DIOSA/XTP V3.1

データ変換・通信オプション

導入の手引

輸出する際の注意事項

本製品（ソフトウェア）は、外国為替及び外国貿易法で規制される規制貨物（または役務）に該当することがあります。

その場合、日本国外へ輸出する場合には日本国政府の輸出許可が必要です。

なお、輸出許可申請手続にあたり資料等が必要な場合には、お買い上げの販売店またはお近くの当社営業拠点にご相談下さい。

はしがき

本書は、業務システムの構築を支援する DIOSA/XTP データ変換・通信オプション製品群の導入の手引です。

本書の読者としては、業務アプリケーション開発を担当し、OS、TPBASE、IM/DB、その他関連 PP の使用法を一通り心得ているシステム技術者を想定しています。

2023 年 6 月 第 3 版

本書の関連説明書としては次のものがあります。

- DIOSA/XTP 導入の手引き
- DIOSA/XTP 利用の手引き
- DIOSA/XTP メモリキャッシュ 利用の手引き
- DIOSA/XTP データストア 利用の手引き
- DIOSA/XTP データ変換・通信オプション 利用の手引き
- DIOSA/XTP API リファレンス
- DIOSA/XTP コマンドリファレンス
- DIOSA/XTP 環境定義リファレンス
- DIOSA/XTP メッセージリファレンス

目次

第1章　概要

1.1　目的と特徴

1.1.1　目的

社会インフラの重要性が増す中、大規模アプリケーションシステムにおいても、高信頼、高性能、高運用・高稼動性、そしてアプリケーション開発の高生産性・即応性への要求が益々高まっています。

本製品は、これらの要件を満足するべく、以下の機能を提供します。

- データベースサーバ障害時でもオンライン業務の継続が必要となるシステムの開発を支援する超高可用性を実現したデータベース制御機能の提供
- フロントシステム／バックアップシステム構成で多重化しているシステムの、業務システム継続性を向上させるためのデータ同期／移行機能の提供

1.1.2　特徴

システム構築における開発作業の効率化と維持管理の簡易化や共通化を可能とするデータベース制御機能を提供します。

1.1.3　データ同期制御

データ同期制御は、ユーザアプリケーションが更新したユーザデータと同期したデータを、同一システム上、他システム上に保持するための機能です。

データ同期制御は以下の特徴があります。

- ユーザアプリケーション処理と同一のトランザクションでは、更新内容の保存のみをおこない、それ以降のデータ同期処理は、即時実行される非同期処理でおこなうため、リアルタイムトランザクションへの影響を抑えることが可能です。
- 保存した更新内容は、トランザクションの原子性や順序性を保持して確実に処理されることが保証されるため、データ同期対象システムの運用状態などを意識する必要がありません。
- TAM 表の同期データを、別システムの TAM 表として保持するといった、単純なレプリケーションだけでなく、TAM 表と同期した DB 表を同一システム上、別システム上に保持するような形態も可能です。
- IM の更新内容については、DB アクセス制御機能と連携して自動的に出力するため、ユーザアプリケーションは、データ同期のための特別な処理をおこなう必要がありません。
- 更新内容は、データ圧縮、データ分割されるため、同期処理中のリソースの消費を抑えることができます。

1.2　諸概念

1.2.1　データ同期制御

IM-DB データ同期制御、センタ間データ同期制御には、以下の共通概念があります。

(1) 更新ログ

更新ログは、フロントシステムにおける IM/DB の 1 更新の更新内容を保持するデータです。

C アプリケーションが、DB アクセス制御機能を利用して IM を更新した場合、DB アクセス制御機能によって、更新ログが自動的に生成されます。WebOTX 上の Java アプリケーションが Oracle を更新した場合、DB アクセス制御機能を使用して更新ログを生成する必要があります。前者を TAM 更新ログ、後者を Oracle 更新ログと呼びます。

(2) ログデータ

1 トランザクションで発生した更新ログは、1 つのデータにまとめられて DIOSA/XTP データストアが管理している、IM/DB 上のプールファイル(データストアの制御表)に格納されます。この、まとめられたデータのことを、ログデータと呼びます。通常 1 トランザクションで、1 ログデータが発生します。C アプリケーションが、DB アクセス制御機能を利用して IM を更新した場合、ログデータの生成、格納は自動的に実行されます。

WebOTX 上の Java アプリケーションが Oracle を更新した場合、生成した更新ログを、センタ間レプリケーション機能を使用して登録し、トランザクション終了時にトランザクション終了処理を呼び出すことで、登録された更新ログを 1 つのログデータとしてプールファイルへ格納します。

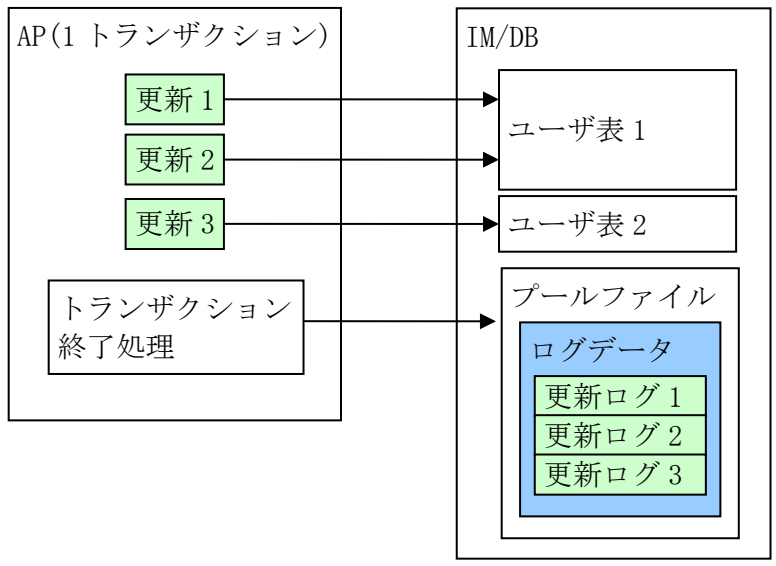


図 1-1 更新ログとログデータ

(3) ストリーム(スーパーストリーム)

ストリームとは、DIOSA/XTP データストアの概念で、ログデータをシーケンシャルに処理する単位です。同一ストリームに発生したログデータは、発生順に転送、処理されることが保証されます。

データストアによるログデータの送受信、更新ログの反映処理はストリーム単位に並列処理されるため、ストリーム数を増やすほど更新ログ反映処理の性能が向上します。

ただし、異なるストリームに発生した更新ログ間の処理順序は保証されないため、更新ログを複数のストリームに分散させて処理する際には注意が必要です。

なお、データストアのリファレンスなどで、スーパーストリームという記述がありますが、ストリームと同一の概念です。

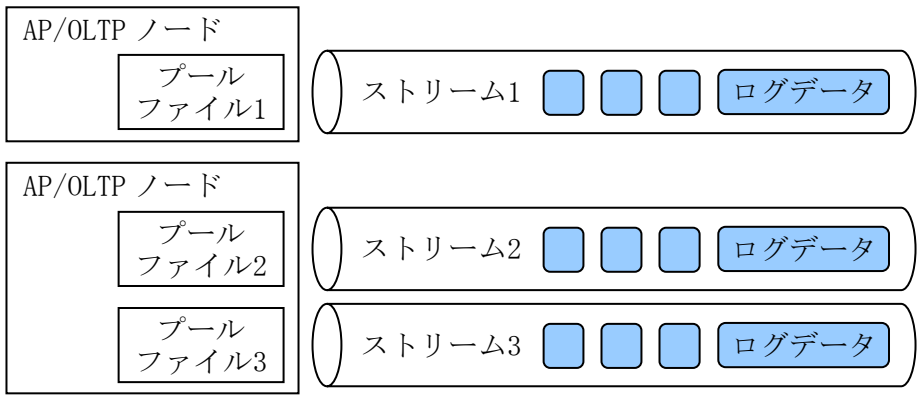


図 1-2 ストリーム

1.2.2 オンライン中メンテナンス

オンライン中メンテナンス機能には、以下の概念があります。

(1) 運用情報

MAP 毎の TAM 再配置(データベース構成変更を行うための手順)の実行状態を管理する情報を運用情報と呼びます。

運用情報には以下の 2 つの要素があります。

- 運用モード
- MAP 構成変更状態

運用モードは、TAM 再配置の進捗状態を把握するための状態です。以下の 5 つの状態があります。

運用モード	説明
IM アクセスモード	通常の運用を行っている状態です。
IM→DB 移行状態	IM アクセスモードから DB アクセスモードに切り替える途中の状態です。
DB アクセスモード	IM のメンテナンスを行う状態です。
DB→IM 移行状態	DB アクセスから IM アクセスに戻す途中の状態です。
未使用状態	MAP の追加直後/削除した後の状態です。

MAP 構成変更状態は MAP の構成を変更する TAM 再配置手順を実施中かを表します。

MAP の構成を変更する TAM 再配置手順を実施中は全データ削除(TRUNCATE)を行えないため、この状態により全データ削除の実行可否を制御します。

- 通常　　：TAM 再配置を行っていない、または MAP の構成を変更しない TAM 再配置中の状態
- 構成変更中　：MAP の構成を変更する TAM 再配置中の状態

(2) ユーザカレントデータ群

ユーザカレントデータ群は DIOSA/XTP が管理する論理表のうち、利用者が DB アクセス制御機能を利用してデータアクセスするものを指します。TAM 再配置機能、セーブロード機能は、ユーザカレントデータ群を処理の対象とします。

ユーザカレントデータ群とするには、以下の定義を行う必要があります。

- DIOSA/XTP メモリキャッシュ機能の環境定義 IMTABLECONF 節に、論理表を「表種別=1～9」(利用者が

使用する論理表)として定義する。

・ DB アクセス制御機能の環境定義 DACENV 節に、論理表を定義します。

(3) データ変換・通信オプション制御情報群

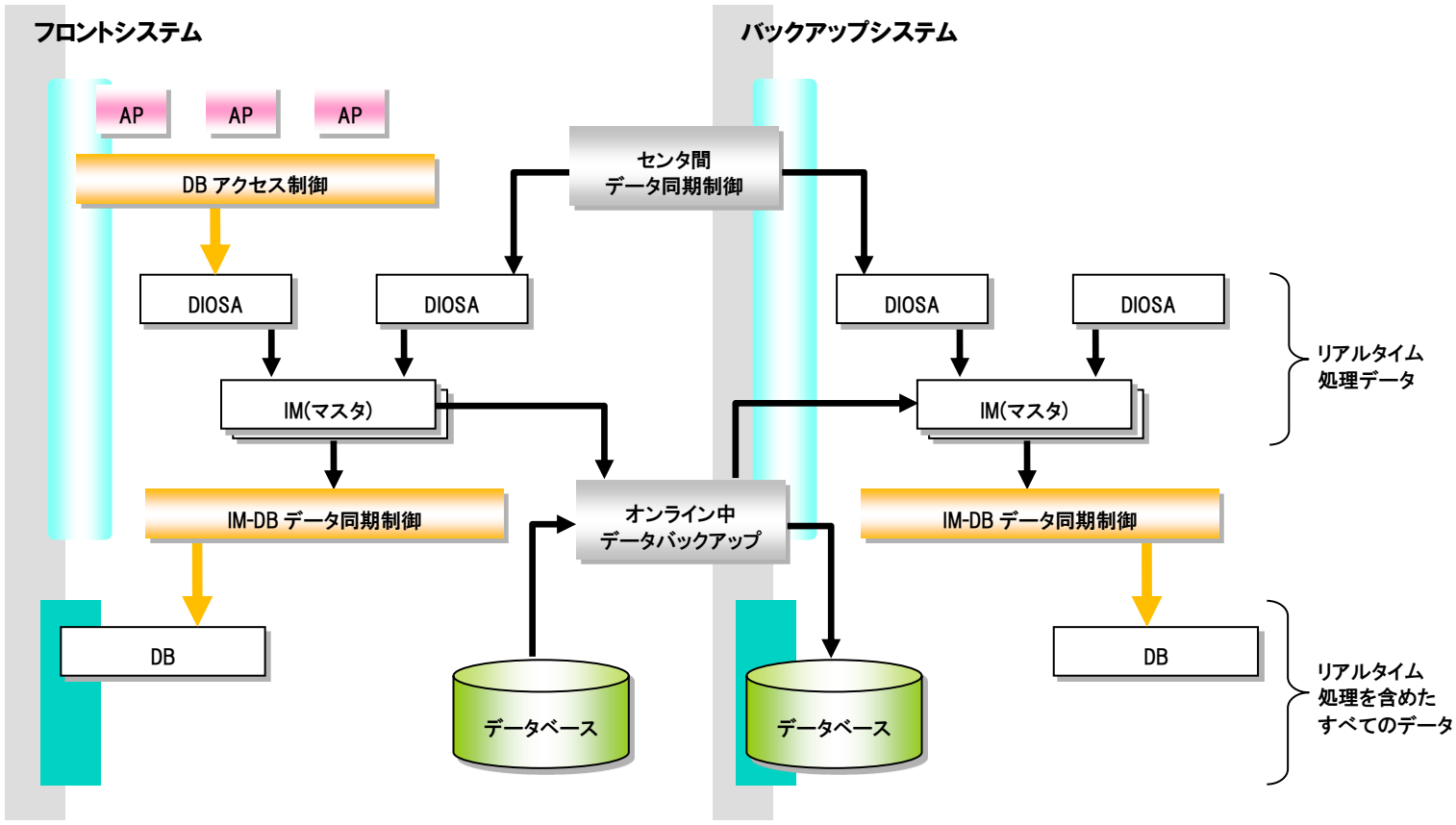
データ変換・通信オプション制御情報群は DIOSA/XTP が管理する論理表のうち、DB アクセス制御機能の制御表を指します。TAM 再配置機能、セーブロード機能は、データ変換・通信オプション制御情報群を処理の対象とします。

第2章　システムの構築

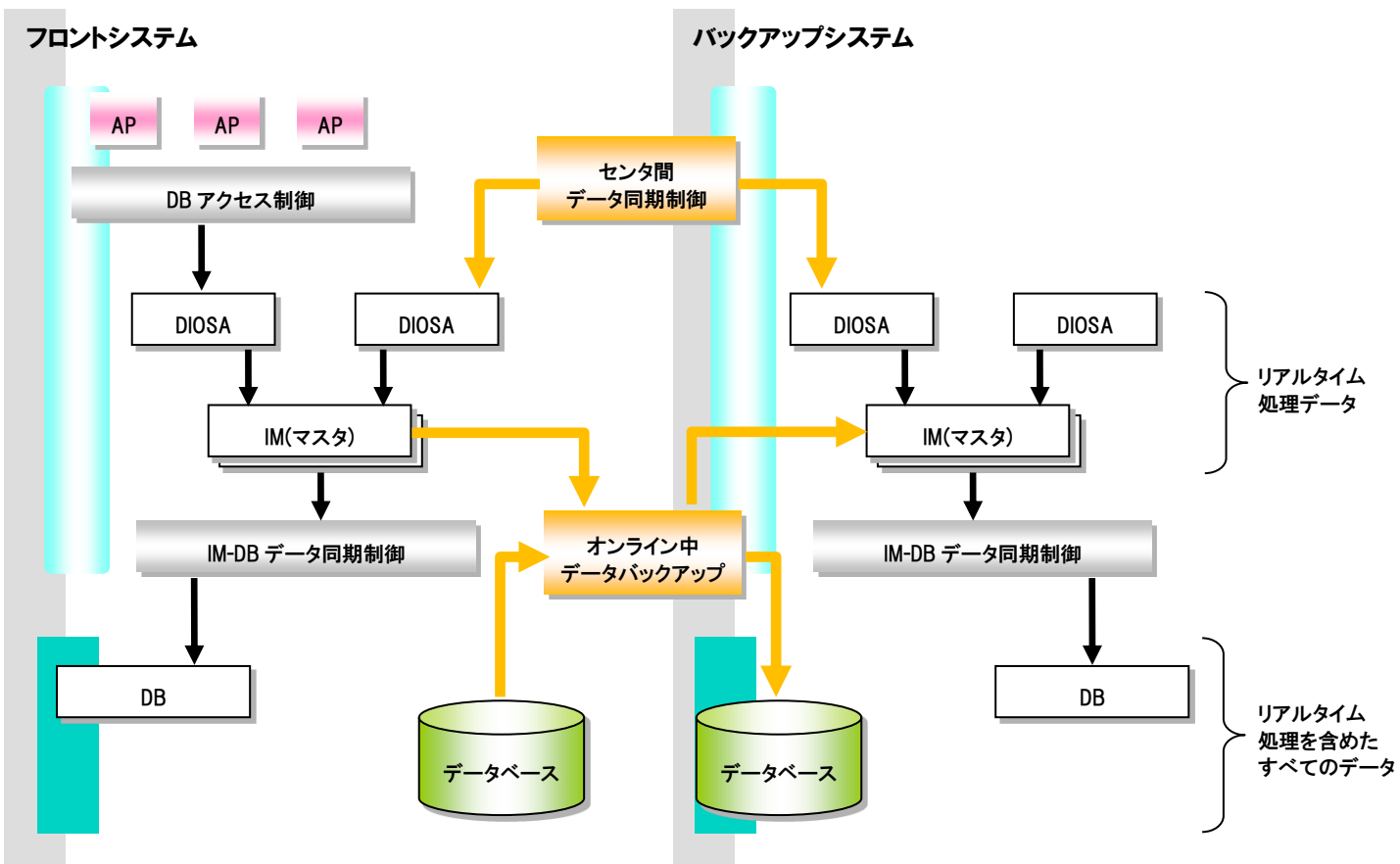
2.1　システム概要

本製品のシステム構成について以下に示します。

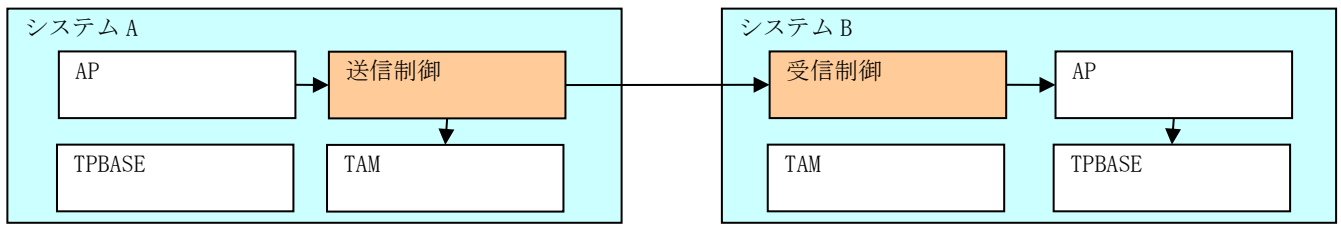
インメモリ DB アクセスユーティリティ システム構成例



ディザスタリカバリユーティリティ システム構成例



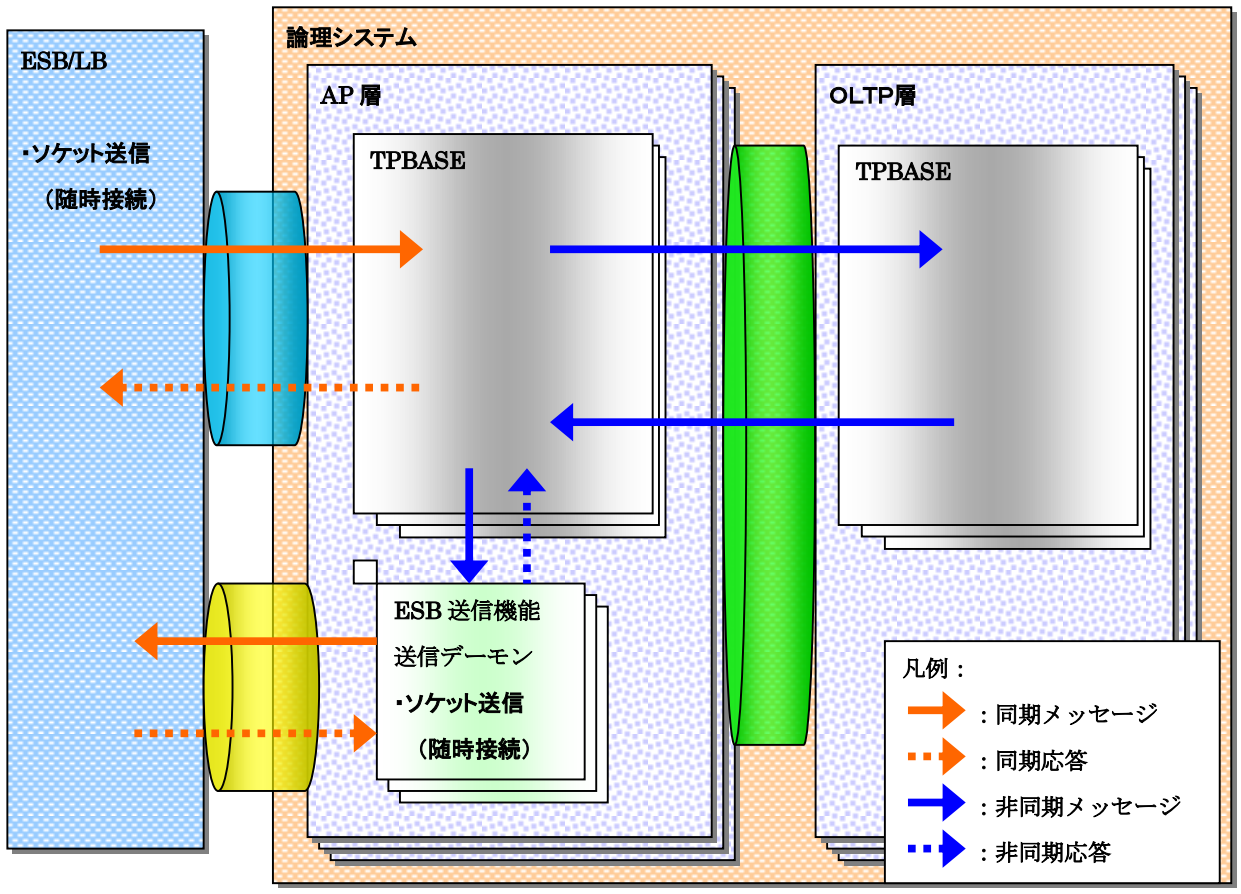
通信接続ライブラリ システム構成例



2.2　環境設計

2.2.1　ノード・ネットワーク構成

論理システム内の通信は DIOSA/XTP、TPBASE を用いた非同期通信を行います。他サブシステムと通信を行う場合は同期通信を行います。他サブシステムから要求メッセージを受信する場合は TPBASE を使用して通信を行いません。他サブシステムへ要求メッセージを送信する場合は本製品が提供する ESB 送信機能の送信デーモンを利用して通信を行います。TPBASE の設定方法については 2.3.7TPBASE の環境定義を参照してください。



2.2.2　　機能設計

(1)　　多重度の設計

(a)　データ同期制御

レプリケーション処理を並列処理させるためには、更新ログを複数のストリームに分散させる必要があります。

更新ログのストリーム分散方法は、IM と DB で異なりますが、いずれの場合も、以下のルールを守る必要があります。

- 同一レコードの更新ログは常に同じストリームに発生させる。
- 1 トランザクションで複数ストリームに更新ログを発生させない。

(i)　IM の場合

IM を更新するトランザクションは、MAP に分散され、トランザクション内では分散された MAP 内のレコードのみを更新します。このため、MAP ごとに 1 ストリームを割り当て、更新ログを分散させれば、上記のルールに従ったストリーム分散が可能です。

更新ログを MAP に対応したストリームに書き込む制御は、DB アクセス制御とセンタ間レプリケーション機能が連携しておこなうため、アプリケーションではストリーム分散を意識する必要はありません。

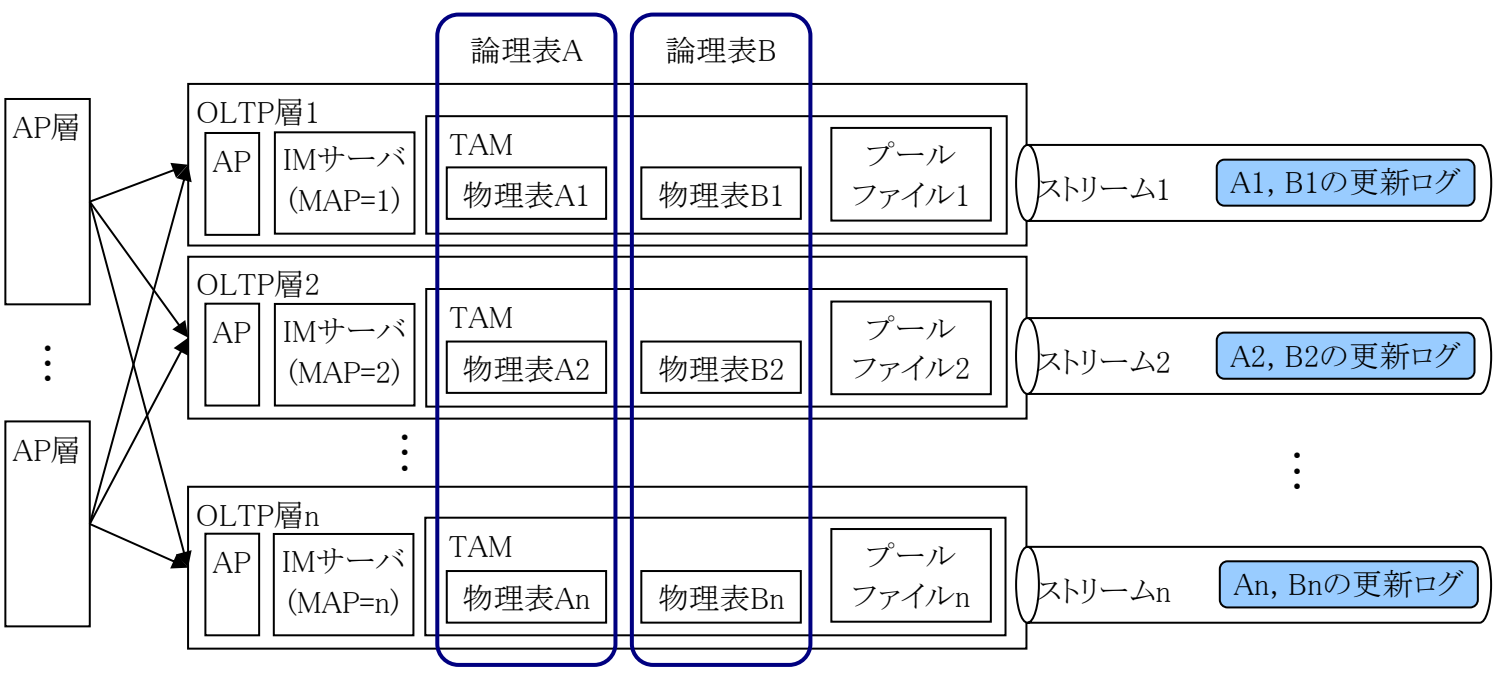


図 3 TAM 表更新ログのストリーム分散

［同一 MAPID の更新ログ分散について］

上記の例では、1 つの MAP に 1 ストリームを割り当てていますが、1 つの MAP に複数のストリームを割り当てることも可能です。その際、更新ログをどのように各ストリームに振り分けるかは、利用者出口によって利用者が指定する方法か、ハッシュ値から自動的に振り分ける方法を利用者が選択することができます。詳細については「2.3.2 (1)(b) MAPID 内ストリーム分割用設定」を参照してください。

