    <scope>
      <variable>
        <value>UEsDBBQACAAIAAp2MzcAAAA...BQAAAAA=</value>
        <updatedDate>2007-09-19T05:41:21.203Z</updatedDate>
```

```
            <variableName>受付-OUTPUT</variableName>
          </variable>
        </scope>
        <engine><engineId>bpel.a.municipal.jp</engineId></engine>
        <machine><machineId>00-0C-29-0A-3E-8E</machineId></machine>
      </processInstance>
      <processInstance>
        <sequenceNo>2</sequenceNo>
        <processInstanceId>1</processInstanceId>
        <status>running</status>
        <processDefId>lgAbp</processDefId>
        <activity>
          <date>2007-09-19T05:41:21.359Z</date>
          <status>end</status>
          <activityId>/process[1]/sequence[1]/reply[1]</activityId>
        </activity>
        <scope>
          <variable>
            <value>UEsDBBQACAAIAAp2MzcAAAAAAAAAAAAAAAAIAAA…DAIAAAAA</value>
            <updatedDate>2007-09-19T05:41:21.359Z</updatedDate>
            <variableName>受領 Ack </variableName>
          </variable>
        </scope>
        <engine><engineId>bpel.a.municipal.jp</engineId></engine>
        <machine><machineId>00-0C-29-0A-3E-8E</machineId></machine>
      </processInstance>
    </BPMAudit>
```

図．付１．７ Audit 推奨モデルにおけるレスポンスのメッセージ例

付録２　実装コンポーネント構成に関する推奨モデル

（１）概要

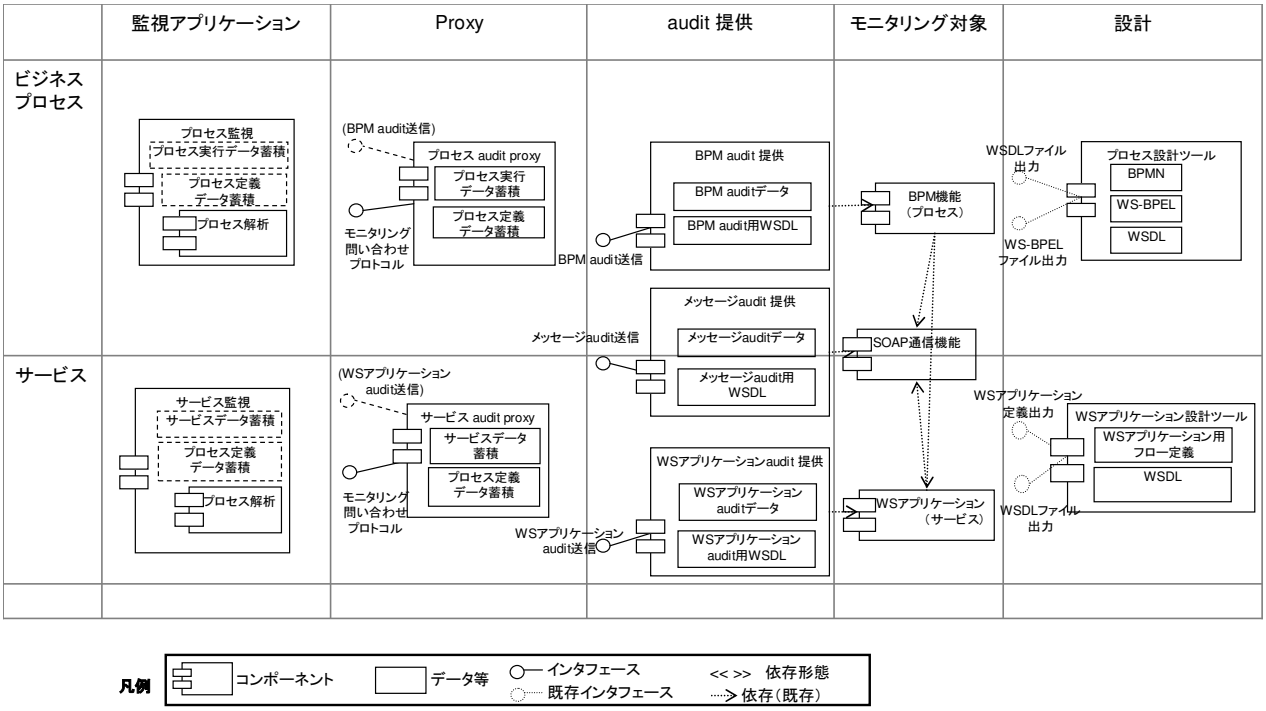
付録として、モニタリング機能を実装するに当たっての実装コンポーネント構成に関する推奨モデルを記載する。アーキテクチャ標準仕様には、モニタリング機能の概略構成、アーキテクチャについて定義があるものの、実装に対する推奨事項は記載されていない。そのため、以下について推奨モデルとして記載する。

(a)奨励コンポーネントの概要

(b)Audit 提供とのカージナリティの関係

（２）奨励コンポーネントの概要

図．付２．１に奨励するコンポーネント群について、その構成要素を、機能階層と伴に展開したものを記す。表．付２．１では、図．付２．１の構成要素について解説する。



図．付２．１　コンポーネント図

表．付２．１　コンポーネントの説明

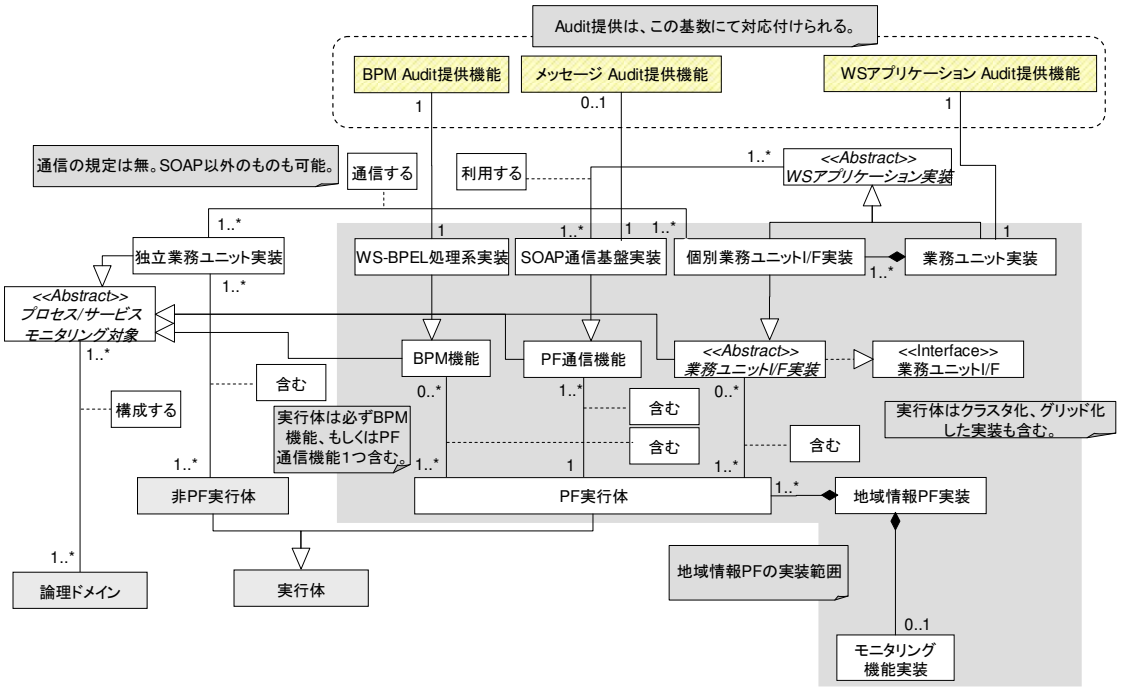
番号	コンポーネント名	概要説明
1.	ビジネスプロセス	BPM(Business Process Management) Audit 提供およびメッセージ Audit 提供から提供される BPM Audit およびメッセージ Audit を、プロセス監視が収集しビジネスプロセスの分析を行う。

