

D I O S A / X T P V2.1

## メモリキャッシュ 利用の手引

#### 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政府の輸出許可が必要です。

なお、輸出許可申請手続き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、D I O S A / X T P メモリキャッシュの利用の手引です。

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、TAM、Oracle、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。

2018 年 2 月 4 版

本書の関連説明書としては次のものがあります。

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

## 備考

- (1) Microsoft、Windows は、米国あるいはその他の国における米国 Microsoft Corporation の商標または登録商標です。
- (2) UNIX は、X/Open カンパニーリミテッドが独占的にライセンスしている米国ならびに他の国における登録商標です。
- (3) HP、HP-UX は、Hewlett-Packard 社の商標または登録商標です。
- (4) Linux は、Linus Torvalds の米国およびその他の国における商標または登録商標です。
- (5) Red Hat は、米国およびその他の国における Red Hat, Inc. の商標または登録商標です。
- (6) Oracle と Java は、Oracle Corporation およびその子会社、関連会社の米国およびその他の国における登録商標です。
- (7) This product includes software developed by the Apache Group for use in the Apache HTTP server project (<http://www.apache.org/>).
- (8) その他、記載されている会社名、製品名は、各社の登録商標または商標です。

# 目次

第1章	概要	1
1.1	目的と特徴	1
1.2	位置づけ	1
1.3	諸概念	2
1.4	機能構成	4
第2章	機能	5
2.1	インメモリサーバ	5
2.1.1	アクセスサーバとブリッジサーバ	6
2.1.2	各種アクセスを行うための API	7
2.1.3	レコード排他制御	8
2.1.4	デッドロック検出機構	9
2.1.5	可変長レコードアクセス機能	10
2.1.6	グループコミット機能	11
2.1.7	表オープン／クローズ機能	11
2.1.8	性能情報採取	12
2.2	インメモリサーバ所在管理	13
2.2.1	分散配置された TAM のマスタが存在するノードへの振分機能	13
2.2.2	TAM およびインメモリサーバの起動／停止制御機能	14
2.2.3	障害検出とマスタ切替制御	15
2.2.4	コマンドによるマスタ切替機能	16
第3章	アプリケーションの開発	17
3.1	アプリケーションの基本構成	17
3.1.1	サービス区間とトランザクション区間	17
3.1.2	アクセス対象の MAP について	18
3.1.3	アクセス対象となる論理表について	18
3.1.4	レコードのキー値の扱いについて (Linux 版)	18
3.1.5	レコードの更新とレコード削除について	18
3.1.6	全レコードの削除	19
3.1.7	更新するレコード数の上限	20
3.1.8	戻り値について	20
3.1.9	マルチスレッドのアプリケーションプログラム	22
3.2	C0	23
3.2.1	プログラムの構造	23
3.2.2	メモリキャッシュにアクセスするプログラムの開発	24
3.3	ユーザアプリケーションプログラム	31
3.3.1	プログラムの構造	31
3.3.2	メモリキャッシュにアクセスするプログラムの開発	33
3.4	利用者出口	43

3.4.1	プログラムの構造.....	43
3.4.2	ハッシュ関数.....	43
3.5	アプリケーションの生成.....	45
第4章	システムの構築.....	46
4.1	環境設計.....	46
4.1.1	TAM 表の配置の決定.....	46
4.1.2	TAM インスタンスの配置の決定.....	51
4.1.3	IMS キューバッファサイズ(IMQUEBUFSIZE)の決定.....	59
4.1.4	アクセスログ蓄積領域サイズ(%ACCESSLOG-BUFSIZE)の目安.....	62
4.1.5	レコードロックエントリの割り当て数.....	63
4.2	環境定義.....	64
4.2.1	メモリキャッシュ関連 (IMENV).....	64
4.2.2	メモリキャッシュ表関連 (IMTABLECONF).....	72
4.2.3	DIOSA 関連 (DIOSAMAP).....	74
4.2.4	TAM の環境定義.....	77
4.2.5	メモリキャッシュ表と TAM 表の定義.....	78
4.3	ノード間ヘルスチェックの信頼性向上.....	80
第5章	システムの運用.....	81
5.1	起動・停止.....	81
5.1.1	メモリキャッシュの起動.....	81
5.1.2	メモリキャッシュの停止.....	83
5.2	環境変更.....	84
5.2.1	環境定義の変更.....	84
5.2.2	表の追加／削除.....	87
5.2.3	表の追加削除以外.....	89
5.3	障害対応.....	92
5.3.1	ノード障害.....	92
5.3.2	通信パス障害.....	93
5.3.3	TAM 障害.....	94
5.3.4	DIOSA 障害.....	95
5.3.5	TAM の更新ログを採取する場合の障害時マスタ切替.....	97
5.3.6	レプリケーション障害からの復旧.....	98
5.3.7	レプリケーション状態が切替中からの復旧.....	99
5.3.8	ノード孤立からの復旧.....	101
5.4	稼動一待機構成における TAM 監視について.....	102
5.4.1	環境定義について.....	102
5.4.2	障害の検出方法について.....	102
5.4.3	補足.....	102
5.5	一時ファイルの削除.....	103
付録A	リソース一覧.....	104

付録 B プロセス一覧..... 104

付録 C 諸元一覧..... 104

付録 D 環境定義例..... 105

    D.1 DIOSA 環境定義例..... 105

        メモリキャッシュ基本動作の設定..... 105

        各種監視動作の設定..... 106

        レプリケーショングループの設定..... 109

        MAP の設定 ..... 110

        ブリッジサーバの設定..... 112

        アプリケーションの設定..... 113

        表の設定..... 114

    D.2 TAM 環境定義例..... 117

        TAM のインスタンスの設定 ..... 117

        TAM のメモリテーブルの設定 ..... 120

付録 E TAM コマンド制限 ..... 125

    E.1 TAM コマンド制限..... 126

# 第1章 概要

## 1.1 目的と特徴

DIOSA/XTP メモリキャッシュは、InfoFrame Table Access Method(以下 TAM) を基盤としてトランザクション処理制御と分散データ管理を付加することで、TAM の使い勝手を向上するとともに適用領域の拡大を実現することを目的としています。

DIOSA/XTP メモリキャッシュは、以下の特長があります。

- TAM を複数のアプリケーションから同時利用可能にし、高速大量処理を可能とします。
- 分散配置されたデータの所在管理とルーティング制御を行い、全データへの透過的なアクセスを可能とします。
- TAM の稼動状態を管理し、障害時はマスタ／スレーブ切り替えを自動的に行うことを可能とします。

## 1.2 位置づけ

DIOSA/XTP メモリキャッシュは、DIOSA/XTP の 1 つの有償プログラムプロダクト(PP)であり、DIOSA/XTP の基本機能、TAM を利用して、データ処理基盤を提供します。

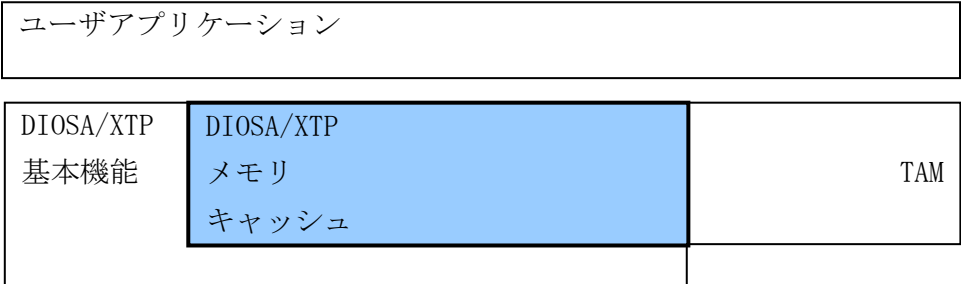


図 1-1 DIOSA/XTP メモリキャッシュの位置付け

## 1.3 諸概念

DIOSA/XTP メモリキャッシュの諸概念について説明します。

- メインキー
- 論理表と物理表
- メモリデータパーティション (MAP)
- ハッシュ値
- レプリケーショングループ

### (1) メインキー

メインキーは、TAM 表の分散を決定するためのキー値です。

種別とデータ内の主キーで構成することを推奨します。

### (2) 論理表と物理表

論理表とは、TAM 分散を行うために TAM 表（物理表）を分割して定義し、その分割された TAM 表を論理的に 1 つにまとめたものを指します。

物理表は、TAM に定義される表の単位になります。

論理表は、アプリケーションが指定する表名になります。ユーザアプリケーションは、物理表を意識せずに分散配置された TAM にアクセスすることが可能です。

### (3) メモリデータパーティション (MAP)

データを格納するメモリ領域の区画をメモリデータパーティション（以下 MAP）と呼びます。

MAP はインメモリサーバのアクセス単位です。

論理表をパーティション分割して MAP を分けるケース（論理表：物理表＝1：n）とカテゴリ別毎に MAP を分けるケース（論理表：物理表＝1：1）の 2 種類があります。また、MAP 分散を行わないことも可能ですが、分散しない場合でも MAP として定義する必要があります。

1 トランザクション処理で更新できるのは 1MAP のみです。

MAP の識別子を MAPID と呼びます。

### (4) ハッシュ値

MAP の決定を行うために必要な情報がハッシュ値です。

利用者が用意するハッシュ関数により、メインキーからハッシュ値に変換します。

メモリキャッシュは、ハッシュ関数から返却されたハッシュ値を元に MAP を決定します。

MAP とハッシュ値の関係は 1:n であり、環境定義で定義します。

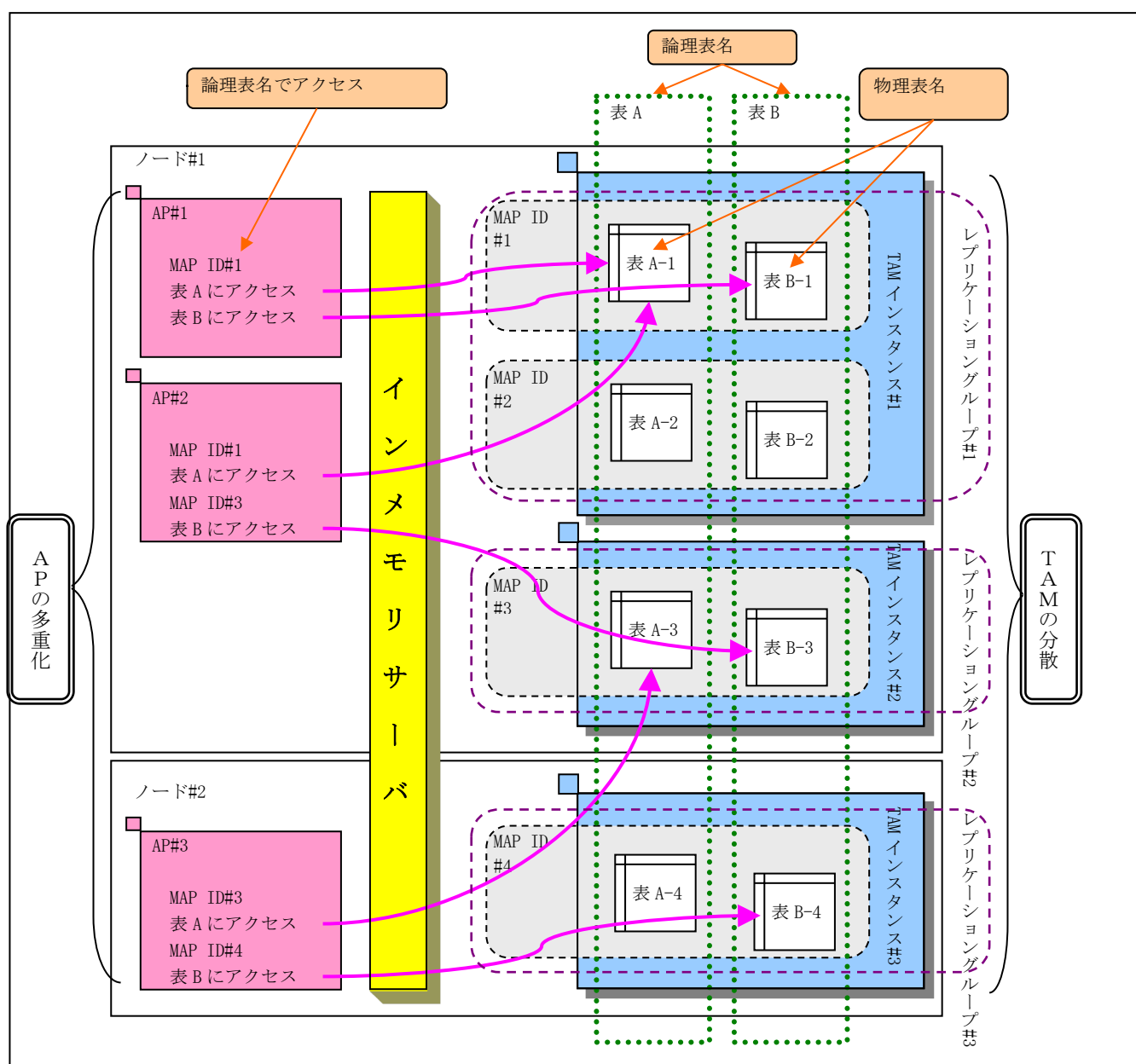
### (5) レプリケーショングループ

マスタインスタンスと、そのメモリテーブルをレプリケーションする複数のスレーブインスタンスによって構成される TAM インスタンス群をレプリケーショングループと呼びます。

レプリケーショングループと MAP の関係は、1:n であり、環境定義で定義します。

レプリケーショングループの識別子をレプリケーショングループ ID と呼びます。





## 1.4 機能構成

メモリキャッシュは、以下の機能より構成されます。

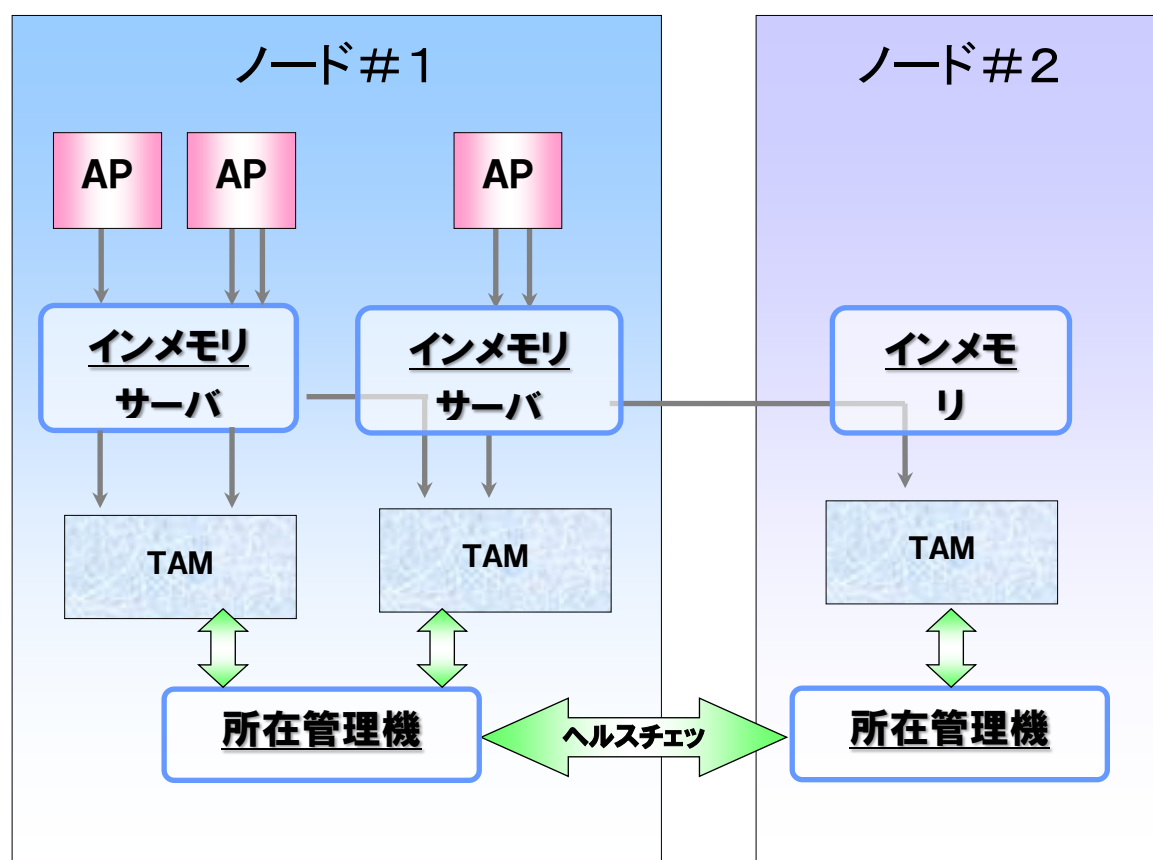
- インメモリサーバ

利用者アプリケーションと TAM との間に位置し、利用者アプリケーションから TAM へのアクセス要求を仲介する役割を担います。

TAM への実アクセスを行うアクセスサーバと、他ノードに配置されたデータへのアクセス要求を転送するためのブリッジサーバにより構成されます。

- インメモリサーバ所在管理機能

分散配置されたインメモリサーバおよび TAM の所在情報を管理します。



## 第2章 機能

本章では、メモリキャッシュ機能が提供する各機能内容について記述します。

### 2.1 インメモリサーバ

インメモリサーバは、アクセスサーバとブリッジサーバで構成され、多重化されたアプリケーションからのアクセス、および、表の所在を意識せずに全ての OLTP ノードからのアクセスを可能とするために必要となる機能を提供します。