(ii) DB の場合

DB の場合、IM のように MAP ごとに物理表が分散されないため、アプリケーションがストリームを意識的に分散させる必要があります。

分散については、前頁に記載したルールが守られていれば任意の分散で構いませんが、例としては以下のような分散方法があります。

[データパーティショニングによる分散]

データがパーティション分割可能で、アプリケーションのトランザクション処理が 1 パーティションに対する処理におさまる場合、パーティション単位にストリームを分けることで、ストリーム分散可能です。(IM の論理表データを物理表に分散し、メインキーのハッシュ値で分散させるのと同様のイメージです。)

この場合、アプリケーションで指定するストリーム名は、パーティションを識別する英数字の文字列が適当です。

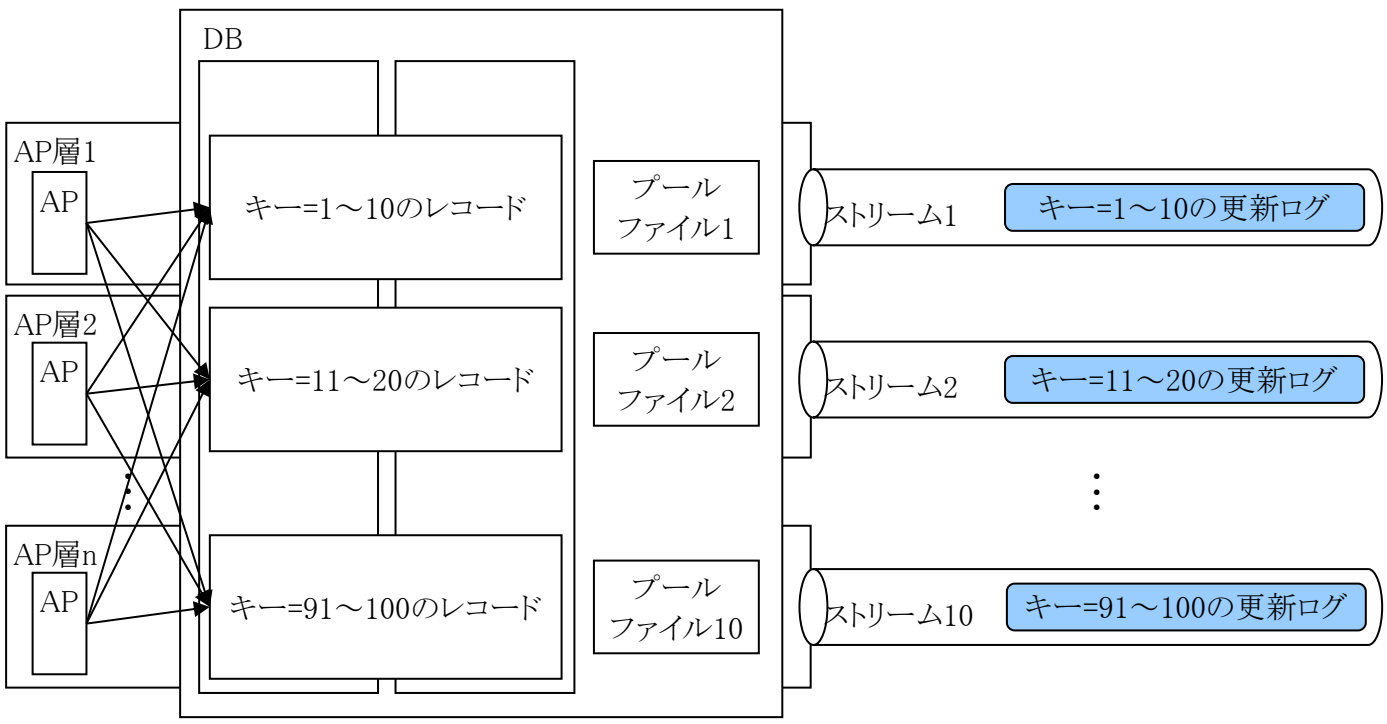


図 4 データパーティショニングによる DB 表更新ログのストリーム分散

［データのカテゴリによる分散］

データをカテゴリに分けることが可能で、アプリケーションのトランザクション処理が 1 カテゴリに対する処理におさまる場合は、カテゴリ単位にストリームを分けることでストリーム分散可能です。

この場合、アプリケーションで指定するストリーム名は、カテゴリを表現する英語名称等が適当です。

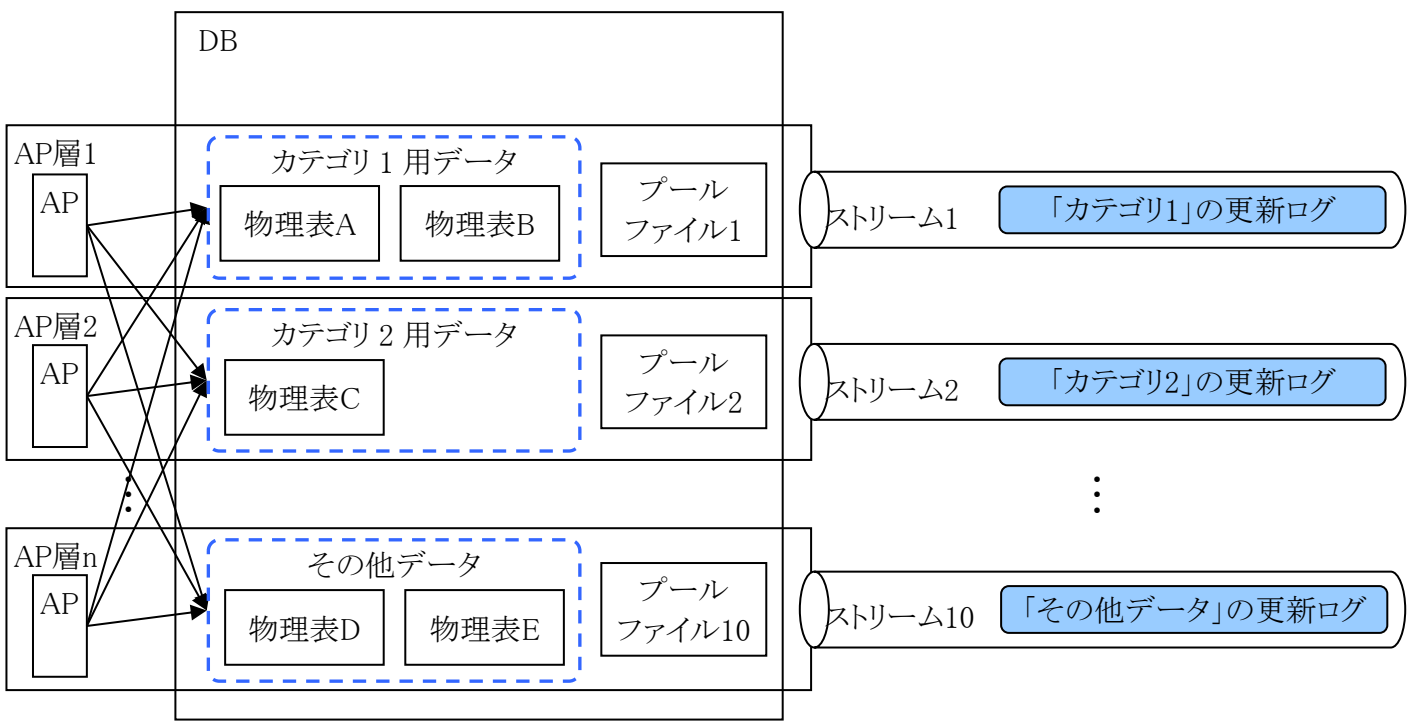


図 5 データのカテゴリによる DB 表更新ログのストリーム分散

［分散しない場合］

ディレード転送機能では、1 つのストリームの転送処理は、1 つの論理ノード上からのみ実行されるため、仮にストリーム分散しなかった場合、DB の更新ログ転送は、ある 1 ノード上でのみおこなわれることになります。

また、更新ログの書き込み処理では、同一ストリームへの書き込み処理は同時におこなえないため、DB 表を更新するアプリケーションはトランザクション終了～コミットまでの処理がシーケンシャルにしか動作できなくなり、分散した場合に比べ、TAT が増大します。

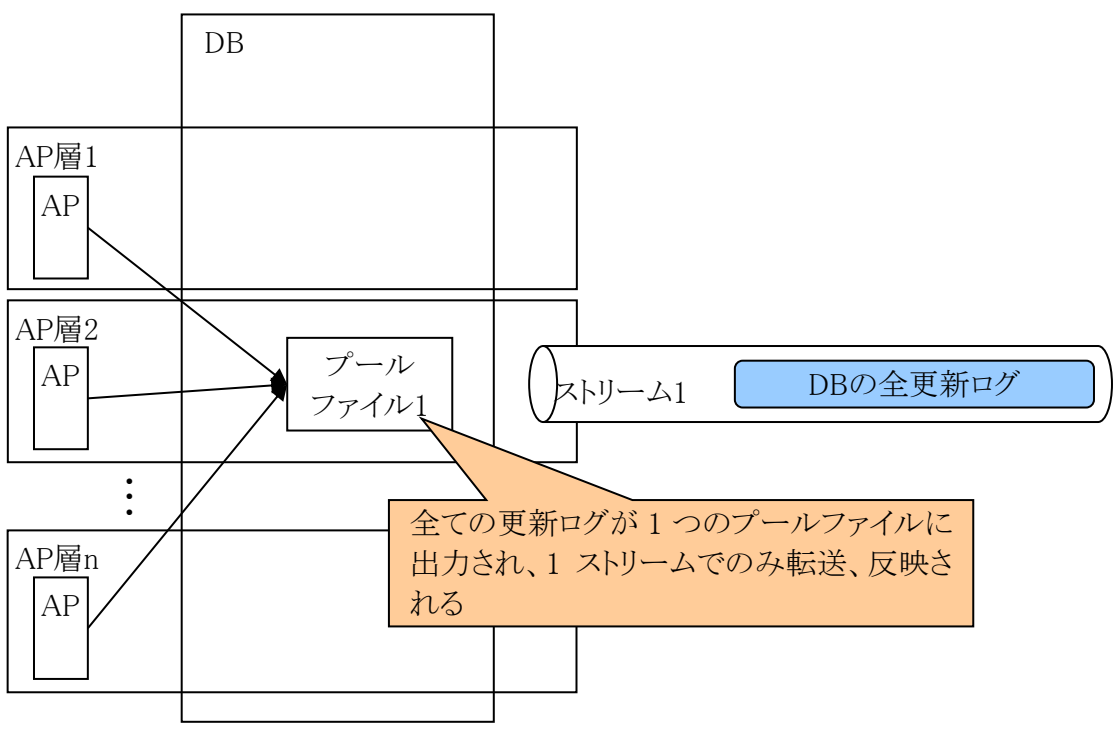


図 6 DB 表更新ログをストリーム分散させない場合

(2)　　動作の抑止

(a)　データ同期制御

データ同期制御をおこなう/おこなわないを、システムごとに変更したい場合や、システム内の特定のストリームのみ、特定のテーブルのみ、特定のデータ更新のみ更新ログの出力を抑止したい場合には以下の対応が必要です。

(i)　システムの更新ログ出力抑止

環境変数の拠点識別情報(DIATC_CENTER_ID)を設定しないと、C のアプリケーションで DB アクセス制御機能を利用したデータ更新をおこなったりしても、更新ログは一切出力されません。

(ii) ストリーム単位の更新ログ出力抑止

システム構築時、抑止対象としたいストリームを、更新ログ出力抑止状態変更コマンド(dadscblock)により出力抑止状態にすることで、拠点識別情報を設定されていても、該当ストリームへは更新ログ出力されなくなります。

(iii)テーブル単位の更新ログ出力抑止

システム構築時、環境定義 DACENV 節の TABLE 項の REPLICATION パラメータでレプリケーションをおこなわない設定にすることで、更新ログを出力するストリームに対しても、該当テーブルの更新ログは出力されなくなります。

(iv) データ更新単位の更新ログ出力抑止

アプリケーションで更新ログ登録要否モード設定関数(diatcdbsetrplmode)を呼び出して更新ログ抑止モードにすることで、更新ログを出力するテーブルに対しても、別のモードに変更するかトランザクション終了まで更新ログは出力されなくなります。

2.2.3　　WebOTX に関する設計

本製品を WebOTX へ適用するために設計する必要があるものはありません。

以下について確認を行い設定してください。

・ログ出力先

2.2.4　　IMに関する設計

(1)　　DBアクセス制御

DB アクセス制御の制御表として、ユーザデータ状態管理 API で登録したメインキーの一覧を保持するユーザデータ状態管理表と呼ばれる IM 表を使用します。ユーザデータ状態管理表の 1 レコードのサイズは 48 バイトで、登録したメインキーの数分のレコードが登録されます。

ユーザデータ状態管理表は更新が発生する全ての MAPID に対して分散させる必要があります。

定義の詳細については、「2.3 環境定義」に記述します。

また、DB アクセス制御機能を通してアクセスする、全てのユーザ IM 表は、レコードの先頭 40 バイトを DB アクセス制御用の制御情報格納領域として空けておく必要があります。

ユーザ IM 表のデータ更新処理や、DIOSA 環境定義(IMTABLECONF 節)／TAM 環境定義(table.conf)に定義するデータオフセットやレコードサイズの情報は、先頭の 40 バイトを考慮してください。

(2)　　データ同期制御

IM/DB 同期制御や、IM 表更新のセンタ間同期制御をおこなう場合、データ同期制御では更新ログ出力抑止用の IM 表を使用します。テーブルのサイズは 1 レコード 16 バイトで、最大で DIOSA/XTP データストアに定義したスーパーストリーム数程度のレコード数を登録します。

IM 表は更新が発生する全ての MAPID に対して分散させる必要があります。

定義の詳細については、「2.3 環境定義」に記述します。

2.2.5　DBに関する設計

(1)　DBアクセス制御

ユーザデータ状態管理 API で登録したメインキーの一覧を保持するユーザデータ状態管理表と呼ばれるDB 表を使用します。ユーザデータ状態管理表の 1 レコードのサイズは 100 バイトで、登録したメインキーの数のレコードが登録されます。

定義の詳細については、「2.3 環境定義」に記述します。

(2)　データ同期制御

DB 表更新のセンタ間同期制御をおこなう場合、データ同期制御では更新ログ出力抑止用の DB 表を使用します。テーブルのサイズは 1 レコード 16 バイト程度で、最大で DIOSA/XTP データストアに定義したスーパーストリーム数程度のレコード数を登録します。

定義の詳細については、「2.3 環境定義」に記述します。

(a)　不正な更新ログによるデータ同期障害復旧の事前準備について

データ同期障害復旧の事前準備として、復旧に使用する SQL ファイルとバインド変数情報ファイルを格納するためのディレクトリを DB ノード上に作成します。また、作成したディレクトリを CREATE DIRECTORY にてディレクトリオブジェクト DIATC_DBDIR として DB 上に作成する必要があります。複数の DB がある場合は、それぞれの DB に対して、ディレクトリを作成し CREATE DIRECTORY を実行する必要があります。

コマンドイメージは以下の通りです。

●ディレクトリオブジェクトを作成します。
CREATE　DIRECTORY　DIATC_DBDIR　AS　‘SQL ファイル格納ディレクトリパス’
※ディレクトリパスは絶対パスで指定してください。

●ディレクトリオブジェクトに権限を与えます。
GRANT　READ　ON　DIRECTORY　DIATC_DBDIR　TO　ユーザ名

2.2.6 IM と DB のマッピングに関する設計

IM/DB 同期制御や TAM ロード機能では、本製品が SQL の自動実行や IM と DB のデータの相互変換を行います。その実行に必要となる環境設計について説明します。

(1) データ型の対応付けについて

環境定義 DACENV 節には IM と DB のマッピングに関する情報として IM レコードの各項目の名前、データ型、サイズを定義します。また、インデックスを構成する項目名を定義します。

本製品で扱うデータ型は以下の 5 種類に分類されます。

・整数

・文字列

・バイナリ

・日付文字列

・可変長項目長

下表は、本製品で扱うデータ型の一覧と、IM レコードの項目の(C 言語における)データ型、DB の項目のデータ型、環境定義 DACENV 節の定義内容をそれぞれ示しています。なお、環境定義 DACENV 節については、環境定義リファレンス「第 II 編 環境定義～2.1 DACENV」を参照してください。

データ型	TAM	Oracle	PostgreSQL	環境定義 DACENV 節	
	C 言語のデータ型			TYPE	SIZE
1byte 整数	char	NUMBER	smallint	NUMBER	1
1byte 整数 (符号なし)	unsigned char	NUMBER	integer	UNSIGNED	1
2byte 整数	short	NUMBER	smallint	NUMBER	2
2byte 整数 (符号なし)	unsigned short	NUMBER	integer	UNSIGNED	2
4byte 整数	int	NUMBER	integer	NUMBER	4
4byte 整数 (符号なし)	unsigned int	NUMBER	bigint	UNSIGNED	4
8byte 整数	long	NUMBER	bigint	NUMBER	8
8byte 整数 (符号なし)	unsigned long	NUMBER	character(8)	UNSIGNED	8
固定長文字列	char[n]	CHAR	character(n)	CHAR	1～2000
可変長文字列	char[n]	VARCHAR2	varchar	STRING	-
		CLOB	-	CLOB	-
可変長バイナリ	char[n]	RAW	-	BINARY	1～32767
		BLOB	-	BLOB	-
		-	bytea	BINARY	1～10485760
日時文字列 (固定長文字列)	char[n]	DATE	timestamp	DATE	1～2000
可変長項目長	char	-	-	VARIABLE_SIZE	1
	short	-	-	VARIABLE_SIZE	2
	int	-	-	VARIABLE_SIZE	4
	long	-	-	VARIABLE_SIZE	8

(2) テーブル名について

DIOSA/XTP インメモリキャッシュではテーブル名(論理表名)に最大 255 文字指定できますが、DB ではテーブル名は最大 30 文字です。そのため、DIOSA/XTP データ変換・通信オプションを利用してデータアクセスを行う場合は、テーブル名は最大 30 文字となります。詳しくは、環境定義リファレンス「第 II 編 環境定義～2.1 DACENV」を参照してください。

(3) 可変長項目について

可変長項目は、レコードごとに長さが変えられる項目で、可変長文字列と可変長バイナリの 2 種類に分かれます。また、Oracle では可変長文字列には STRING と CLOB の 2 つのデータ型があり、可変長バイナリは BINARY と BLOB の 2 つのデータ型があります。どのデータ型を選択するかはアプリケーションが扱うデータの種類や最大長で判断します。

PostgreSQL の可変長文字列は varchar 型に対応しています。可変長バイナリは bytea 型に対応しています。

本製品特有の規則として、可変長項目の位置の前に可変長項目の長さを示す項目が必要です。この項目には、VARIABLE_SIZE と呼ばれる特殊なデータ型を指定します。VARIABLE_SIZE は整数型の項目であり、NUMBER 型と同様に 1, 2, 4, 8 バイトから長さを選択します。VARIABLE_SIZE を省略すると 8 バイト長の項目が設定されます。

IM と DB のマッピングを行うために、アプリケーションは VARIABLE_SIZE 項目に可変長項目の長さを正しく設定しなければなりません。なお、本製品が矛盾を検出するとエラーが発生してメッセージ ID:DADAC201 のメッセージを出力します。

なお、VARIABLE_SIZE の項目は IM レコードには必要ですが、DB には必要ありません。

例として、IM レコード上の可変長項目を Oracle にマッピングするために必要な定義を説明します。ただし、実際にはインメモリ DB アクセスユーティリティ用の制御用項目の定義も必要ですが、下記の説明では簡略化のためにそれらを省略しています。詳しくは 2.3.1 項と 2.3.8 項を参照してください。

TAM のレコードイメージは以下の通りです。

	←レコード先頭				
項目名	ID	DATA1_VARIABLE	DATA1	DATA2_VARIABLE	DATA2
データ	固定長文字列 16 バイト	可変長項目長 4 バイト	可変長文字列 4096 バイト	可変長項目長 4 バイト	可変長バイナリ 8192 バイト

環境定義 DACENV 節は以下のように定義します。

```
$DACENV
(途中省略)
    %TABLE
        NAME = TABLE01,  ID = 1
        %COLUMN NAME = ID              ,TYPE = CHAR           ,SIZE = 16 ;
        %COLUMN NAME = DATA1_VARIABLE  ,TYPE = VARIABLE_SIZE  ,SIZE = 4 ;
        %COLUMN NAME = DATA1           ,TYPE = CLOB ;
        %COLUMN NAME = DATA2_VARIABLE  ,TYPE = VARIABLE_SIZE  ,SIZE = 4 ;
        %COLUMN NAME = DATA2           ,TYPE = BLOB ;
```

```
        %INDEX NAME = I_PK_TABLE01  ,COLUMN = ″ID″ ;
    ;
```

DB 表は以下のように定義します。

```
CREATE TABLE TABLE01 (
    ID                  CHAR(16) PRIMARY KEY,
    DATA1               CLOB,
    DATA2               BLOB
);
```

(4) DATE 型について

C 言語には日時データを扱うデータ型が存在しないため、IM レコードには決められた書式の文字列で日時データを格納します。一方、Oracle では DATE 型、PostgreSQL では timestamp 型に日時データを格納します。

本製品が DATE 型項目にアクセスする時に DB の TO_DATE 関数や TO_CHAR 関数を実行します。これらの関数を実行する際、環境変数 DIATC_DATE_FORMAT または環境定義 DACENV 節に指定された日付書式をパラメータに指定することで IM と DB のデータを相互に変換します。環境変数 DIATC_DATE_FORMAT や環境定義 DACENV 節については環境定義リファレンスを参照してください。また、日付書式は各 DB の仕様に依存しますので各 DB のマニュアルを参照してください。

例として、IM レコードに格納した「2012-01-23 12:34:56」という日時データ(固定長文字列)を Oracle にマッピングするために必要な定義を説明します。

IM レコード中の日時データを以下に示します。

	←レコード先頭																		
文字	2	0	1	2	-	0	1	-	2	3		1	2	:	3	4	:	5	6
コード(16 進)	32	30	31	32	2d	30	31	2d	32	33	20	31	32	3a	33	34	3a	35	36

環境定義 DACENV 節は以下のように定義します。

```
%COLUMN
NAME=COL_DATE,  TYPE=DATE,  SIZE=19
;
```

各パラメータの指定内容を説明します。

- NAME パラメータ
  項目名を指定します。対応する DB の項目と同じ名前を指定してください。
- TYPE パラメータ
  DATE を指定します。
- SIZE パラメータ
  IM レコードにおける項目長をバイト単位で指定します。上記の例では 1 バイト文字 19 文字なので 19 を指定します。

(5)　整数型のサイズについて

環境定義 DACENV 節では IM レコードの項目のサイズをバイト単位で指定するのに対して、DB のテーブル定義では NUMBER 型のサイズを桁数で指定します。そのため、整数型における環境定義 DACENV 節のサイズ指定と DB の型のサイズ指定は下表のような関係があり、両者の値は同じにならない点に注意が必要です。

整数の種類	有効範囲	環境定義 DACENV 節		Oracle	PostgreSQL
		TYPE	SIZE		
1byte 整数	-128～127	NUMBER	1	NUMBER(3)以上	smallint
1byte 整数 (符号なし)	0～255	UNSIGNED	1	NUMBER(3)以上	integer
2byte 整数	-32,768～32,767	NUMBER	2	NUMBER(5)以上	smallint
2byte 整数 (符号なし)	0～65,535	UNSIGNED	2	NUMBER(5)以上	integer
4byte 整数	-2,147,483,648 ～2,147,483,647	NUMBER	4	NUMBER(10)以上	integer
4byte 整数 (符号なし)	0～4,294,967,295	UNSIGNED	4	NUMBER(10)以上	bigint
8byte 整数	-(2 の 63 乗) ～(2 の 63 乗-1)	NUMBER	8	NUMBER(20)以上	bigint
8byte 整数 (符号なし)	0～(2 の 64 乗-1)	UNSIGNED	8	NUMBER(20)以上	character(8)

(6)　文字列型の注意事項について

文字列型を使用した際、データ型の実サイズに対して余った領域の扱いが DB の仕様により異なります。

Oracle では NULL('¥0')でパディングされます。

PostgreSQL ではスペース('¥20')でパディングされます。

TAM では C 言語で指定された領域通りに登録されます。

(7)　　レコード全件削除実行時の DB の更新方法について

IM では一つの論理表のデータが複数の MAP に分散配置されている場合でも、DB では論理表の全てのデータが一つの表に格納されています。

アプリケーションが DB アクセス API のレコード全件関数を実行することで削除されるデータが IM と DB で同じになるように、IM-DB データ同期制御で DB を更新する方法を論理表の分散配置の有無やレコード数に応じて下記の 3 種類から選択します。なお、環境定義 DACENV 節で明示的に指定しない場合は、下記の方法 1 が選択されます。