2.		プロセス監視	監視アプリケーションである。BPM Audit を観測することで、BPM 機能における業務実行の状況監視などを行う。さらに、メッセージ Audit を BPM Audit と関連させて分析することで、より精度の高い解析を行うこともできる。内部コンポーネントとしてプロセス解析(進捗管理)等の外にプロセス実行データ蓄積、プロセス定義データ蓄積を持ち得る。
3.		プロセス実行データ蓄積	BPM Audit およびメッセージ Audit を蓄積するもので DBMS を想定する。監視アプリケーションが持っても良い。
4.		プロセス定義データ蓄積	モニタリング対象の把握・管理を実施するためメタデータ、種々の情報モデルを定義、管理、維持する。監視アプリケーションが持っても良い。
5.		プロセス解析(進捗管理)	蓄積した Audit データとプロセス定義データから、ビジネスプロセスの進行状況等を解析する。
6.		プロセス Audit Proxy	プロセス監視と BPM Audit 提供の間に配置され、BPM Audit のキャッシュをはじめとする情報統合化機能を提供する。通常は、モニタリング問い合わせインタフェースのクエリーレスポンスのインタフェースでアクセスするが、プロセス監視からは BPM Audit 提供のように見える設定も可能である。
7.		プロセス実行データ蓄積	BPM Audit およびメッセージ Audit を蓄積するもので DBMS を想定する。
8.		プロセス定義データ蓄積	モニタリング対象の把握・管理を実施するためメタデータ、種々の情報モデルを定義、管理、維持する。
9.		BPM Audit 提供	観測対象である BPM の WS-BPEL(Web Services Business Process Execution Language 2.0)処理系の Audit データを提供する。Audit データには WS-BPEL 処理系で業務を実行する際に発生する状態遷移に関する情報が含まれる。WSDL を通じてインタフェースが公開される。
10.		BPM Audit データ	観測対象である BPM の WS-BPEL 処理系が生成する各種のログ、BPM の内部状態等に基づく Audit データ。
11.		BPM Audit 用 WSDL	BPM Audit を提供する Web サービスの WSDL。
12.		メッセージ Audit 提供	観測対象である SOAP(Simple Object Access Protocol)通信機能の Audit データを提供する。Audit データには BPM 機能である WS-BPEL 処理系、ないし WS アプリケーション間でやり取りされる、業務に関する SOAP 通信の送信元、受信先、送信時刻、受信時刻、やり取りされた SOAP メッセージなどの情報が含まれる。WSDL でインタフェースが公開される。
13.		メッセージ Audit データ	観測対象である SOAP 通信機能が生成する各種のログ、SOAP 通信の内部状態等に基づく Audit データ。
14.		メッセージ Audit 用 WSDL	メッセージ Audit を提供する Web サービスの WSDL。
15.		BPM 機能 (プロセス)	観測対象である BPM の処理系のことで、WS-BPEL 処理系等が相当する。
16		SOAP 通信機能	観測対象である SOAP 通信を実施する処理系のこと。BPM 機能である WS-BPEL 処理系、ないし WS アプリケーション間でやり取りされる業務に関する SOAP 通信から、メッセージ Audit 提供でメッセージ Audit が生成される。
17.		プロセス設計ツール	BPM 機能に関連する設計ツール群。例えば、BPM 機能である WS-BPEL 処理系で実行する WS-BPEL の定義ファイル、及び WSDL の定義ファイルを生成・編集するツールなどが相当する。
18.		BPMN	ビジネスプロセスを定義した BPMN(Business Process Modeling Notation)定義。
19.		WS-BPEL	BPMN 定義に従いビジネスプロセスを定義した WS-BPEL の定義ファイル。