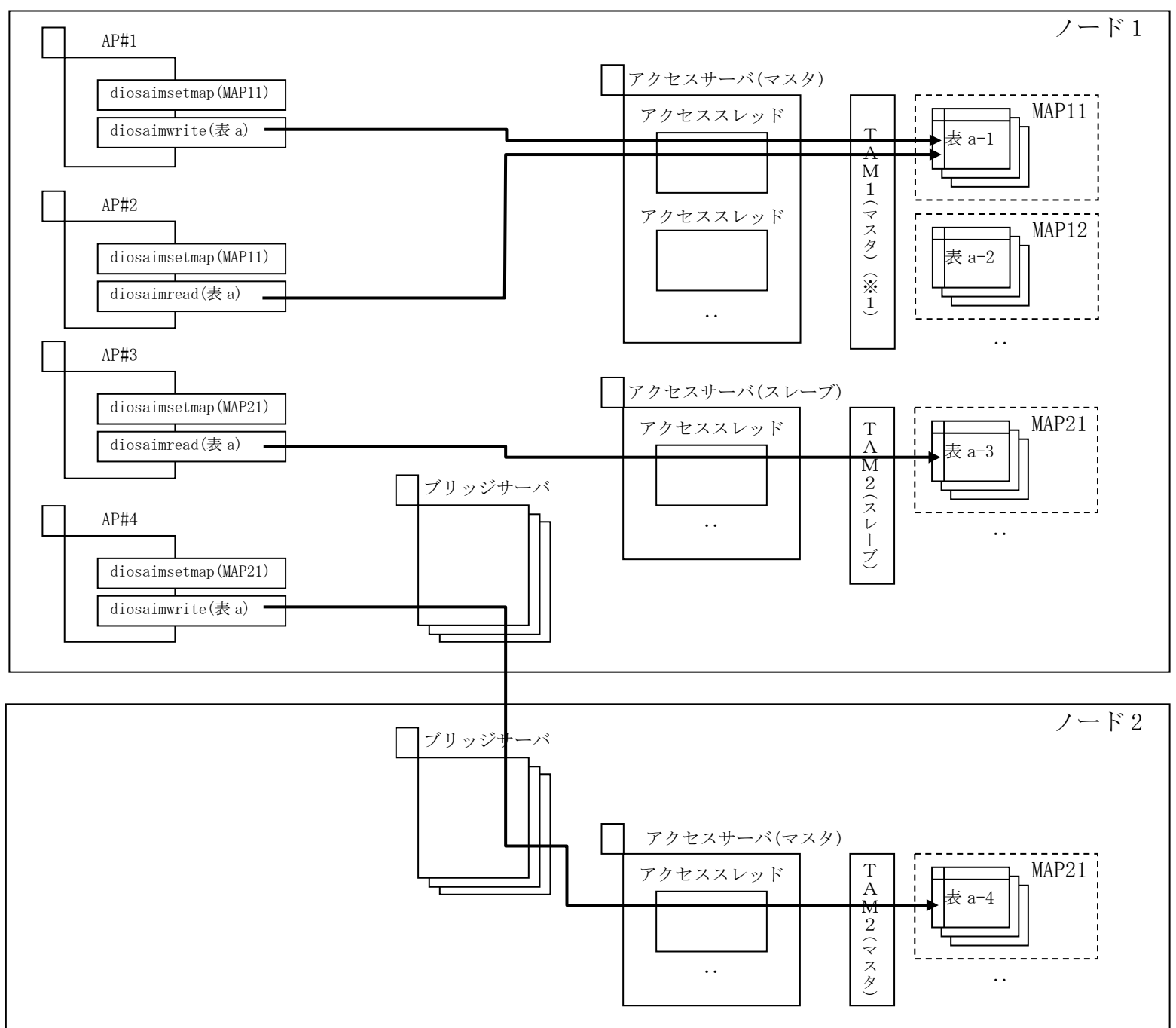
- 各種アクセスを行うための API
- レコード排他制御
- デッドロック検出機構
- 可変長レコードアクセス機能
- グループコミット機能
- 表オープン／クローズ機能
- アクセス統計情報とアクセスログ情報照会機能

## 2.1.1 アクセスサーバとブリッジサーバ

アクセスサーバは、多重化されたアプリケーションからのアクセスを実現するために存在するサーバであり、TAM インスタンスごと(マスタとスレーブごと)に起動されます。アクセスサーバは、MAP ごとにスレッド(アクセススレッドと呼称)を起動するマルチスレッド構造であり、アクセススレッド単位に起動、停止することが可能です。アクセススレッドが起動されて、その MAP へのアクセスが可能となります。

ブリッジサーバは、他ノードに存在する表へのアクセスを実現するために存在するサーバです。他ノードへのアクセス発生頻度に応じて、ブリッジサーバの多重化を図れるよう、各 OLTP ノードに最大 16 多重まで起動することができます。ブリッジサーバが起動されて、他ノードへのアクセスが可能となります。但し、ブリッジサーバが起動していなくとも、自ノード内におけるアクセスは可能です。

なお、アプリケーションからアクセスサーバへのアクセス要求、またはブリッジサーバを経由してアクセス要求を行う際、IMS キューを介して電文の送受信を行います。



※1 TAM インスタンスを表す

## 2.1.2 各種アクセスを行うための API

アプリケーションは、以下のアクセス API を使用して、表が存在するノードを意識することなく、表に対するレコードアクセスを行うことができます。

カテゴリ	アクセス種別	API 名	アクセス先
参照系	キー指定レコード読込(ロックなし)	diosaimreadl	マスタ、スレーブアクセス可
	複数レコード読込	diosaimread	マスタ、スレーブアクセス可
更新系	キー指定レコード読込(ロックあり)	diosaimreadl	マスタのみアクセス可
	レコード追加	diosaimwrite	マスタのみアクセス可
	レコード更新	diosaimrewrite	マスタのみアクセス可
	キー指定レコード削除	diosaimdeletexl	マスタのみアクセス可
	レコード削除	diosaimdelete	マスタのみアクセス可
	全レコード削除	diosaimtruncate	マスタのみアクセス可
トランザクション制御	コミット	diosaimcommit	マスタのみアクセス可
	ロールバック	diosaimrollback	マスタのみアクセス可
その他	アクセス先 MAP 宣言	diosaimsetmap	—
	アクセス先 MAP 取得	diosaimgetmap	—

＜レコードキーについて＞

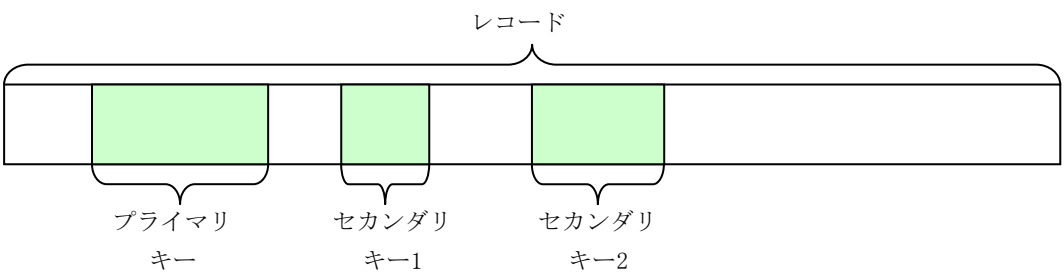
表ごとに、プライマリキーとセカンダリキーを設定することができます。

プライマリキーは、表内においてレコードを一意に管理するためのキーであり(キー値の重複は不可)、必ず、設定する必要があります。なお、キーの値は、レコード更新時に変更することはできません。

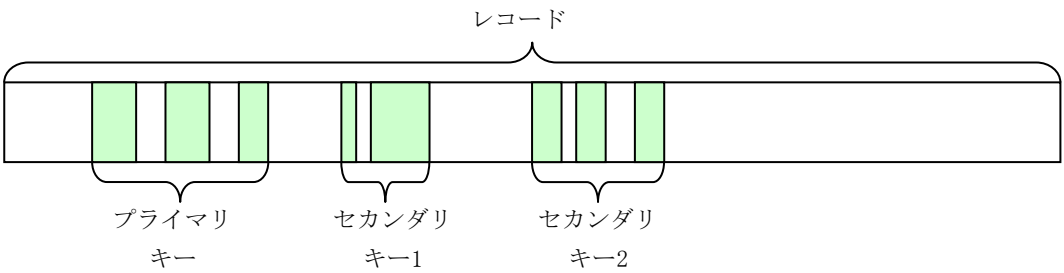
セカンダリキーは、効率的にレコードの検索を行うためのキーであり(キー値の重複は可)、任意に設定することができます。設定できる数は、0～62 個です。複数のセカンダリキーを設定した場合、セカンダリキーに優先順位はありません。なお、キーの値は、レコード更新時に変更することができます。

プライマリキーとセカンダリキーとも、以下の通り、連続領域、不連続領域のいずれの形式で設定することができます。

＜連続領域による設定＞



＜不連続領域による設定＞

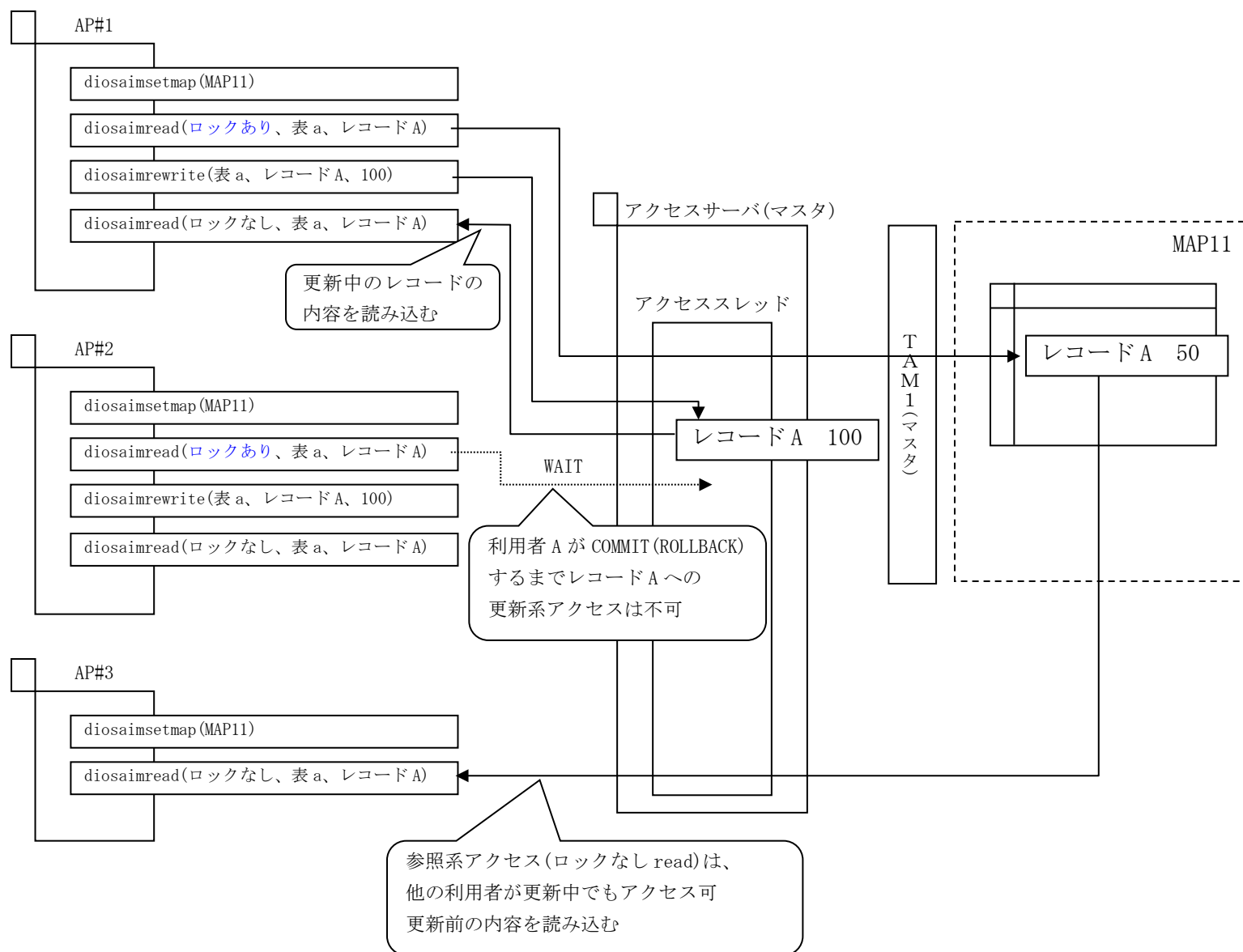


## 2.1.3 レコード排他制御

インメモリサーバでは、レコード単位の排他制御を行います。

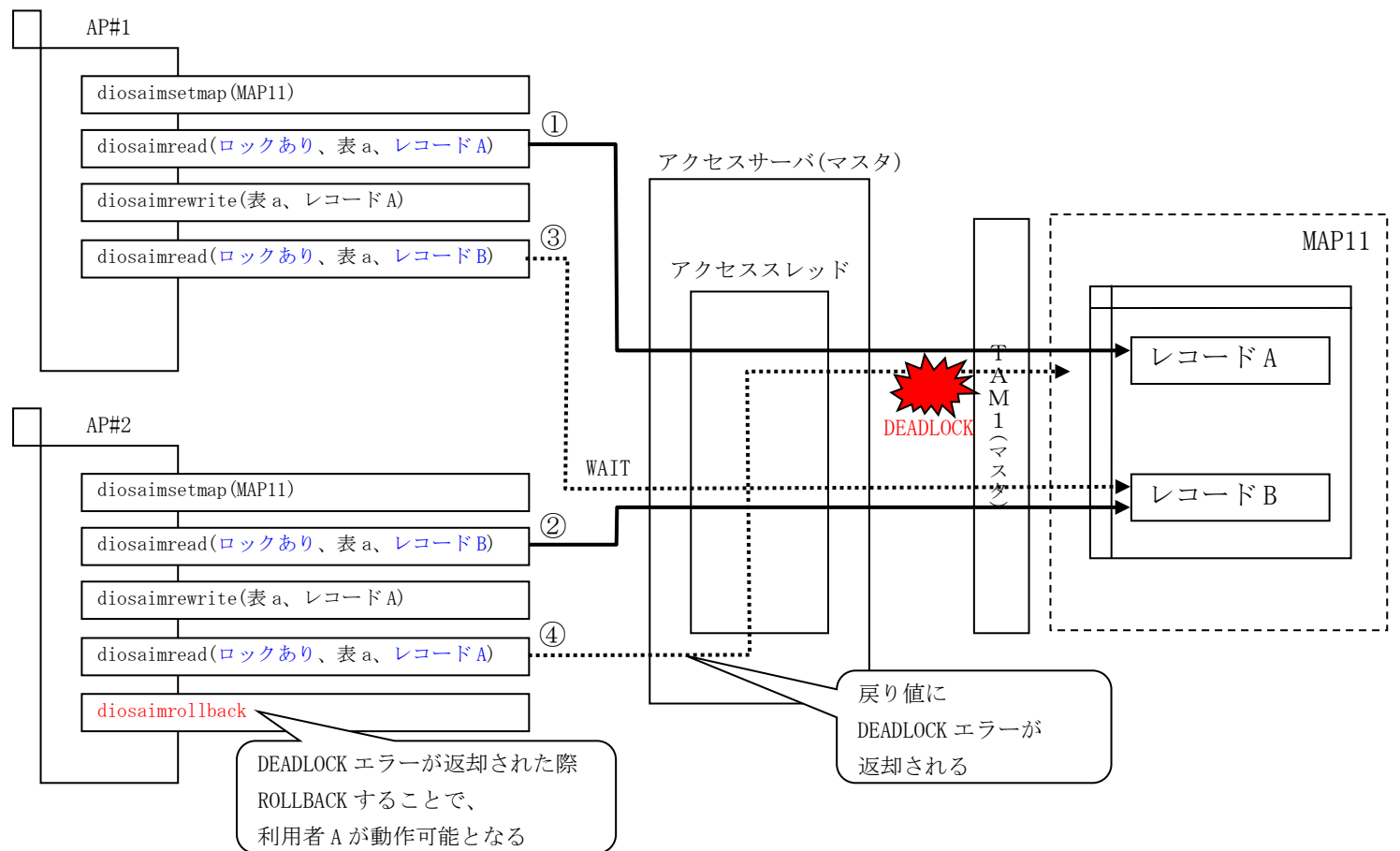
複数のアプリケーションから同一表の同一レコードに対して、更新系のアクセス要求を行われた場合、最初にアクセス要求を行ったアプリケーションがコミット(またはロールバック)要求を行うまで、後続のアプリケーションは、そのレコードに対するアクセスを行うことができません。この時、最初にアクセス要求を行ったアプリケーションがコミット(またはロールバック)要求を行うまで待ち合わせるか、待ち合わせずにエラー終了するかをAPIのパラメータで指定することができます。

なお、更新系のアクセス要求を行っているアプリケーションがいる時に、参照系のアクセス要求は、待ち合わせすることなく、並列にアクセスすることができます。この時、参照系のアクセス要求を行ったアプリケーションは、更新前のレコードを読み込むことになります。



## 2.1.4 デッドロック検出機構

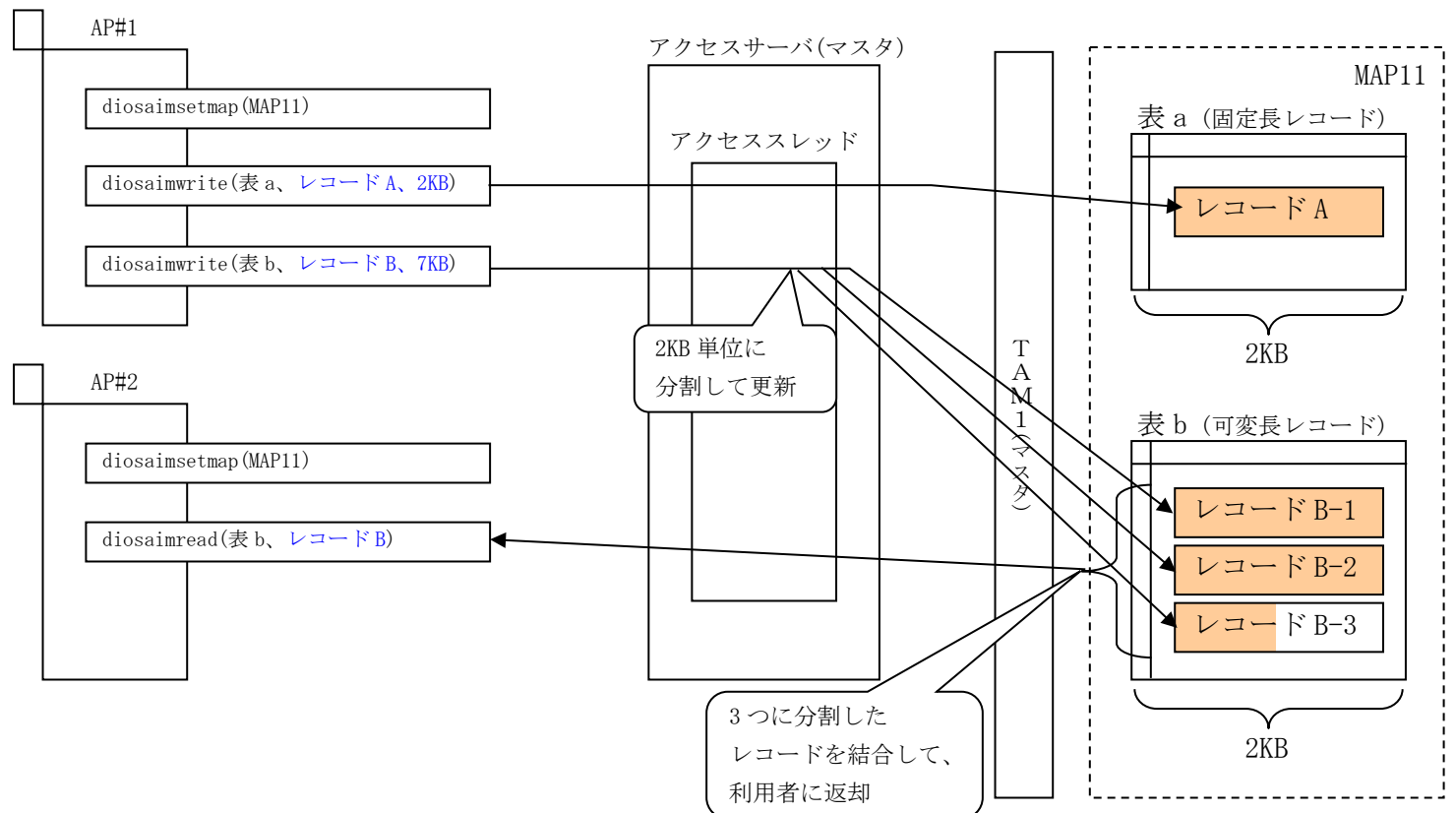
インメモリサーバでは、レコード単位の排他制御を行うため、アプリケーションの造りによってデッドロックが発生する可能性があります。アプリケーションからのアクセス要求により、デッドロックを検出した際は、デッドロックが発生したことを戻り値として返却し、アプリケーションに通知します。デッドロックエラーが返却されたアプリケーションは、ROLLBACK を実行することで、デッドロック状態を解消することができます。