方法	特徴	SQL 自動生成	MAP 分割 可否	IM-DB データ同期 の処理時間
1	DELETE 文で削除する	あり	可能	レコード数が大量にあると遅い
2	TRUNCATE 文で削除する	あり	不可	レコード数にほぼ依存せず高速
3	SQL 雛形ファイルの利用者指定 SQL 文で削除する	なし	可能	利用者指定 SQL 文に依存

以下に各方法の詳しい説明と定義例を示します。

【方法 1】DELETE 文で削除する

DB アクセス API のレコード全件削除関数で IM から削除したデータを、IM-DB データ同期制御では DB の DELETE 文で削除します。

DELETE 文の性質上、削除対象のレコードが大量にあると処理時間がかかります。

指定したレコード数の削除毎にコミットを実行することも可能です。詳しくは、環境定義リファレンスの環境変数「DIATC_DELETE_ROWNUM」の説明を参照してください。

環境定義 DACENV 節は以下のように定義します。

```
$DACENV
(途中省略)
    %TABLE
        NAME = TABLE01,  ID = 1
        TRUNCATETYPE = DELETE
(以下省略)
```

"DELETE"を指定する

【方法 2】TRUNCATE 文で削除する

DB アクセス API のレコード全件削除関数で IM から削除したデータを、IM-DB データ同期制御では DB の TRUNCATE 文で削除します。

TRUNCATE 文の性質上、削除対象のレコードが大量でも短時間で処理できます。

IM では一つの MAP のデータが全件削除されるのに対して、DB ではすべての MAP のデータが全件削除されるので、この方法を指定する表では MAP 分割できません。

環境定義 DACENV 節は以下のように定義します。

```
$DACENV
(途中省略)
    %TABLE
        NAME = TABLE01,  ID = 1
        TRUNCATETYPE = TRUNCATE
(以下省略)
```

"TRUNCATE"を指定する

【方法 3】SQL 雛形ファイルの利用者定義の SQL 文で削除する

DB アクセス API のレコード全件削除関数で IM から削除したデータを、IM-DB データ同期制御では利用者が SQL 雛形ファイルに定義した SQL 文で削除します。

SQL 雛形ファイルの SQL から利用者定義のストアドプロシージャを呼び出すこともでき、DB 表の更新を柔軟に行うことができます。

SQL 雛形には、レコード全件削除の対象の MAPID をバインドすることができます。環境定義リファレンスの「SQL 雛形」の説明を参照してください。

環境定義 DACENV 節は以下のように定義します。

```
$DACENV
(途中省略)
    %TABLE
        NAME = TABLE01,  ID = 1
        TRUNCATETYPE = TEMPLATE
(途中省略)
    ;
    %SQLTEMPLATE
        FILE =〈SQL 雛形ファイルのパス〉
    ;
(以下省略)
```

"TEMPLATE"を指定する

SQL 雛形ファイルは以下のように定義します。

```
[TABLE]
NAME = TABLE01
TRUNCATE1 = CALL PRC_TRUNCATE('TABLE01', :DIATC_MAPID)
```

環境定義 IMENV 節は以下のように定義します。

```
$IMENV
(途中省略)
    %MAP
        ID = 1
        %HASHRANGE START = 100 END = 199;
    ;
```

```
(途中省略)
    %MAP
        ID = 2
        %HASHRANGE START = 200 END = 299;
        %HASHRANGE START = 400 END = 499;
    ;
(途中省略)
    %MAP
        ID = 3
        %HASHRANGE START = 300 END = 399;
        %HASHRANGE START = 500 END = 599;
        %HASHRANGE START = 700 END = 799;
    ;
(以下省略)
```

DB 表の定義例を示します。

```
CREATE TABLE TABLE01
(
  DIATC_MAINKEY RAW(32),
  DIATC_OPNUM   NUMBER(10),
  DIATC_VERSION NUMBER(10),
  DIATC_HASH    NUMBER(10),
  DATA1         CHAR(16),
(途中省略)
CONSTRAINT I_PK_TABLE01
  PRIMARY KEY (DATA1, DIATC_HASH) USING INDEX LOCAL
  (
      PARTITION TABLE01_0000000001_01,
      PARTITION TABLE01_0000000002_01,
      PARTITION TABLE01_0000000003_01,
      PARTITION TABLE01_0000000002_02,
      PARTITION TABLE01_0000000003_02,
      PARTITION TABLE01_0000000003_03
  )
)
PARTITION BY RANGE(DIATC_HASH)
(
    /* DIATC_HASH 列をキーとするレンジ・パーティション表を作成 */
    /* 各 MAP に対し IMENV-REPGROUP-MAP-HASHRANGE 項の定義数(1～10)だけ作成する */
    PARTITION TABLE01_0000000001_01 VALUES LESS THAN(200),
    PARTITION TABLE01_0000000002_01 VALUES LESS THAN(300),
```

```
    PARTITION TABLE01_0000000003_01 VALUES LESS THAN(400),
    PARTITION TABLE01_0000000002_02 VALUES LESS THAN(500),
    PARTITION TABLE01_0000000003_02 VALUES LESS THAN(600),
    PARTITION TABLE01_0000000003_03 VALUES LESS THAN(800)
)
ENABLE ROW MOVEMENT;
```

SQL 雛形から呼ばれるストアドプロシージャの例を示します。

```
CREATE OR REPLACE PROCEDURE PRC_TRUNCATE (
    TABLE_NAME IN VARCHAR2, /* 削除対象のテーブル名 */
    MAPID      IN NUMBER    /* 削除対象の MAPID */
)
AS
    SQL_STMT       VARCHAR2(1024);      /* 実行する SQL */
    PARTITION_NAME VARCHAR2(32);        /* 削除対象のパーティション名 */
    PARTITION_NUM  PLS_INTEGER := 0;    /* 削除対象 MAP のパーティション数 */
    LOOPCNT        PLS_INTEGER := 1;    /* カウンタ */

BEGIN
    /* 削除対象 MAP のパーティション数を設定 */
    /* パーティション数 = IMENV-REPGROUP-MAP-HASHRANGE 項の定義数(1～10) */
    IF MAPID = 1 THEN
        PARTITION_NUM := 1;
    ELSIF MAPID = 2 THEN
        PARTITION_NUM := 2;
    ELSIF MAPID = 3 THEN
        PARTITION_NUM := 3;
    ELSE
        /* MAPID が不正 */
        RETURN;
    END IF;

    /* 削除対象 MAP の全パーティションを TRUNCATE */
    WHILE LOOPCNT <= PARTITION_NUM LOOP
        /* テーブル名、MAPID、パーティション ID からパーティション名を生成 */
        PARTITION_NAME := TABLE_NAME || '_' || LPAD(MAPID, 10, '0') || '_'
                          || LPAD(LOOPCNT, 2, '0');

        /* パーティションの TRUNCATE を行う SQL を生成 */
        SQL_STMT := 'ALTER TABLE ' || TABLE_NAME || ' TRUNCATE PARTITION '
                    || PARTITION_NAME || ' UPDATE INDEXES';
```

```
        /* 動的 SQL を実行する */
        EXECUTE IMMEDIATE SQL_STMT;

        LOOPCNT := LOOPCNT + 1;
    END LOOP;
END;
/
```

2.3　環境定義

2.3.1　DB アクセス制御

DB アクセス制御機能に関する環境定義について説明します。

(1)　環境定義 DACENV 節

アプリケーションが DB アクセス API を利用してアクセスする表に関する情報を定義します。本機能を利用する場合は必ず定義してください。

DB アクセス API で IM にアクセスしたり、IM-DB データ同期制御で DB にアクセスしたりする関係上、環境定義 DACENV 節で定義するものは DIOSA/XTP　メモリキャッシュの表の定義(IMTABLECONF 節)や DB の表の定義と一致する必要があります。

(2)　SQL 雛形ファイル

SQL 雛形ファイルは、IM-DB データ同期制御で DB の更新を行う時に合わせて利用者が付加的に実行したい SQL を記述するためのファイルです。付加的に実行したい SQL が無い場合は定義する必要はありません。

SQL 雛形ファイルには任意の名前を付けることができます。環境定義 DACENV 節に SQL 雛形ファイルのパスを定義します。

(3)　環境変数 DIATC_DATE_FORMAT

インメモリ DB アクセスユーティリティは、IM レコード上の日付文字列と DB レコード上の DATE　型の項目の相互変換に DB の TO_DATE 関数と TO_CHAR 関数を利用します。本環境変数を定義すると、TO_DATE 関数と TO_CHAR 関数の第 2 パラメータに本環境変数の値を指定します。また、環境定義 DACENV 節でテーブル毎に日付書式を指定することもできます。

本環境変数が省略されている、かつ環境定義 DACENV 節-TABLE 項の DATEFORMAT パラメータが省略されている場合は、DB の仕様によって日付書式が決定されます。

(4)　環境変数 DIATC_DATE_NLSPARAM

インメモリ DB アクセスユーティリティは、IM レコード上の日付文字列と DB レコード上の DATE　型の項目の相互変換に DB の TO_DATE 関数と TO_CHAR 関数を利用します。本環境変数を定義すると、TO_DATE 関数と TO_CHAR 関数の第 3 パラメータに本環境変数の値を指定します。また、環境定義 DACENV 節でテーブル毎に NLS パラメータを指定することもできます。

本環境変数が省略されている、かつ環境定義 DACENV 節-TABLE 項の DATENLSPARAM パラメータが省略されている場合は、DB の仕様によって NLS パラメータが決定されます。

(5)　環境変数 DIATC_RECORDCHK_ENABLE

DB アクセス API で指定されたレコードサイズと、環境定義 DACENV 節に定義した表の構造から求めたレコードサイズを比較して矛盾を検出する機能の有効／無効を指定します。本環境変数を定義しない場合、本機能は有効になります。

(6)　環境変数 DIATC_SQLLOG_NORECORD_CHECK

Oracle 更新ログを処理するスーパーストリームで、更新ログ(UPDATE 文)が対象とするレコードが存在しない場合に正常とするか、エラーとするかを指定します。本環境変数を省略した場合は正常とします。

2.3.2　　データ同期制御

(1)　　IM-DB データ同期制御、センタ間データ同期制御共通

(a)　拠点識別情報の設定

データ同期制御の更新ログ出力機能は、環境変数の拠点識別情報(DIATC_CENTER_ID)を設定することで利用可能となります。

拠点識別情報には 1 または 2 を指定します。フロント-バックアップ構成のシステムでは、それぞれの拠点に 1 か 2 を割り当てて設定する必要があります。

フロントのみの構成の場合は、固定で 1 を指定してください。

環境変数未設定の場合、更新ログは収集されません。

(b)　MAPID 内ストリーム分割用設定

同一 MAPID の更新ログを、さらに複数のスーパーストリームに分割したい場合、下記のいずれかの環境変数を設定してください。なお、MAPID 内ストリーム分割をおこなう場合、MAPID の定義値上限が、9999999 となりますので注意してください。

(i)　ストリーム分割内通番決定利用者出口(DIATC_STREAM_PARTEXIT)

MAPID に対して、割り当てられた複数のストリームの何番目のストリームに更新ログを出力するかを決定するための出口関数名

定義した場合、出口関数も作成する必要があります。

(ii) MAP 内ストリーム分割数(DIATC_STREAM_PARTNUM)

MAPID をいくつのストリームに分割するかを指定します。

分割内の通番は、分割数とハッシュ値から計算して決定します。MAPID 内がハッシュ値分散されている場合は、利用者出口を定義するより簡単に分割することができます。

2.3.3　　オンライン中メンテナンス

オンライン中メンテナンスは 3 つの機能を持ち、それぞれ以下の環境変数を定義します。

オンライン中 DB リカバリ支援機能

　　オンライン中 DB リカバリ支援機能に環境変数はありません。

TAM 再配置機能

- DIATC_MULTIPLE_NUMBER_HASH　　1 プロセスで並列に行う処理の多重度を指定します。

セーブロード機能

- DIATC_MULTIPLE_NUMBER_SLU　　1 プロセスで並列に行う処理の多重度を指定します。

また、セーブロード機能では DB のない環境でのデータベース構成変更に対応するため、構成変更情報取得出口をファイル入力 TAM ロードコマンドのオプションに指定します。

(1)　　TAM 再配置機能

(a)　環境変数 DIATC_MULTIPLE_NUMBER_HASH

ハッシュ値更新コマンドが 1 プロセスで並列に行う処理の多重度を指定します。ハッシュ値更新コマンドは処理対象となる論理表ごとにスレッドを起動し、並列に処理を行います。処理対象となる論理表数が本環境変数より多い場合、ハッシュ値更新コマンドから複数の子プロセスを起動し、子プロセスにそれぞれスレッドを割り当てて処理を行います。省略時は、1 プロセスあたり 48 の多重度で処理します。

環境により設定された 1 プロセスあたりのスレッド数の上限を超えない範囲で論理表数より大きくなるよう設定します。論理表数がスレッド数の上限を超える場合は、1 プロセスあたりのスレッド数の上限値を設定します。

(2)　　セーブロード機能

(a)　環境変数 DIATC_MULTIPLE_NUMBER_SLU

セーブロード機能の各コマンドが 1 プロセスで並列に行う処理の多重度を指定します。各コマンドは処理対象となる論理表ごとにスレッドを起動し、並列に処理を行います。処理対象となる論理表数が本環境変数より多い場合、各コマンドから複数の子プロセスを起動し、子プロセスにそれぞれスレッドを割り当てて処理を行います。省略時は、1 プロセスあたり 48 の多重度で処理します。

対象コマンド：TAM セーブコマンド
　　TAM ロードコマンド
　　ファイル出力 TAM セーブコマンド
　　ファイル入力 TAM ロードコマンド

環境により設定された 1 プロセスあたりのスレッド数の上限を超えない範囲で論理表数より大きくなるよう設定します。論理表数がスレッド数の上限を超える場合は、1 プロセスあたりのスレッド数の上限値を設定します。

(b)　利用者出口構成変更情報取得出口

セーブロード機能のファイル入力 TAM ロードコマンドを利用してデータベース構成変更を行う場合に、ハッシュ値計算やロードするユーザデータの修正を行う利用者出口を呼び出します。詳細は「データ変換・

通信オプション 利用の手引」の「セーブロードプログラミング」を参照してください。

2.3.4　その他 DIOSA/XTP 関連定義

(1)　機能ごとの必須定義

データ変換・通信オプションの各機能を利用するために必要な DIOSA/XTP の環境定義、環境変数は以下の通りです。

(a)　環境定義

	DBアクセス制御	オンライン中メンテナンス（TAM再配置）	データ同期制御	オンライン中メンテナンス（セーブロード、オンライン中DBリカバリ支援）
APLIB		○	○	○
CMDSEND		○	○	○
COCENV				○
DBCTRL		○	○(※1)	○
DELAYED		○	○	○
DIOSAMAP	○	○	○	○
IMENV	○(※1)	○	○(※1)	○
IMTABLECONF	○(※1)	○	○(※1)	○
MMG	○	○(※2)	○	○(※2)
SYSMAP		○	○	○
TPATHENV		○		

※1 IM/DB の使用有無により必須

※2 dasluftamload の出口関数でメモリ管理機能を利用しない場合は不要

(b)　環境変数

	DBアクセス制御	オンライン中メンテナンス（TAM再配置）	データ同期制御	オンライン中メンテナンス（セーブロード、オンライン中DBリカバリ支援）
DIOSA_APP 系				
DIOSA_DBG 系	○	○	○	○
DIOSA_IRMROOT	○	○	○	○
DIOSA_LNODENAME	○	○	○	○
DIOSA_MSG 系	○	○	○	○
DIOSA_ROOT	○	○	○	○
DIOSA_TMP	○	○	○	○

※1 ジャーナル出力時必須

(2) DIOSA/XTP 基盤機能

(a) 基本機能

以下の環境定義は、DIOSA/XTP が動作するために必要となるものです。

DIOSA/XTP 導入の手引、DIOSA/XTP 環境定義リファレンスを参照し、環境の設定を行ってください。

環境定義

DIOSAMAP	論理システム構成定義
SYSMAP	論理システム間通信構成定義

環境変数

DIOSA_ROOT	DIOSA/XTP インストールパス
DIOSA_IRMROOT	DIOSA/XTP 環境定義ファイル格納パス
DIOSA_LNODENAME	DIOSA/XTP 論理ノード名
DIOSA_TMP	DIOSA/XTP 作業用ディレクトリ
DIOSA_MSG 系変数	メッセージ管理機能定義
DIOSA_DBG 系変数	トレース機能定義

(b) AP 動的置換機能

AP 動的置換機能の環境定義作成、または環境変数設定をおこなう必要があります。

LM 名	ライブラリ	関数名	備考
daslufltamload	ユーザ作成	定義関数名	ファイル入力 TAM ロードコマンドで、構成変更情報取得出口を使用してデータベース構成変更を行う場合
TAM の更新ログ出力プロセス	ユーザ作成	定義関数名	同一 MAPID 内複数ストリームの分割内通番を出口で決定する場合
dicoctpp	ユーザ作成	GetDateExit	ジャーナルに設定する時刻を出口で決定する場合

(c)　ロック制御機能

各機能が制御のために使用するため、以下の範囲のロック識別子は使用できません。

(i)　ファイルロック(範囲：1～1024)

ファイルロック識別子 641～1024 は各機能が使用するため予約されています。

機能名	ファイルロック使用数	ファイルロック割当	
		開始	終了
TAM 再配置機能	1	641	641
セーブロード機能	2	642	643
DB アクセス制御機能	3	644	646
予備	110	915	1024

(ii) DB ロック(範囲：1～4096)

DB ロック識別子 4065～4096 は各機能が使用するため予約されています。

機能名	DB ロック使用数	DB ロック割当	
		開始	終了
DB アクセス制御機能	1	4065	4065
予備	31	4066	4096

(d)　メッセージ出力機能

各機能は、DIOSA/XTP のメッセージ出力機能を利用して、メッセージリファレンスのメッセージを出力します。そのため、各機能のメッセージを出力するために、ユーザ用原型メッセージファイルに関する環境変数の定義やメンテナンス作業を行う必要があります。

なお、DIOSA/XTP のメッセージ出力機能を利用してユーザ作成のメッセージを出力するか否かにより、ユーザ用原型メッセージファイルに関する環境変数の定義やメンテナンス作業が変わります。

(i)　原型メッセージファイル名の定義

原型メッセージファイル名を DIOSA/XTP の環境変数「DIOSA_MSG_FORM_FILEPATH(ユーザ用原型メッセージファイル名)」に定義します。

DIOSA/XTP のメッセージ出力機能を利用してユーザ作成のメッセージを出力する場合は、ユーザ任意の原型メッセージファイルパス名を定義してください。

```
setenv　DIOSA_MSG_FORM_FILEPATH　ユーザ任意のユーザ原型メッセージファイルのパス
```

DIOSA/XTP のメッセージ出力機能を利用してユーザ作成のメッセージを出力しない場合は、省略可能です。

(ii) 原型メッセージファイルの作成/更新(メンテナンス作業)

DIOSA/XTP のメッセージ出力機能を利用してユーザ作成のメッセージを出力する場合、ユーザ作成のユーザ原型メッセージファイルを作成する必要があります。

DIOSA 起動前に以下の作業を実施してください。なお、運用中に原型メッセージファイルを更新する場合も同作業を行ってください。

・メッセージ一覧を作成します。

・dimsgmtn(原型メッセージファイルメンテナンスコマンド)を利用して、作成したメッセージ一覧からユーザ用原型メッセージファイルを作成します。

```
dimsgmtn -i メッセージ一覧ファイル
```

(3) インメモリキャッシュ

(a) DB アクセス制御 ユーザデータ状態管理表(DIATC_DAC_MAINKEY)

DB アクセス制御を利用するためには、DIOSA/XTP のインメモリキャッシュ機能に、DB アクセス制御の制御用のテーブル定義をおこなう必要があります。

```
%LTABLE
    TYPE        = 0
    NAME        = DIATC_DAC_MAINKEY
    ID          = 任意の ID
    %RECORDCONF
        TYPE  = FIXED
        %PRIMARYKEY
            NAME = DIATC_DAC_MAINKEY_IDX
            %KEYDEF
                OFFSET = 0
                LENGTH = 32
            ;
        ;
    ;

    %PTABLE                                          MAPID 数分繰り返す
        NAME    = 任意の IM 表名
        MAPID   = スーパーストリームに割り当てた MAPID
    ;
;
```

(b) データ同期制御 更新ログ出力抑止用テーブル(DIATC_DSC_OUTPUTCTRL)

IM-DB データ同期制御や、IM 表を同期元としたセンタ間データ同期制御を利用するためには、DIOSA/XTP のインメモリキャッシュ機能に、更新ログ出力抑止用のテーブル定義をおこなう必要があります。

また、データ同期制御をおこなうためには、DIOSA/XTP データストア用の定義も必要なため、DIOSA/XTP データストア 利用の手引きを参照してテーブル定義をおこなってください。

```
%LTABLE
    TYPE        = 0
    NAME        = DIATC_DSC_OUTPUTCTRL
    ID          = 任意の ID
    %RECORDCONF
        TYPE  = FIXED
        %PRIMARYKEY
            NAME = DIATC_DSC_OUTPUTCTRL_IDX
            %KEYDEF
                OFFSET = 0
                LENGTH = 16
            ;
        ;
    ;

    %PTABLE                                  データストアに定義した MAPID 数分繰り返す
        NAME    = 任意の IM 表名
        MAPID   = スーパーストリームに割り当てた MAPID
    ;
;
```

(c)　利用者が使用する論理表

TAM 再配置機能、セーブロード機能が提供する以下のコマンドを利用する場合、利用者が使用する論理表をメモリキャッシュ機能の環境定義 IMTABLECONF 節 LTABLE 項に登録する際に、表種別を 1～9 の値に設定する必要があります。また、ここで表種別を 1～9 に設定した論理表は、DB アクセス制御機能の環境定義 DACENV 節に論理表の定義を行う必要があります。

TAM 再配置機能

- ハッシュ更新コマンド

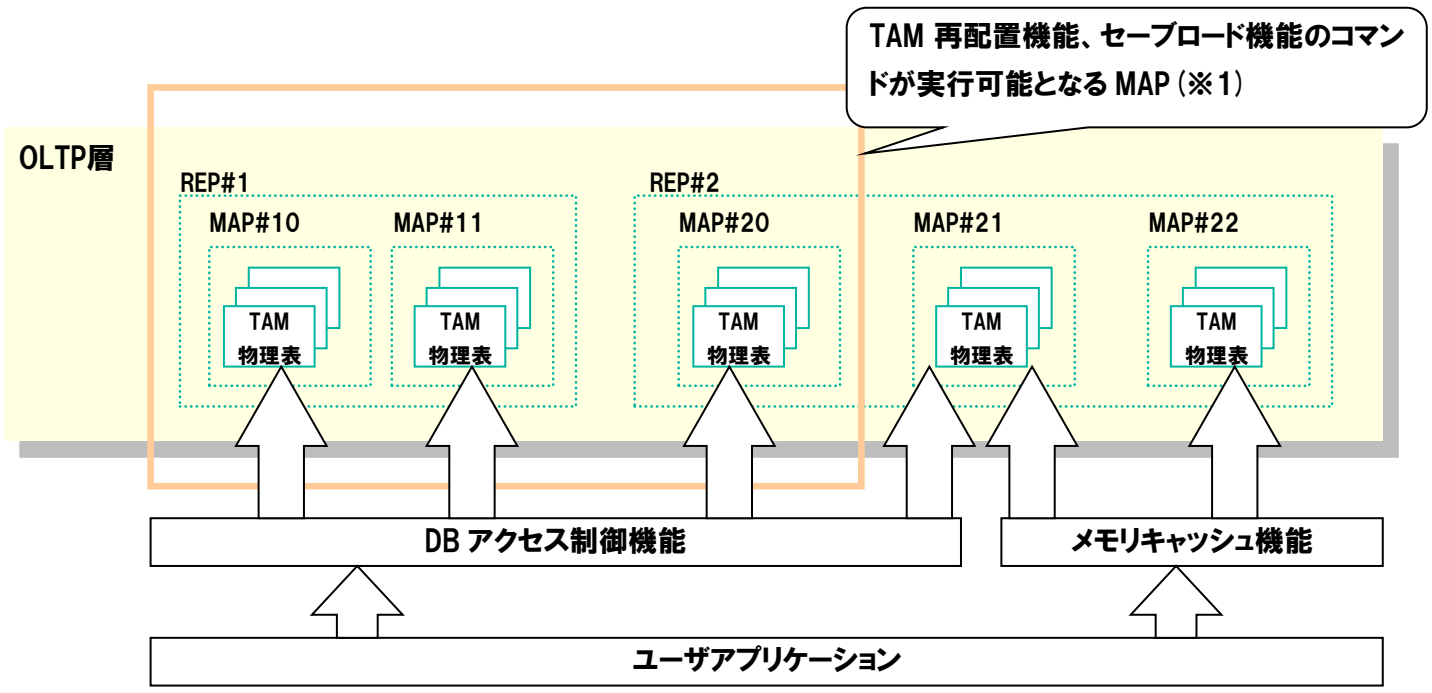
セーブロード機能

- TAM セーブコマンド
- TAM ロードコマンド
- ファイル出力 TAM セーブコマンド
- ファイル入力 TAM ロードコマンド

論理システム内に DB アクセス制御機能を使用せず、メモリキャッシュ機能を使用してデータアクセスする論理表が存在する場合、以下のような制限があります。

- メモリキャッシュ機能を使用してデータアクセスする論理表が存在する MAP に対しては、上記で挙げられた TAM 再配置機能、セーブロード機能の各コマンドは利用できない(下図では、MAP#21、#22 が該当します)。
- メモリキャッシュ機能を使用してデータアクセスする論理表が存在するレプリケーショングループに対しては、TAM 再配置手順の容量変更手順が実施できない(下図では REP#2 に属する MAP#20、#21、#22 が該当します)。

TAM 再配置手順については、「第 3 章 システムの運用」を参照してください。



(※1) MAP#20 については、TAM 再配置手順　容量変更の実施不可

(4)　　データストア

IM-DB データ同期制御、センタ間データ同期制御を利用するためには、DIOSA/XTP のデータストアの環境定義をおこなう必要があります。

(a)　IM-DB データ同期

IM-DB データ同期をおこなう場合、同期をおこないたいスーパーストリームに対してログリーダユニットを定義します。

ユーザアプリケーションによる IM 更新を DB とデータ同期したい場合

定義項目		転送元
プールファイル種別		WRITER
ログデータ格納先		IM
ログリーダユニット		
	ユーザデータ更新先	DB
	プロセス初期化出口	UpdPrcInitExit
	プロセス終了出口	UpdPrcTermExit
	トランザクション初期化出口	UpdTrnInitExit
	トランザクション終了出口	UpdTrnTermExit
	メイン AP	LogUpdOraExit