20.		WSDL	BPMN 定義および BPEL 定義を提供する Web サービスの WSDL の定義ファイル。
21.	サービス		WS アプリケーション Audit 提供から提供される WS アプリケーション Audit を、サー

			ビス監視が収集し、サービスの分析を行う。
22.		サービス監視	監視アプリケーションである。WS アプリケーション Audit を観測することで、WS アプリケーションにおける業務実行の状況監視などを行う。内部コンポーネントとしてサービスデータ蓄積、プロセス定義データ蓄積、プロセス解析を持つ。サービスデータ蓄積は WS アプリケーション Audit を蓄積するものでデータベースを想定する。
23.		サービスデータ蓄積	WS アプリケーション Audit を蓄積するものでデータベースを想定する。監視アプリケーションが持っても良い。
24.		プロセス定義データ蓄積	WS アプリケーション内の動作を擬似的に WS-BPEL を用いて記述した疑似 WS-BPEL 定義と種々の情報モデルを定義、管理、維持する。監視アプリケーションが持っても良い。
25.		プロセス解析	収集した WS アプリケーション Audit とプロセス定義データを関連づけることで、WS アプリケーション内の業務実行を監視するための各種情報を生成する。
26.		サービス Audit proxy	サービス監視と WS アプリケーション Audit 提供の間に配置され、WS アプリケーション Audit のキャッシュをはじめとする情報統合化機能を提供する。通常は、モニタリング問い合わせインタフェースのクエリーレスポンスのインタフェースでアクセスするが、サービス監視からは WS アプリケーション Audit 提供のように見える設定も可能である。
27.		サービスデータ蓄積	WS アプリケーション Audit を蓄積するものでデータベースを想定する。
28.		プロセス定義データ蓄積	WS アプリケーション内の動作を擬似的に WS-BPEL を用いて記述した疑似 WS-BPEL 定義と種々の情報モデルを定義、管理、維持する。
29.		WS アプリケーション Audit 提供	観測対象である WS アプリケーションの Audit データを提供する。Audit データには WS アプリケーションで業務を実行する際の状態遷移に関する情報が含まれる。WSDL を通じてインタフェースが公開される。
30.		WS アプリケーション Audit データ	モニタリング対象である WS アプリケーションが生成する各種のログ、WS アプリケーションの内部状態等に基づく Audit データ。
31.		WS アプリケーション Audit 用 WSDL	WS アプリケーション Audit を提供する Web サービスの WSDL の定義ファイル。
32.		WS アプリケーション(サービス)	観測対象である WS アプリケーション。
33.		WS アプリケーション設計ツール	WS アプリケーションに関する設計ツール群。例えば、WS アプリケーションが提供するサービスの WSDL 定義ファイルを生成・編集するツールなどが相当する。
34.		WS アプリケーション用フロー定義	WS アプリケーション内の動作を WS-BPEL を用いて記述した疑似 WS-BPEL 定義。
35.		WSDL	WS アプリケーション用フロー定義を提供する Web サービスの WSDL の定義ファイル。

（３）Audit 提供とのカージナリティの関係

Audit 提供とのカージナリティの関係に関しては、図．付２．２の様に推奨する。モニタリング機能の実装に対して、実装コンポーネント構成に関する推奨モデルを図．付２．１に奨励されているが、これに対して、Audit 提供等をどの様に対応付けるか、を図．付２．２では記載する。図中、灰色にて網掛けされているクラスが地域情報プラットフォーム内に実装されるクラス群である。各 Audit 提供は、各処理系実装と１対１の関係で実装されることが望ましい。表．付２．２は、図．付２．２の情報モデルのクラスを説明したものである。



図．付２．２　Audit 提供との基数関係、クラス図

表．付２．２　Audit 提供との基数関係、クラス