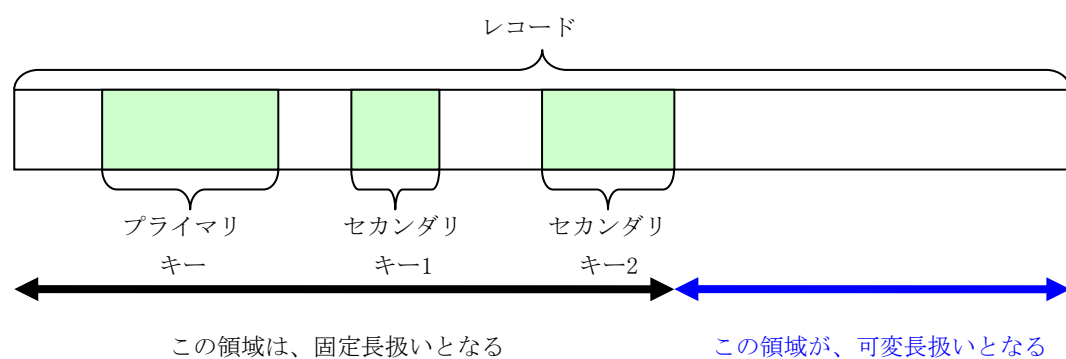
## 2.1.5 可変長レコードアクセス機能

TAM では、固定長レコードのみ扱うことができます。このため、レコードサイズが処理によって変動する表については、インメモリサーバが TAM のレコードサイズに従って、アプリケーションから要求されたレコードを分割して表へ書き込み、また分割したレコードを表から読み込んで結合した形でアプリケーションへ返却する可変長レコードアクセス機能を提供します。これにより、アプリケーションは、表に分割格納されていることを意識することなく、固定長レコードと同様、1レコードとしてアクセスすることが可能となります。

表ごとに、固定長レコード、可変長レコードのいずれのレコード形式かを指定します。その指定されたレコード形式に従って、インメモリサーバは、レコードの扱い方を切り替えます。



なお、可変長にできる領域は、レコード内において、最後に存在するキー(プライマリキー、またはセカンダリキー)より後ろに存在する領域となります。





## 2.1.6 グループコミット機能

マスタインスタンスとスレーブインスタンスでレプリケーショングループが構成されている場合、コミット時に、マスタインスタンスとスレーブインスタンス間での同期処理を行うため、コミットの処理時間が大きくなります。このため、インメモリサーバでは、複数のアプリケーションからのコミット要求を一つにまとめて、TAMへコミット要求を行う機能を提供します。

グループコミット機能として、以下の内容を指定することができます。

- コミット要求数
  - 1 コミットとしてグループ化するコミット要求数を指定します。
- コミット要求保留時間
  - コミット要求数に満たない場合に、コミット要求の到着を待ち合わせる(保留にする)時間を指定します。
  - この時間が経過するとコミット要求数に満たない状態でもコミットを行います。
- 滞留電文チェック有無
  - コミット要求数に満たない場合に、IMS キューに滞留している電文をチェックするか否かを指定します。
  - チェックする場合は、滞留している電文がなくなった時点で、コミット要求数に満たなくても、その時点で要求のあったコミット要求をグループ化します。チェックしない場合は、コミット要求保留時間までコミット要求の到着を待ち合わせます。

## 2.1.7 表オープン／クローズ機能

運用中に、アクセスサーバに対し、表のオープン、またはクローズ指示を行うコマンドを提供します。

表は、アクセスサーバ起動時に、自動的にオープンしますが、運用開始後、表に対するメンテナンスを行う場合に、アクセスサーバから表を切り離すためのクローズ要求、そして、メンテナンス完了後に、アクセスサーバへ表を取り込むようオープン要求を行うことを可能としています。

なお、クローズした表に対するアクセス要求は、全てエラーとなるため、アプリケーション側でクローズ中にアクセス要求を行わないような対応が必要です。

本コマンドは、ノードごとに、そして全ノードに実行する必要があります。本コマンドにより変更したオープン／クローズの状態は、アクセスサーバが何らかの原因で再起動になった場合は引き継がれますが、一旦、運用を停止した場合(マスタのアクセスサーバをコマンドで停止)は引き継がれず、次の運用開始時に、自動的にオープンされます。

2.1.8 性能情報採取

インメモリサーバでは性能解析をするための情報として以下の3つの情報を採取可能です。

(1) アクセス統計

採取可能な情報	<ul style="list-style-type: none"><li>サーバプロセス (MAP 単位) の稼働率</li><li>MAP 単位のロック待ち、デッドロックの検出回数</li><li>アクセス要求種別ごとのサーバプロセスでの処理時間 (平均、最小、最大)</li></ul>
主な使用用途	<ul style="list-style-type: none"><li>ロック待ちやデッドロックが不必要に発生していないかを確認する</li><li>処理時間が大きいアクセス要求を特定する</li></ul>
採取可否の設定方法	<ul style="list-style-type: none"><li>環境定義 IMENV 節 MAP 項、または diimmod コマンドで設定。詳細は環境定義リファレンスを参照</li></ul>
照会方法	<ul style="list-style-type: none"><li>diimeprfref コマンドで照会可能です。</li></ul>

(2) アクセスログ

採取可能な情報	<ul style="list-style-type: none"><li>1 アクセス要求ごとのサーバプロセスでの処理開始時刻、処理完了時刻</li><li>アクセス要求元の情報 (ノード ID、プロセス ID、スレッド ID)</li></ul>
主な使用用途	<ul style="list-style-type: none"><li>サーバプロセスの処理のどの部分で処理が遅延しているかを確認する</li></ul>
採取可否の設定方法	<ul style="list-style-type: none"><li>環境定義 IMENV 節の ACCESSLOG 項、または、diimmod コマンドで設定</li></ul>
照会方法	<ul style="list-style-type: none"><li>diimeprfref コマンドで照会可能です。</li></ul>

(3) API 性能情報

採取可能な情報	<ul style="list-style-type: none"><li>アクセス要求種別ごとのアプリケーションプロセス (または C0) での処理時間 (平均、最小、最大)</li><li>グループコミットの結果</li><li>ロック待ち、デッドロックをした場合の相手プロセスの情報</li></ul>
主な使用用途	<ul style="list-style-type: none"><li>アクセスログと合わせて確認することで、アプリケーション (C0)、サーバプロセス間の通信部分のオーバーヘッド (CPU 割り当て待ち) を確認する</li></ul>
採取可否の設定方法	<ul style="list-style-type: none"><li>(ユーザアプリケーションプログラムの場合) diosatrinit で指定します。</li><li>(C0 の場合) 環境定義 COCENV 節の TRANS 項で設定、または、dicocmod コマンドで設定。</li></ul>
照会方法	<ul style="list-style-type: none"><li>(ユーザアプリケーションプログラムの場合) diosaimgetperf() で情報を取得し、printf などによる出力をアプリケーションに作り込む必要があります。diosaimgetperf() の呼び出しタイミングについては 3.3.1 の図を参照してください。</li><li>(C0 の場合) 稼働統計機能のコマンド (diopsedit, diopsflush, diopsgather, diopsmrg) で照会可能です。C0 で diosaimgetperf() を呼び出す必要はありません (呼び出した場合 DIOSA_EFUNCNAV (-34) が返却されます)。</li></ul>

# 2.2 インメモリサーバ所在管理

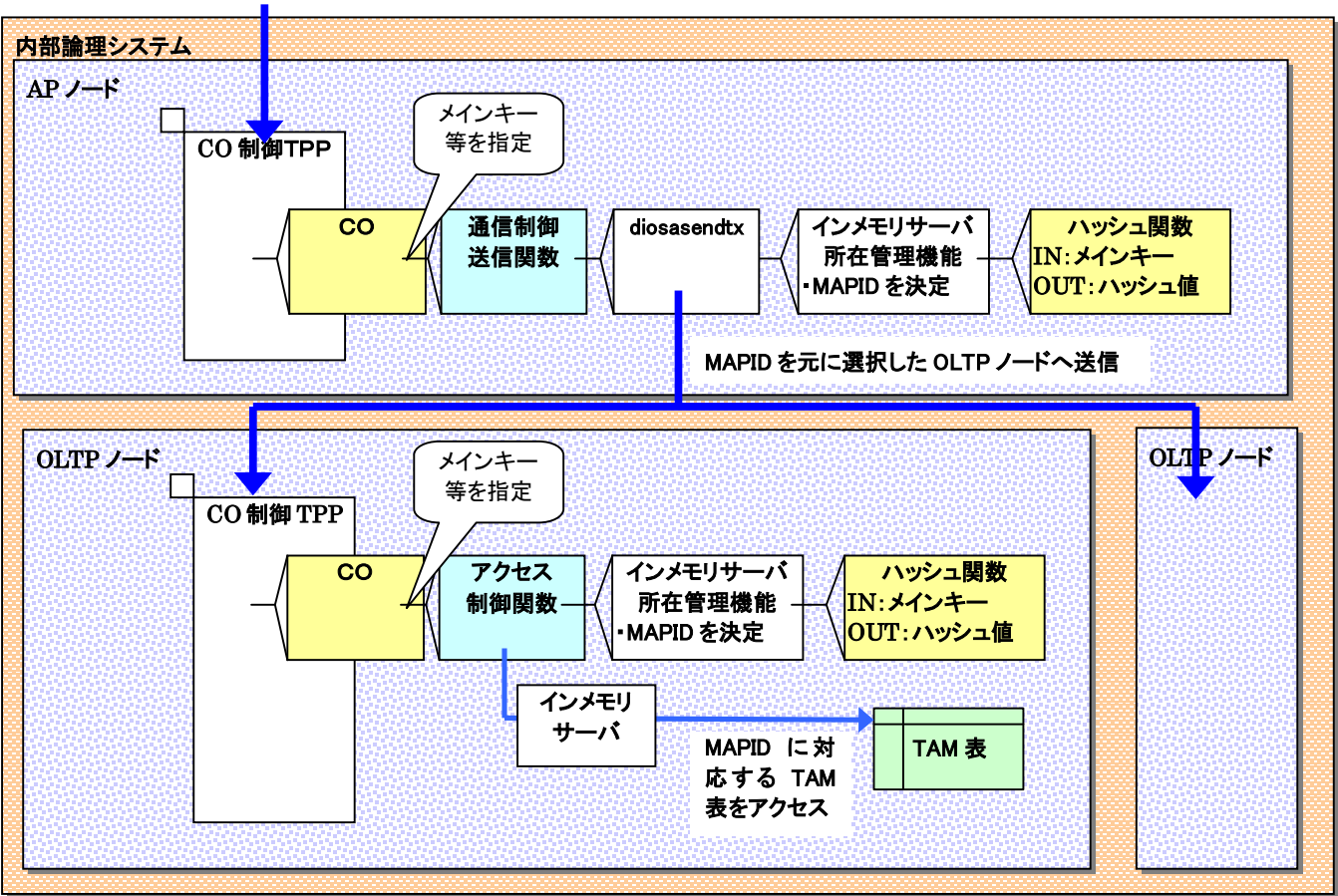
インメモリサーバ所在管理機能（以下、IMS 所在管理機能）は、分散配置されたインメモリサーバおよび TAM の所在情報を管理し、以下の機能を提供します。

- 分散配置された TAM のマスタが存在するノードへの振分機能
- TAM およびインメモリサーバの起動／停止制御機能
- 障害検出とマスタ切替制御
- コマンドによるマスタ切替機能

## 2.2.1 分散配置された TAM のマスタが存在するノードへの振分機能

### (1) メインキーから MAP への変換機能

TAM へのデータ振分は、利用者が提供するハッシュ関数にてメインキーよりハッシュ値を算出し、このハッシュ値により該当 MAP を決定します。決定した MAP をインメモリサーバに渡すことにより、対応する TAM 表にアクセスします。



### (2) AP ノードから OLTP ノードへの振分

入力情報としてメインキーを指定された場合、AP ノードはメインキーを使用し、対応するキー値の TAM 表（マスタ）が存在する OLTP ノードを選択し、OLTP ノードへ電文送信を行います。

入力情報としてメインキーを指定されない場合、AP ノードは共有メモリ上で管理している OLTP ノード情報一覧をルーティング制御に返却します。この際返却する OLTP ノード情報には、そのノード上にマスタ TAM が存在するかどうかの情報が含まれており、ルーティング制御が正常なマスタの TAM インスタンスが存在する OLTP ノードを選択し、OLTP ノードへ電文送信を行います。

### (3) OLTP ノードでの再振分

AP ノードから OLTP ノードへ電文を送信している間にマスタ TAM の切替が起こった場合や、AP ノードが存在しないシステムでは、IMS ブリッジサーバ経由でマスタ TAM が存在する OLTP ノードへアクセスすることが可能です。OLTP ノードでは、インメモリサーバが決定された MAP の物理表へのアクセスを行います。

## 2.2.2 TAM およびインメモリサーバの起動／停止制御機能

起動／停止制御機能は、インメモリサーバ所在管理デーモン（以下、IMS 所在管理デーモン）の起動・停止、及び共有メモリの生成・削除を行います。OLTP ノードでは、共有メモリの状態に応じて、TAM インスタンス、インメモリサーバの起動・停止も行います。

- 起動機能

IMS 所在管理デーモンの起動、共有メモリの生成を行い、未起動状態の TAM およびインメモリサーバを起動します。展開済みの共有メモリの内容に従い、TAM インスタンス、インメモリサーバの起動および共有メモリのインメモリサーバ起動状態更新を行います。また、インメモリサーバ起動済みの場合は、インメモリサーバに通知を行い、追加された MAP のアクセススレッド起動要求を行うことも可能です。

COLD 起動の場合は、環境定義情報を使用し、共有メモリ上にテーブルを生成します。また、引き継ぎファイルにテーブルの情報を保存します。

WARM 起動の場合は、引継ぎファイルの定義情報を使用して共有メモリ上にテーブルを生成します。

テーブル生成後、IMS 所在管理デーモンを起動します。IMS 所在管理デーモンは、他ノードで起動済みの IMS 所在管理デーモンから、TAM のマスタ・スレーブ配置等の情報を取得し、自ノードの共有メモリ情報を最新化します。これにより、ノード障害等でマスタ切替が行われた場合でも、切替後の状態で該当ノードを復旧させることが可能となります。

IMS 所在管理デーモンは、最新化された配置情報に従い、TAM インスタンスおよびインメモリサーバの起動を行います。ただし、起動オプションでインメモリサーバを起動しないようにすることも可能です。

起動処理完了後、他ノード上の IMS 所在管理デーモンへ起動通知を行います。

また、TAM の更新ログ機能利用時は、TAM の更新ログ起動も本機能が自動的に行います。ただし、TAM の更新ログ起動は warm モードで行いますので、初回起動時は、TAM のコマンドを利用して TAM の更新ログ起動を行ってください。

- 停止機能

DIOSA/XTP 稼働中に任意のノードの TAM およびインメモリサーバを停止し、IMS 所在管理デーモンの停止、共有メモリ情報の削除を行います。ノード上の TAM が全てスレーブであれば IMS 所在管理デーモンは停止処理を受け付けることが可能です。他ノード上の IMS 所在管理デーモンと情報を同期しながら、スレーブ TAM のレプリケーション切り離し、インメモリサーバ、TAM インスタンスの停止を行います。その後、IMS 所在管理デーモンの停止、共有メモリ情報の削除を行います。

ノード内にマスタ TAM が存在している場合でも強制的にノード停止を行うことが可能です。ただし、この場合、マスタ TAM の停止は行いません。マスタ TAM の停止が必要な場合は、利用者が TAM のコマンドを使って停止してください。

また、TAM の更新ログ機能利用時は、TAM の更新ログ停止も本機能が自動的に行います。

## 2.2.3 障害検出とマスタ切替制御

障害検出機能は、TAM 障害、インメモリサーバ障害および OLTP ノード障害を検出します。マスタ切替制御機能は、障害を検出した場合に、自動的にマスタ切替等の復旧処理を行います。ただし、環境定義の IMENV 節 COMMON 項 SWITCH パラメータに MANUAL を定義した場合は、障害検出時のマスタ切替は自動的には行いません。

TAM の更新ログ機能を利用する場合には、障害検出時の自動マスタ切替制御は利用できません。

- TAM 障害検出

TAM インスタンスを監視し、TAM 障害を検出します。

環境定義により、一定間隔で TAM の死活監視を行うことが可能です。また、インメモリサーバ経由の TAM アクセスで、マスタ切替が必要な障害が発生した場合や、スレーブへのレプリケーション異常が発生した場合も、TAM 障害とみなします。

自動切替ありの環境定義で閉塞等の異常状態が検出された TAM がマスタであった場合、マスタ切替を指示します。スレーブの場合は、各ノードへスレーブ障害通知を送信し、スレーブ切り離しを行います。

自動切替なしの環境定義の場合は、障害検出のメッセージ出力とステータス変更を行います。この状態では、マスタ昇格コマンドを実行することにより、マスタ切替を行うことが可能です。