(b)　センタ間データ同期

センタ間データ同期をおこなう場合、転送元、転送先のシステムにスーパーストリーム定義が必要です。代表的な処理パターンについて、定義例を示します。

IM の更新を、別システムの IM とデータ同期する場合

定義項目		転送元	転送先
プールファイル種別		WRITER	RECEIVER
ログデータ格納先		IM	IM
データ転送用ユニット		センダ	レシーバ
ログリーダユニット		定義不要	
	ユーザデータ更新先	-	IM
	プロセス初期化出口	-	UpdPrcInitExit
	プロセス終了出口	-	UpdPrcTermExit
	トランザクション初期化出口	-	UpdTrnInitExit
	トランザクション終了出口	-	UpdTrnTermExit
	メイン AP	-	LogUpdTamExit

IM の更新を、別システムの DB とデータ同期する場合

定義項目	転送元	転送先
プールファイル種別	WRITER	RECEIVER
ログデータ格納先	IM	IM/DB(※)
データ転送用ユニット	センダ	レシーバ
ログリーダユニット	定義不要	
ユーザデータ更新先	-	DB
プロセス初期化出口	-	UpdPrcInitExit
プロセス終了出口	-	UpdPrcTermExit
トランザクション初期化出口	-	UpdTrnInitExit
トランザクション終了出口	-	UpdTrnTermExit
メイン AP	-	LogUpdOraExit

※　転送先のログデータ格納先は、IM、DB どちらでも可

DB の更新を、別システムの DB とデータ同期する場合

定義項目	転送元	転送先
プールファイル種別	WRITER	RECEIVER
ログデータ格納先	DB	DB
データ転送用ユニット	センダ	レシーバ
ログリーダユニット	定義不要	
ユーザデータ更新先	-	DB
プロセス初期化出口	-	定義不可(※)
プロセス終了出口	-	定義不可(※)
トランザクション初期化出口	-	定義不可(※)
トランザクション終了出口	-	定義不可(※)
メイン AP	-	LogUpdOraExit

※　スーパーストリームが動作するノードで dadacinit を実行する場合は、他の場合と同様の出口関数を定義しても問題ありません。

(c)　スーパーストリーム名について

データ同期制御の更新ログ出力先スーパーストリーム名は、命名規則によって決定します。データストアには、環境に合わせた名前のスーパーストリーム名で定義をおこなう必要があります。

なお、MAPID 内でスーパーストリームを分割する場合、各スーパーストリームに異なるプールファイル ID を割り当てる必要があります。

更新ログの種類	スーパーストリーム名
IM 表	TAM{MAPID(0 埋め 10 桁)_{拠点識別情報}
IM 表(MAPID 内分割時)	TAM{MAPID(0 埋め 7 桁)_{分割内通番(0 埋め 2 桁)}_{拠点識別情報}
DB 表	ORA_{指定ストリーム名}_{拠点識別情報}

(d)　その他

- IM-DB データ同期とセンタ間データ同期を両方おこないたい場合、同一のスーパーストリームに対してログリーダユニットとセンダユニットを両方記述します。
- ログリーダユニットの経過時間監視や、CPU 時間監視を利用すると、更新ログの反映処理に想定以上の時間がかかっている場合に、メッセージを出力したり、処理を中断したりすることが可能です。設定時

間の目安としては、最低でも 1 トランザクションの更新ログ反映に要する時間は必要なため、最も更新量の多いトランザクションの反映処理時間に対して、ある程度の余裕を持たせた時間を設定してください。

2.3.5　WebOTX の環境定義

(1)　センタ間データ同期制御

WebOTX 上で動作するアプリケーションで、更新ログ出力をおこなう場合、データ同期制御の環境設定、Java 環境設定に加え、以下の設定が必要です。

DIOSA/XTP データストア用の WebOTX 環境設定については、DIOSA/XTP データストア 利用の手引きを参照してください。

(a)　セキュリティポリシーファイル(server.policy)の更新

対象ドメイン配下の config/server.policy ファイルに、以下の記述を追加してください。

```
grant {
    permission java.lang.RuntimePermission "getenv.DIATC_CENTER_ID";
};
```

(b)　データベース接続用設定

更新ログ出力には、業務アプリケーションと同一のデータソース(DB 接続コネクション)を使用します。

データソースは、applicationContext.xml ファイル(contextConfigLocation に定義したファイルでも可)に、bean 名に「dataSource」を指定して定義してください。

/WEB-INF/applicationContext.xml

```
<bean id="dataSource" class="org.springframework.jndi.JndiObjectFactoryBean">
  <property name="jndiName">
    <value>${jndi.name}</value>
  </property>
</bean>
```

(c)　メッセージ出力用設定

(i)　log4j を使用する場合

対象ドメイン/config ディレクトリ配下に、commons-logging.properties ファイル、および log4j.xml ファイルを格納します。

(ii) simplelog を使用する場合

対象ドメイン/config ディレクトリ配下に、simplelog.properties ファイルを格納します。

2.3.6 Java 環境設定

(1) センタ間データ同期制御

Java アプリケーションで、更新ログ出力をおこなう場合、データ同期制御の環境設定に加えて、以下の設定が必要です。

DIOSA/XTP データストア用の Java 環境設定については、DIOSA/XTP データストア 利用の手引きを参照してください。

(a) クラスパスの設定

以下のファイルをクラスパスに追加してください。バージョン部分はインストールされているバージョンに合わせて適宜書き換えてください。

diatc.jar

commons-logging-1.1.1.jar

org.springframework.core-3.0.5.RELEASE.jar

org.springframework.beans-3.0.5.RELEASE.jar

org.springframework.context-3.0.5.RELEASE.jar

org.springframework.web-3.0.5.RELEASE.jar

(b) log4j を使用する場合の設定

commons-logging.properties ファイル、log4j.xml ファイルを記述します。

(i) commons-logging.properties ファイル

```
org.apache.commons.logging.Log=org.apache.commons.logging.impl.Log4JLogger
```

(ii) log4j.xml ファイル

下記設定例の赤字部分を設定します。

ログファイルの絶対パスには任意のディレクトリを指定可能です。

ログ出力レベルは、出力レベルの高い順に、fatal/error/warn/info/debug のいずれかを指定します。

下記では info を指定しているので、debug 以外の全てのログが出力されます。

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">
<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/" >
  <appender name="ApplicationLog" class="org.apache.log4j.FileAppender">
    <param name="File" value="(ログファイルの絶対パス)"/>
    <param name="Append" value="true"/>
    <layout class="org.apache.log4j.PatternLayout">
      <param name="ConversionPattern" value="%d %-5p %-52c [%-12t] %m (%F:%L)%n"/>
    </layout>
  </appender>
  <logger name="com.nec.jp.diatc" >
    <level value ="info" />
    <appender-ref ref="ApplicationLog" />
  </logger>
```

```
</log4j:configuration>
```

※log4j の詳細は、Apache log4j のホームページをご参照ください。

(c) simplelog を使用する場合の設定

simplelog.properties ファイルを記述します。

(i) simplelog.properties ファイル

下記設定例の赤字部分にログ出力レベルを設定します。

出力レベルの高い順に、fatal/error/warn/info/debug/trace のいずれかを指定してください。

下記では info を指定しているので、trace/debug 以外の全てのログが出力されます。

```
org.apache.commons.logging.simplelog.defaultlog=info
```

※simpleog の詳細は、Apache commons logging のホームページをご参照ください。

2.3.7　　TAM の環境定義

DIOSA/XTP インメモリキャッシュの環境定義(IMTABLECONF 節)に記述した IM 表定義は、TAM のテーブル定義(table.conf)にも記述する必要があります。

見出しは論理表名で記述しますが、各論理表名配下の PTABLE 節に記述したテーブル名が、実際に定義する必要がある IM 表名で、1 つの論理表に対して複数の PTABLE 項を記述した場合、PTABLE 項の定義数分の IM 表定義が必要です。太字の項目については、定義例に書かれている通りに記述してください。

(1)　　DB アクセス制御 ユーザデータ状態管理表(DIATC_DAC_MAINKEY)

DB アクセス制御を利用するためには、DB アクセス制御の制御用テーブルの定義を行う必要があります。

```
[table]
    tam_table_name          = メモリテーブル名
    tam_table_file          = TAM ファイル名
    backup                  = yes
    share                   = yes
    auto_extend             = yes
    auto_extend_size        = 1
    max_auto_extend_size    = 5
    record_size             = 48
    init_record_num         = 10
    index_number            = 1
    index_name1             = DIATC_DAC_MAINKEY_IDX
    keys1                   = 0:32
    index_entry_num1        = 10
    auto_extend_key_num1    = 20
    max_auto_extend_count1  = 0
    init_key_num1           = 10
    auto_release_index1     = yes
```

(2)　　データ同期制御 更新ログ出力抑止用テーブル(DIATC_DSC_OUTPUTCTRL)

IM-DB データ同期制御や、IM 表を同期元としたセンタ間データ同期制御を利用するためには、更新ログ出力抑止用のテーブル定義をおこなう必要があります。

また、データ同期制御をおこなうためには、DIOSA/XTP データストア用の定義も必要なため、DIOSA/XTP データストア 利用の手引きを参照してテーブル定義をおこなってください。

```
[table]
    tam_table_name          = メモリテーブル名
    tam_table_file          = TAM ファイル名
    backup                  = yes
    share                   = yes
    auto_extend             = yes
    auto_extend_size        = 1
    max_auto_extend_size    = 5
    record_size             = 16
    init_record_num         = 10
    index_number            = 1
    index_name1             = DIATC_DSC_OUTPUTCTRL_IDX
    keys1                   = 0:16
    index_entry_num1        = 10
    auto_extend_key_num1    = 20
    max_auto_extend_count1  = 0
    init_key_num1           = 10
    auto_release_index1     = yes
```

2.3.8 DBの環境定義

(1) DBアクセス制御

DB表を作成するため、サンプルとして提供しているSQLファイルを実行してください。

Oracleの場合：{プロダクトインストールディレクトリ}/sample/sql/create_table_dac.sql

PostgreSQLの場合：{プロダクトインストールディレクトリ}/sample/sql_pg/create_table_dac.sql

また、環境定義DACENV節に定義したのと同名の表に対して下記の項目を含めなければなりません。

項目	項目名	型(Oracle)	型(PostgreSQL)
メインキー	DIATC_MAINKEY	RAW(32)	bytea
ハッシュ値	DIATC_HASH	NUMBER(10)	integer
運用番号	DIATC_OPNUM	NUMBER(10)	integer
レコード版数	DIATC_VERSION	NUMBER(10)	integer

(2) データ同期制御

DB表を作成するため、サンプルとして提供しているSQLファイルを実行してください。

Oracleの場合：{プロダクトインストールディレクトリ}/sample/sql/create_table_dsc.sql

PostgreSQLの場合：{プロダクトインストールディレクトリ}/sample/sql/create_table_dsc.sql

第3章　システムの運用

3.1　定義生成

3.1.1　DB アクセス制御

作成した環境定義は、DIOSA の起動前にオブジェクトの生成を行う必要があります。

生成したオブジェクトファイルの内容に従い動作します。

(a)　環境定義オブジェクトファイル作成

はじめて環境定義オブジェクトを生成するときに以下のコマンドを実行します。

```
diirmadd -E 環境定義ファイル名 ...
```

(b)　環境定義オブジェクトファイル更新

すでに環境定義オブジェクトファイルを生成しているときに以下のコマンドで更新します。