クラス名	定義、デプロイメント制約
実行体	地域情報プラットフォーム実装の実行環境については、プラットフォーム通信標準仕様における OS 事項などの制限以外、デプロイメント・構成に関する制約事項は存在しない。 この「実行体」クラスとは、CPU 資源に代表される計算機資源を持ち、BPM 機能の実行やサービスの実行提供等を行う計算機を一般化したものである。 さらに、このクラスは、地域情報プラットフォームを実装、実現する際の資源を意味する「PF 実行体」クラスと、地域情報プラットフォームから呼び出されて関連する機能を提供するための資源を意味する「非 PF 実行体」クラスの２つに分類される。
PF 実行体	「実行体」クラスの一つで、地域情報プラットフォームを実装、実現する際の資源を意味する。そのため「地域情報 PF 実装」クラスは、これを複数含んで構成される。「PF 実行体」クラスの一つのインスタンスは、０以上の「BPM 機能」クラスのインスタンス、もしくは１以上の「PF 通信機能」クラスのインスタンス、もしくは０以上の「業務ユニット I/F 実装」クラスのインスタンスを含み得る。 具体的には、地域情報プラットフォームの機能がインストール、実装されたサーバ等が該当する。
非 PF 実行体	「実行体」クラスの一つで、地域情報プラットフォームから呼び出されて関連する機能を提供するための資源を意味する。具体的には、「業務ユニット I/F 実装」クラスの任意インスタンスから呼び出されて、プロプラエタリの通信手段により、サービス実施を行うアプリケーションを意味する「独立業務ユニット実装」クラスの複数種の複数インスタンスを保持し得る。

	具体的には、業務ユニットやバックエンドの業務アプリケーション機能がインストール、実装されたサーバが該当する。これらは、地域情報プラットフォームの実装基盤からサービス呼出を受けるが、地域情報プラットフォーム実装の範囲内のサーバとしては扱われていないものである。
(Abstract)プロセス/サービスモニタリング対象	プロセス、サービスについてのモニタリング対象を意味する抽象クラスである。この具体的な実装が、「BPM 機能」「PF 通信機能」「業務ユニット I/F 実装」「独立業務ユニット実装」の各クラスである。
BPM 機能	地域情報プラットフォームの BPM 機能を意味するクラスであり、「プラットフォーム通信標準仕様、第４章、プラットフォーム通信標準のビジネスプロセス制御定義仕様」を満足する。BPM 機能の実装方式には規定がないため、「PF 実行体」クラスの１インスタンス上に１インスタンス以上を含めることができる。またクラスタ化された実装も考えられるため、逆に、このクラスの１インスタンスは、１以上の「PF 実行体」クラスのインスタンスに展開されることもある。 この「BPM 機能」クラスは、抽象クラスである「プロセス/サービス観測対象」クラスを継承する。さらに、「BPEL 処理系実装」クラスは、この「BPM 機能」の一つである。
PF 通信機能	プラットフォーム通信標準仕様を満たす通信機能を意味するクラスであり、「プラットフォーム通信標準仕様、第２章のプラットフォーム通信仕様」を満足する。PF 通信機能の実装方式には規定はないため、「PF 実行体」クラスの１インスタンス上に１インスタンス以上を含めることができる。 この「標準通信機能」クラスは、抽象クラスである「プロセス/サービス観測対象」クラスを継承する。さらに通常、「SOAP 通信基盤実装」クラスとして継承され、扱われる。
(Interface)業務ユニット I/F	「アーキテクチャ標準仕様、第４章　実現するためのアーキテクチャ」で定義される、汎化されたインタフェースであり、業務ユニットが持つべきインタフェースとして定義される。
(Abstract)業務ユニット I/F 実装	「業務ユニット I/F」インタフェースを実装したクラスであり、抽象クラスである「プロセス/サービス観測対象」クラスを継承したものである。これは、実際のサービスを汎化したものとして定義され、実際のサービスを意味する「個別業務ユニット I/F 実装」クラスが、継承されて定義される。 「PF 実行体」クラスの一つのインスタンスは、０以上の「業務ユニット I/F 実装」クラスのインスタンスを含み得る。これは、実際のサービスを意味する「個別業務ユニット I/F 実装」クラスのインスタンス数に相当する。