- インメモリサーバ障害検出

インメモリサーバを監視し、インメモリサーバ障害を検出します。

IMS アクセスサーバのプロセス障害は、ソケットの監視で行います。また、一定間隔でキューへの滞留件数チェックによる IMS アクセスサーバのストールを監視することも可能です。プロセス停止を検出した IMS アクセスサーバがマスタの場合は、マスタ切替もしくは、プロセス再起動を指示します。スレーブの場合は、スレーブ障害通知の発行もしくは、プロセス再起動を指示します。障害検出時の処理(マスタ切替/プロセス再起動等)の選択は、IMENV 節の環境定義によって定義されます。

また、環境定義によって、一定間隔で IMS ブリッジサーバのプロセス死活監視を行うことが可能です。プロセス停止を検出した場合、IMS ブリッジサーバの再起動を試みます。自動切替ありの環境定義で全 IMS ブリッジサーバが停止した場合、マスタ切替を行います。

- OLTP ノード障害検出

AP ノードから OLTP ノードへ定期的に電文を送信し、正常応答が送信されるかどうかで OLTP ノードの状態を判別するヘルスチェック処理を行い、OLTP ノード障害を検出することが可能です。

ヘルスチェック処理で OLTP ノード障害を検出した場合、自動的にマスタ切替処理を行い、正常稼動している OLTP ノードにマスタ TAM を切り替えます。

また、T パス管理機能と連携して、T パス障害を検出することが可能です。

AP ノードにてノード障害、及び T パス障害を検出した場合、障害となった OLTP ノードから、ノード番号昇順のラウンドロビンで選択した OLTP ノードにノード障害通知を送信します。OLTP ノードは、全 AP ノードからノード障害通知を受けた場合、障害と判定します。ただし、環境定義により、一部の AP ノードからのノード障害通知を受信した時点で障害と判定させることも可能です。自動切替ありの環境定義でノード障害や T パス障害を検出した場合、マスタ切替を自動的行います。

## 2.2.4 コマンドによるマスタ切替機能

### (1) マスタ切替

ノードの保守など、計画的にノードを停止したい場合は、本コマンドにより停止するノードから別ノードへマスタを切り替えた後に、メモリキャッシュの停止、DIOSA/XTP の停止を行います。環境定義の状態にマスタを戻すオプションも提供します。

本コマンドは、マスタが存在するノードで実行します。

### (2) マスタ昇格

自動切替なしの環境定義で、マスタ TAM が使用できない状態になった場合に、スレーブをマスタに昇格させるためのコマンドを提供します。

本コマンドは新マスタとしたい TAM が存在するノードで実行します。

TAM の更新ログ機能を利用する場合は、マスタ昇格コマンドを投入後に、tamsave、更新ログ起動、インメモリサーバ起動が必要となります。(詳細は、「5.3.5 TAM の更新ログを採取する場合の障害時マスタ切替」を参照。)

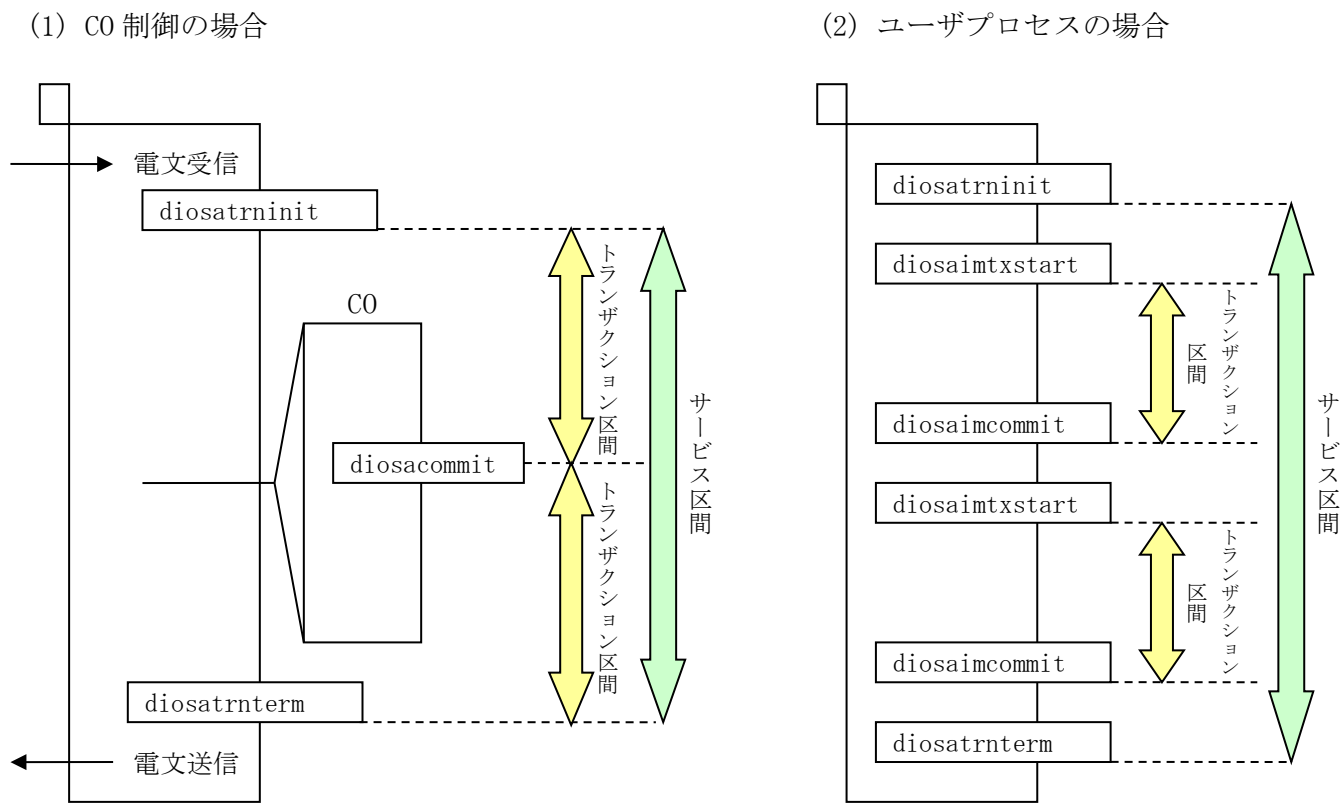
# 第3章 アプリケーションの開発

## 3.1 アプリケーションの基本構成

### 3.1.1 サービス区間とトランザクション区間

メモリキャッシュでは、DIOSA トランザクション開始 API(diosatrnrninit)から DIOSA トランザクション終了 API(diosatrnrterm) までの区間をサービスと定義し、1 コミット区間 (diosaimtxstart 呼出後から diosaimcommit(または diosaimrollback)呼出まで)をトランザクションと定義しています。

C0 制御の場合、これら API の呼び出しは、C0 呼出前後や diosacommit 実行時に C0 制御機能が行います。



メモリキャッシュへアクセスを行うアプリケーションは、サービス区間内でアクセスを行う必要があります。

以下の API で更新系のアクセス(ロックありでレコード読込も含む)を行う場合は、トランザクション区間内でアクセスを行う必要があります(以降、下記 API を更新系アクセス API と呼称する)。

diosaimread1 (排他オプションが DIOSA\_LOCK\_WAIT、DIOSA\_LOCK\_NOWAIT)、

diosaimrewrite、diosaimwrite、diosaimdelete、diosaimdeletex1、diosaimtruncate

以下の API で参照系のアクセスを行う場合は、サービス区間内であれば、トランザクション区間内であるか否かに関わらず、アクセスを行うことができます(以降、下記 API を参照系アクセス API と呼称する)。

diosaimread1 (排他オプションが DIOSA\_NOLOCK)、diosaimread

### 3.1.2 アクセス対象の MAP について

アプリケーションにおいて、アクセス要求を行う前に、アクセス先 MAP 宣言 API(diosaimsetmap)を用いて、アクセス対象となる表が所属する MAP を宣言しておく必要があります。

宣言した MAP はサービス区間内で有効であり、アクセス MAP 宣言 API でアクセス先 MAP を切り替えることで、複数の MAP に対しアクセスすることが可能です。

但し、1 トランザクション区間内で、更新系アクセス API によるアクセスが行える MAP は、一つのみであるため、複数の MAP に対し、更新系アクセスを行う場合は、トランザクションを区切る必要があります。

### 3.1.3 アクセス対象となる論理表について

表には、論理表名と論理表 ID があり、アクセス要求を行う際は、論理表 ID を各アクセス API に指定します。これは、アクセス対象となる表の検索処理を効率的に行うことを目的としています。

アプリケーションにおいて、論理表 ID 照会 API(diosaimgettblid)を用いて、論理表名に対応する論理表 ID を取得し、以降、取得した論理表 ID を指定してアクセスを行うことができます。

但し、論理表名に対応する論理表 ID は、サービス区間内でのみ有効となるため、サービス単位に論理表 ID 照会 API で論理表 ID を取得する必要があります。同一サービス区間内であれば、最初に 1 回のみ論理表 ID を取得して、以降は同じ論理表 ID を使用してアクセスすることができます。

### 3.1.4 レコードのキー値の扱いについて (Linux 版)

複数レコード読込(diosaimread)はネットワークバイトオーダーでレコードをソートするため、複数レコード読込の検索キーとなるレコードのプライマリキー、セカンダリキー部分に整数型のデータがある場合は、レコード追加(diosaimwrite)時、または、レコード更新(diosaimrewrite)時に、整数型のデータをネットワークバイトオーダーに変換した状態で格納しておく必要があります。

なお、セカンダリキーを指定した複数レコード読込時でも、セカンダリキーはプライマリキーを包含しているため、プライマリキー内に整数型のデータがある場合は、レコード追加時、または、レコード更新時にネットワークバイトオーダーに変換する必要があります。

そのため、1 件読込(diosaimread1)や複数レコード読込の検索キーを設定する際に、整数部分はネットワークバイトオーダーに変換する必要があります。また、レコードを読み込んだ後に該当キーの整数型の値を参照する場合は、必要に応じてホストバイトオーダーに変換する必要があります。

### 3.1.5 レコードの更新とレコード削除について

アプリケーションにおいて、レコード更新 API(diosaimrewrite)、またはレコード削除 API(diosaimdelete)でアクセス要求を行う場合は、事前に更新(削除)対象となるレコードに対し、レコード排他をしておく必要があります。

レコード排他は、キー指定レコード読込 API(排他オプションに DIOSA\_LOCK\_WAIT、DIOSA\_LOCK\_NOWAIT を指定した diosaimread1)、およびレコード追加 API(diosaimwrite)でアクセスすることで行われます。

上記 API で返却されたレコード情報構造体(t\_diosa\_recinfo)は、何も変更せずにそのままレコード更新 API、またはレコード削除 API に渡してください。



### 3.1.6 全レコードの削除

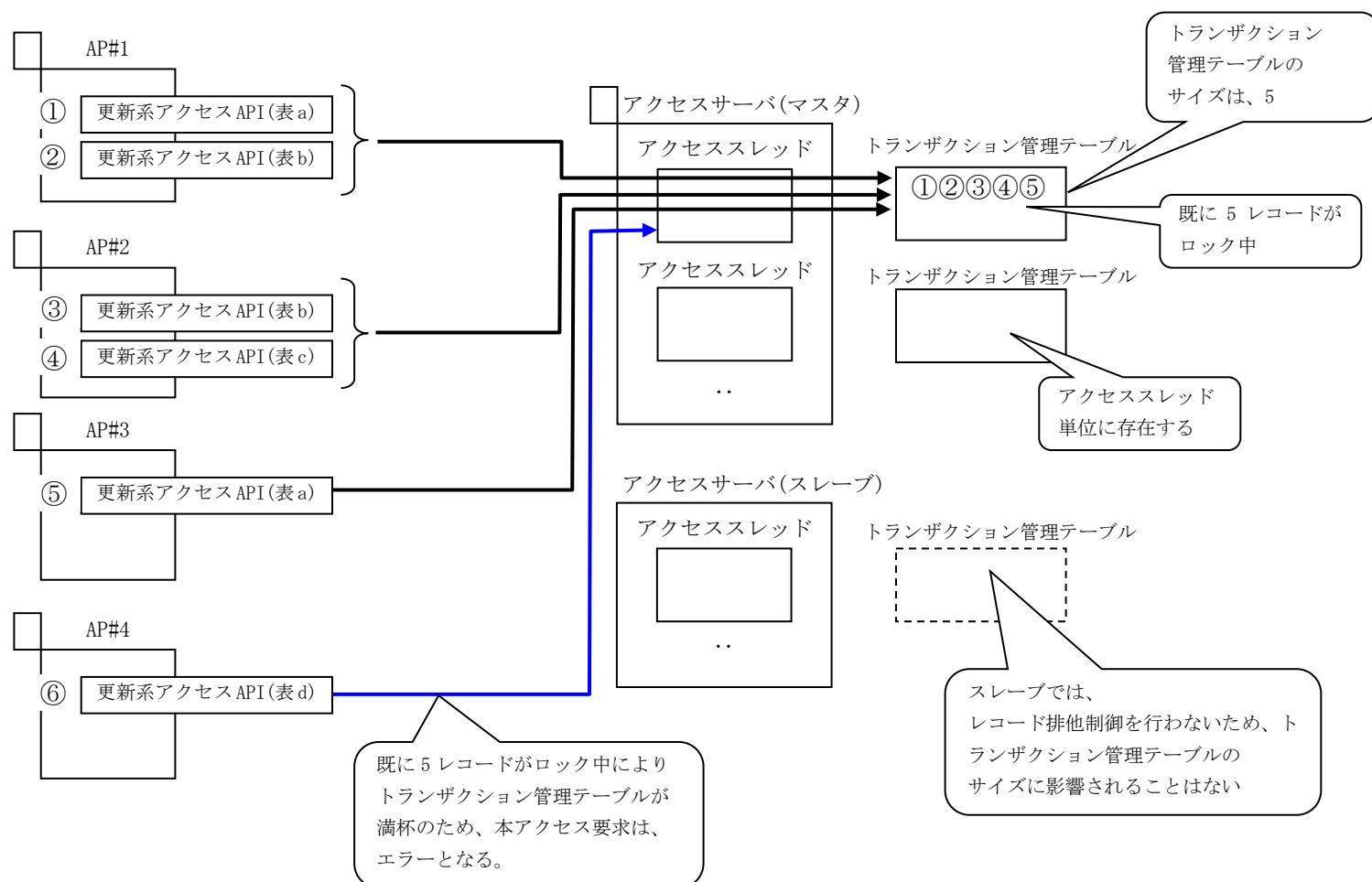
全レコード削除 API (diosaimtruncate) を用いて、表内の全レコードを削除する場合は、同一トランザクション区間内で、全レコード削除 API 以外の更新系アクセス API を使用することはできません。

同一サービス区間内で、全レコード削除 API の呼び出し後に、全レコード削除 API 以外の更新系アクセス API を使用する場合は、コミット API (またはロールバック API) を呼び出し、トランザクションを区切ることでアクセス可能となります。

### 3.1.7 更新するレコード数の上限

アクセスサーバでは、レコード排他制御を行うため、ロック中のレコードを管理しており、この管理するためのテーブルをトランザクション管理テーブルと呼びます。トランザクション管理テーブルは、MAPID 単位に存在し、テーブルサイズは、環境定義(IMENV 節 MAP 項 TXTBLSIZE パラメータ)に設定された値となります。このトランザクション管理テーブルのサイズを超えて、更新系アクセス API によるアクセス要求を受け付けることはできません。

このため、複数のアプリケーションから同時にロックされるレコードアクセス数を考慮して、トランザクション管理テーブルのサイズを設定する必要があります。



なお、更新中のレコードはコミットで確定させるため、コミット要求が行われるまで、更新中レコード(※1)をメモリ上に蓄積しています。この蓄積するための領域は、アプリケーション(スレッド)単位に存在し、領域サイズは、環境定義(IMENV 節 USERAP 項 IMQUEBUFSIZE パラメータ)に設定します。この領域サイズを超えて、更新要求を受け付けることはできません。

同一トランザクション区間内で更新するレコードサイズの合計値を考慮して、本領域サイズを設定する必要があります。

(※1) キー指定レコード読込 API (排他オプションに DIOSA\_LOCK\_WAIT、DIOSA\_LOCK\_NOWAIT を指定した diosaimread1) によるアクセスも含まれる。

### 3.1.8 戻り値について

各アクセス API においてアクセス失敗時に返却される戻り値について、以下に記載します。

戻り値により、同一サービス区間内におけるアクセスが抑止されるケース(以下、サービス抑止と呼称する)があります。サービス抑止となる戻り値が返却された場合は、(エラー原因を解決後、)新たなサービスに切替えることでアクセスが可能となります。

分類		戻り値	原因
サービス抑止にならない戻り値		DIOSA_ELOCK、DIOSA_EBUSY DIOSA_EDEADLOCK、DIOSA_NOENT DIOSA_EEXIST	API リファレンスを参照
		DIOSA_ESTATE、DIOSA_ENOINIT DIOSA_EPARAM、DIOSA_EFUNCNAV DIOSA_ECOND、DIOSA_EINVAL DIOSA_ESIZE、DIOSA_ENOBUFS	アプリケーションのプログラム誤り
サービス抑止となる戻り値		DIOSA_SWITCH(※1)	計画マスタ切替中
	サービスを切り替えて リトライすれば、アクセ スが成功する可能性が あるケース	DIOSA_ESWITCH(※1)	障害によるマスタ切替中
		DIOSA_EREADY(※2)	アクセスサーバが再起動中や環境定義動的 置換中など
	サービスを切り替えて も、原因を解消するまで はアクセス不可能なケ ース	DIOSA_EINACTIVE	アクセス対象の MAP(アクセススレッド)のマ スタが停止している
		DIOSA_BLOCK	アクセス対象の MAP が閉塞されている
		DIOSA_EDOWN	アクセス対象の MAP(アクセススレッド)のス レーブが停止している
		DIOSA_EACCES	アクセス対象の表がクローズされている、又 は更新ログ出力機能が準備できてないため 更新できない
		DIOSA_ETAM	TAM におけるエラー発生
		DIOSA_EEXTEND DIOSA_ECONNECT DIOSA_ETIMEOUT DIOSA_EOVERFLOW DIOSA_EMEM、DIOSA_ESYS DIOSA_EMSQ、DIOSA_ESOCKET DIOSA_ERROR	API リファレンスを参照

(※1) C0 制御上で動作する C0 からのアクセス要求において、DIOSA\_SWITCH、DIOSA\_ESWITCH が返却された場合は、C0 制御の電文保留機能を利用することで、マスタ切替が完了するまでアクセス要求を保留し、マスタ切替が完了した時点でアクセス処理を再開することができます。(詳細は、DIOSA\_XTP の利用の手引きを参照のこと)