```
diirmrep -E 環境定義ファイル名 ...
```

3.1.2　TPBASE 環境定義

作成した環境定義は、変換操作は必要ありません。

環境定義の詳細については、TPBASE のマニュアルを参照してください。

3.1.3　TAM 環境定義

作成した環境定義は、tamcfgmaint コマンドや tamcreate を使って変換や反映する操作が必要となります。

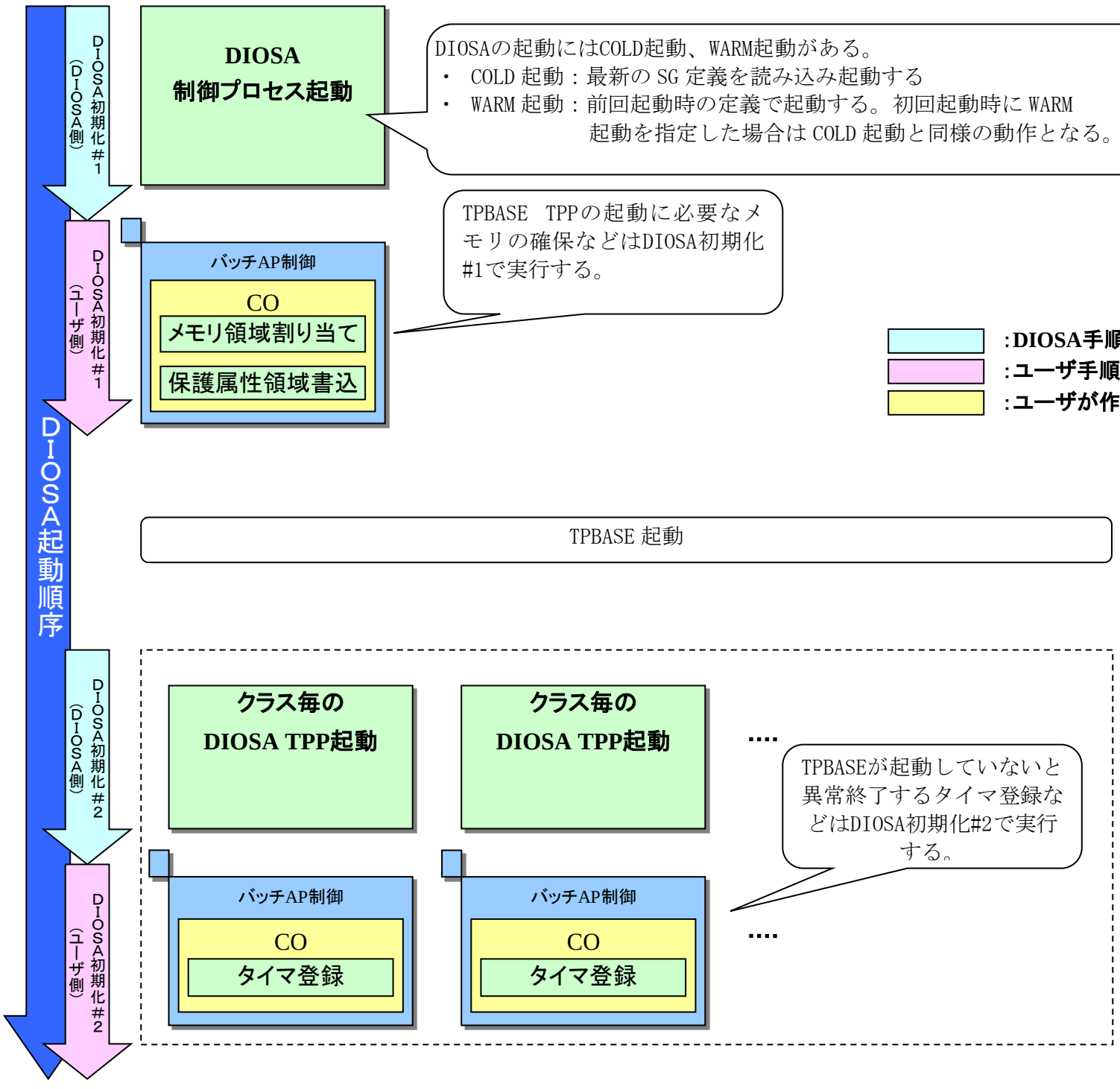
環境定義の詳細については、TAM のマニュアルを参照してください。

3.2　起動・停止

データ変換・通信オプションの起動・停止処理を、DIOSA/XTP の起動・停止シーケンス内に組み込みます。

3.2.1　DIOSA/XTP の起動シーケンス

DIOSA/XTP の起動シーケンスは、以下のように 2 フェーズあります。データ変換・通信オプションは、各フェーズの DIOSA 初期化(ユーザ側)のタイミングで、起動に必要となる処理を行います。



3.2.2　　起動コマンド一覧

以降の表にノード毎の起動コマンド一覧を記します。

(1)　　APノード

コマンド名	オプション	実行有無	概要
dadacinit	dadacinit [-c|-w]	選択	DBアクセス制御の起動を行います。
daolminit	daolminit	選択	オンライン中メンテナンスの起動を行います。

※各コマンドの詳細は、コマンドリファレンスを参照してください。

(2)　　OLTPノード

コマンド名	オプション	実行有無	概要
dadacinit	dadacinit [-c|-w]	選択	DBアクセス制御の起動を行います。
daolminit	daolminit	選択	オンライン中メンテナンスの起動を行います。
datrlrefopdata	datrlrefopdata	選択	運用情報を照会します。この照会結果から自ノードがマスタである MAPID を取得し、下記ロードコマンドを実行します。
daslutamload	daslutamload -m MAPID [-t TIMEOUT] [-n COMMITNUMBER] [-r READSTRECNUM] [-k]	選択	上記で取得したMAPID(自ノードがマスタであるMAPID)に対して、DBのデータをIMへロードします。

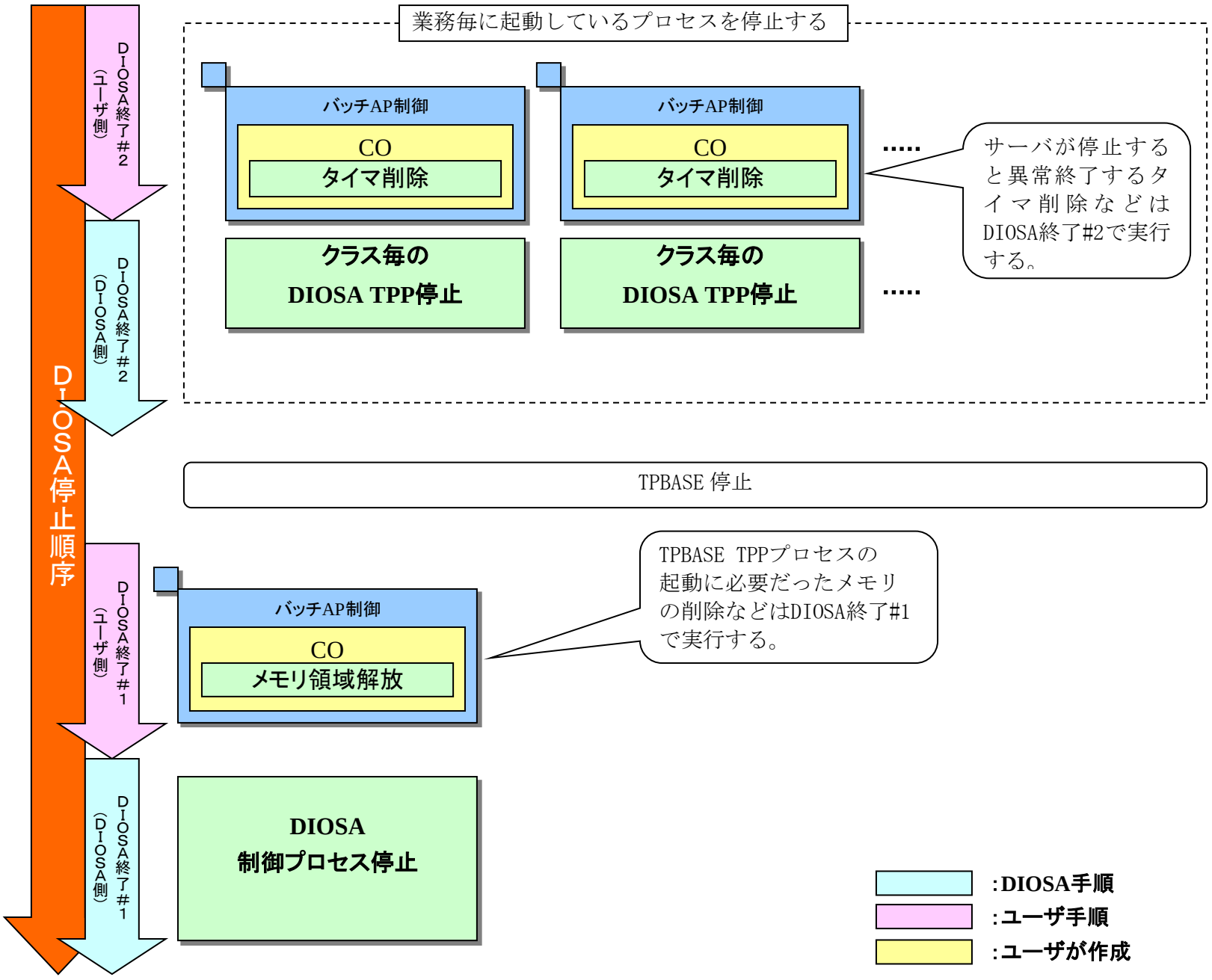
※各コマンドの詳細は、コマンドリファレンスを参照してください。

(3)　　システム構築時のみ実行

コマンド名	オプション	実行有無	概要
dadscblock	dadscblock -b {-o | -t MAPID} -s SuperStreamName	選択	指定されたストリームの更新ログ出力を抑止します。 システム構築時に実行すれば、DIOSA の起動/停止の度に実行する必要はありません。 -o オプションは DB アクセス可能なノード、-t オプションは IM アクセス可能なノードで実行してください。

3.2.3　　DIOSA/XTP の停止シーケンス

DIOSA/XTP の停止シーケンスは、以下のように 2 フェーズあります。データ変換・通信オプションは、各フェーズの DIOSA 終了(ユーザ側)のタイミングで、停止に必要となる処理を行います。



3.2.4　　停止コマンド一覧

以降の表にノード毎の停止コマンド一覧を記します。

(1)　　APノード

コマンド名	オプション	実行有無	概要
daolmterm	daolmterm	選択	オンライン中メンテナンスの停止を行います。
dadacterm	dadacterm	選択	DBアクセス制御の停止を行います。

※各コマンドの詳細は、コマンドリファレンスを参照してください。

(2)　　OLTPノード

コマンド名	オプション	実行有無	概要
daolmterm	daolmterm	選択	オンライン中メンテナンスの停止を行います。
dadacterm	dadacterm	選択	DBアクセス制御の停止を行います。

※各コマンドの詳細は、コマンドリファレンスを参照してください。

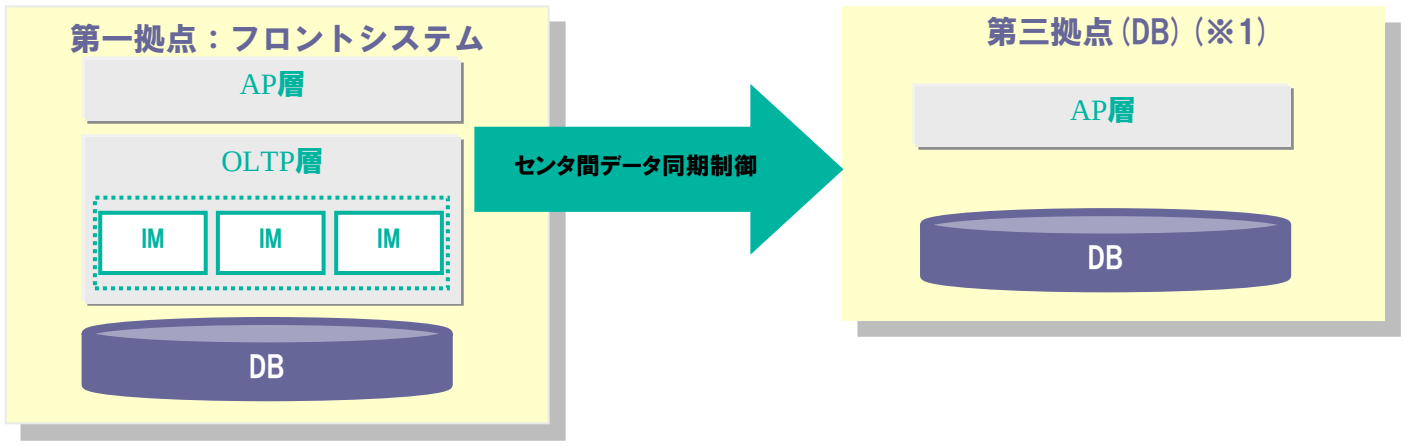
3.3　データベース構成変更

V3.0 ではデータベース構成変更は機能未提供です。

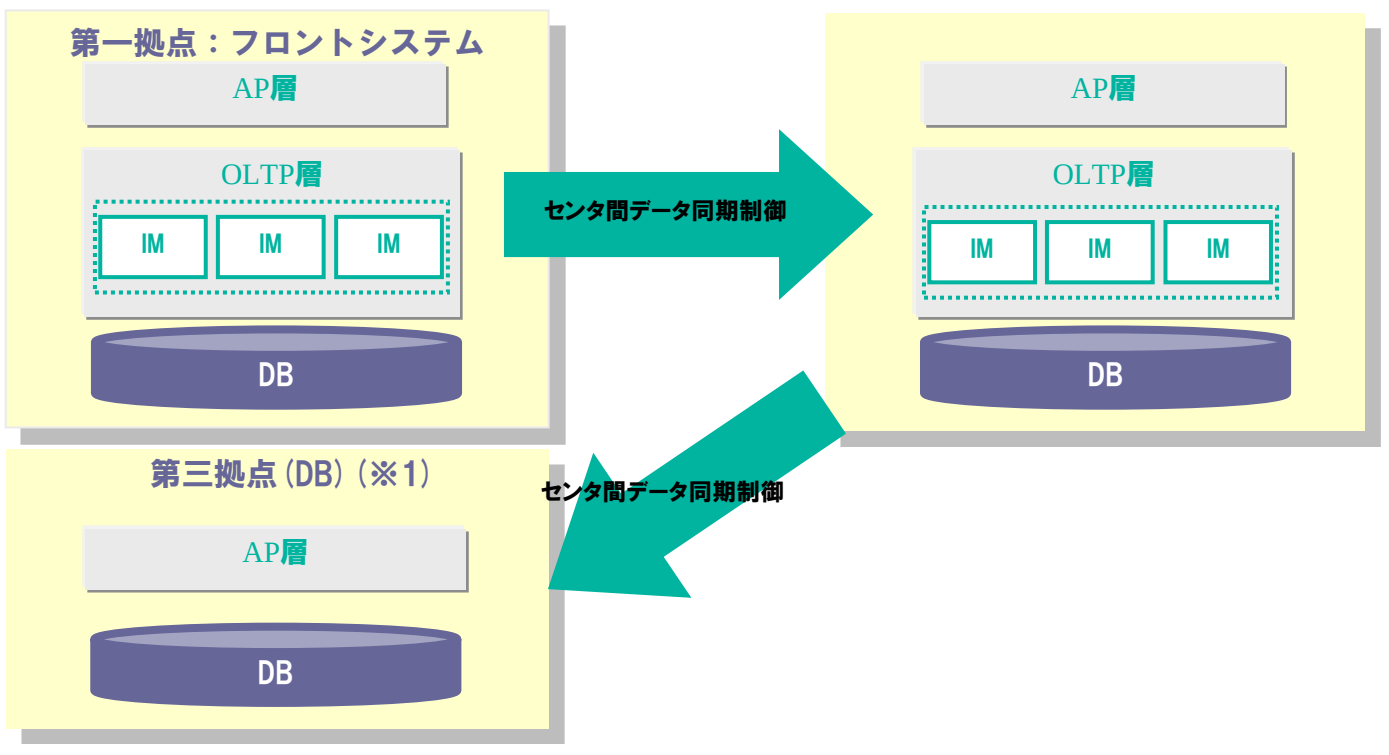
MAP 構成変更や IM のメンテナンスを行う手順をまとめた機能として、TAM 再配置機能があります。
本章では、TAM 再配置機能を利用したデータベース構成変更について説明します。

本章で扱うデータベース構成変更手順は、下記のシステム構成モデルをベースに手順化しています。

＜システム構成モデル①　フロントシステムのみの構成＞



＜システム構成モデル②　フロントシステム／バックアップシステムの構成＞



（※1）第三拠点（DB）はセンタ間データ同期制御を通してデータのバックアップを行います。

3.3.1　　構成変更パターン

データベース構成変更は、MAP のデータ構成を変更するパターン(MAP 構成変更)と、TAM(IM 表、TAM インスタンス)を変更するパターン(TAM メンテナンス)の大きく 2 つに分類されます。

TAM 再配置で取り扱うデータベース構成変更とその概要について以下に示します。

分類	データベース構成変更	概要
MAP 構成変更	MAP 追加／削除	MAP の追加／削除を行い、各 MAP に割り当てられたハッシュ値を変更することで、MAP 構成を変更します。
	レプリケーショングループ(REPG) 追加／削除	REPG の追加／削除を行い、各 MAP に割り当てられたハッシュ値を変更することで、MAP 構成を変更します。
	OLTP ノード追加／削除	ローリングアップデートによって実施します。
	ハッシュ関数置換	ハッシュ関数を置換し、DB のデータのハッシュ値を更新することで、MAP 構成を変更します。
	ハッシュ値追加	既存のメインキーとハッシュ値の対応は変更せず、新しいメインキーとハッシュ値の対応を追加します。
TAM メンテナンス	レコードサイズ変更(既存データ影響あり) (項目追加／削除／領域長変更)(※1)	論理表のレコード項目の追加／削除やレコード項目の領域長変更を行います。
	レコードサイズ変更(既存データ影響なし) (項目追加／削除／領域長変更)(※1)	論理表のレコード項目の追加／削除やレコード項目の領域長変更を行います。
	論理表追加／削除	論理表の追加／削除を行います。
	インデックスキー追加／削除	論理表のインデックスキーの追加／削除を行います。
	容量サイズ変更	IM 表、及び TAM インスタンスの容量サイズ変更を行います。

(※1) 論理表定義の変更により IM 上の既存レコード項目のオフセット、及び IM 上のレコード長の定義に変更が無い場合、IM の定義変更は不要となり、既存データに影響はありません。

3.3.2　　TAM 再配置方式

データベース構成変更は、データベース構成、業務状態、論理システム構成に合わせた構成変更手順が必要となります。TAM 再配置は、それぞれの構成に合わせた構成変更手順を提供します。

データベース構成、業務状態、論理システム構成に対応する TAM 再配置方式について以下に示します。

データベース構成	業務状態(※1)	論理システム構成	TAM 再配置方式
IM と DB の両方	オンライン中	フロントシステムのみ	DB 同期しない TAM 再配置(※2)
		フロントシステム／バックアップシステム	DB 同期しない TAM 再配置(※2) 計画切替を利用した TAM 再配置(※3)
	オフライン中	フロントシステムのみ	オフライン中の TAM 再配置(DB あり)
		フロントシステム／バックアップシステム	
IM のみ	オンライン中	フロントシステムのみ	DB 同期しない TAM 再配置(※2)
		フロントシステム／バックアップシステム	
	オフライン中	フロントシステムのみ	オフライン中の TAM 再配置(DB なし)
		フロントシステム／バックアップシステム	

(※1) 業務状態にはオンライン中とオフライン中があり、業務運用中の状態をオンライン中と呼び、業務を停止した状態をオフライン中と呼びます。

(※2)「DB 同期しない TAM 再配置手順」は既存データに影響がないため、計画切替なしでデータベース構成変更ができる手順です。

(※3)「計画切替を利用した TAM 再配置」は、計画切替を行う際に短時間の業務停止が発生します。

以下にデータベース構成変更パターンと TAM 再配置方式との対応を記載します。TAM 再配置手順番号は、「付録 TAM 再配置手順」の手順番号に対応します。データベース構成変更を行うシステムの構成とデータベース構成変更から TAM 再配置手順を選択します。

分類	データベース構成変更	TAM 再配置方式毎の TAM 再配置手順番号			
		計画切替を利用した TAM 再配置	DB 同期しない TAM 再配置	オフライン中の IM 再配置(DB あり)	オフライン中の IM 再配置(DB なし)
MAP 構成変更	MAP 追加／削除	1-1	-	3-2	4-1
	レプリケーショングループ(REPG)追加／削除	1-2	-	3-2	4-1
	OLTP ノード追加／削除 （ローリングアップデートで実施する）	1-3	-	3-2	4-1
	ハッシュ関数置換	1-4	-	3-1	4-1
	ハッシュ値追加	-	2-1	3-2	4-1
TAM メンテナンス	論理表追加／削除	-	2-2	3-2	4-1
	レコードサイズ変更(既存データ影響あり) (項目追加／削除／領域長変更)	1-5	-	3-2	4-1
	レコードサイズ変更(既存データ影響なし) (項目追加／削除／領域長変更)	-	2-3	3-2	4-1
	インデックスキー追加／削除	1-6	-	3-2	4-1
	容量サイズ変更	1-7	-	3-2	4-1

以下に、各 TAM 再配置方式の概要を記載します。

(1)　　計画切替を利用したTAM再配置

計画切替を利用したTAM再配置は、フロントシステムとバックアップシステム、第三拠点(DB)で構成される論理システム構成でのデータベース構成変更手順です。バックアップシステムのデータベース構成変更を行った後、計画切替(※1)を利用してフロントシステムとバックアップシステムを切り替え、旧フロントシステムのデータベース構成変更をバックアップ側で行います。

通常の計画切替ではフロントシステムとバックアップシステムの切り替え後にセンタ間データ同期を再開しますが、TAM再配置中に実施する場合は拠点間のデータの整合性を保つため、切り替え後のデータ同期は行わず、全拠点のメンテナンス完了後に再開します。

また、フロントシステムでの業務に影響を与えないようにするため、MAP構成変更のTAM再配置では定義変更などの操作を計画切替の中で行う必要があります。

以下に処理の流れを記載します。

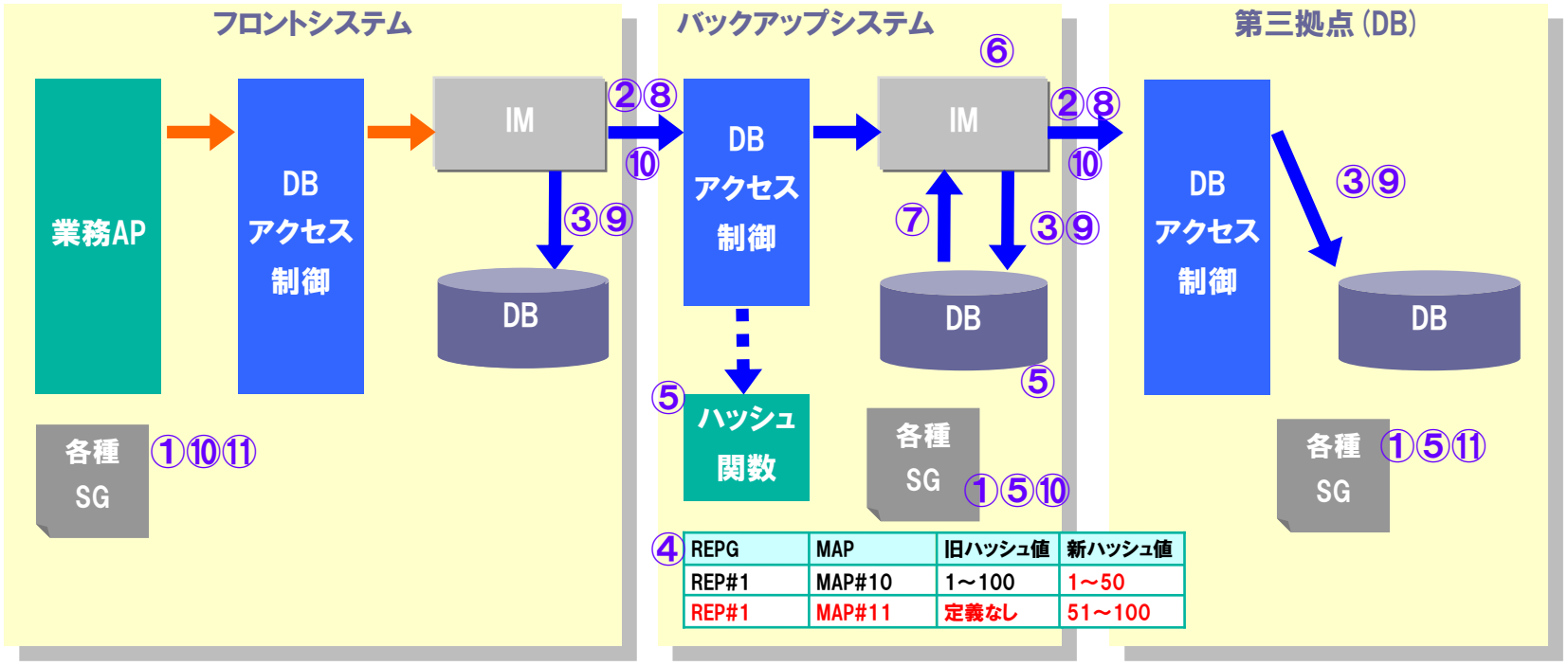
(※1)計画切替については「3.4 バックアップ・リストア」を参照してください。

(注意)

MAP構成変更のTAM再配置中に全データ削除(TRUNCATE)を行うことはできません。

MAP 構成変更の TAM 再配置中に全データ削除を行うと、バックアップ拠点のデータストア ログデータ実行制御(ログリーダ)が異常停止します。この場合はTAM再配置前の環境に戻してログリーダがTRUNCATEの更新ログを処理した後、再度TAM再配置を行う必要があります。

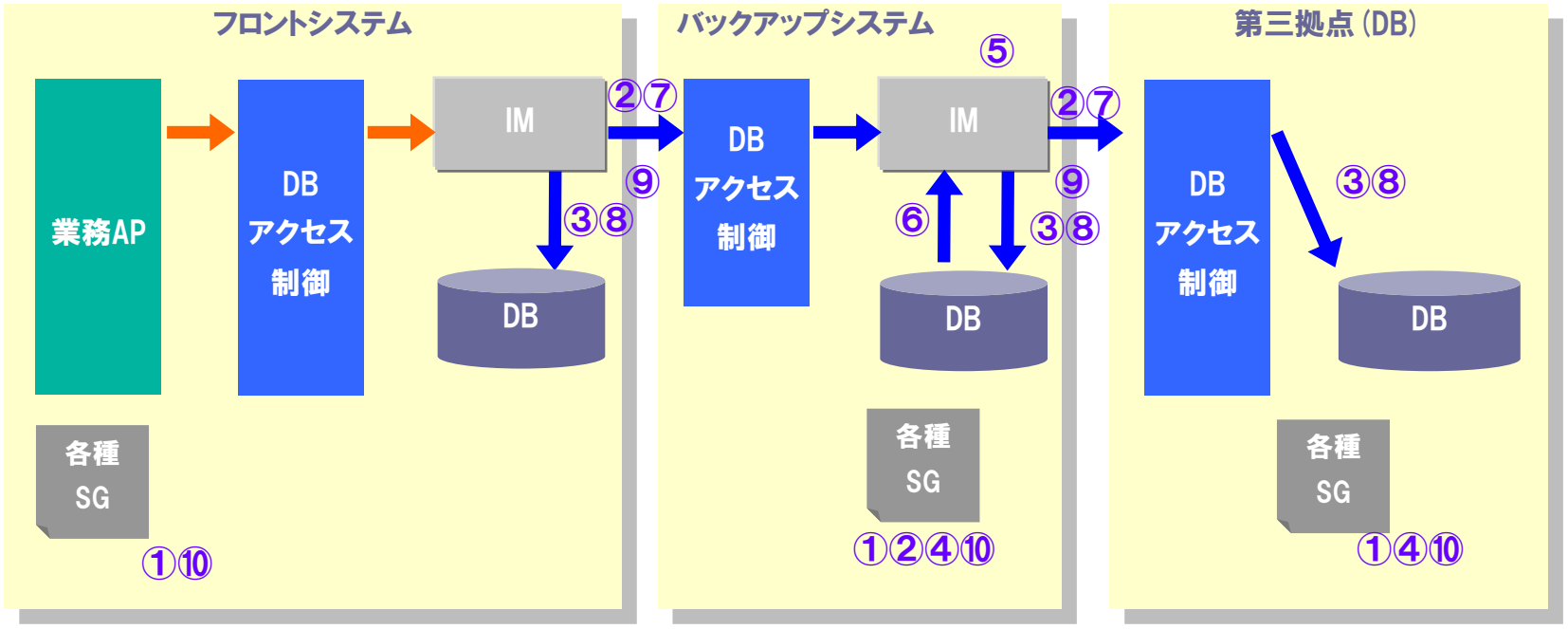
(a)　MAP 構成変更



NO	対象拠点	手順	MAP		REPG		ハッシュ関数置換
			追加	削除	追加	削除	
事前準備(主にこれから追加される項目の定義変更)							
①	全拠点	各種環境定義(メモリキャッシュ)の変更を行います。	○	○	○	○	－
		各種環境定義(データストア等)の変更を行います。	○	－	○	－	－
DB アクセス切替							
②	全拠点	センタ間データ同期(IM)の更新ログ転送を停止します。	○	○	○	○	○
③	全拠点	IM-DB データ同期が完了するまで待ち合わせます。	○	○	○	○	○
メンテナンス							
④	バックアップ	MAP に割り当てられたハッシュ値を切り替えます。	○	○	○	○	－
⑤	バックアップ／ 第三拠点(DB)	各種環境定義(IMENV、APLIB)の変更を行います。 ※この時点までに、ハッシュ関数を含むライブラリの置換を行う必要があります。	－	－	－	－	○
	バックアップ	DB のハッシュ値を更新します。	－	－	－	－	○
⑥	バックアップ	IM 上のデータを初期化します。	○	○	○	○	○
IM アクセス切替							
⑦	バックアップ	DB のデータを IM にロードします。	○	○	○	○	○
⑧	全拠点	センタ間データ同期(IM)の更新ログ転送を再開します。	○	○	○	○	○
⑨	全拠点	IM-DB データ同期の更新ログ反映を再開します。	○	○	○	○	○
拠点計画切替							
⑩	全拠点	計画切替を行い、フロントシステムとバックアップシステムを切り替えます。(以降、計画切替前のフロント拠点を新バックアップ拠点とします) 計画切替後、センタ間データ同期制御(IM)は停止したままの状態にしておきます。(新バックアップ側での IM アクセス切替後に再開します)	○	○	○	○	○

	フロント	これまでフロントであった拠点の各種環境定義(メモリキャッシュ)の変更を行います。 フロントでの業務に影響を与えないため、バックに切り替わる際に処理を行います。	○	○	○	○	－
		これまでフロントであった拠点の各種環境定義(データストア等)の変更を行います。 フロントでの業務に影響を与えないため、バックに切り替わるタイミングで処理を行います。	○	－	○	－	－
	バックアップ	これからフロントとなる拠点の各種環境定義(メモリキャッシュ、データストア等)の変更を行います。 フロントでの業務に影響を与えないため、フロントに切り替わる前に定義変更を完了させます。	－	○	－	○	－
新バックアップの構成変更(「DB アクセス切替」以降の手順を、計画切替後の新バックアップシステムに対して行う)							
後処理(主に削除された項目の定義変更)							
⑪	新バックアップ／第三拠点(DB)	各種環境定義(メモリキャッシュ、データストア以外)の変更を行います。	－	○	－	○	－