個別業務ユニット I/F 実装	「業務ユニット I/F 実装」クラスを継承したもので、実際のサービス毎に実装される。例えば、A 業務、B 業務、C 業務と３つのインタフェースが存在する場合、これに基づき３インスタンス生成される。 これは、「PF 実行体」クラスの同じインスタンス上に置かれる、「業務ユニット実装」クラスのインスタンスに取り込まれるか、もしくは別の実行体である「非 PF 実行体」クラスのインスタンス上に置かれる、「独立業務ユニット実装」クラスのインスタンスと通信を介して、サービスを提供するものである。
業務ユニット実装	「アーキテクチャ標準仕様、第４章　実現するためのアーキテクチャ」で定義される、業務ユニットを実装したクラスであり、汎化されたものである。このクラスの各インスタンスが、具体的なサービスの処理を提供するものである。この場合、「PF 実行体」クラスのインスタンス上の、複数「個別業務ユニット I/F 実装」クラスのインスタンスを組み込んで構成される。
独立業務ユニット実装	「アーキテクチャ標準仕様、第４章　実現するためのアーキテクチャ」で定義される、業務ユニットを実装したクラスであり、汎化されたものである。このクラスの各インスタンスが、具体的なサービスの処理を提供するものである。このインスタンスは「非 PF 実行体」クラスのインスタンス上に配置されるもので、別の実行体である、「PF 実行体」クラスのインスタンス上に配置される「個別業務ユニット I/F 実装」クラスのインスタンスに対して通信を介して処

	理を行う。
(Abstract) WS アプリケーション実装	サービスを実施する主体を抽象的に表現しているクラスであり、実際には、「アーキテクチャ標準仕様、第４章　実現するためのアーキテクチャ」で定義される、業務ユニットを実装したクラスに相当する「業務ユニット実装」クラス、もしくは「個別業務ユニット I/F 実装」クラスに相当する。
WS-BPEL 処理系実装	「BPM 機能」クラスを継承したクラスであり、「プラットフォーム通信標準仕様、第４章、プラットフォーム通信標準のビジネスプロセス制御定義仕様」に関する規定事項を満たす。
SOAP 通信基盤実装	「PF 通信機能」クラスを継承したクラスであり、プラットフォーム通信標準仕様を満たす。
地域情報 PF 実装	地域情報プラットフォームのプラットフォーム通信標準仕様を満たす、具体的な実装であり、１以上の「PF 実行体」クラスのインスタンスを含んで構成される。さらに、「モニタリング機能実装」インスタンスも、含み得るが、シングルトンを仮定する。
論理ドメイン	「プロセス/サービスモニタリング対象」クラスを継承する複数のインスタンスを含んで構成される、プロセス/サービス監視の単位対象範囲のことで、同じ「プロセス/サービスモニタリング対象」のインスタンスは複数の「論理ドメイン」に参加できる。１以上の任意個数の「標準通信機能」のインスタンスは、必ず登録される。 １つの「論理ドメイン」のインスタンスは、それを管理するシステム管理者がローカルに定める、セキュリティポリシーにより管理され、同一のポリシーで扱われる。従って、通常は組織単位で定義されるが、組織に跨って定義されても良い。
モニタリング機能実装	本仕様書の要件を満たす実装を意味するクラスであり、１つの「登録定義」クラスのインスタンスのもと、「プロセス/サービス監視対象範囲」を１つ保持することができる。
BPM Audit 提供機能	BPM 処理系の動作を表す Audit を提供する。BPEL 処理系実装と１対１で関係付けられる。
メッセージ Audit 提供機能	ネットワーク上で流通する業務データの内容、状態を表す Audit を提供する。SOAP 通信基盤実装と１対１で関係付けられる。但し、メッセージ Audit 提供機能自身は、オプションであるため、基数的には１対(0..1)となる。
WS アプリケーション Audit 提供機能	業務ユニット等のアプリケーションの動作を表す Audit を提供する。業務ユニット実装と１対１で関係付けられる。