C0 制御以外の場合は、マスタ切替が完了するまで、サービスを切り替えてリトライしても、DIOSA\_SWITCH(DIOSA\_ESWITCH) が返却し続けられます。

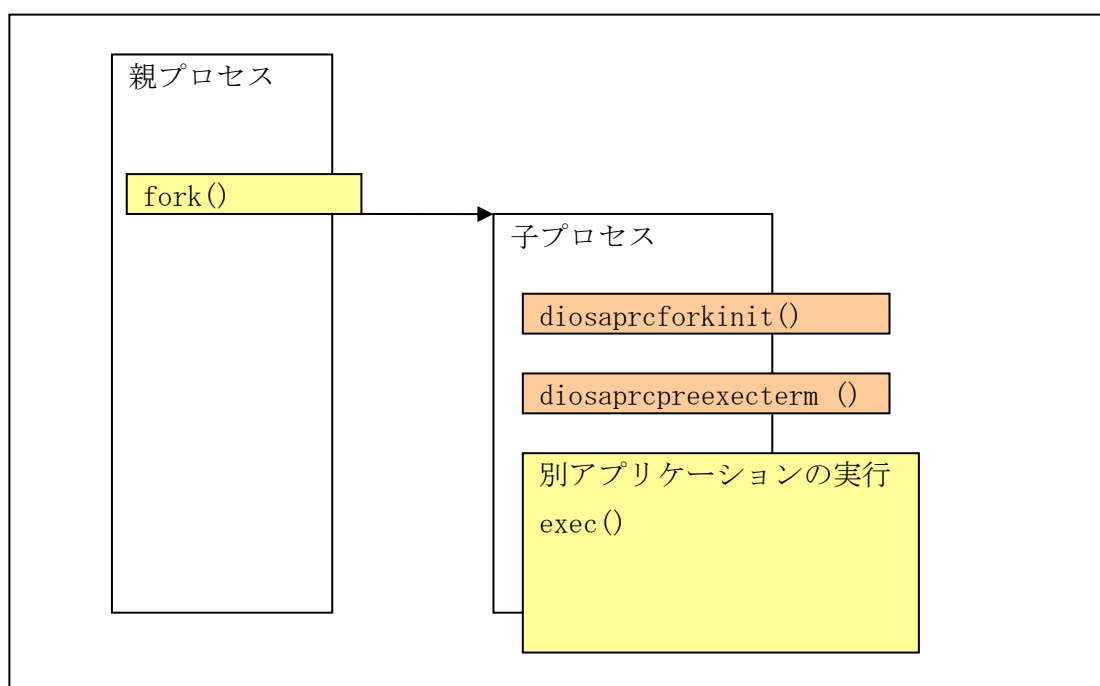
(※2) DIOSA\_EREADY が返却された場合は、サービスを切り替えてリトライすることで、タイミングによっては再度 DIOSA\_EREADY が返却される可能性があります。アクセスサーバの起動完了、もしくは環境定義の動的置換が完了すれば、アクセスが成功します。

### 3.1.9 マルチスレッドのアプリケーションプログラム

マルチスレッドのアプリケーションプログラム上で `fork` を行う場合は、スレッド間ロック情報がロック状態のまま引き継がれる可能性があるなど、問題を引き起こす可能性があります。

DIOSA の API を呼び出しているプログラムで、マルチスレッドのアプリケーションを作成する場合は、子プロセス上の `fork` 直後に `diosaprcforkinit` を呼び出しておく必要があります。

また、`fork` 後の子プロセス上で別アプリケーションの実行(`exec`)を行う場合は、`exec` 前に `diosaprcpreexecterm` を呼び出す必要があります。

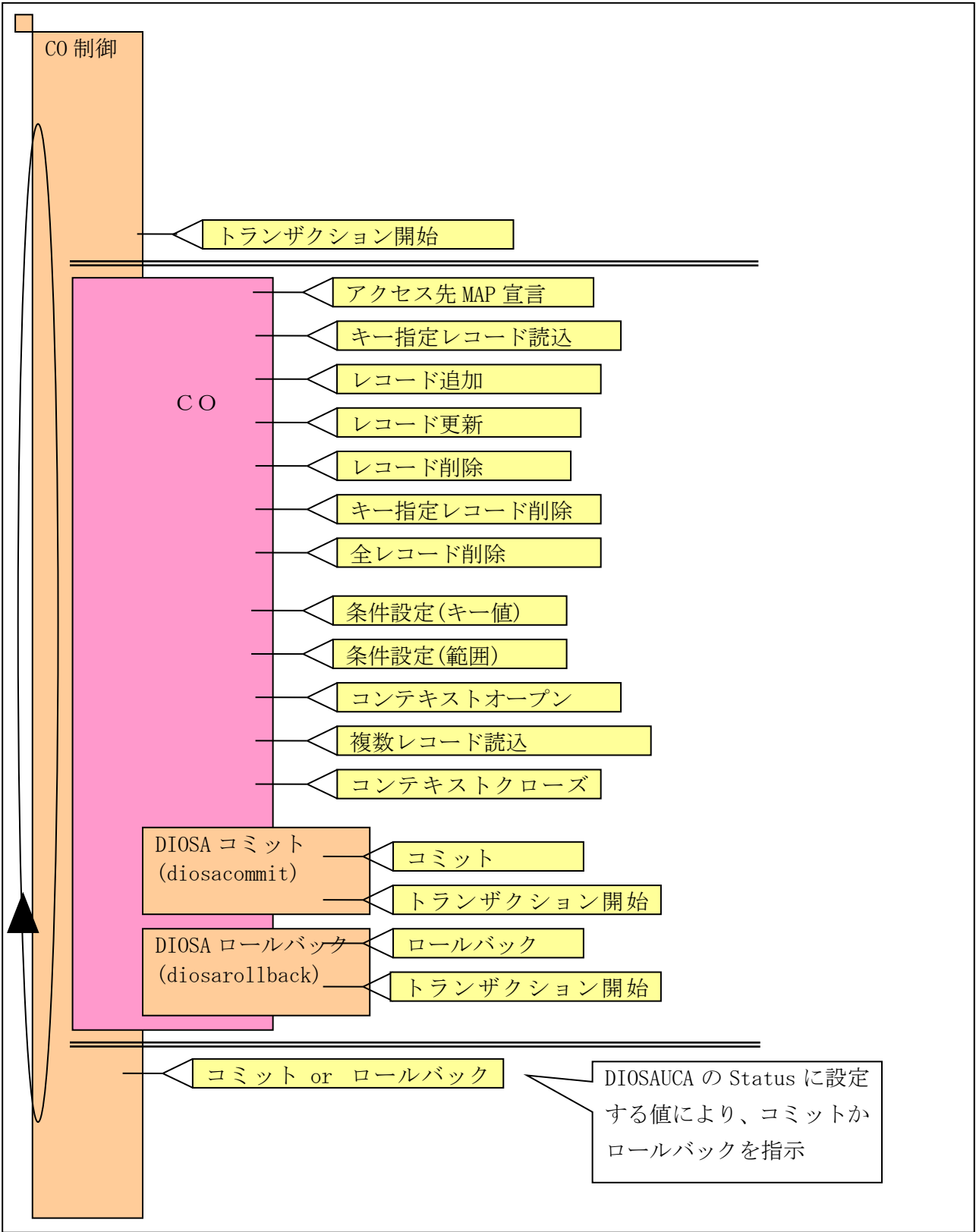


## 3.2 C0

### 3.2.1 プログラムの構造

C0 でのアプリケーションプログラムでは DIOSA プロセス初期化／終了、IM サーバオープン／クローズ、DIOSA トランザクション初期化／終了相当の処理は C0 制御機能により実行されるため、アプリケーションプログラムが実行する必要はありません。

トランザクション開始は C0 制御機能により実行されるため、アプリケーションプログラムが実行する必要はありません。diosacommit、diosarollback 実行時にもトランザクション開始を C0 制御機能が実行します。



## 3.2.2 メモリキャッシュにアクセスするプログラムの開発

### (1) レコード更新を行うサンプルプログラム

以下に複数の MAP に対してレコード追加、レコード更新を行う C0 プログラムのサンプルを記します。

```
void SAMPLEC0(t_diosa_uca *DiosaUca) {

    t_diosa_recinfo Recinfo;
    t_diosa_imcond Cond;
    char *BuffPtr;
    char BuffArea[1024];
    size_t BuffSize;
    int Ret;
    int TblId;
    int UserStatus;
    struct index2 {          /* セカンダリキー */
        char Date[8];
        char Rfu[4];
        int ReceiptNumber;
    };
    struct index2 *SecondaryKey;

    /*-----*/
    /* MAP1 の表にレコードを追加 */
    /*-----*/
    /* アクセス先 MAP 宣言 */
    Ret = diosaimsetmap( 1 );
    if(DIOSA_DONE != Ret) {
        /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
        /* 以下にアクセス先 MAP 宣言の例を記載する */
        switch(Ret) {
            case DIOSA_BLOCK:
            case DIOSA_SWITCH:
            case DIOSA_ECONNECT:
            case DIOSA_ESWITCH:
            case DIOSA_EINACTIVE:
            case DIOSA_EREADY:
            case DIOSA_EEXTEND:
                /* 別サービスであれば再実行可能なエラー */
                /* プロセス継続の異常終了を設定する */
                DiosaUca->Status = DIOSA_ST_ABORTCONT;
        }
    }
}
```

```

        return;
    case DIOSA_EPARAM:
    case DIOSA_EINVAL:
    case DIOSA_ESTATE:
        /* API の利用方法誤り、または定義に存在しない値が指定された。 */
        /* アプリケーション修正が必要な可能性があるためプロセス終了する */
        /* プロセス終了の異常終了を設定する */
        DiosaUca->Status = DIOSA_ST_ABORTSTOP;
        return;
    default:
        /* その他致命的なエラー */
        /* プロセス終了の異常終了を設定する */
        DiosaUca->Status = DIOSA_ST_ABORTSTOP;
        return;
    }
}

/* 論理表 ID 照会 */
Ret = diosaimgettblid("SAMPLE_TABLE_A", &TblId);
if(DIOSA_DONE != Ret){
    /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
    /* 詳細は 3.1.8 戻り値について を参照 */
}

/* 追加レコードの設定 */
/* レコードイメージ設定 */
strncpy(BuffArea, "SAMPLE_WRITE_RECORD", 32);
/* セカンダリキーを設定(整数型の値はネットワークバイトオーダーに変換)(Linux 版は必須) */
SecondaryKey = (struct index2*) &BuffArea[32];
memset(SecondaryKey, '¥0', sizeof(struct index2));
strncpy(SecondaryKey->Date, "20110601", 8);
SecondaryKey->ReceiptNumber = htonl(100);
...
Recinfo.RecordPtr = BuffArea;
/* 更新レコードサイズ設定 */
/* 可変長レコードの場合は取得レコードサイズからの増減可能 */
Recinfo.RecordSize = 128;

/* レコード追加 */
Ret = diosaimwrite(TblId, &Recinfo, 1, DIOSA_LOCK_WAIT);

```

```

if(DIOSA_DONE != Ret) {
    /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
    /* 詳細は 3.1.8 戻り値について を参照 */
}

/* 複数 MAP 更新時は COMMIT してから次の MAP の処理を行う */
/* コミット */
Ret = diosacommit(DIOSA_YES);
if(DIOSA_DONE != Ret) {
    /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
    /* 詳細は 3.1.8 戻り値について を参照 */
}

/*-----*/
/* MAP2 の表のレコードを更新 */
/*-----*/
/* アクセス先 MAP 宣言 */
Ret = diosaimsetmap( 2 );
if(DIOSA_DONE != Ret) {
    /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
    /* 詳細は 3.1.8 戻り値について を参照 */
}

/* 論理表 ID 照会 */
Ret = diosaimgettblid("SAMPLE_TABLE_B", &TblId);
if(DIOSA_DONE != Ret) {
    /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
    /* 詳細は 3.1.8 戻り値について を参照 */
}

/* 更新対象のレコードのロック取得 */
Ret = diosaimcondsetkey(&Cond, "SAMPLE_PKEY", 11);
if(DIOSA_DONE != Ret) {
    /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
    /* 詳細は 3.1.8 戻り値について を参照 */
}

/* レコードが格納できる領域を確保する(この例では 26byte のレコードとする) */
BuffSize = 26;
Ret = diosamalloc(NULL, NULL, DIOSA_SVCALL, BuffSize, &BuffPtr);
if(DIOSA_DONE != Ret) {

```



```

        /* プロセス終了の異常終了を設定する */
        DiosaUca->Status = DIOSA_ST_ABORTSTOP;
        return;
    }

    /* キー指定レコード読込 */
    Ret = diosaimread1(TblId, &Cond, &Recinfo, BuffPtr, BuffSize, DIOSA_LOCK_WAIT);
    if(DIOSA_DONE != Ret){
        /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
        /* 詳細は 3.1.8 戻り値について を参照 */
    }

    /* 対象レコードの更新 */
    /* 更新レコードイメージ設定(プライマリキーは変更不可) */
    strncpy(BuffArea, "SAMPLE_PKEY_UPDATE_RECORD", 32 );
    Recinfo.RecordPtr = BuffArea;
    /* 更新レコードサイズ設定 */
    /* 可変長レコードの場合は取得レコードサイズからの増減可能 */
    Recinfo.RecordSize = strlen(BuffArea);

    /* レコード更新 */
    Ret = diosaimrewrite(TblId, &Recinfo, 1);
    if(DIOSA_DONE != Ret){
        /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
        /* 詳細は 3.1.8 戻り値について を参照 */
    }

    /* 正常終了を設定する */
    DiosaUca->Status = DIOSA_ST_DONE;
    return;
}

```

## (2) レコード参照を行うサンプルプログラム

以下に複数 MAP のレコードを複数レコード取得する C0 プログラムのサンプルを記します。

```
void SAMPLEC0(t_diosa_uca *DiosaUca) {

    t_diosa_imcond Cond;
    t_diosa_recinfo RecInfo[10];
    int    CtxId;
    int    FetchedNum;
    char   *BuffPtr;
    size_t BuffSize;
    int     Ret;
    int     TblId;
    int     i;
    struct index2 {          /* セカンダリキー */
        char Date[8];
        char Rfu[4];
        int  ReceiptNumber;
    };
    struct index2 KeyMin, KeyMax;

    /*-----*/
    /* MAP1 のレコードを検索      */
    /*-----*/
    /* アクセス先 MAP 宣言 */
    Ret = diosaimsetmap( 1 );
    if(DIOSA_DONE != Ret) {
        /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
        /* 詳細は 3.1.8 戻り値についてを参照                                     */
    }

    /*
     * SG 定義上、長さ 8 バイトの文字列(YYYYMMDD)と 4 バイトの整数型のキー対する範囲指定検索を行う
     * 整数型の値はネットワークバイトオーダーで指定する必要あり (Linux 版は必須)
     */
    memset(&KeyMin, '¥0', sizeof(KeyMin));
    memset(&KeyMax, '¥0', sizeof(KeyMax));
    strncpy(KeyMin.Date, "20110601", 8);
    strncpy(KeyMax.Date, "20110601", 8);
    KeyMin.ReceiptNumber = htonl(100);
    KeyMax.ReceiptNumber = htonl(200);
    Ret = diosaimcondsetrange(&Cond, DIOSA_COND_GE, (char*)&KeyMin, sizeof(KeyMin))
```

```

        , DIOSA_COND_LE, (char*)&KeyMax, sizeof(KeyMax));

if(DIOSA_DONE != Ret){
    /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
    /* 詳細は 3.1.8 戻り値について を参照 */
}

Ret = diosaimgettblid("SAMPLE_TABLE", &TblId);
if(DIOSA_DONE != Ret){
    /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
    /* 詳細は 3.1.8 戻り値について を参照 */
}

Ret = diosaimctxopen(TblId, "index2", &Cond, DIOSA_ASCEND, DIOSA_NOLOCK, &CtxId);
if(DIOSA_DONE != Ret){
    /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
    /* 詳細は 3.1.8 戻り値について を参照 */
}

/* レコード 10 件分が格納できる領域を確保する */
BuffSize = 10240;
Ret = diosamalloc(NULL, NULL, DIOSA_SVCALL, BuffSize, &BuffPtr);
if(DIOSA_DONE != Ret){
    /* プロセス終了の異常終了を設定する */
    DiosaUca->Status = DIOSA_ST_ABORTSTOP;
    return;
}

Ret = DIOSA_DONE;
while(DIOSA_DONE == Ret){
    /* 条件に合致するものを 10 件ずつ取得する */
    Ret = diosaimread(CtxId, RecInfo, 10, &FetchedNum, BuffPtr, BuffSize);
    if((DIOSA_DONE != Ret) && (DIOSA_NOENT != Ret)){
        /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
        /* 詳細は 3.1.8 戻り値について を参照 */
    }
    if(DIOSA_NOENT == Ret){
        /* 条件に合致するレコードが一件もない */
        break;
    }
    for(i=0; FetchedNum>i; i++){
        /* 取得したレコードに対する処理を行う */

```

```

        /* 取得レコード件数分、レコード情報(RecInfo)に格納されています */
        /* レコードイメージ格納アドレスは、RecInfo[i].RecordPtr          */
        /* レコードサイズは、RecInfo[i].RecordSize でアクセスします      */
    }
}

Ret = diosaimctxclose(CtxId);
if(DIOSA_DONE != Ret){
    /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
    /* 詳細は 3.1.8 戻り値について を参照                                */
}

/*-----*/
/* MAP2 のレコードを検索      */
/*-----*/
/* アクセス先 MAP 宣言 */
Ret = diosaimsetmap( 2 );
if(DIOSA_DONE != Ret){
    /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
    /* 詳細は 3.1.8 戻り値について を参照                                */
}

/* MAP1 のレコード検索と同様にレコード取得を行う */
/* diosaimread、diosaimreadl かつ排他オプションでロックを行わない場合は */
/* 複数 MAP へのアクセスであっても diosacommit の実行は不要。          */

/* 正常終了を設定する */
DiosaUca->Status = DIOSA_ST_DONE;
return;
}