(b)　TAM メンテナンス



NO	対象拠点	手順	レコードサイズ変更			インデックス変更		容量変更	
			追加	削除	領域長変更	追加	削除	インスタンス	物理表
事前準備(主にこれから追加される項目の定義変更)									
①	全拠点	環境定義(DB アクセス制御)を変更し、業務 AP を置換します。	―	―	―	―	○	―	―
		環境定義(IM)を変更します。	○	○	○	○	○	○	○
		環境定義(DB) を変更します。	○	―	○	―	―	―	―
DB アクセス切替									
②	全拠点	センタ間データ同期(IM)の更新ログ転送を停止します。	○	○	○	○	○	○	○
③	全拠点	IM-DB データ同期が完了するまで待ち合わせます。	○	○	○	○	○	○	○
メンテナンス									
④	バックアップ／第三拠点(DB)	環境定義(DB アクセス制御)を変更し、業務 AP を置換します。	○	○	○	○	―	―	―
		環境定義(メモリキャッシュ)を変更します。	○	○	○	○	○	―	―
⑤	バックアップ	IM 上のデータを初期化します。	○	○	○	○	○	○	○
IM アクセス切替									
⑥	バックアップ	DB のデータを IM にロードします。	○	○	○	○	○	○	○
⑦	全拠点	センタ間データ同期(IM)の更新ログ転送を再開します。	○	○	○	○	○	○	○
⑧	全拠点	IM-DB データ同期の更新ログ反映を再開します。	○	○	○	○	○	○	○
拠点計画切替									
⑨	全拠点	計画切替を行い、フロントシステムとバックアップシステムを切り替えます。 計画切替後、センタ間データ同期制御(IM)は停止したままの状態にしておきます。(新バックアップ側での IM アクセス切替後に再開します)	○	○	○	○	○	○	○

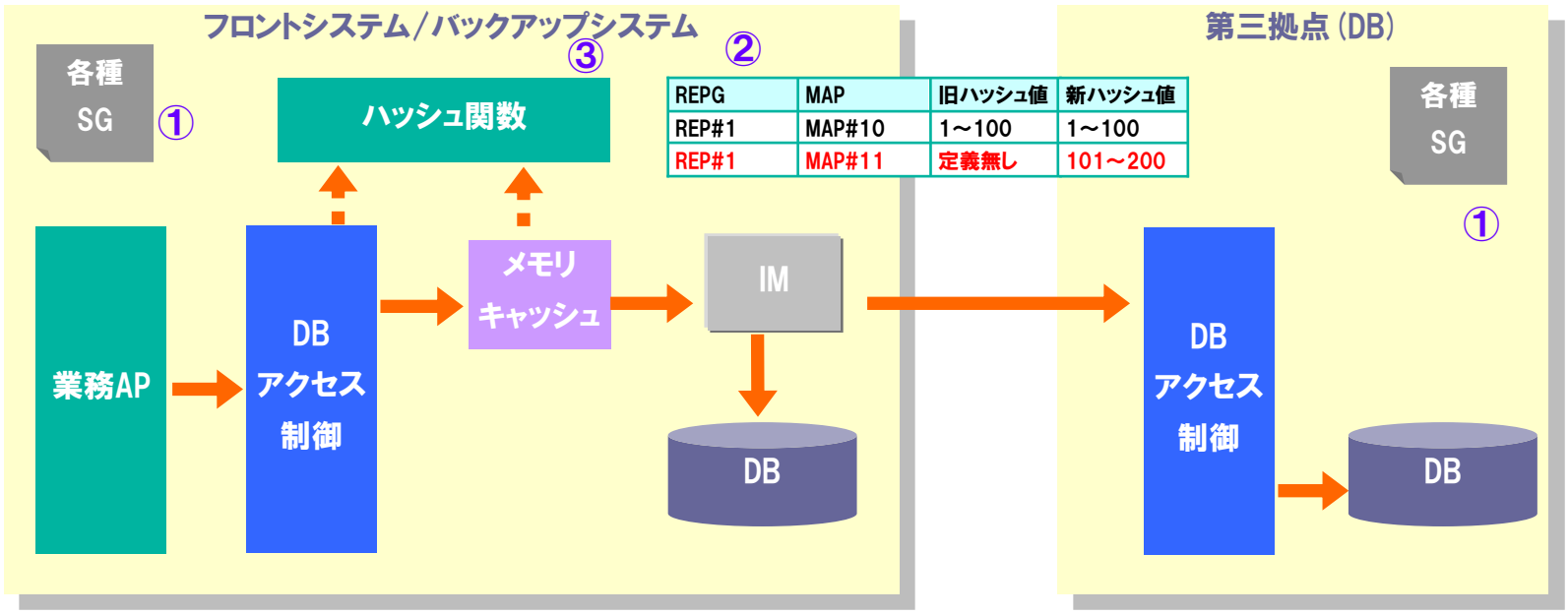
新バックアップの構成変更(「DB アクセス切替」以降の手順を、計画切替後の新バックアップシステムに対して行う)									
後処理(主に削除された項目の定義変更)									
⑩	全拠点	環境定義(DB) を変更します。	－	○	－	－	－	－	－

(2)　　DB 同期しない TAM 再配置

DB 同期しない TAM 再配置では、各拠点で同期することなくデータベース構成変更を行うことができますが、更新ログ転送先での反映エラーを防止するため、センタ間データ同期制御の転送先論理システムから順に行う必要があります。

(a)　MAP 構成変更

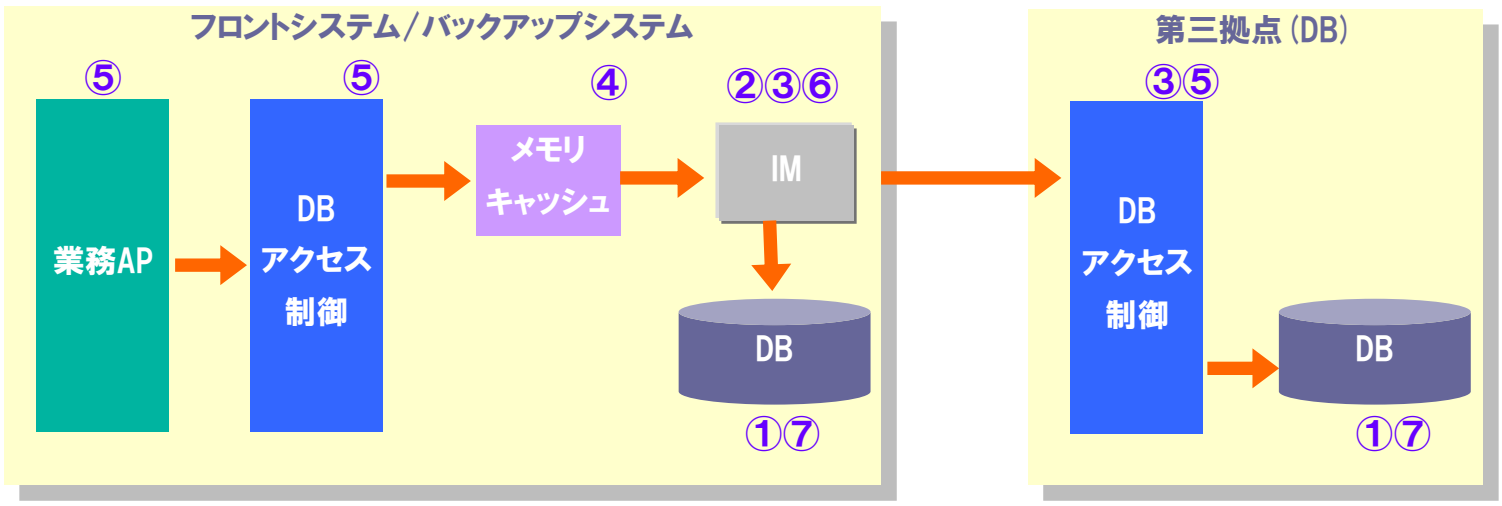
データベース構成変更パターンの「ハッシュ値追加」を行うことができます。「ハッシュ値追加」では MAP、レプリケーショングループの追加も行うことができます。以下にその手順を記載します。



NO	対象拠点	手順	ハッシュ値追加		
			構成変更なし	MAP 追加	REPG 追加
①	全拠点	各種環境定義の変更を行います。	―	○	○
②	フロント／ バックアップ	MAP に割り当てられたハッシュ値を切り替えます。	○	○	○
③	フロント／ バックアップ	ハッシュ関数を含むライブラリの置換を行います。	○	○	○

(b)　TAM メンテナンス

データベース構成変更パターンの「レコードサイズ変更(既存データ影響なし)」と「論理表追加／削除」を行うことができます。以下にその手順を記載します。



NO	対象拠点	手順	レコードサイズ変更			論理表	
			項目追加	項目削除	項目変更	追加	削除
事前準備(主にこれから追加される項目の定義変更)							
①	全拠点	環境定義(DB) を変更します。	○	―	○	○	―
②	フロント／バックアップ	環境定義(IM)を変更します。	―	―	―	○	―
メンテナンス							
③	全拠点	環境定義(DB アクセス制御)を変更し、業務 AP を置換します。	―	―	―	―	○
④	フロント／バックアップ	環境定義(メモリキャッシュ)を変更し、TAM 物理表をオープン／クローズします。	―	―	―	○	○
⑤	全拠点	環境定義(DB アクセス制御)を変更し、業務 AP を置換します。	○	○	○	○	―
後処理(主に削除された項目の定義変更)							
⑥	フロント／バックアップ	環境定義(IM)を変更します。	―	―	―	―	○
⑦	全拠点	環境定義(DB) を変更します。	―	○	―	―	○

(3)　　オフライン中の TAM 再配置(DB あり)

DB がある環境でのオフライン中の TAM 再配置は、「ハッシュ関数置換」と、その他のデータベース構成変更に分けられます。以下にその手順を記載します。

(a)　ハッシュ関数置換

NO	対象拠点	手順
業務処理の停止		
①	フロント／ バックアップ	オンライン業務の停止処理を行います。
DB アクセス切替		
②	全拠点	センタ間データ同期(IM)の更新ログ転送を停止します。
③	全拠点	IM-DB データ同期が完了するまで待ち合わせます。
メンテナンス		
④	全拠点	各種環境定義(IMENV、APLIB)の変更を行います。
	フロント／ バックアップ	DB のハッシュ値を更新します。
DIOSA/XTP の停止		
⑤	全拠点	各拠点の DIOSA/XTP を停止します。
DIOSA/XTP の起動		
⑥	全拠点	各拠点の DIOSA/XTP を起動します。
業務処理の開始		
⑥	全拠点	オンライン業務の開始処理を行います。

(b)　その他のデータベース構成変更

NO	対象拠点	手順
業務処理の停止		
①	全拠点	オンライン業務の停止処理を行います。
DIOSA/XTP の停止		
②	全拠点	各拠点の DIOSA/XTP を停止します。
定義変更		
③	全拠点	各種環境定義の変更を行います。
DIOSA/XTP の起動		
④	全拠点	各拠点の DIOSA/XTP をコールドモードで起動します。
業務処理の開始		
⑤	全拠点	オンライン業務の開始処理を行います。

(4)　　オフライン中の TAM 再配置(DB なし)

DB がない環境でのオフライン中の TAM 再配置では、停止時に IM のデータをファイルに出力する処理が必要となります。以下にその手順を記載します。

NO	対象拠点	手順
業務処理の停止		
①	全拠点	オンライン業務の停止処理を行います。
TAM のデータを退避		
②	フロント／ バックアップ	ファイル出力 TAM セーブコマンドを利用し、TAM 上のデータをファイルに出力します。 ファイル出力先ディレクトリが全OLTP ノードから参照できる共有ディレクトリでない場合、後述する TAM にデータを反映する処理のために任意の 1OLTP ノードで全 MAP に対して実行します。
DIOSA/XTP の停止		
③	全拠点	各拠点の DIOSA/XTP を停止します。
定義変更		
④	全拠点	各種環境定義の変更を行います。
DIOSA/XTP の起動		
⑤	全拠点	各拠点の DIOSA/XTP をコールドモードで起動します。
TAM にデータを反映		
⑥	フロント／ バックアップ	②でファイル出力先ディレクトリが共有ディレクトリでない場合､TAMのマスタノードを TAM のデータを退避したファイルの存在する OLTP ノードに移動します。
⑦	フロント／ バックアップ	ファイル入力 TAM ロードコマンドを利用し、ファイルのデータを TAM 上に反映します。 この際、構成変更情報取得出口を利用して MAP 構成変更、TAM メンテナンスを行います。
⑧	フロント／ バックアップ	②でファイル出力先ディレクトリが共有ディレクトリでない場合、TAM のマスタノードを各 OLTP ノードに分散させます。
業務処理の開始		
⑨	全拠点	オンライン業務の開始処理を行います。

3.3.3　　環境定義の変更

データベース構成変更を行うには、各機能の環境定義を変更する必要があります。以下にデータベース構成変更を行う際に必要となる環境定義を記載します。各定義の変更内容については、各機能のマニュアルを参照してください。

データベース構成変更 ファイル名	MAP 追加／削除	REPG 追加／削除	OLTP ノード 追加／削除 (※5)	ハッシュ関数 置換	ハッシュ値 追加	論理表 追加／削除	レコード サイズ変更	インデックス キー追加／削除	容量サイズ 変更
アプリケーション実行制御(※5)									
DIOSAMAP 節	－	－	○	－	○(※1)	－	－	－	－
CMDSEND 節	－	－	○	－	○(※1)	－	－	－	－
アプリケーション実行制御(ユーザアプリケーションの定義)									
APLIB 節	－	－	－	○(※3)	○(※3)	○(※3)	○(※3)	○(※3)	－
データストア基盤									
DELAYED 節	○	○	○	－	○(※2)	－	－	－	－
メモリキャッシュ									
IMENV 節(REPGROUP 項)	○	○	○	－	○(※2)	－	－	－	－
IMENV 節(USERAP 項)	－	－	－	○	○	－	－	－	－
IMTABLECONF 節	○	○	○	－	○(※2)	○	○	○	－
インメモリ DB アクセスユーティリティ									
DACENV 節	－	－	－	－	－	○	○	○	－
Table Access Method (TAM)									
table.conf	○	○	○	－	－	○	○	○	○(※4)
tammng.conf	－	○	○	－	－	－	－	－	○(※4)
TPBASE(※5)									
*.vd	－	－	○	－	○(※1)	－	－	－	－
*.term	－	－	○	－	○(※1)	－	－	－	－
DB									
DIOSA_DELAYED_POOL	○	○	○	－	○(※2)	－	－	－	－
ユーザ論理表	－	－	－	－	－	○	○	○	－

＜表の見方＞

○：定義変更が必要な定義ファイル

－：定義変更が不要な定義ファイル

(※1) ハッシュ値追加と同時に OLTP ノードを追加する場合のみ必要となります。

(※2) ハッシュ値追加と同時に MAP、またはレプリケーショングループを追加する場合のみ必要となります。

(※3) ハッシュ関数、またはユーザアプリケーションのライブラリを APLIB 節に定義している場合のみ必要となります。

(※4) 物理表の容量サイズを変更する場合は table.conf の、TAM インスタンス全体の容量サイズを変更する場合は tammng.conf の定義を変更します。

(※5) ローリングアップデートで実施します。

3.4　バックアップ・リストア

V3.0 では PostgreSQL のバックアップ・リストアは機能未提供です。

本章では、以下 4 つの復旧手順について説明します。

・IM の復旧

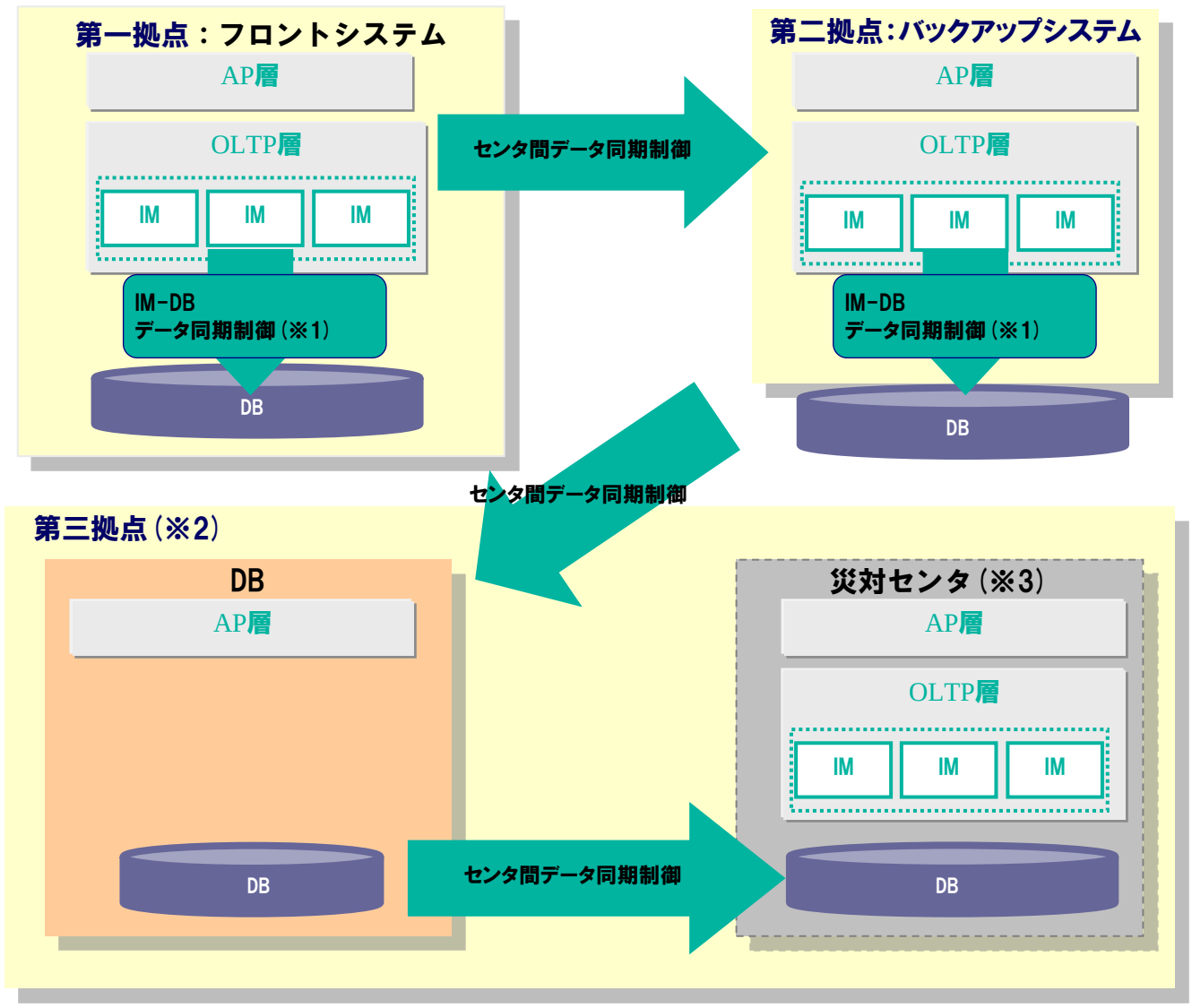
・DB の復旧

・プールファイル障害復旧

・システム障害復旧

各復旧手順は、下記のシステム構成モデルをベースに手順化しています。

＜システム構成モデル＞



（※1） 拠点間のデータベースの復旧は DB の機能を用いて行うため、ユーザアプリケーションから IM にデータアクセスする際には DB アクセス制御の API を利用し、全ての更新を DB にも反映する必要があります。

（※2） 第三拠点はセンタ間データ同期制御を通してデータのバックアップを行う DB と、災害時などのフロントシステム、バックアップシステムの障害に備える災対センタから構成されます。

（※3） 通常時の災対センタは非稼動状態ですが、第三拠点(DB)のログデータ実行制御機能により災対センタの DB へのデータ同期が行われ、最新の状態を保たれます。

3.4.1　　IMの復旧

DIOSA/XTP 起動時にセーブロード機能の TAM ロードコマンドを利用して DB から IM にデータをロードすることで、DIOSA/XTP 停止時の状態に IM を復旧することができます。

TAM ロードコマンドを利用して IM にデータをロードする場合、データ変換・通信オプションの各機能（DB アクセス制御機能、オンラインメンテナンス機能）が起動している必要があります。

TAM ロードコマンドは、MAP のマスタノード上で実行する必要があります。MAP のマスタノードは TAM 再配置機能の運用情報照会コマンドで確認することができます。

ロード実行中は、MAP へのデータアクセスは行わないでください。

（補足）

DB のない環境では、セーブロード機能の TAM ロードコマンドを利用した IM の復旧はできません。この場合、TAM の機能を利用して復旧を行います。

詳細については、「メモリキャッシュ 利用の手引」を参照してください。

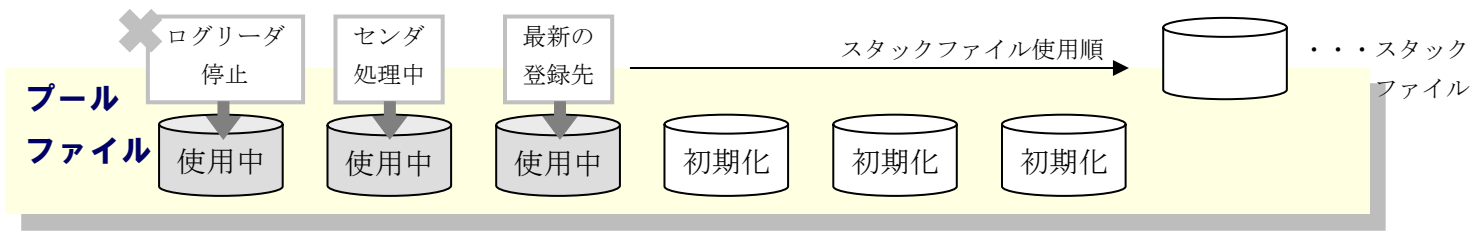
3.4.2　　DB の復旧

運用中に DB が障害となった場合、IM のユーザデータから DB のユーザデータを復旧します。

DB で障害が発生した場合、データストア ログデータ実行制御(ログリーダ)による更新ログの DB 反映(非同期更新)が停止し、IM のみでの運用となります。この間、更新ログを格納する IM のプールファイルに更新ログが蓄積し続けます。このプールファイルの状態に応じて、DB 復旧完了後に行う DB のユーザデータの復旧手順を以下のいずれかから選択します。

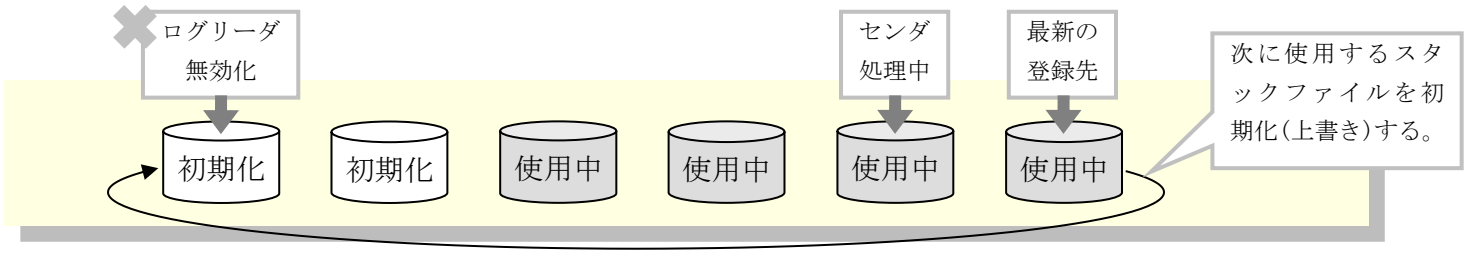
X1. プールファイルの更新ログが失われていない状態

ログリーダ停止時の更新ログから最新の更新ログまでが全てプールファイル上に存在している場合、DB の復旧後にログリーダを再開し、DB のユーザデータを復旧します。



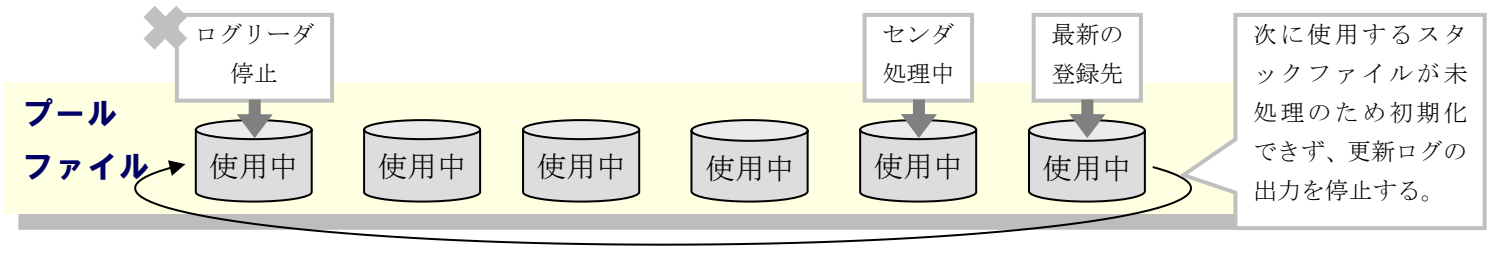
X2-①. プールファイルの更新ログが失われている状態(ケース 1)

ログリーダ無効化中にログリーダ停止時の更新ログを格納したスタックファイルが初期化(上書き)された場合、DB の復旧後に IM のユーザデータを DB に反映してユーザデータを復旧します。



X2-②. プールファイルの更新ログが失われている状態(ケース 2)

スタックファイルが全て使用中(オーバーフロー)となったため、新たな更新ログの出力を停止している場合、DB の復旧後に IM のユーザデータを DB のユーザデータを反映して復旧します。また、オーバーフローが発生している間の更新ログが失われるため、フロントシステム以外の拠点についてもデータベース(IM/DB 両方)の復旧が必要となります。フロントシステム以外のデータベースの復旧は、後述する「システム障害からの復旧」の手順で行います。