付録３　共通ヘッダ、受領 Ack、添付書類の XML スキーマについて

本付録では、プラットフォーム通信標準仕様により定められている下記の地域情報 PF において共通的な XML スキーマについて説明する。

・共通ヘッダ
・受領 Ack
・添付書類

（１）共通ヘッダ

メッセージ共通のヘッダ（共通ヘッダの項目セット）は、プラットフォーム通信標準仕様における「７．１．２　共通ヘッダの項目のＸＭＬスキーマ作成仕様」で規定した以下の形式を使用する。

表．付３．１　共通ヘッダのＸＭＬスキーマ仕様

No.	データ項目	XML データ型	最小文字数	最大文字数	最小出現回数	最大出現回数
1	共通ヘッダ	共通ヘッダ情報				
2	To	string	0	1024	0	1
3	MsgID	string	0	1024	0	1
4	RelatesTo	string	0	1024	0	1
5	ReplyTo	string	0	1024	0	1
6	受付番号	string	0	256	1	1
7	共通コリレーションセット	string	0	256	0	1
8	ビジネスプロセス制御情報	string	0	100	0	1
9	業務サービス結果情報	string	0	10	0	1
10	結果情報	string	0	1	0	1
11	システムエラー報告	string	0	1024	0	1

（２）受領 Ack

受領 Ack は、下図に従う。受領 Ack は、共通ヘッダとリクエストやレスポンスを受領できたかの成否のための受領ステータス、受領日時から構成される。

表．付３．２　受領 Ack のＸＭＬスキーマ仕様

No.	データ項目	XML データ型	最小文字数	最大文字数	最小出現回数	最大出現回数
1	受領 Ack	受領 Ack 情報				
2	共通ヘッダ	共通ヘッダ情報				
3	受領ステータス	string	1	1	1	1
4	受領日時				1	1
	年	string	4	4	0	1
	月	string	2	2	0	1
	日	string	2	2	0	1
	時	string	2	2	0	1
	分	string	2	2	0	1
	秒	string	2	2	0	1

No.	データ項目	XML データ型	最小文字数	最大文字数	最小出現回数	最大出現回数
5	備考	string	0	1024	0	1

（３）添付書類

添付ファイルをメッセージ本文で扱う場合の 2 種類の方法を下記に示す。添付書類がある場合は、以下のいずれかの形式を、メッセージ設計時にメッセージの最後に追加する。

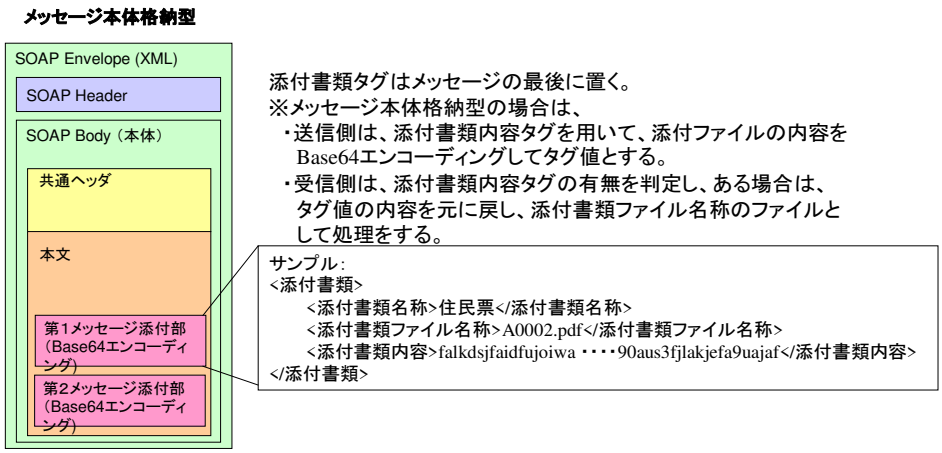


図. 付３. １　添付ファイルのための XML タグ形式

表．付３．３　添付書類のＸＭＬスキーマ作成仕様

No.	データ項目	XML データ型	最小 文字数	最大 文字数	最小 出現回数	最大 出現回数
1	添付書類	添付書類情報				
2	添付書類名称	string	(※1)	(※1)	1	1
3	添付書類ファイル名称	string	(※1)	(※1)	1	1
4	添付書類内容	string	(※1)	(※1)	1	1
5	添付書類参照情報	Ref.swaRef	(※1)	(※1)	1	1

(※1)最小文字数・最大文字数に関しては、プラットフォーム通信標準仕様 V2.1 では規定していない。

（４）ＸＭＬスキーマファイルに関して

前述した、共通ヘッダ、受領 Ack、添付情報の３要素の定義を、プラットフォーム通信標準仕様 V2.1 の一部として、単一のＸＭＬスキーマファイルで提供する。

プラットフォーム通信標準仕様の「３．２　（５）ＸＭＬに関する取り決め」に示してあるとおり、共通ヘッダ、受領 Ack、添付書類のいずれか、もしくは全てを扱う場合には、プラットフォーム通信標準仕様として提供しているスキーマを使用することとする。

ＸＭＬスキーマファイルの名称は、「プラットフォーム通信標準仕様 V2.1」の「３．２.（７）④ファイル命名規約（※2)」に従い、次に示すとおりである。

ファイル命名例）common-2010-01.xsd

（※2）次のファイル命名形式が示されている。

・ファイル命名形式

{地域情報PF仕様種別の文字列}＋{定義識別子}＋{s}＋{-バージョン文字列}.{拡張子}