```

## 3.3 ユーザアプリケーションプログラム

本章では、ユーザアプリケーションからメモリキャッシュを利用するプログラムの開発方法について説明します。

### 3.3.1 プログラムの構造

ユーザアプリケーションを作るためには、必要な初期化処理や終了処理を適切な順番に従って実行する必要があります。

メモリキャッシュにアクセスを行う場合は、以下の処理を行う必要があります。

＜プロセス単位に実施が必要な処理＞

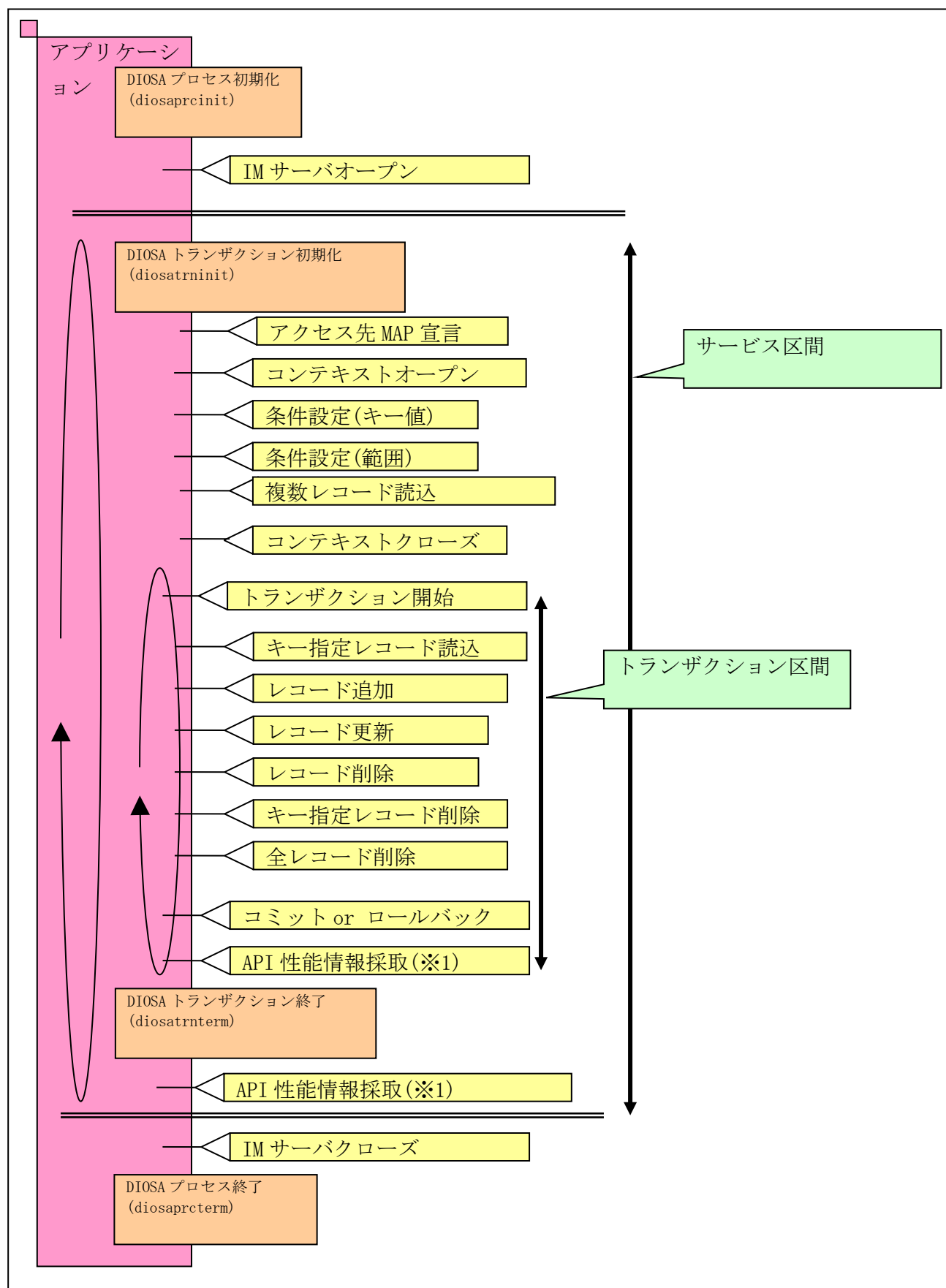
①DIOSA プロセス初期化／終了

＜スレッド単位に実施が必要な処理＞

①DIOSA スレッド初期化／終了 ※シングルスレッドの場合は不要

②IM サーバオープン／クローズ

③DIOSA トランザクション初期化／終了



※1. API 性能情報採取は任意のタイミングで呼び出しできますが、以下のいずれかの呼び出しを推奨します。

なお、C0 の場合はアプリケーション実行制御によりトランザクション区間単位で情報を採取しています。

- ・ コミット、またはロールバックの直後(トランザクション区間単位での情報採取になります)
- ・ DIOSA トランザクション終了の直前、または直後(サービス区間単位での情報採取になります)

### 3.3.2 メモリキャッシュにアクセスするプログラムの開発

#### (1) レコード更新を行うサンプルプログラム

以下に複数の MAP に対してレコード追加、レコード更新を行うプログラムのサンプルを記します。

```
int main() {

    t_diosa_recinfo Recinfo;
    t_diosa_imcond Cond;
    char  *BuffPtr;
    char  BuffArea[1024];
    size_t BuffSize;
    int    Ret;
    int    TblId;
    int    PrcEnd;
    int    Mode;
    struct index2 {                      /* セカンダリキー */
        char Date[8];
        char Rfu[4];
        int  ReceiptNumber;
    };
    struct index2 *SecondaryKey;

    /* DIOSA プロセス初期化 */
    Ret = diosaprcinit( NULL );
    if(DIOSA_DONE != Ret) {
        diosaprcterm( NULL );
        return 1;
    }

    /* IM サーバオープン */
    Mode = DIOSA_CONN_INADVANCE;
    Ret = diosaimopen( Mode );
    if(DIOSA_DONE != Ret) {
        diosaimclose();
        diosaprcterm( NULL );
        return 1;
    }

    Ret = DIOSA_DONE;
    while(DIOSA_DONE == Ret) {
```

```

/* DIOSA トランザクション初期化/終了の区間内でメモリキャッシュの API を実行し */
/* レコード操作を行う */

/* DIOSA トランザクション初期化 */
Ret = diosatrnninit( NULL );
if(DIOSA_DONE != Ret){
    break;
}

Ret = DIOSA_DONE;
while(DIOSA_DONE == Ret){
    /*-----*/
    /* MAP1 の表にレコードを追加 */
    /*-----*/
    /* アクセス先 MAP 宣言 */
    Ret = diosaimsetmap( 1 );
    if(DIOSA_DONE != Ret){
        /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
        /* 以下にアクセス先 MAP 宣言の例を記載する */
        switch(Ret){
            case DIOSA_BLOCK:
            case DIOSA_SWITCH:
            case DIOSA_ECONNECT:
            case DIOSA_ESWITCH:
            case DIOSA_EINACTIVE:
            case DIOSA_EREADY:
            case DIOSA_EEXTEND:
                /* 別サービスであれば再実行可能なエラー */
                break;
            case DIOSA_EPARAM:
            case DIOSA_EINVAL:
            case DIOSA_ESTATE:
                /* API の利用方法誤り、または定義に存在しない値が指定された。 */
                /* アプリケーション修正が必要な可能性があるためプロセス終了する */
                PrcEnd=1;
                break;
            default:
                /* その他致命的なエラー */
                /* プロセス終了する */
                PrcEnd=1;
                break;
        }
    }
}

```



```

    }

    /* 二つ目の while ループを抜ける */
    break;
}

/* トランザクション開始 */
Ret = diosaimtxstart();
if(DIOSA_DONE != Ret) {
    /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
    /* 詳細は 3.1.8 戻り値について を参照 */
}

/* 論理表 ID 照会 */
Ret = diosaimgettblid("SAMPLE_TABLE_A", &TblId);
if(DIOSA_DONE != Ret) {
    /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
    /* 詳細は 3.1.8 戻り値について を参照 */
}

/* 追加レコードの設定 */
/* レコードイメージ設定 */
strncpy(BuffArea, "SAMPLE_WRITE_RECORD", sizeof(BuffArea) );
/* セカンダリキーを設定(整数型の値はネットワークバイトオーダーに変換) (Linux 版は必須) */
SecondaryKey = (struct index2*) &BuffArea[32];
memset(SecondaryKey, '¥0', sizeof(struct index2));
strncpy(SecondaryKey->Date, "20110601", 8);
SecondaryKey->ReceiptNumber = htonl(100);
...

Recinfo.RecordPtr = BuffArea;
/* 更新レコードサイズ設定 */
/* 可変長レコードの場合は取得レコードサイズからの増減可能 */
Recinfo.RecordSize = 128;

/* レコード追加 */
Ret = diosaimwrite(TblId, &Recinfo, 1, DIOSA_LOCK_WAIT);
if(DIOSA_DONE != Ret) {
    /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
    /* 詳細は 3.1.8 戻り値について を参照 */
}

```

```

/* コミット */
Ret = diosaimcommit();
if(DIOSA_DONE != Ret) {
    /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
    /* 詳細は 3.1.8 戻り値について を参照 */
}

/*-----*/
/* MAP2 の表のレコードを更新 */
/*-----*/
/* 複数 MAP 更新時は COMMIT してから再度トランザクションを開始し次の MAP の処理を行う*/
/* トランザクション開始 */
Ret = diosaimtxstart();
if(DIOSA_DONE != Ret) {
    /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
    /* 詳細は 3.1.8 戻り値について を参照 */
}

/* アクセス先 MAP 宣言 */
Ret = diosaimsetmap( 2 );
if(DIOSA_DONE != Ret) {
    /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
    /* 詳細は 3.1.8 戻り値について を参照 */
}

/* 論理表 ID 照会 */
Ret = diosaimgettblid("SAMPLE_TABLE_B", &TblId);
if(DIOSA_DONE != Ret) {
    /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
    /* 詳細は 3.1.8 戻り値について を参照 */
}

/* 更新対象のレコードのロック取得 */
Ret = diosaimcondsetkey(&Cond, "SAMPLE_PKEY", 11);
if(DIOSA_DONE != Ret) {
    /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
    /* 詳細は 3.1.8 戻り値について を参照 */
}

/* レコードが格納できる領域を確保する(この例では 26byte のレコードとする) */
BuffSize = 26;

```

```

Ret = diosamalloc(NULL, NULL, DIOSA_SVCALL, BuffSize, &BuffPtr);
if(DIOSA_DONE != Ret) {
    return 1;
}

/* キー指定レコード読込 */
Ret = diosaimreadl(TblId, &Cond, &Recinfo, BuffPtr, BuffSize, DIOSA_LOCK_WAIT);
if(DIOSA_DONE != Ret) {
    /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
    /* 詳細は 3.1.8 戻り値について を参照 */
}

/* 対象レコードの更新 */
/* 更新レコードイメージ設定(プライマリキーは変更不可) */
strcpy(BuffArea, "SAMPLE_PKEY_UPDATE_RECORD", sizeof(BuffArea) );
Recinfo.RecordPtr = BuffArea;
/* 更新レコードサイズ設定 */
/* 可変長レコードの場合は取得レコードサイズからの増減可能 */
Recinfo.RecordSize = strlen(BuffArea);

/* レコード更新 */
Ret = diosaimrewrite(TblId, &Recinfo, 1);
if(DIOSA_DONE != Ret) {
    /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
    /* 詳細は 3.1.8 戻り値について を参照 */
}

/* コミット */
Ret = diosaimcommit();
if(DIOSA_DONE != Ret) {
    /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
    /* 詳細は 3.1.8 戻り値について を参照 */
}
}

/* DIOSA トランザクション終了 */
Ret = diosatrnterm( NULL );
if(DIOSA_DONE != Ret) {
    break;
}

```

```

        /* 致命的なエラーが発生した場合はプロセスを終了する */
        if(1 == PrcEnd) {
            break;
        }
    }

    /* IM サーバクローズ */
    diosaimclose();

    /* DIOSA プロセス終了 */
    diosaprcterm( NULL );

    return 0;
}

```

## (2) レコード参照を行うサンプルプログラム

以下に複数 MAP のレコードを複数レコード取得するプログラムのサンプルを記します。

```

int main() {

    t_diosa_imcond Cond;
    t_diosa_recinfo RecInfo[10];
    int    CtxId;
    int    FetchedNum;
    char   *BuffPtr;
    size_t BuffSize;
    int    Ret;
    int    TblId;
    int    PrcEnd;
    int    Mode;
    int    i;
    struct index2 {                /* セカンダリキー */
        char Date[8];
        char Rfu[4];
        int  ReceiptNumber;
    };
    struct index2 KeyMin, KeyMax;

    /* DIOSA プロセス初期化 */
    Ret = diosaprcinit( NULL );
    if(DIOSA_DONE != Ret) {

```

```

        diosaprcterm( NULL );
        return 1;
    }

/* IM サーバオープン */
Mode = DIOSA_CONN_INADVANCE;
Ret = diosaimopen( Mode );
if(DIOSA_DONE != Ret) {
    diosaimclose();
    diosaprcterm( NULL );
    return 1;
}

Ret = DIOSA_DONE;
while(DIOSA_DONE == Ret) {
    /* DIOSA トランザクション初期化/終了の区間内でメモリキャッシュの API を実行し */
    /* レコード操作を行う */
    /*
    /* DIOSA トランザクション初期化 */
    Ret = diosatrinit( NULL );
    if(DIOSA_DONE != Ret) {
        break;
    }

    Ret = DIOSA_DONE;
    while(DIOSA_DONE == Ret) {
        /*-----*/
        /* MAP1 のレコードを検索 */
        /*-----*/
        /* アクセス先 MAP 宣言 */
        Ret = diosaimsetmap( 1 );
        if(DIOSA_DONE != Ret) {
            /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
            /* 詳細は 3.1.8 戻り値について を参照 */
        }

        /*
        * SG 定義上、長さ 8 バイトの文字列(YYYYMMDD)と 4 バイトの整数型のキー対する
        * 範囲指定検索を行う。整数型の値はネットワークバイトオーダで指定する必要あり
        * (Linux 版は必須)
        */

```

```

memset(&KeyMin, '¥0', sizeof(struct index2));
memset(&KeyMax, '¥0', sizeof(struct index2));
strncpy(KeyMin.Date, "20110601", 8);
strncpy(KeyMax.Date, "20110601", 8);
KeyMin.ReceiptNumber = htonl(100);
KeyMax.ReceiptNumber = htonl(200);
Ret = diosaimcondsetrange(&Cond, DIOSA_COND_GE, (char*)&KeyMin, sizeof(KeyMin)
                          , DIOSA_COND_LE, (char*)&KeyMax, sizeof(KeyMax));

if(DIOSA_DONE != Ret) {
    /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
    /* 詳細は 3.1.8 戻り値について を参照 */
}

Ret = diosaimgettblid("SAMPLE_TABLE", &TblId);
if(DIOSA_DONE != Ret) {
    /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
    /* 詳細は 3.1.8 戻り値について を参照 */
}

Ret = diosaimctxopen(TblId, "index2", &Cond, DIOSA_ASCEND, DIOSA_NOLOCK, &CtxId);
if(DIOSA_DONE != Ret) {
    /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
    /* 詳細は 3.1.8 戻り値について を参照 */
}

/* レコード 10 件分が格納できる領域を確保する */
BuffSize = 10240;
Ret = diosamalloc(NULL, NULL, DIOSA_SVCALL, BuffSize, &BuffPtr);
if(DIOSA_DONE != Ret) {
    return 1;
}

Ret = DIOSA_DONE;
while(DIOSA_DONE == Ret) {
    /* 条件に合致するものを 10 件ずつ取得する */
    Ret = diosaimread(CtxId, RecInfo, 10, &FetchedNum, BuffPtr, BuffSize);
    if((DIOSA_DONE != Ret) && (DIOSA_NOENT != Ret)) {
        /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
        /* 詳細は 3.1.8 戻り値について を参照 */
    }
    if(DIOSA_NOENT == Ret) {

```

```

        /* 条件に合致するレコードが一件もない */
        break;
    }

    for(i=0; FetchedNum>i; i++){
        /* 取得したレコードに対する処理を行う */
        /* 取得レコード件数分、レコード情報(RecInfo)に格納されています*/
        /* レコードイメージ格納アドレスは、RecInfo[i].RecordPtr          */
        /* レコードサイズは、RecInfo[i].RecordSize でアクセスします      */
    }
}

Ret = diosaimctxclose(CtxId);
if(DIOSA_DONE != Ret){
    /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
    /* 詳細は 3.1.8 戻り値について を参照                                */
}

/*-----*/
/* MAP2 のレコードを検索      */
/*-----*/
/* アクセス先 MAP 宣言 */
Ret = diosaimsetmap( 2 );
if(DIOSA_DONE != Ret){
    /* API の異常終了時には戻り値によってその後の動作を判断する必要あり */
    /* 詳細は 3.1.8 戻り値について を参照                                */
}

/* MAP1 のレコード検索と同様にレコード取得を行う */
/* diosaimread、diosaimreadl かつ排他オプションでロックを行わない場合は */
/* 複数 MAP へのアクセスであっても diosacommit の実行は不要          */
}

/* DIOSA トランザクション終了 */
Ret = diosatrnterm( NULL );
if(DIOSA_DONE != Ret){
    break;
}

/* 致命的なエラーが発生した場合はプロセスを終了する */
if(1 == PrcEnd){
    break;
}

```

```
        }  
    }  
  
    /* IM サーバクローズ */  
    diosaimclose();  
  
    /* DIOSA プロセス終了 */  
    diosaprcterm( NULL );  
  
    return 0;  
}
```