X2 の復旧方法では、DB のデータ復旧時、IM-DB データ同期制御で付加的に行う処理(SQL 雛形ファイル)は実行されません。

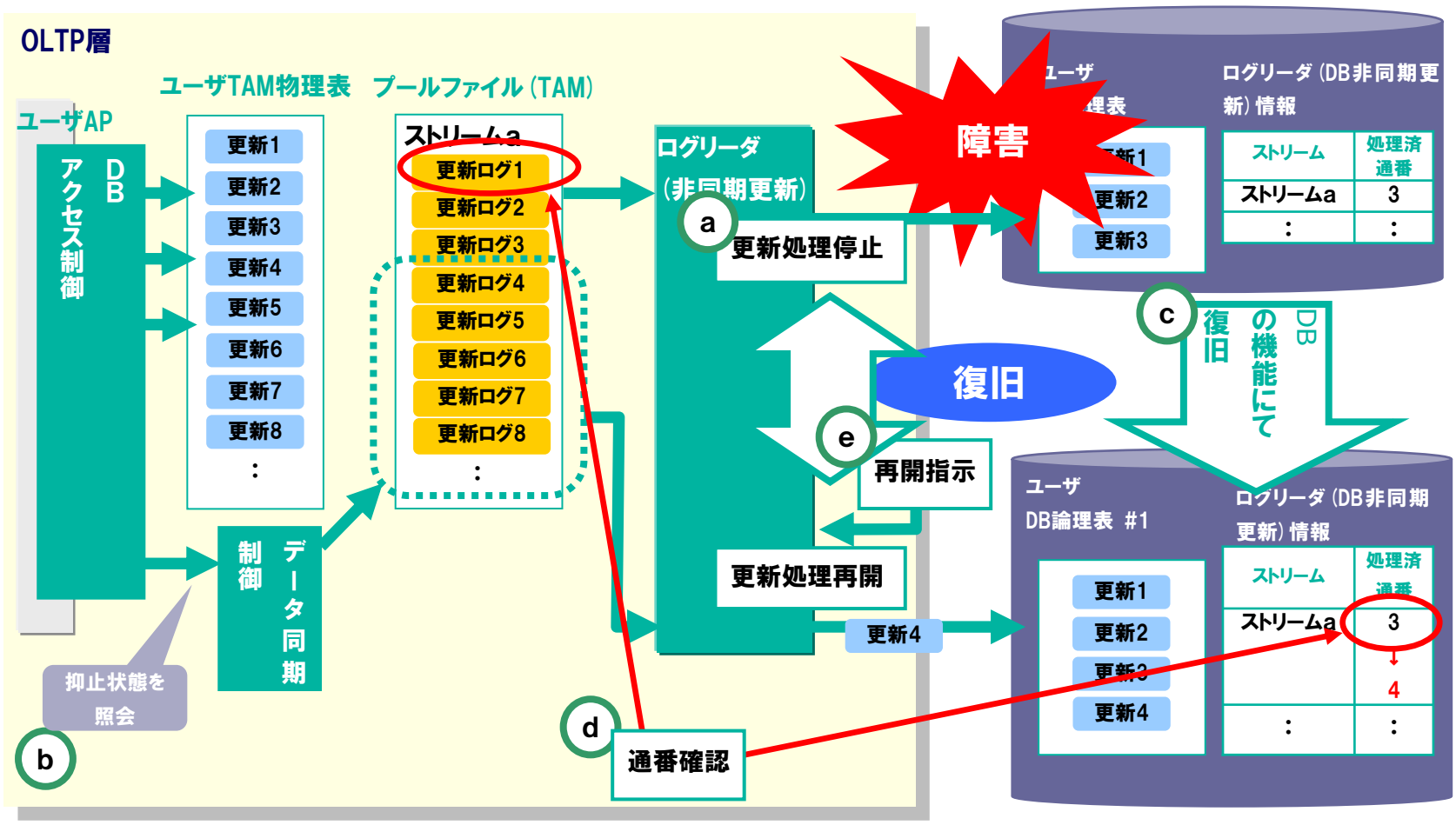
プールファイルの動作の詳細については「データストア 利用の手引」を参照してください。

(1)　X1. プールファイルの更新ログが失われていない場合

プールファイルの更新ログが失われていない場合の復旧は、以下の手順で実施します。

DB障害（プールファイルの更新ログが失われていない場合）

プールファイルの更新ログが失われていない場合、DB復旧後にログリーダの更新処理を再開する



(a)　更新ログのDB反映停止

ログリーダを無効化します。無効化された機能の処理が完了していなくてもスタックファイルは再利用可能となるため、復旧中にプールファイルがオーバーフローすることを防ぐことができます。

(b)　抑止状態確認

プールファイルがオーバーフローし、更新ログ出力抑止状態になっていないか確認します。更新ログ出力抑止状態になっている場合は、「(2) X2.プールファイルの更新ログが失われている場合」の手順(c)以降を実施します。

(c)　DB復旧

DBの復旧を行います。この際、ユーザデータは全て削除しておく必要があります。

(d)　通番確認

プールファイルのスタックファイルが再利用されているか、プールファイルIM表の最も古い更新ログの通番と、ログリーダが最後に処理した通番(処理済み通番)を比較して確認します。

DB復旧でディレード制御情報を復旧できなかった場合は、「(2) プールファイルの更新ログが失われている場合」の手順(f)以降を実施します。

ログリーダが最後に処理した通番の方が古い場合は「(2) X2.プールファイルの更新ログが失われている場合」の手順(g)以降を実施します。

(e)　更新ログの DB 反映再開

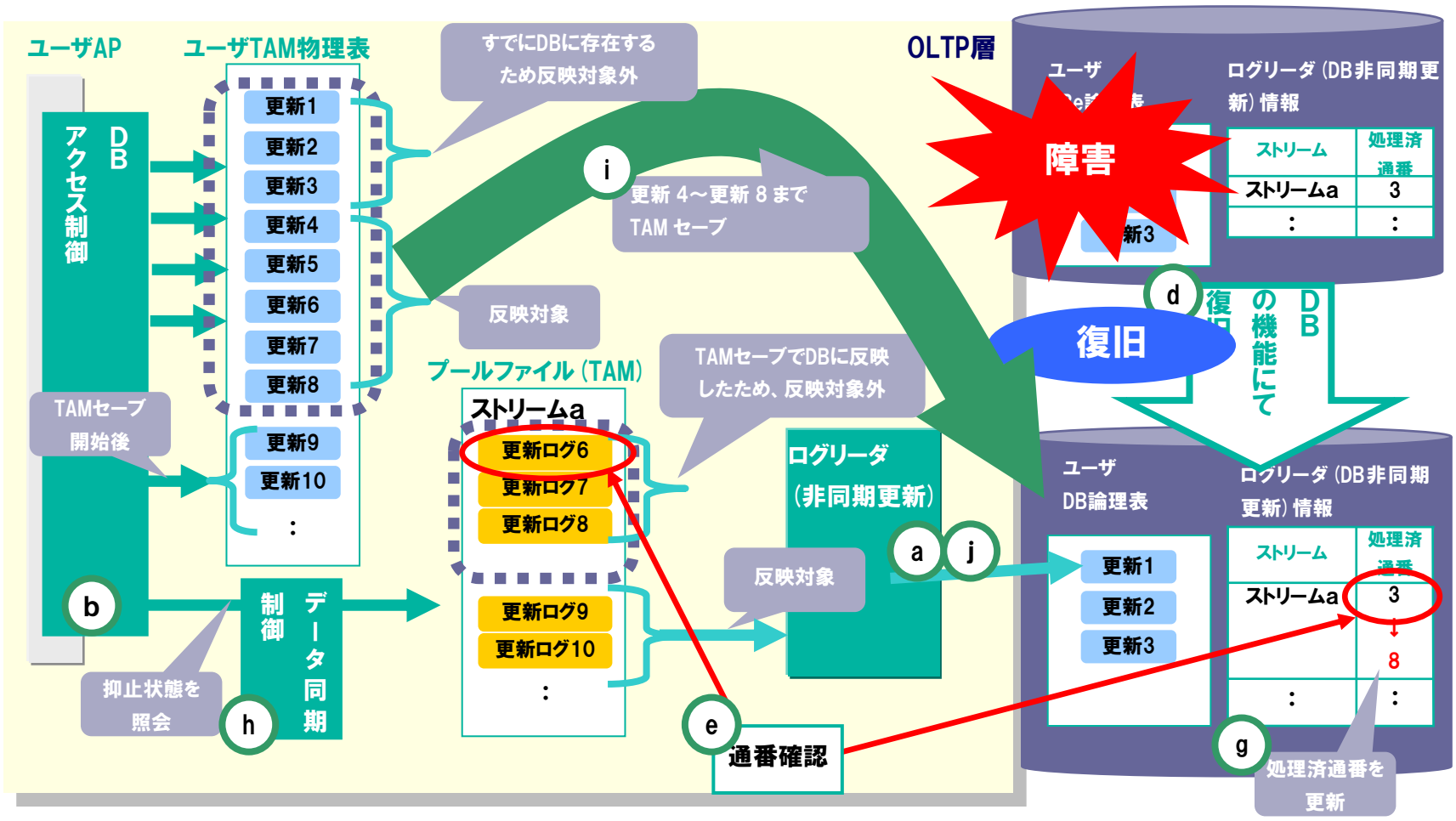
ログリーダの無効化を解除し、DB 反映処理を再開します。DB 反映処理によりプールファイル IM 表に滞留していた更新情報が DB に反映され、IM と DB のデータが同期されます。

(2)　　X2. プールファイルの更新ログが失われている

プールファイルの更新ログが失われている場合の復旧は、以下の手順で実施します。

DB障害（プールファイルの更新ログが失われている場合）

プールファイルの更新ログが失われている場合、TAMセーブによりDBを復旧する



(a)　更新ログの DB 反映停止

(b)　抑止状態確認

(a)～(b) は「(1) プールファイルの更新ログが失われていない場合」の手順と同じです。

(c)　障害先切り離し

手順(b)で 1 件でも更新ログ出力抑止状態のストリームが存在する場合は、他拠点の切り離しを行います。

拠点の切り離しについては、後述する「システム障害からの復旧」を参照してください。

(d)　DB 復旧

(e)　通番確認

(d)～(e) は「(1) プールファイルの更新ログが失われていない場合」の手順と同じです。

(f)　ディレード制御情報生成

手順(d)でDBのディレード制御情報が復旧出来なかった場合は､DBのディレード制御情報を生成します。

(g)　強制開始データ通番変更

プールファイルの最も新しい更新ログの通番を、ログリーダが最後に処理した通番(処理済み通番)に上書きし、ログリーダの無効化を解除します。これにより、ログリーダ再開時には 手順(i)で実行する TAM セーブコマンド開始以降に発生した更新から処理を始めます。

(h)　更新ログ出力抑止解除

手順(b)で1件でも更新ログ出力抑止状態のストリームが存在する場合は、ここで処理済更新ログの削除を行い、更新ログ出力抑止状態を解除します。これにより、プールファイルのオーバーフロー状態が解除され、更新ログの出力が再開します。

(i)　IMからDBへセーブ

TAMセーブコマンドを実行し、IMのユーザデータをDBに反映します。

(j)　更新ログのDB反映再開

DB反映処理を再開します。DB反映処理によりプールファイルに滞留していた更新情報がDBに反映され、IMとDBのデータが同期されます。

3.4.3　　プールファイル障害復旧

運用中にプールファイルがオーバーフローした場合の復旧について扱います。

プールファイルのオーバーフローは、DB 障害時や拠点間の通信障害等により発生することがあります。

フロントシステムのプールファイルがオーバーフローした場合は、プールファイルが IM/DB どちらのデータベースに存在するかに応じて復旧方法を選択します。

X3. IM のプールファイルがオーバーフローしている場合

フロントシステムの IM に存在するプールファイルがオーバーフローした場合は、DB のユーザデータと、フロントシステム以外の拠点のデータベース復旧が必要になります。

X4. DB のプールファイルがオーバーフローしている場合

フロントシステムの DB に存在するプールファイルがオーバーフローした場合は、フロントシステム以外の拠点のデータベース復旧が必要になります。

どちらの復旧方法でも、DB のデータ復旧時、IM-DB データ同期制御で付加的に行う処理(SQL 雛形ファイル)は実行されません。

フロントシステム以外の拠点に存在するプールファイルがオーバーフローした場合は、オーバーフローした時点の更新ログが転送元拠点に残っているかに応じて復旧方法を選択します。

更新ログが残っている場合は、転送元拠点から更新ログの転送を再開することで復旧することができます。

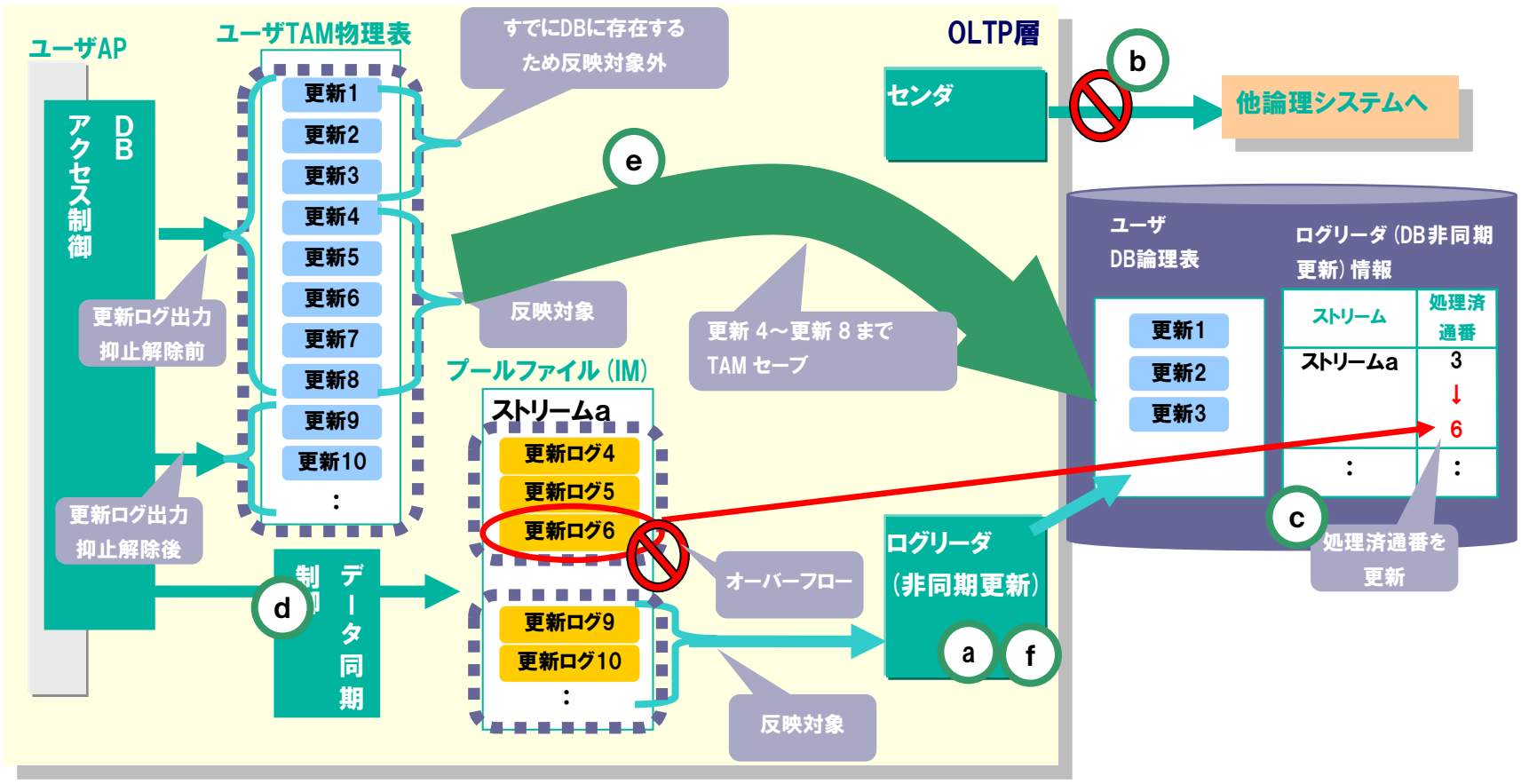
更新ログが残っていない場合は、プールファイルがオーバーフローした拠点をシステム障害と見なし、拠点切り離しを行います。拠点切り離しについては、後述する「システム障害からの復旧」を参照してください。

(1)　X3. フロントシステムの IM のプールファイルがオーバーフローしている場合

フロントシステムの IM のプールファイルがオーバーフローしている場合の復旧は、以下の手順で実施します。

プールファイル障害（フロントシステムのIMのプールファイルがオーバーフローしている場合）

フロントシステムのIMのプールファイルがオーバーフローしている場合、更新ログ出力抑止状態を解除し、TAMセーブによりDBを復旧する



(a)　更新ログの DB 反映停止

ログリーダを無効化します。無効化された機能の処理が完了していなくてもスタックファイルは再利用可能となるため、古い更新ログが削除され、オーバーフロー状態が解消します。

(b)　障害先切り離し

拠点の切り離しについては、後述する「システム障害からの復旧」を参照してください。

(c)　強制開始データ通番変更

プールファイルの最も新しい更新ログの通番を、ログリーダが最後に処理した通番(処理済み通番)に上書きし、ログリーダの無効化を解除します。これにより、ログリーダ再開時には 手順(e)で実行する TAM セーブコマンド開始以降に発生した更新から処理を始めます。

(d)　更新ログ出力抑止解除

処理済更新ログの削除を行い、更新ログ出力抑止状態を解除します。これにより、プールファイルのオーバーフロー状態が解除され、更新ログの出力が再開します。

(e)　IM から DB へセーブ

TAM セーブコマンドを実行し、IM のユーザデータを DB に反映します。

(f)　更新ログの DB 反映再開

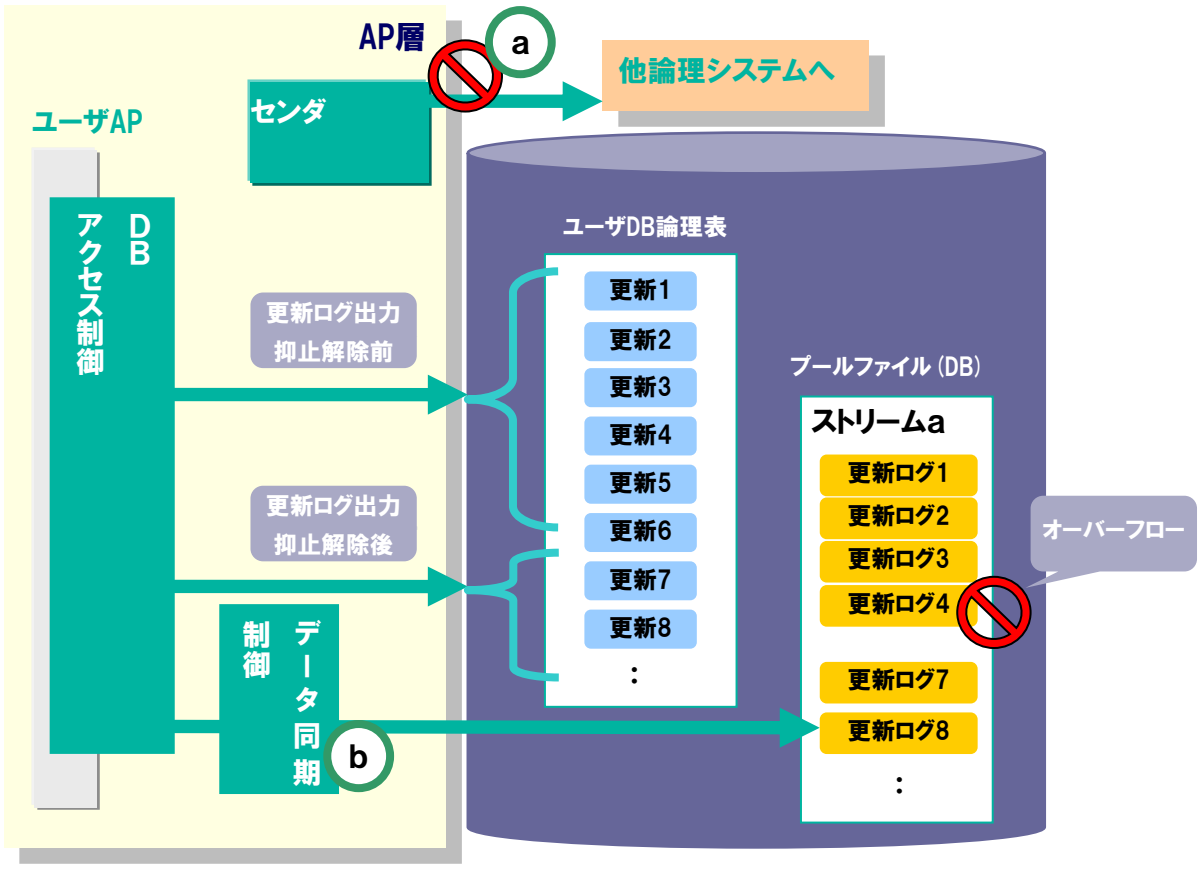
DB 反映処理を再開します。DB 反映処理によりプールファイルに滞留していた更新情報が DB に反映され、IM と DB のデータが同期されます。

(2)　X4. フロントシステムの DB のプールファイルがオーバーフローしている場合

フロントシステムの DB のプールファイルがオーバーフローている場合の復旧は、以下の手順で実施します。

プールファイル障害（フロントシステムのDBのプールファイルがオーバーフローしている場合）

フロントシステムのDBのプールファイルがオーバーフローしている場合、更新ログ出力抑止状態を解除する。



(a)　障害先切り離し

(b)　更新ログ出力抑止解除

「(1) フロントシステムの IM のプールファイルがオーバーフローしている場合」の手順と同じです。

3.4.4　　システム障害からの復旧

運用中にシステム障害が発生した場合、フロントシステムでの業務継続ができる状態を維持する必要があります。オンライン中 DB リカバリ手順に従って実施することで、フロントシステム障害以外の場合は、フロントシステムでの業務を停止することなく継続運転したままの状態で障害を復旧することができます。フロントシステム障害の場合は、短時間でフロントシステムでの業務を開始できます。

次頁より、復旧パターンの流れについて説明します。

システム障害復旧手順は、以下の 2 段階に分けてシステムを復旧します。

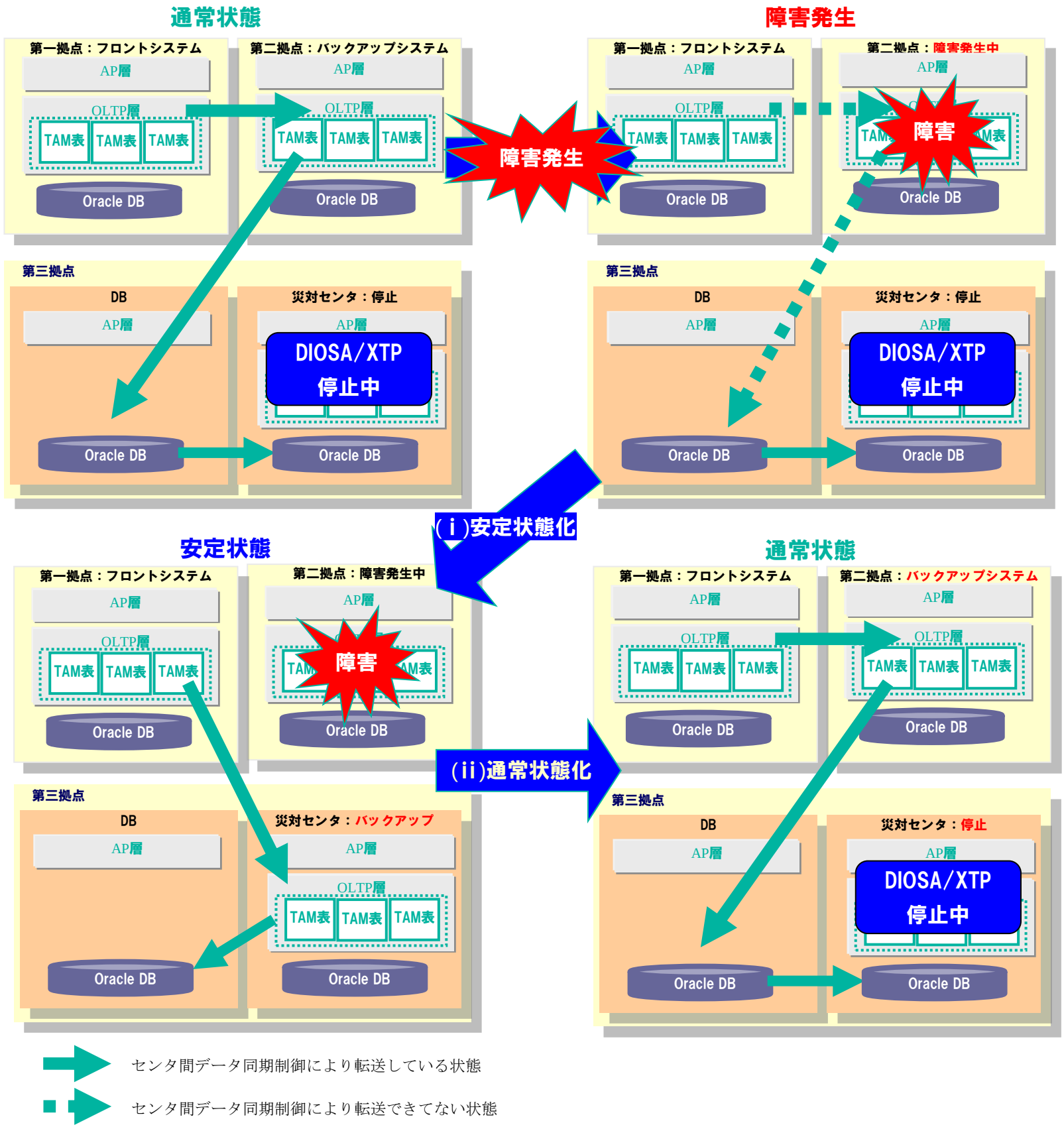
（ｉ）安定状態化：障害状態から安定状態への復旧

（ii）通常状態化(完全復旧)：安定状態から通常状態への復旧

※安定状態とは、フロントシステムで業務が継続運転できる状態。

例として、バックアップシステムでの障害発生から通常状態に戻す復旧の流れ(イメージ)を以下に示します。

＜復旧の流れ＞



(1)　　安定状態化

障害発生直後の状態から、フロントシステムで業務が継続運転できる状態にすることを安定状態化と呼びます。安定状態化では、以下の方針に従って 9 パターンから実施する手順を選択し、短時間でフロントシステムでの業務を開始します。

- フロントシステムが正常動作中にバックアップシステムが障害となった場合、センタ間データ同期制御を継続するため、転送先を変更します。優先順は第三拠点(災対センタ)、第三拠点(DB)です。転送先になることのできる拠点がない場合は障害拠点を切り離し、フロントシステムのみの運用とします。
- フロントシステムが障害となった場合、正常動作している拠点をフロントシステムに切り替えます。優先順はバックアップシステム、第三拠点(災対センタ) です。フロントシステムになることのできる拠点がない場合は、障害となっている拠点を復旧し、フロントシステムとして起動します。
- フロントシステム、バックアップシステムが正常動作中に第三拠点(DB)が障害となった場合、第三拠点(DB)、第三拠点(災対センタ)を切り離します。
- バックアップシステム、または第三拠点(DB)が正常動作中の場合、フロントシステム以外の DB のデータを利用して DB の復旧を行います。

復旧パターン		障害発生時				安定状態			
		第一拠点	第二拠点	第三拠点		第一拠点	第二拠点	第三拠点	
				DB	災対			DB	災対
A	災対センタにバックアップ構築(DB 正常時)	フロント	×	○	停止	フロント	×	○	バック
B	災対センタにバックアップ構築(DB 障害時)	フロント	×	×	停止	フロント	×	×	バック
C	フロント-DB 間転送開始	フロント	×	○	×	フロント	×	○	×
D	バックアップ切替	×	バック	○	停止	×	フロント	○	停止
E	災対センタにフロント構築(DB 正常時)	×	×	○	停止	×	×	○	フロント
F	災対センタにフロント構築(DB 障害時)	×	×	×	停止	×	×	×	フロント
G	DB からのフロント構築	×	×	○	×	フロント	×	○	×
H	障害先切り離し	フロント	×	×	×	フロント	×	×	×
I	第三拠点切り離し	フロント	バック	×	停止	フロント	バック	×	×

＜表の見方＞

フロント：フロントシステムとして動作中

バック　：バックアップシステムとして動作中

停止　　：災対センタの DIOSA/XTP を停止し、DB－災対用 DB 間でレプリケーションされている状態
　　DB からリソースグループセット指定で災対用 DB にデータアクセスできる設定とします

○　　：正常動作中

×　　：システム障害発生中

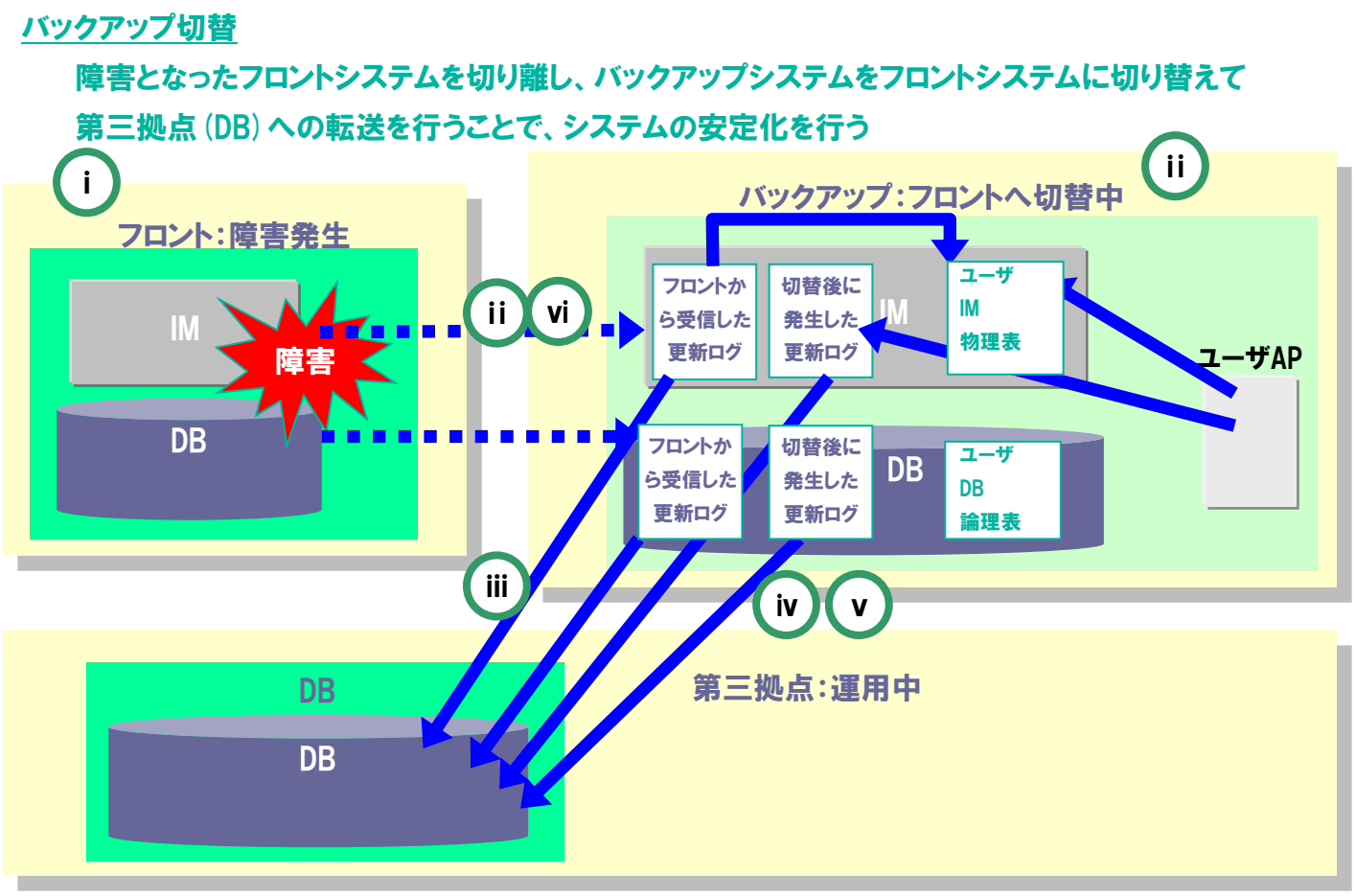
網掛け　：安定状態化により状態が変更された箇所

以下に (D)バックアップ切替の実行手順を記載します。その他の実行手順については「付録 オンライン

中 DB リカバリ手順」を参照してください。

(a)　バックアップ切替

バックアップ切替では、フロントシステムで障害が発生した場合に、バックアップシステムをフロントシステムに切り替えることで安定状態化を行います。



(i)　障害拠点対応

・ フロントシステムのルーティング変更を行います。

(ii) バックアップ切替

・ レシーバ転送停止コマンドを実行し、レシーバを停止します。これにより、フロントシステムからのレプリケーションが停止します。
・ レシーバ強制ディビジョン終了コマンドを実行します。
・ 無効化状態変更コマンドを実行し、フロントシステムからバックアップシステムへのスーパーストリームを無効化します。
・ ルーティング変更を行い、バックアップシステムをフロントシステムに切り替えます。以降ユーザアプリケーションは切り替え後のフロントシステムにデータアクセスを行います。
以後、切り替え後のフロントシステムを新フロントシステムと称します。

(iii)センタ間データ同期制御停止確認

・ ディレード転送機能の状況照会コマンドを実行し、センダが停止していることを確認します。
・ 第三拠点(DB)でディレード転送機能の状況照会コマンドを実行し、レシーバとログリーダが停止していることを確認します。

(iv) センタ間データ同期制御方向切替

・ 第三拠点(DB)でディレード無効化状態変更コマンドを実行し、新フロントシステムから第三拠点

(DB)へのスーパーストリームを無効化します。

- 新フロントシステム、第三拠点(DB)でディレード無効化状態変更コマンドを実行し、新フロントシステムから旧フロントシステム、第三拠点(DB)へのスーパーストリームを無効化解除します。
- センダ動作変更コマンドを実行し、新フロントシステムで動作するセンダユニットの相手論理システムを第三拠点(DB)に変更します。
- 第三拠点(DB)でレシーバ動作変更コマンドを実行し、第三拠点(DB)で動作するレシーバユニットの相手論理システムを新フロントシステムに変更します。