・共通ヘッダ、受領 Ack、添付書類の３定義を単一のファイルとして提供するため、定義識別子は設定しない。

（５）「common-YYYY-NN.xsd」のファイルの更新規約に関して

本スキーマファイルは、共通ヘッダ、受領 Ack、添付書類のいずれかの定義に変更が行われた場合、スキーマファイルの更新に合わせ、スキーマファイル内の名前空間 URI のバージョン文字列と、ファイル名に使用されるバージョン文字列の同期をとって更新する。

そのため、前述した「common-YYYY-NN.xsd」が更新された場合は、他の仕様でこれをを参照しているスキーマ内の「名前空間識 URI」と「参照先ファイル名」の更新が必要となる。

（６）使用する定義に関して

各業務仕様で共通ヘッダ、受領 Ack、添付書類のＸＭＬスキーマを使用する場合は、そのスキーマファイル内でグローバル要素定義されている、共通ヘッダ、受領 Ack、添付書類の要素を原則使用する。

使用例）

○ <xsd:element ref="common:共通ヘッダ"/>
○ <xsd:element ref="common:受領 Ack"/>
○ <xsd:element ref="common:添付書類"/>

（７）タグの値省略に関して

共通ヘッダ、受領 Ack、添付書類のＸＭＬスキーマでは、タグの値が無い（NULL 値）の場合は、プラットフォーム通信標準仕様の「表３．２．２　データ項目の値が省略された場合のＸＭＬ表現方法」の「No.１タグを省略する」を使う。xsi:nil での NULL 値表現は不可とする。

（８）ＸＭＬスキーマと対応する標準仕様のバージョン情報について

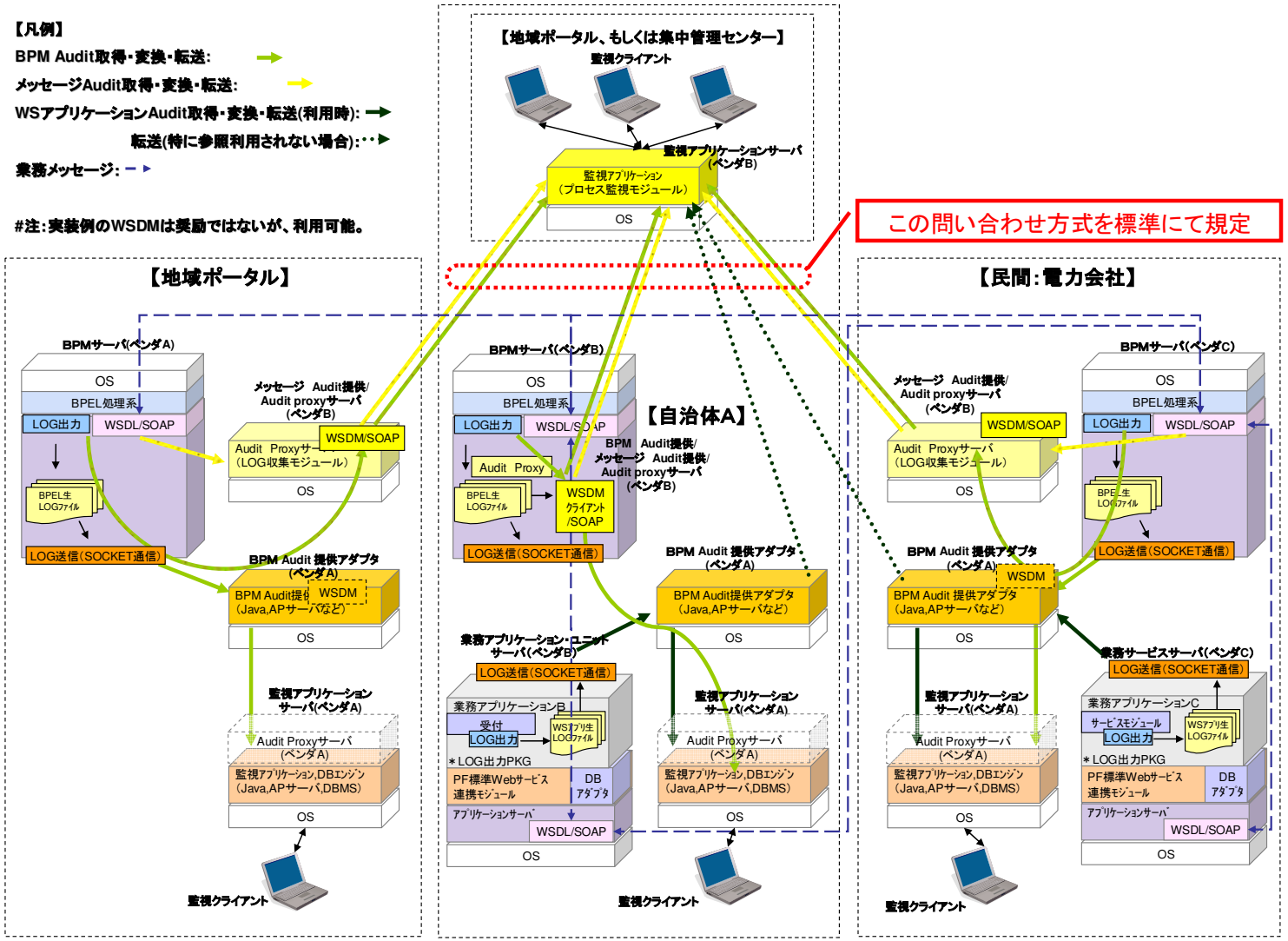
「プラットフォーム通信標準仕様 V2.1」の「３．２（５）ＸＭＬに関する取り決め　⑫」に「『XML スキーマの<xsd:documentaion>の領域に、そのスキーマを提供している地域情報プラットフォーム標準仕様情報』について記述する」表記ルールが示されている。

その表記ルールを踏まえた「common-YYYY-NN.xsd」における情報の記載例を次に示す。

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
  targetNamespace="urn:applic:xmlns:pf:common:schema:2010-01"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:common="urn:applic:xmlns:pf:common:schema:2010-01"
  xmlns:ref="http://ws-i.org/profiles/basic/1.1/xsd"
  elementFormDefault="qualified">

  <xsd:annotation>
    <xsd:documentation>
      本文書は「共通ヘッダ」と「受領 Ack」と「添付書類」に関する XML スキーマである。
      管理主体：APPLIC
      著作権：APPLIC
      ファイル名：common-2010-01.xsd
      PF 仕様:プラットフォーム通信標準仕様　V2.1
      作成日：2010/05/31
    </xsd:documentation>
  </xsd:annotation>

  <xsd:import namespace="http://ws-i.org/profiles/basic/1.1/xsd"
schemaLocation="http://ws-i.org/profiles/basic/1.1/swaref.xsd" />

  <xsd:element name="共通ヘッダ" type="common:共通ヘッダ情報" />
  <xsd:element name="受領 Ack" type="common:受領 Ack 情報" />
  <xsd:element name="添付書類" type="common:添付書類情報"/>
```