## 3.4 利用者出口

メモリキャッシュから呼び出す利用者出口について説明します。

### 3.4.1 プログラムの構造

インメモリサーバ所在管理機能が MAP を決定する際に、ハッシュ関数を呼び出します。

### 3.4.2 ハッシュ関数

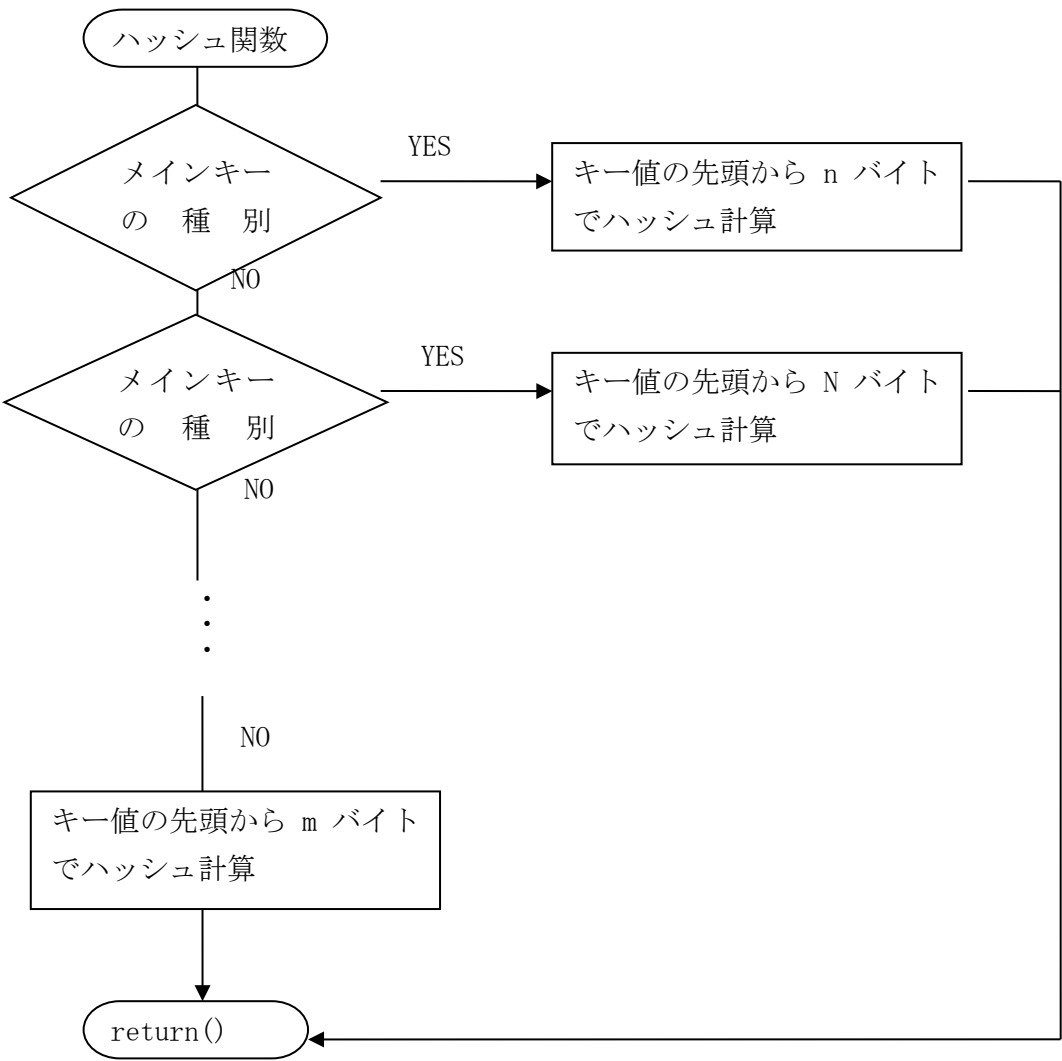
ハッシュ関数は、ユーザデータの格納先を決定するためのハッシュ値を計算する利用者出口です。

ハッシュ関数は、AP 層および OLTP 層で、MAPID を決定する際に呼び出されます。

ハッシュ関数は、入力パラメータで渡されたメインキーを使用してハッシュ計算し、ハッシュ値を返却します。

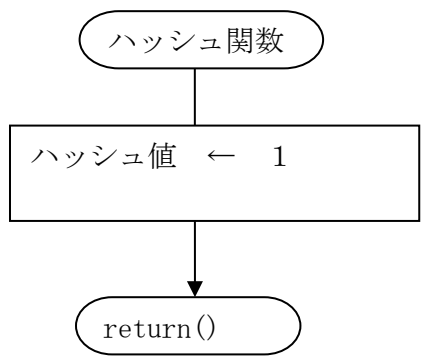
#### (1) ハッシュ関数の実装例 1

キー値が複数存在し、キー毎にキー値長が異なる場合、下記のように実装します。



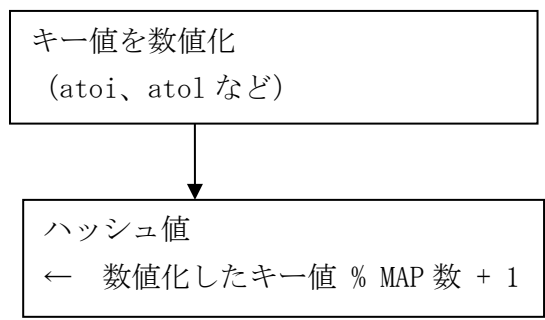
(2)      **ハッシュ関数の実装例 2**

メインキー分散を行わない場合、下記のように実装します。  
※環境定義で、唯一定義されている MAP のハッシュ値が 1 の場合



(3)      **ハッシュ計算方法について**

最も簡単なハッシュ計算方法は、キー値を MAP 数で割った余りをハッシュ値とする方法です。



この方式では、キー値の最後が固定値になっている場合などは、同じハッシュ値に偏りができてしまいます。

例えば、nnnnn000 (n は任意の数値) であるキー値で、MAP 数が 12 の場合、上記方式で計算すると、全てハッシュ値が 1 となってしまいます。このような場合、nnnnn の部分のみを使用したハッシュ計算を行う必要があります。

## 3.5 アプリケーションの生成

アプリケーションの生成について、メモリキャッシュ固有の指定等はありません。

DIOSA/XTP 利用の手引きを参照して、アプリケーションを生成してください。

# 第4章 システムの構築

## 4.1 環境設計

メモリキャッシュの環境構築を行う場合、利用形態に合わせて、論理ノードに TAM インスタンスをどのように配置するか、TAM インスタンスにどのように TAM 表を配置するかを決定し、システム構築を行う必要があります。

### 4.1.1 TAM 表の配置の決定

まず、どのように TAM 表を配置するかを設計します。  
TAM 表毎に分散するかしないかを決定し、分散させる場合は、複数の MAP 構成で定義します。  
そして、MAP をどのレプリケーショングループとするかを決定します。  
レプリケーショングループは複数 MAP で構成することも可能です。

下図は、表 A、表 B、表 C は表を分散させ、表 D、表 E は表の分散せず TAM 表を構成し、これらを 6 のレプリケーショングループに配置する例です。

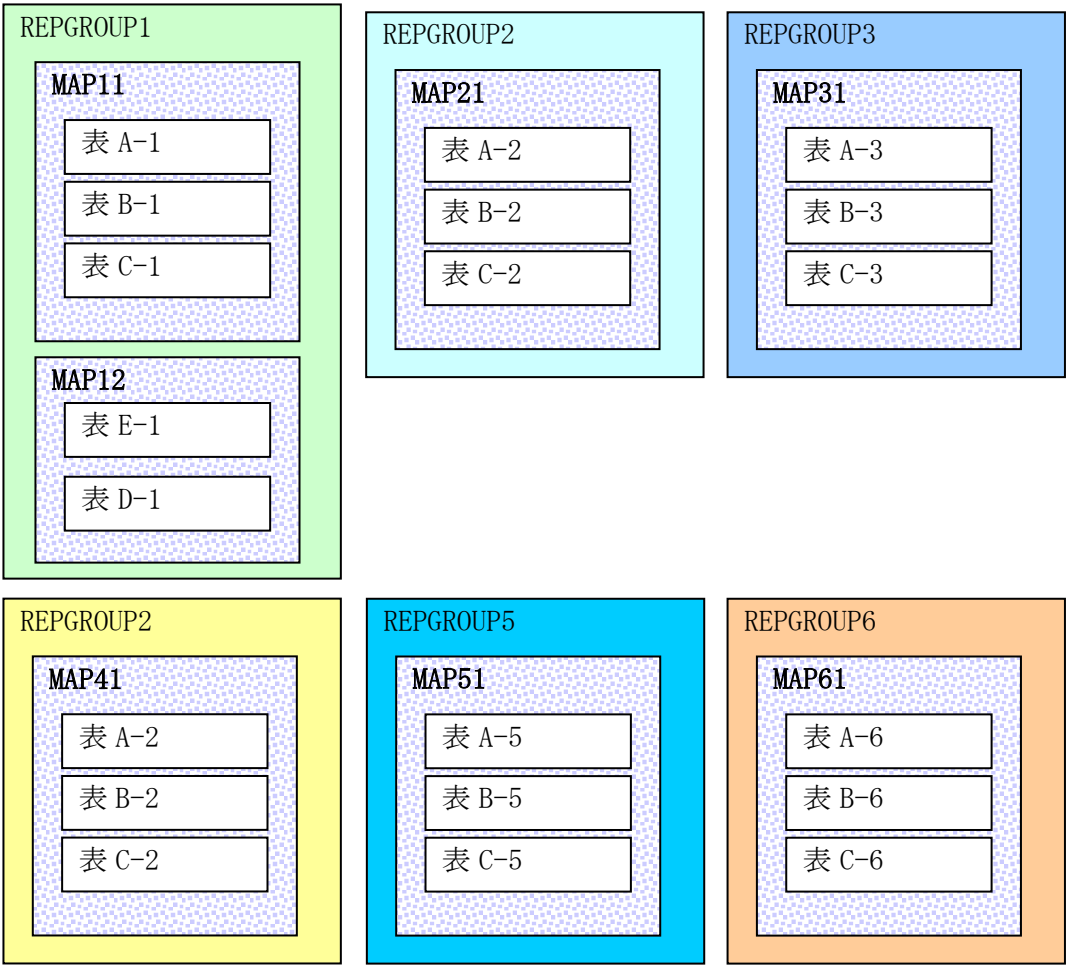


図 4-1 TAM 表の配置の決定

又、これらの構成をメモリキャッシュ環境定義に記述する場合の定義イメージを示します。

(定義イメージ)

```
$IMENV
%COMMON ALLOWCOMMIT = REPGTHALF ;
%DEF_MAP
%GROUPCOMMIT TMOUT = 0 ;
;
%REPGROUP
ID = 1, INSTANCE = INS001
%MAP
ID = 11 %HASHRANGE START = 100, END = 149 ;
;
%MAP
ID = 12 %HASHRANGE START = 150, END = 199 ;
;
%MASTER LNODE = 論理ノード 1, TAMVNODE = MASTER ;
;
%REPGROUP
ID = 2, INSTANCE = INS002
%MAP
ID = 21 %HASHRANGE START = 200, END = 299 ;
;
%MASTER LNODE = 論理ノード 2, TAMVNODE = MASTER ;
;
%REPGROUP
ID = 3, INSTANCE = INS003
%MAP
ID = 31 %HASHRANGE START = 300, END = 399 ;
;
%MASTER LNODE = 論理ノード 3, TAMVNODE = TAMLN003MASTER ;
;
～省略～
;
```

```
$IMTABLECONF
%LTABLE
TYPE = 1, NAME = 表 A, ID = 1
%RECORDCONF
TYPE = VARLEN
%PRIMARYKEY NAME = I_PK_TABLE_A %KEYDEF OFFSET = 40, LENGTH = 16; ;
;
%PTABLE NAME = 表 A-1, MAPID = 11 ;
%PTABLE NAME = 表 A-2, MAPID = 21 ;
%PTABLE NAME = 表 A-3, MAPID = 31 ;
%PTABLE NAME = 表 A_4, MAPID = 41 ;
%PTABLE NAME = 表 A-5, MAPID = 51 ;
%PTABLE NAME = 表 A-6, MAPID = 61 ;
;
%LTABLE
TYPE = 1, NAME = 表 B, ID = 2
```

```
%RECORDCONF
    TYPE = VARLEN
    %PRIMARYKEY  NAME = I_PK_TABLE_B    %KEYDEF OFFSET = 40, LENGTH = 16; ;
;
%PTABLE  NAME = 表 B-1, MAPID = 11 ;
%PTABLE  NAME = 表 B-2, MAPID = 21 ;
%PTABLE  NAME = 表 B-3, MAPID = 31 ;
%PTABLE  NAME = 表 B-4, MAPID = 41 ;
%PTABLE  NAME = 表 B-5, MAPID = 51 ;
%PTABLE  NAME = 表 B-6, MAPID = 61 ;
;
%LTABLE
    TYPE = 1, NAME = 表 C, ID = 3
    %RECORDCONF
        TYPE = VARLEN
        %PRIMARYKEY  NAME = I_PK_TABLE_C    %KEYDEF OFFSET = 40, LENGTH = 16; ;
;
%PTABLE  NAME = 表 C-1, MAPID = 11 ;
%PTABLE  NAME = 表 C-2, MAPID = 21 ;
%PTABLE  NAME = 表 C-3, MAPID = 31 ;
%PTABLE  NAME = 表 C-4, MAPID = 41 ;
%PTABLE  NAME = 表 C-5, MAPID = 51 ;
%PTABLE  NAME = 表 C-6, MAPID = 61 ;
;
%LTABLE
    TYPE = 1, NAME = 表 D, ID = 4
    %RECORDCONF
        TYPE = VARLEN
        %PRIMARYKEY  NAME = I_PK_TABLE_D    %KEYDEF OFFSET = 40, LENGTH = 16; ;
;
%PTABLE  NAME = 表 D-1, MAPID = 12 ;
;
%LTABLE
    TYPE = 1, NAME = 表 E, ID = 5
    %RECORDCONF
        TYPE = VARLEN
        %PRIMARYKEY  NAME = I_PK_TABLE_E    %KEYDEF OFFSET = 40, LENGTH = 10; ;
;
%PTABLE  NAME = 表 E-1, MAPID = 12;
;
;
```

(TAM コンフィグファイル)

```
#table.conf
[common]
update_log_size =1
update_log_extend_size =1
max_update_log_size =50
trace_entry =0
```

```

record_duplicate_check =no
check_record_num =0
mutex_spin_num =0
dirty_read =no
#-----
[table]
tam_table_name =表 D-1
tam_table_file =/diosa_xtp/tam/table/TABLE_D_1.tam
index_name1 =I_PK_TABLE_D
keys1 =24:16,6:2
~省略~
#-----
[table]
tam_table_name =表 E-1
tam_table_file =/diosa_xtp/tam/table/TABLE_E_1.tam
index_name1 =I_PK_TABLE_E
index_entry_num1 =41
keys1 =24:10,6:2
auto_extend_key_num1 =1290
~省略~
#-----
[table]
tam_table_name =表 A-1
tam_table_file =/diosa_xtp/tam/table/TABLE_A_1.tam
index_name1 =I_PK_TABLE_A
index_entry_num1 =18
keys1 =24:16,6:2
~省略~
#-----
[table]
tam_table_name =表 B-1
tam_table_file =/diosa_xtp/tam/table/TABLE_B_1.tam
index_name1 =I_PK_TABLE_B
keys1 =24:16,6:2
~省略~
#-----
[table]
tam_table_name =表 C-1
tam_table_file =/diosa_xtp/tam/table/TABLE_C_1.tam
index_name1 =I_PK_TABLE_C
keys1 =24:16,6:2
~省略~
#-----
[table]
tam_table_name =表 A-2
tam_table_file =/diosa_xtp/tam/table/TABLE_A_2.tam
index_name1 =I_PK_TABLE_A
index_entry_num1 =18

```

```
keys1 =24:16,6:2
#-----
[table]
tam_table_name =表 B-2
tam_table_file =/diosa_xtp/tam/table/TABLE_B_2.tam
index_name1 =I_PK_TABLE_B
index_entry_num1 =18
keys1 =24:16,6:2
～省略～
#-----
-----
[table]
tam_table_name =表 C-2
tam_table_file =/diosa_xtp/tam/table/TABLE_C_2.tam
index_name1 =I_PK_TABLE_C
index_entry_num1 =18
keys1 =24:16,6:2
～省略～
#-----
-----
[table]
tam_table_name =表 A-3
tam_table_file =/diosa_xtp/tam/table/TABLE_A_3.tam
index_name1 =I_PK_TABLE_A
index_entry_num1 =18
keys1 =24:16,6:2
～省略～
```