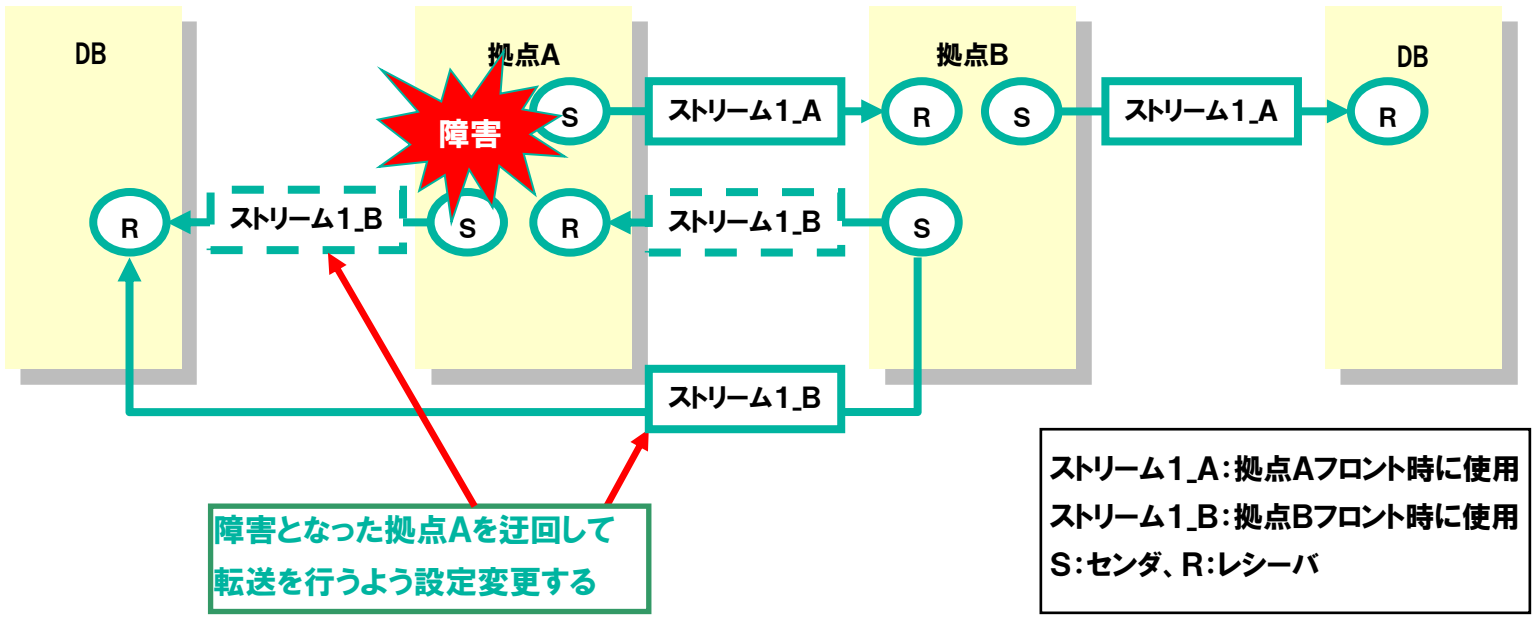
(補足)センタ間データ同期制御方向切替について

障害発生時に使用しているスーパーストリームは、障害となった旧フロントシステムで業務を行うために使用するスーパーストリームです。

新フロントシステムをフロントシステムとするためには、使用するスーパーストリームを新フロントシステムで業務を行うためのスーパーストリームに切り替え、新フロントシステムから第三拠点(DB)へ直接レプリケーションするよう設定変更する必要があります。

センタ間データ同期制御方向切替

拠点ごとに、自拠点がフロントシステムとなった場合に使用するスーパーストリームを定義する。
フロントが切り替わった場合、フロントシステムとなった拠点で業務を行うためのスーパーストリームに切り替える。
転送先の拠点が障害となっている場合には、障害拠点を迂回するよう転送先の変更を行う。



(v)　センタ間データ同期制御再開

- 新フロントシステムでセンダ転送開始コマンドを実行し、新フロントシステムから第三拠点(DB)へのレプリケーションを開始します。これにより、更新ログのディビジョン更新以降に発生した更新ログの転送が開始します。
- 新フロントシステムでログリーダ転送開始コマンドを実行し、IM から DB への更新ログの反映を開始します。
- 第三拠点(DB)でレシーバ転送開始コマンドを実行し、新フロントシステムからの更新ログの受信を開始します。
- 第三拠点(DB)でログリーダ転送開始コマンドを実行し、DB への更新ログの反映を開始します。

(vi) 後処理

- 新フロントシステムでディレード転送機能の定義生成コマンドを実行します。
- 第三拠点(DB)でディレード転送機能の定義生成コマンドを実行します。バックアップ切替前に使用していたスーパーストリームを初期化し、不要となったリソースを解放します。

(2)　　通常状態化(完全復旧)

安定状態から完全復旧に向けてシステムを復旧します。以下の方針に従って 8 パターンから実施する手順を選択します。複数の論理システムが障害となっている場合には全ての論理システムが正常動作している状態になるまで各パターンの手順を実施します。

- フロントシステムが正常動作しており、かつバックアップシステムが存在しない場合、バックアップシステムを構築します。優先順は第一拠点、第二拠点、第三拠点(災対センタ) です。
- フロントシステム、バックアップシステムが正常動作しており、かつ第三拠点(DB)でシステム障害が発生している場合、第三拠点(DB)を構築します。
- フロントシステム、バックアップシステム、第三拠点(DB)が正常動作しており、かつ第三拠点(災対センタ)でシステム障害が発生している場合、第三拠点(災対センタ)を構築します。
- 拠点を作り出せない場合、またはフロントシステムとバックアップシステムを入れ替えたい場合は計画切替を行います。
- バックアップシステム、または第三拠点(DB)が正常動作中の場合、障害拠点の DB のデータ復旧にはフロントシステム以外のデータベースのデータを使用します。

復旧パターン		安定状態				復旧後			
		第一拠点	第二拠点	第三拠点		第一拠点	第二拠点	第三拠点	
				DB	災対			DB	災対
a	災対センタにバックアップ構築(フロント-DB 間転送時)	フロント	×	○	停止	フロント	×	○	バック
b	DB からのバックアップ構築(災対センタ正常時)	フロント	×	○	停止	フロント	バック	○	停止
c	DB からのバックアップ構築(災対センタ障害時)	フロント	×	○	×	フロント	バック	○	×
d	DB からのバックアップ構築(災対センタがバックアップとして稼動時)	フロント	×	○	バック	フロント	バック	○	停止
e	フロントシステムからのバックアップ構築	フロント	×	×	×	フロント	バック	×	×
f	災対センタ構築	フロント	バック	○	×	フロント	バック	○	停止
g	DB 構築	フロント	バック	×	×	フロント	バック	○	×
h	計画切替	バック	フロント	○	停止	フロント	バック	○	停止

＜表の見方＞

フロント：フロントシステムとして動作中

バック　：バックアップシステムとして動作中

停止　　：災対センタの DIOSA/XTP を停止し、DB－災対用 DB 間でレプリケーションされている状態
　　DB からリソースグループセット指定で災対用 DB にデータアクセスできる設定とします

○　　：正常動作中

×　　：システム障害発生中

網掛け　：復旧処理により状態が変更された箇所

以下に、(b) DB からのバックアップ構築(災対センタ正常時) 及び(h) 計画切替の実行手順を記載します。その他の実行手順については「付録 オンライン中 DB リカバリ手順」を参照してください。

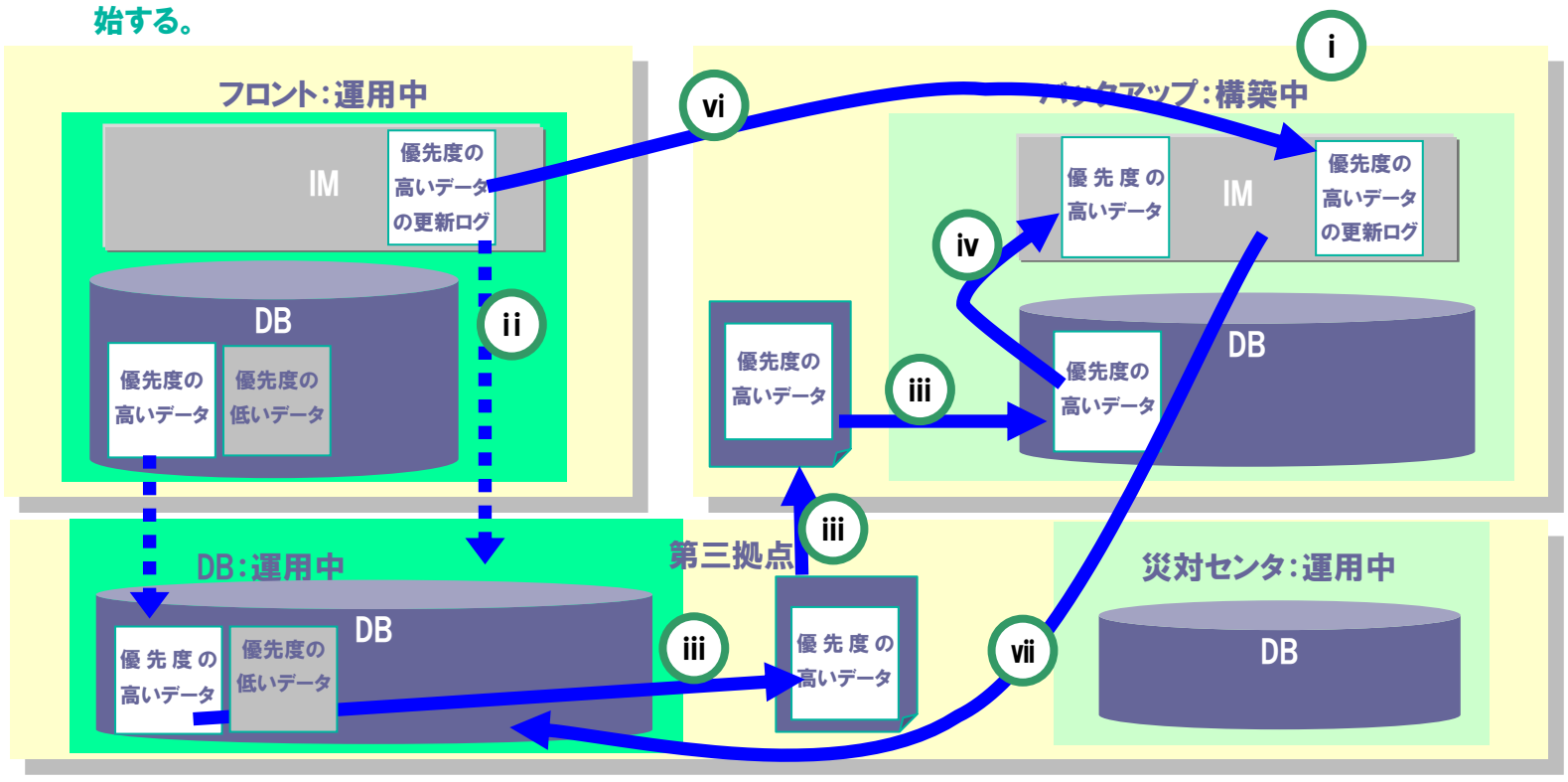
(a)　DB からのバックアップ構築(災対センタ正常時)

DB からのバックアップ構築では、第三拠点(DB)からシステム障害拠点にデータを転送してバックアップシステムを構築します。この手順は優先度の高いデータと優先度の低いデータに分類し、2 段階に分けて行います。ここではユーザアプリケーションが IM への更新を行い、IM-DB データ同期制御により IM と DB の同期を行っているデータを優先度の高いデータとし、それ以外のデータを優先度の低いデータとします。

DB からのバックアップ構築では、第三拠点(DB)の DB データを利用してバックアップシステムの DB データを復旧します。DB の機能を利用したバックアップ／リカバリ方法は複数ありますが、本章では Oracle Data Pump Export / Import ユーティリティを利用した手順を記載します。

DBからのバックアップ構築 (1)

最初にTAM上に存在するデータ(優先度の高いデータ)の復旧を行う。復旧が完了後、ディレード転送機能を開始する。



(i)　システムバックアップから各ノード/OracleDB を復旧

- OS や OracleDB インスタンス等の復旧を行います。

(ii) センタ間データ同期制御停止(フロント→DB)

- フロントシステムでセンダ転送停止コマンドを実行します。以降、手順(g) で転送を再開するまでの間に発生した更新ログはフロントシステムのプールファイル IM 表に蓄積します。
- 第三拠点(DB)でレシーバ転送停止コマンドを実行します。
- 第三拠点(DB)でレシーバ停止前に受信した更新ログが OracleDB に反映されたことを確認します。
- 第三拠点(DB)でセンタ間同期制御機能の更新状況同期コマンドを災対用 DB のログリーダユニット指定で実行し、制御情報ファイルを出力します。

(iii) Oracle セーブ、ロード(優先度の高いデータ)

- 第三拠点(DB)で OracleDB の機能を利用して優先度の高いデータをエクスポートします。
- FTP 等を利用して、第三拠点(DB)からバックアップシステムにダンプ・ファイルを転送します。
- バックアップシステムで OracleDB の機能を利用してダンプ・ファイルをインポートします。

(iv) バックアップシステム開始

- バックアップシステムで DIOSA/XTP を起動します。起動の際にはセーブロード機能の TAM ロードコマンドを実行し、OracleDB のデータを TAM にロードします。
- バックアップシステムでセンタ間同期制御機能の更新状況同期コマンドを実行し、出力した制御情

報ファイルの内容をログリーダに反映します。これにより第三拠点(DB)でディレード転送機能を再開した際に、「センタ間データ同期制御停止」以降に発生した更新から処理を始めるようにします。
- バックアップシステムでルーティング変更を行い、バックアップシステムとして動作するように設定します。

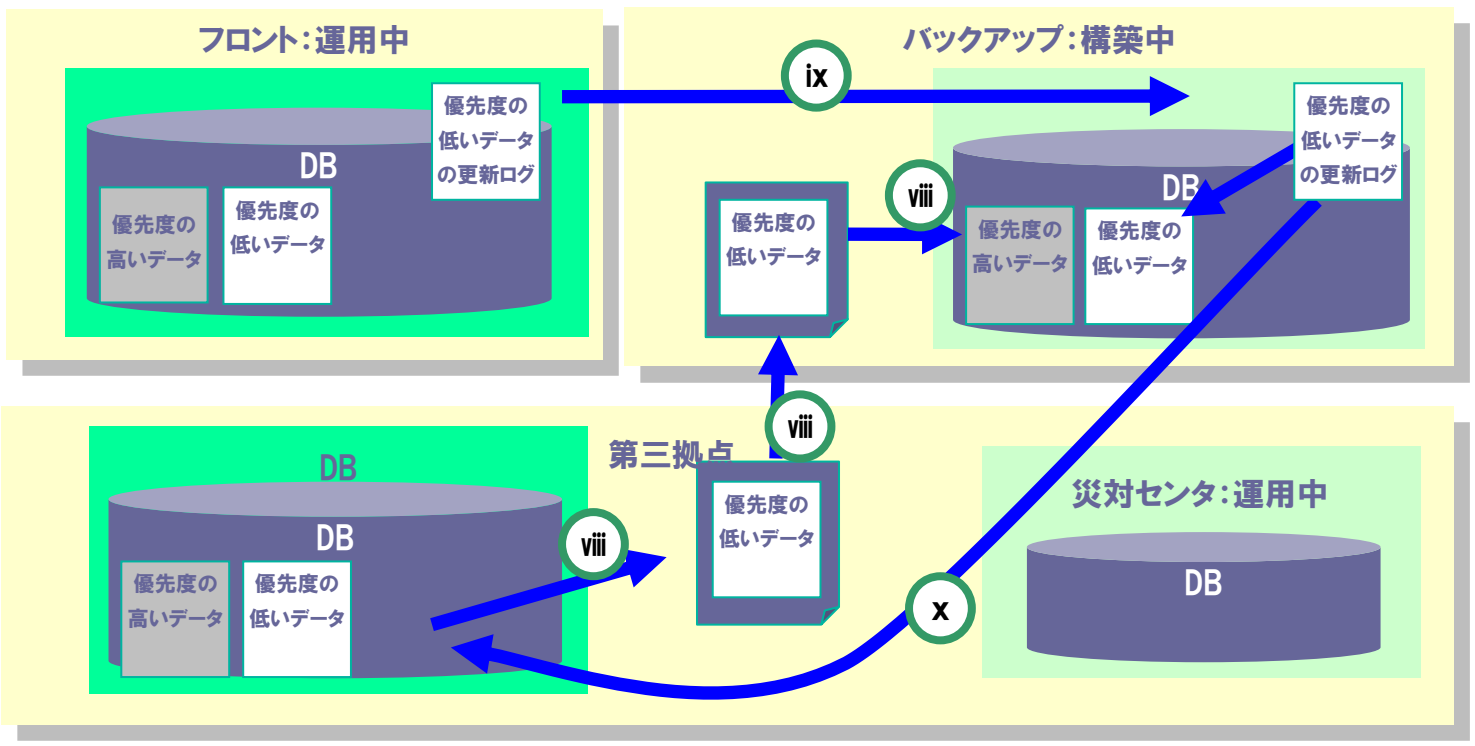
(v) センタ間データ同期制御方向切替
- フロントシステムでセンダ動作変更コマンドを実行し、センダユニットの相手論理システムをバックアップシステムに変更します。これにより、フロントシステムからバックアップシステムへのレプリケーションが行えるようになります。
- 第三拠点(DB)でレシーバ動作変更コマンドを実行し、レシーバユニットの相手論理システムをバックアップシステムに変更します。これにより、バックアップシステムから第三拠点(DB)へのレプリケーションが行えるようになります。

(vi) センタ間データ同期制御再開(フロント→バックアップ) －TAM 更新ログ－
- バックアップシステムでレシーバ転送開始コマンドを実行します。
- フロントシステムでセンダ転送開始コマンドを実行し、フロントシステムからバックアップシステムへのレプリケーションを開始します。

(vii)センタ間データ同期制御再開(バックアップ→第三拠点(DB)) －TAM 更新ログ－
- 第三拠点(DB)でレシーバ転送開始コマンドを実行します。
- バックアップシステムでセンダ転送開始コマンドを実行し、バックアップシステムから第三拠点(DB)へのレプリケーションを開始します。

DBからのバックアップ構築 (2)

DB上のみに存在するデータ(優先度の低いデータ)の復旧を行う。



(viii) Oracle セーブ、ロード(優先度の低いデータ)

- 第三拠点(DB)で OracleDB の機能を利用して優先度の低いデータをエクスポートします。
- FTP 等を利用して、第三拠点(DB)からバックアップシステムにダンプ・ファイルを転送します。
- バックアップシステムで OracleDB の機能を利用してダンプ・ファイルをインポートします。

(ix) センタ間データ同期制御再開(フロント→バックアップ) ーOracle 更新ログー

- バックアップシステムでレシーバ転送開始コマンドを実行します。
- フロントシステムでセンダ転送開始コマンドを実行し、フロントシステムからバックアップシステムへのレプリケーションを開始します。

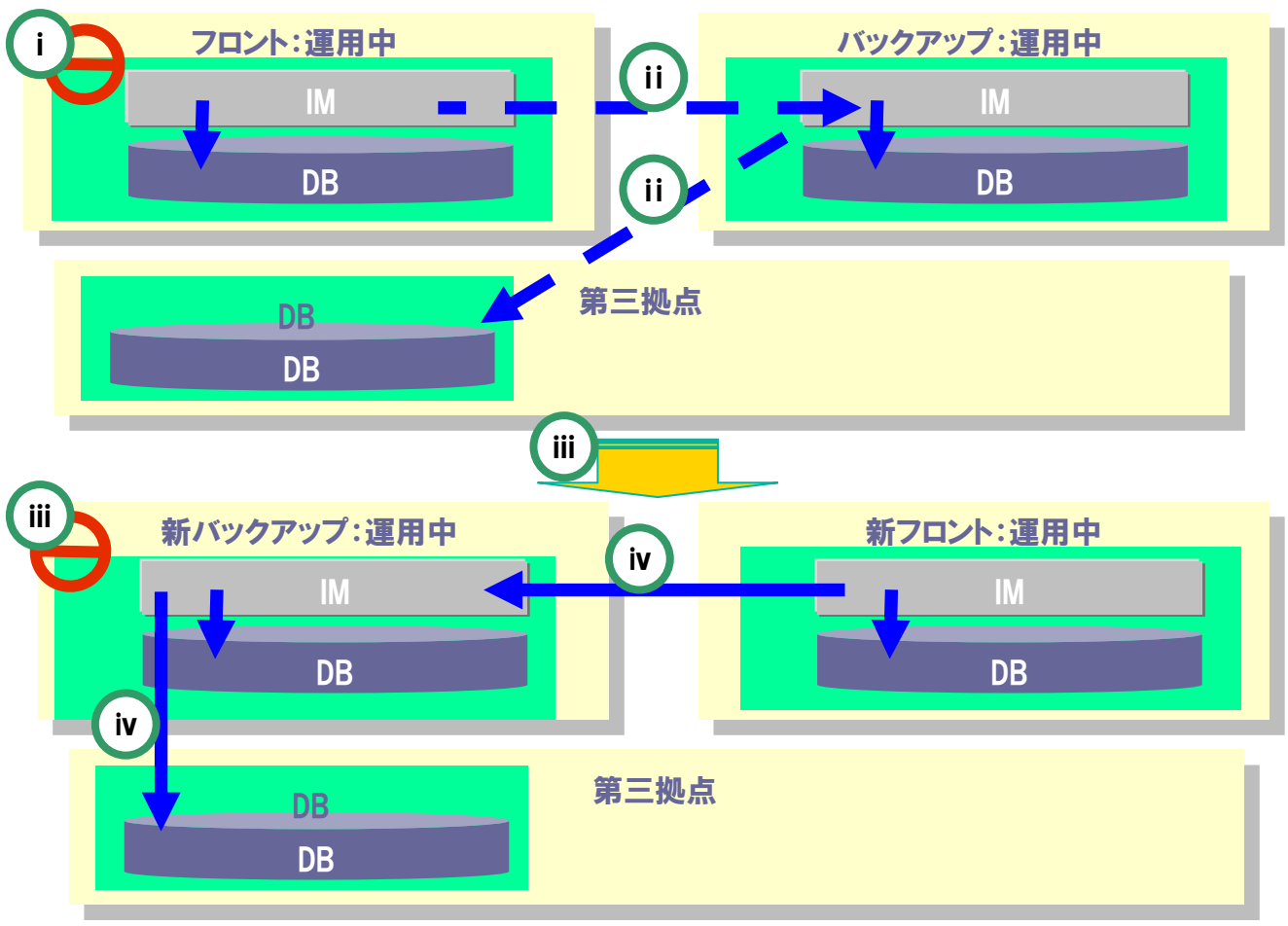
(x) センタ間データ同期制御再開(バックアップ→第三拠点(DB)) ーOracle 更新ログー

- 第三拠点(DB)でレシーバ転送開始コマンドを実行します。
- バックアップシステムでセンダ転送開始コマンドを実行し、バックアップシステムから第三拠点(DB)へのレプリケーションを開始します。

(b)　計画切替

計画切替では、フロントシステムとバックアップシステムの切り替えを行います。この手順はシステム障害からの復旧によりフロントシステムとバックアップシステムが入れ替わってしまった場合や、TAM 再配置手順においてバックアップシステムのデータベース構成変更を先に行った場合に実施します。

この手順の実行中は業務を停止する必要があります。



(i)　業務停止

- 業務処理を停止します。
- フロントシステムで閉塞管理機能の閉塞状態変更コマンドを実行し、全 OLTP ノードを予閉塞します。
- CO 制御機能の稼動状況照会コマンドを実行し、滞留している電文が 0 件になるのを確認します。
- フロントシステムで閉塞管理機能の閉塞状態変更コマンドを実行し、全 OLTP ノードを閉塞します。

(ii) センタ間データ同期制御停止

- フロントシステムでディレード転送機能のディビジョン切替コマンドを実行します。
- 各拠点でディレード転送機能の状況照会コマンドを実行し、センダ、レシーバ、ログリーダが停止したことを確認します。

(iii) 計画切替

- ルーティングを変更し、フロントシステムをバックアップシステムに切り替えます。またバックアップシステムをフロントシステムに切り替えます。
- 業務処理を開始します。

(iv) センタ間データ同期制御開始

- 各拠点でディレード転送機能の無効化状態変更コマンドを実行し、計画切替前に使用していたスーパーストリームを無効化します。
- 各拠点でディレード転送機能の無効化状態変更コマンドを実行し、計画切替前に無効化していたスーパーストリームを無効化解除します。

・ 各拠点でディレード転送機能の各ユニットの転送開始コマンドを実行します。

(v) 後処理

新バックアップシステムで閉塞管理機能の閉塞状態変更コマンドを実行し、全 OLTP ノードを閉塞解除します。

・ 各拠点でディレード転送機能の定義生成コマンドを実行し、無効化したスーパーストリームのリソースを解放します。

付録A　データベース表一覧

A.1　IM表

ユーザデータ状態管理表							
概要	メインキーの一覧を管理する						
論理表名	DIATC_DAC_MAINKEY					物理表名	{論理表名}_{MAPID(10桁)}
データ／制御	制御	分割	MAPID	レコードサイズ	48byte	サイズ計算式	登録メインキー数 × レコードサイズ

更新ログ出力抑止							
概要	更新ログの出力を抑止中のスーパーストリーム情報を管理する						
論理表名	DIATC_DSC_OUTPUTCTRL					物理表名	{論理表名}_{MAPID(10桁)}
データ／制御	制御	分割	MAPID	レコードサイズ	16byte	サイズ計算式	スーパーストリーム数 × レコードサイズ

A.2　DB表

ユーザデータ状態管理表									
概要	メインキーの一覧を管理する								
表名	DIATC_DAC_MAINKEY								
データ／制御	制御	分割	無	バッファ	－	レコードサイズ	52 byte	サイズ計算式	登録メインキー数 × レコードサイズ

更新ログ出力抑止表									
概要	更新ログの出力を抑止中のスーパーストリーム情報を管理する								
表名	DIATC_DSC_OUTPUTCTRL								
データ／制御	制御	分割	無	バッファ	－	レコードサイズ	16 byte	サイズ計算式	スーパーストリーム数 × レコード

付録B　諸元一覧

B.1　インメモリ DB アクセスユーティリティ

項目名称	単位	諸元	備考
IM レコードの構成項目数 (テーブルカラム数)	個数	1～1000	
論理表数	個数	1～256	

DIOSA/XTP V3.1
データ変換・通信オプション
導入の手引

2023 年 6 月 第 3 版

日本電気株式会社
東京都港区芝五丁目 7 番 1 号
TEL (03)3454-1111(大代表)