図．付３．２　「common-YYYY-NN.xsd」における<xsd:documentation >領域の記述例

付録４　モニタリング機能の実装例

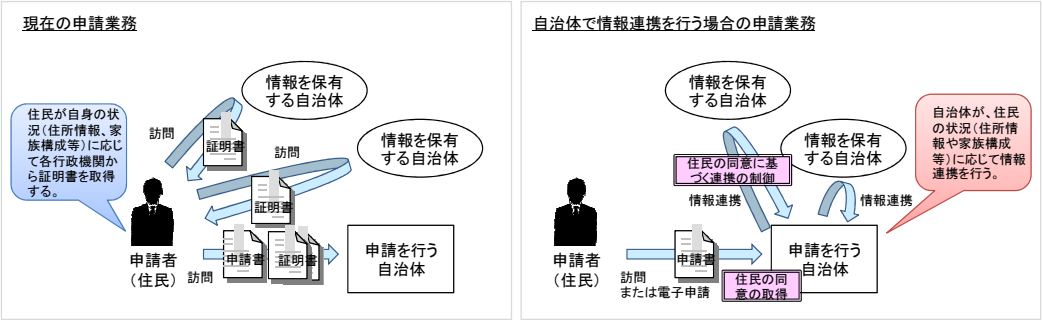


付録５　採用候補の技術仕様の検討状況について

本章で、実現方法の１つとして ID-WSF を候補としている記載がある。ID-WSF については、具体的なユースケースや、技術仕様の標準化動向を踏まえた検討が必要であるとされた。APPLIC における検討状況について以下に示す。

(1) 具体的なユースケースに基づく必要性の検討

自治体が保有する情報を自治体間で相互に連携することで、例えば、現行の申請手続きにおいて、住民が各行政機関を訪問し、必要な証明書を取得するといった手続きが不要となるなどの効果が期待できる。



図．付５．１　自治体で情報連携を行う場合の申請業務

自治体が保有する情報に、無条件で他の自治体が連携することにより、住民にとって新たな脅威となる可能性もあるため、住民の同意に基づく連携の制御や、住民の情報に対するアクセス記録を確認できる仕組みが必要と考える。

上記の仕組みの必要性については、具体的なユースケースに基づき、適用する範囲や内容を検討する必要がある。検討ポイントを以下に示す。

- 法制度による情報連携(公用照会)等、住民同意によらない情報連携の検討
- 情報連携制御に求められる要件を明確化するための、ユースケースの具体化
- 本人同意の範囲、同意の取得のタイミング、住民の解り易さの検討
- 本人同意に基づく連携を制御する仕組みの検討
- 技術面および運用面の観点
- 既存技術と組み合わせた実装の検討（※）

※　補足

例えば、住民情報へのアクセスについて、照会先自治体の職員が承認許可をする為には、ID-WSF と PF サービス認可仕様や非同期 2WAY など MEP と組み合わせた実装が必要となる。

この規格の組み合わせについては、ユースケースへの適合性、実装が複雑にならないか、性能、信頼性等について十分に検討する必要がある。

（２） 技術仕様の標準化動向の検討

地域情報プラットフォーム標準仕様書では､技術標準採用基準として以下の 4 つの観点を挙げており、観点に基づく技術評価が必要となる。

(a) 標準化団体にて標準化が確定しているまたは検討中の仕様
(b) ライセンスフリーの考え方を基本とした仕様
(c) 自治体および地域情報化で実現性が高い仕様
(d) 標準仕様や相互接続ガイドラインで、相互接続の仕様が存在する仕様

ID-WSF について 4 つの観点の状況を示す。
※ ID-WSF2.0、2010 年時点

(a) 標準化団体にて標準化が確定しているまたは検討中の仕様
Liberty Alliance による標準化がなされた後、現在は、Liberty Alliance から引き継がれた Kantara Initiative により ITU への提案活動を行っている。ただし、Kantara Initiative は自らを標準化団体として位置づけておらず、他団体への草案提出を活動内容としている。

(b)ライセンスフリーの考え方を基本とした仕様
Liberty Alliance の活動は特許使用許諾（ライセンス）の条件として RF 及び RAND を認めており、RAND 宣言猶予期間は 45 日間と定められている。
ID-WSF2.0 策定期間において RAND 宣言はされていない。

(c)自治体および地域情報化で実現性が高い仕様
オープンソースでの実装は存在するが、実運用はされていない。現在国内で(4)に示す相互接続認定を受けているのは 1 社である（世界では 2 社)。今後の普及動向を見ていく必要がある。

(d) 標準仕様や相互接続ガイドラインで、相互接続の仕様が存在する仕様
相互接続試験を行う団体として、Liberty Alliance、2009 年以降は Kantara Initiative が存在する。ただし、ID-WSF 2.0 の相互接続認定試験は 2006 年 12 月以降実施されていない。