## 4.1.2 TAM インスタンスの配置の決定

論理ノードに配置する TAM インスタンスについて設計します。

TAM インスタンスは、各レプリケーショングループのマスタ／スレーブ毎に異なる OLTP ノードに配置します。

すなわち、同一レプリケーショングループのマスタとスレーブインスタンスを 1 つの論理ノードに配置することはできません。

構成は TAM コンフィグファイル、メモリキャッシュ環境定義 (IMENV, IMTABLECONF) に指定します。

図 4-2 は、複数レプリケーション構成、OLTP ノードに複数マスタ TAM を配置する例です。

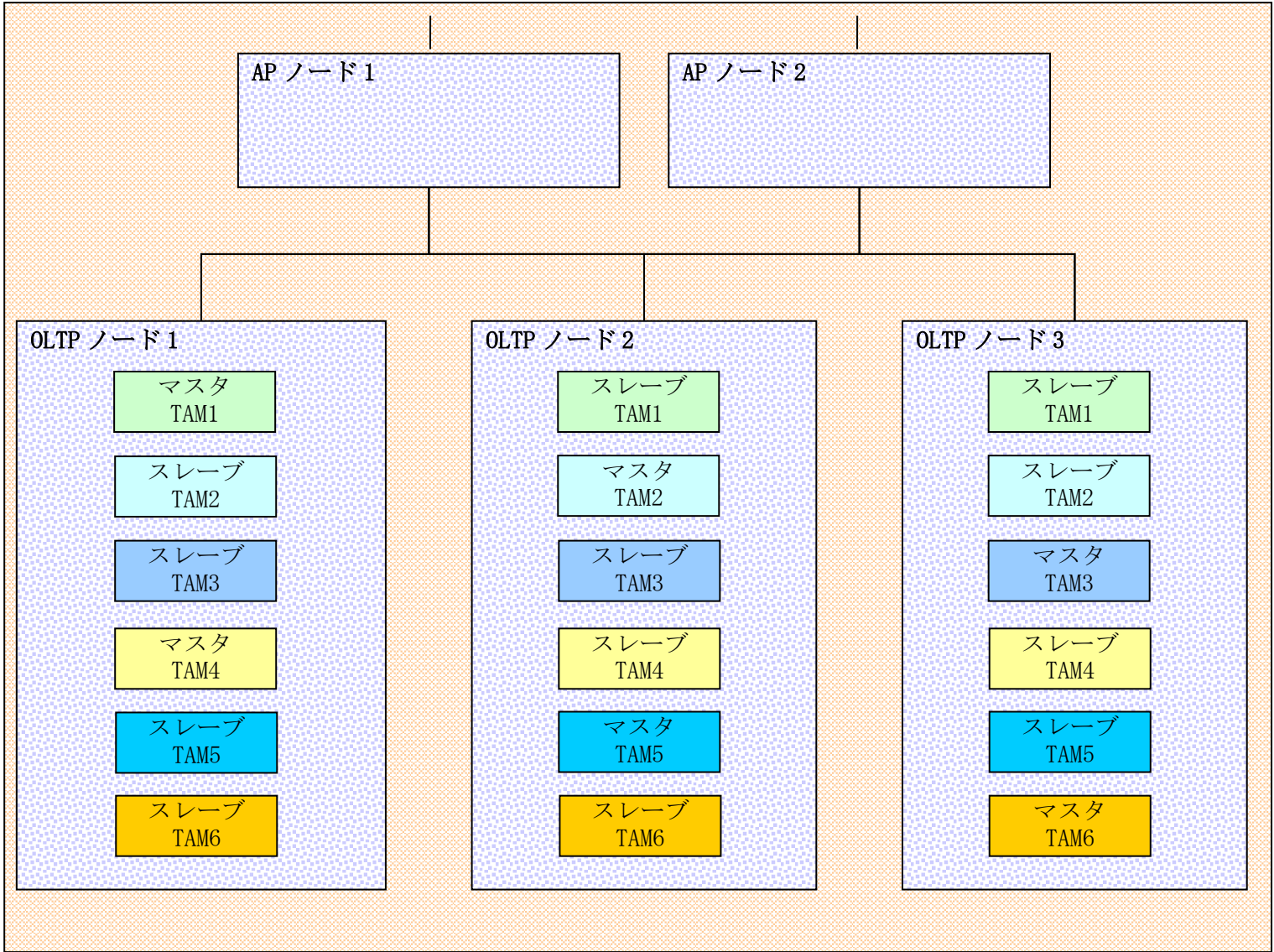


図 4-2 TAM インスタンスの配置の決定

以下に 4.1.1 で決定した 6 つのレプリケーショングループの構成を、マスタ TAM 1 ～ 6 として記述するイメージを示します。論理ノードの定義は DIOSAMAP 節に定義します。

```
$DIOSAMAP
%DEFAULT
PORT_BCM = 1100
PORT_CDD = 1101
PORT_DLT = 1102
PORT_IIC = 1106
PORT_IMS = 1108
;
```

```
%LOGSYSTEM
NAME = LS11
%LNODE
NAME = AP ノード 1, ID = 101, TYPE = AP
IPADDR = 192.168.1.11
IPCKEY = 0001
;
%LNODE
NAME = AP ノード 2, ID = 102, TYPE = AP
IPADDR = 192.168.1.12
IPCKEY = 0001
;
%LNODE
NAME = OLTP ノード 1, ID = 201, TYPE = OLTP
IPADDR = 192.168.1.21
IPCKEY = 0001
;
%LNODE
NAME = OLTP ノード 2, ID = 202, TYPE = OLTP
IPADDR = 192.168.1.22
IPCKEY = 0001
;
%LNODE
NAME = OLTP ノード 3, ID = 203, TYPE = OLTP
IPADDR = 192.168.1.23
IPCKEY = 0001
;
;
;
```

```
$IMENV
%REPGROUP
ID = 1, INSTANCE = INS001
%MAP ID = 11 %HASHRANGE START = 100, END = 149 ; ;
%MAP ID = 12 %HASHRANGE START = 150, END = 199 ; ;

%MASTER LNODE = OLTP ノード 1, TAMVNODE = INS001MASTER ;
%SLAVE LNODE = OLTP ノード 2, TAMVNODE = INS001SLAVE1 ;
%SLAVE LNODE = OLTP ノード 3, TAMVNODE = INS001SLAVE2 ;
;
%REPGROUP
ID = 2, INSTANCE = INS002
%MAP ID = 21 %HASHRANGE START = 200, END = 299 ; ;
%MASTER LNODE = OLTP ノード 2, TAMVNODE = INS002MASTER ;
%SLAVE LNODE = OLTP ノード 3, TAMVNODE = INS003SLAVE1 ;
%SLAVE LNODE = OLTP ノード 1, TAMVNODE = INS003SLAVE2 ;
;
%REPGROUP
```

```

        ID = 3, INSTANCE = INS003
        %MAP ID = 31 %HASHRANGE START = 300, END = 399 ; ;
        %MASTER LNODE = OLTP ノード 3, TAMVNODE = INS003MASTER ;
        %SLAVE LNODE = OLTP ノード 1, TAMVNODE = INS003SLAVE1 ;
        %SLAVE LNODE = OLTP ノード 2, TAMVNODE = INS003SLAVE2 ;
    ;
    %REPGROUP
        ID = 4, INSTANCE = INS004
        %MAP ID = 41 %HASHRANGE START = 400, END = 499 ; ;
        %MASTER LNODE = OLTP ノード 1, TAMVNODE = INS004MASTER ;
        %SLAVE LNODE = OLTP ノード 2, TAMVNODE = INS004SLAVE1 ;
        %SLAVE LNODE = OLTP ノード 3, TAMVNODE = INS004SLAVE2 ;
    ;
    %REPGROUP
        ID = 5, INSTANCE = INS005
        %MAP ID = 51 %HASHRANGE START = 500, END = 599 ; ;
        %MASTER LNODE = OLTP ノード 2, TAMVNODE = INS005MASTER ;
        %SLAVE LNODE = OLTP ノード 3, TAMVNODE = INS005SLAVE1 ;
        %SLAVE LNODE = OLTP ノード 1, TAMVNODE = INS005SLAVE2 ;
    ;
    %REPGROUP
        ID = 6, INSTANCE = INS006
        %MAP ID = 61 %HASHRANGE START = 600, END = 699 ; ;
        %MASTER LNODE = OLTP ノード 3, TAMVNODE = INS006MASTER ;
        %SLAVE LNODE = OLTP ノード 1, TAMVNODE = INS006SLAVE1 ;
        %SLAVE LNODE = OLTP ノード 2, TAMVNODE = INS006SLAVE2 ;
    ;
;
;

```

(TAM コンフィグファイル)

```

#tammng.conf
[cluster]
multi_instance          =yes
cluster_name            =CL_00
instance_group1         =(INS001@OLTP ノード 1, INS001@OLTP ノード 2, INS001@OLTP ノード 3)
instance_group2         =(INS002@OLTP ノード 1, INS002@OLTP ノード 2, INS002@OLTP ノード 3)
instance_group3         =(INS003@OLTP ノード 1, INS003@OLTP ノード 2, INS003@OLTP ノード 3)
instance_group4         =(INS004@OLTP ノード 1, INS004@OLTP ノード 2, INS004@OLTP ノード 3)
instance_group5         =(INS005@OLTP ノード 1, INS005@OLTP ノード 2, INS005@OLTP ノード 3)
instance_group6         =(INS006@OLTP ノード 1, INS006@OLTP ノード 2, INS006@OLTP ノード 3)
#-----
[node]
node_name               =OLTP ノード 1
ip_memory_backup       =(192.168.2.21)
[node]
node_name               =OLTP ノード 2
ip_memory_backup       =(192.168.2.22)
[node]
node_name               =OLTP ノード 3

```

```

ip_memory_backup          =(192. 168. 2. 23)
#-----
[vnode]
virtual_node_name          =INS001MASTER
backup_role                =master
[vnode]
virtual_node_name          =INS002MASTER
backup_role                =master
[vnode]
virtual_node_name          =INS003MASTER
backup_role                =master
[vnode]
virtual_node_name          =INS004MASTER
backup_role                =master
[vnode]
virtual_node_name          =INS005MASTER
backup_role                =master
[vnode]
virtual_node_name          =INS006MASTER
backup_role                =master
#-----
[vnode]
virtual_node_name          =INS001SLAVE1
backup_role                =slave
[vnode]
virtual_node_name          =INS002SLAVE1
backup_role                =slave
[vnode]
virtual_node_name          =INS003SLAVE1
backup_role                =slave
[vnode]
virtual_node_name          =INS004SLAVE1
backup_role                =slave
[vnode]
virtual_node_name          =INS005SLAVE1
backup_role                =slave
[vnode]
virtual_node_name          =INS006SLAVE1
backup_role                =slave
#-----
[vnode]
virtual_node_name          =INS001SLAVE2
backup_role                =slave
[vnode]
virtual_node_name          =INS002SLAVE2
backup_role                =slave
[vnode]
virtual_node_name          =INS003SLAVE2

```

```

backup_role                =slave
[vnode]
virtual_node_name          =INS004SLAVE2
backup_role                =slave
[vnode]
virtual_node_name          =INS005SLAVE2
backup_role                =slave
[vnode]
virtual_node_name          =INS006SLAVE2
backup_role                =slave
#-----
[instance]
instance_name              =INS001
base_node                  =OLTP ノード 1
i_init_shm_size            =2590
i_init_shm_num             =1
i_max_shm_num              =16
i_shm_extend_size         =260
i_port                     =41000
i_port_range               =20
i_start_check_port         =40000
i_ccm_thread_num           =10
i_send_thread_num          =2
i_recv_thread_num          =5
i_sendproc_thread_num      =2
i_recvproc_thread_num      =2
i_max_proc_request_num     =64
i_send_piece_size          =62816
i_stc_set_priority         =no
i_send_ratio               =3:1
i_send_delay               =0
i_timer_max_queue_size     =4000
i_auto_lock_check          =no
i_max_tpid_num             =1024
i_net_send_buffer_size     =256
i_net_recv_buffer_size     =256
i_auto_lock_check_syslog   =summary
i_blockage_to_down         =no
i_do_backup                =yes
i_tx_shm_table_size        =50
i_tx_thread_num            =100
#-----
[instance]
instance_name              =INS002
base_node                  =OLTP ノード 1
i_init_shm_size            =2590
i_init_shm_num             =1
i_max_shm_num              =16

```

```

i_shm_extend_size      =260
i_port                 =41020
i_port_range           =20
i_start_check_port     =40001
i_ccm_thread_num       =10
i_send_thread_num      =2
i_recv_thread_num      =5
i_sendproc_thread_num  =2
i_recvproc_thread_num  =2
i_max_proc_request_num =64
i_send_piece_size      =62816
i_stc_set_priority     =no
i_send_ratio           =3:1
i_send_delay           =0
i_timer_max_queue_size =4000
i_auto_lock_check      =no
i_max_tpid_num         =1024
i_net_send_buffer_size =256
i_net_recv_buffer_size =256
i_auto_lock_check_syslog =summary
i_blockage_to_down     =no
i_do_backup            =yes
i_tx_shm_table_size    =50
i_tx_thread_num        =100
#-----
[instance]
instance_name          =INS003
base_node              =OLTP ノード 1
~~省略~~
#-----
[instance]
instance_name          =INS006
base_node              =OLTP ノード 1
i_init_shm_size        =2590
i_init_shm_num         =1
i_max_shm_num          =16
i_shm_extend_size      =260
i_port                 =41080
i_port_range           =20
i_start_check_port     =40005
i_ccm_thread_num       =10
i_send_thread_num      =2
i_recv_thread_num      =5
i_sendproc_thread_num  =2
i_recvproc_thread_num  =2
i_max_proc_request_num =64
i_send_piece_size      =62816
i_stc_set_priority     =no

```

```

i_send_ratio          =3:1
i_send_delay          =0
i_timer_max_queue_size =4000
i_auto_lock_check     =no
i_max_tpid_num        =1024
i_net_send_buffer_size =256
i_net_recv_buffer_size =256
i_auto_lock_check_syslog =summary
i_blockage_to_down    =no
i_do_backup           =yes
i_tx_shm_table_size   =50
i_tx_thread_num       =100
#-----
[instance]
instance_name         =INS001
base_node              =OLTP ノード 2
i_init_shm_size       =2590
i_init_shm_num        =1
i_max_shm_num         =16
i_shm_extend_size     =260
i_port                =41000
i_port_range          =20
i_start_check_port     =40000
~~省略~~
#-----
[instance]
instance_name         =INS001
base_node              =OLTP ノード 3
i_init_shm_size       =2590
i_init_shm_num        =1
i_max_shm_num         =16
i_shm_extend_size     =260
i_port                =41000
i_port_range          =20
i_start_check_port     =40000
~~省略~~
#-----
[instance]
instance_name         =INS006
base_node              =OLTP ノード 3
i_init_shm_size       =2590
i_init_shm_num        =1
i_max_shm_num         =16
i_shm_extend_size     =260
i_port                =41080
i_port_range          =20
i_start_check_port     =40005
i_ccm_thread_num      =10

```

i_send_thread_num	=2
i_recv_thread_num	=5
i_sendproc_thread_num	=2
i_recvproc_thread_num	=2
i_max_proc_request_num	=64
i_send_piece_size	=62816
i_stc_set_priority	=no
i_send_ratio	=3:1
i_send_delay	=0
i_timer_max_queue_size	=4000
i_auto_lock_check	=no
i_max_tpid_num	=1024
i_net_send_buffer_size	=256
i_net_recv_buffer_size	=256
i_auto_lock_check_syslog	=summary
i_blockage_to_down	=no
i_do_backup	=yes
i_tx_shm_table_size	=50
i_tx_thread_num	=100



### 4.1.3 IMS キューバッファサイズ(IMQUEBUFSIZE)の決定

アプリケーション、アクセスサーバ、ブリッジサーバ間のプロセス間通信で使用する IMS キューバッファのサイズについて説明します。

ひとつの IMS キューバッファは、ひとつの共有メモリセグメントとして確保され、アプリケーション、アクセスサーバ、ブリッジサーバごとに用意されます。それぞれの場合において、格納されるレコード情報や保持期間は異なります。

アプリケーションの IMS キューバッファは、メモリキャッシュへアクセスするスレッドごとに確保されるため、『アプリケーションの多重度数』分確保されます。また、アクセスサーバの IMS キューバッファは『( MAP ごとのアクセススレッド多重度 + 1 )の全 MAP 分の合計』分、ブリッジサーバの IMS キューバッファは『ブリッジサーバ多重度数 × 2』分確保されます。

#### (1) アプリケーション側の IMS キューバッファサイズ計算

アプリケーション側の IMS キューバッファは、アプリケーションからアクセスサーバへアクセスする情報と更新したレコード情報が格納されます(ブリッジサーバ経由でアクセスする場合も同様)。

更新したレコード情報とは、diosaimread1(ロックあり)、diosaimwrite、diosaimrewrite、diosaimdelete、diosaimdeletex1 の API を使用して更新したレコードの情報です。

この更新したレコード情報は、アプリケーションがコミットまたはロールバックするまで IMS キューバッファ上に保持され、コミットまたはロールバック処理が完了した時に解放されます。

また、アプリケーションからアクセスサーバへアクセス要求する情報は、参照系のアクセス要求であれば、インメモリサーバが要求を受け付けた時点で解放されます。

そのため、アプリケーション側の IMS キューバッファのサイズは、1 トランザクションで更新するレコードサイズとそれに付随する DIOSA の制御情報サイズ、コミットまたはロールバック要求電文のサイズを合計したサイズを用意する必要があります。

アクセス時の各サイズの計算方法を下に記述します。

- 固定長レコードサイズ計算

■計算に必要な情報

- (A) 更新対象表の table.conf の record\_size
- (B) 書込むレコードのプライマリキーのサイズ

■計算式

- (C) 補正レコードサイズ =  $((A + 7) / 8) \times 8$  ※8 バイトバウンダリにあわせませす
- (D) 固定長レコードサイズ =  $40 + C + 392 + B$

- 可変長レコードサイズ計算

■計算に必要な情報

- (E) 更新対象表の table.conf の record\_size
- (F) 更新対象表に定義された全キーのサイズ合計
- (G) API に指定する実際に書込むレコードのサイズ
- (H) 書込むレコードのプライマリキーのサイズ

■計算式

- (I) 分割レコード数 =  $G \div (E - F - 24)$  ※少数点以下は繰り上げます
- (J) 補正レコードサイズ =  $((E + 7) / 8) \times 8$  ※8 バイトバウンダリにあわせませす
- (K) 可変長レコードサイズ =  $(40 + J) \times I + 392 + H$

- コミット要求電文のサイズ

■計算式

(L) コミット要求のサイズ = 304

上記の各サイズと 1 トランザクションで更新するレコード数により、アプリケーション側の IMS キューバッファのサイズを計算します。

■計算に必要なもの

(M) 最大レコードサイズ = D, K で最大のもの

(N) レコード数 = 1 トランザクション内で更新するレコードの「のべ数」(※)

※例)diosaimreadl(ロックあり) した後 diosaimrewrite する場合は「2」とカウントする

■計算式

(O) トランザクションごとに必要なサイズ= M × N + L

(P) IMS キューバッファのサイズ(バイト) = O のうち最大のもの × 安全率

(注)M, N, O は、トランザクションごとに計算する

(注)安全率は、タイムアウト時に複数トランザクションが格納される可能性を考慮し、タイムアウトの発生率に応じて1〜2程度を指定する

(注)SG ではキロバイト単位で設定する必要があります。

(2)      **アクセスサーバの IMS キューバッファサイズ計算**

アクセスサーバの IMS キューバッファは、アプリケーションが読込要求を行ったレコード情報とインメモリサーバからアプリケーションへのアクセス応答情報が格納されます。

読込要求を行ったレコード情報とは、diosaimread、diosaimreadl(ロックあり、ロックなし)、diosaimdeletexl(内部で読込を行っています)の API を使用して読込したレコードの情報です。

この読込要求を行ったレコード情報は、アプリケーションにレコード情報が受け渡された時点で解放されます。また、その他の API のアクセス応答情報も同様にアプリケーションへ情報を受け渡した時点で解放されます。

そのため、アクセスサーバ側の IMS キューバッファのサイズは、1つのアクセスサーバに対して、複数のアプリケーションが同時に読込要求を行った場合を考慮して、レコード情報のサイズとそれに付随する DIOSA の制御情報サイズ、アクセス応答のサイズをアプリケーションの多重度数分合計したサイズを用意する必要があります。

アクセス時の各サイズの計算方法を下に記述します。

- 1 件読込時(diosaimreadl)固定長レコード、または、キー指定レコード削除(diosaimdeletexl)のサイズ計算

■計算に必要な情報

(A) 読込対象表の table.conf の record\_size

■計算式

(B) 1 件読込時固定長レコードサイズ = 40 + A + 392

- 複数件読込時(diosaimread)固定長レコードサイズ計算

■計算に必要な情報

(C) 読込対象表の table.conf の record\_size

(D) 読込要求時に指定するバッファサイズ(diosaimread の BuffSize に指定する値)

■計算式

(E) 補正レコードサイズ = (( C + 7 ) / 8) × 8    ※8 バイトバウンダリにあわせませす

- (F) 格納数 =  $D / E$
- (G) 複数件読込時固定長レコードサイズ =  $(40 + E) \times F + 392$

- 1 件/複数件読込時(diosaimread1, diosaimread)可変長レコードサイズ計算

■計算に必要な情報

- (H) 更新対象表の table.conf の record\_size
- (I) 更新対象表に定義された全キーのサイズ合計
- (J) 読込要求時に指定するバッファサイズ  
(diosaimread、または、diosaimread1 の BuffSize に指定する値)

■計算式

- (K) 分割レコードボディ長 =  $H - I - 24$
- (L) 補正レコードサイズ =  $((H + 7) / 8) \times 8$  ※8 バイトバウンダリにあわせませす
- (M) 格納可能分割レコード数 =  $J \div K$  ※少数点以下は繰り上げます
- (N) 可変長レコードサイズ =  $(40 + L) \times M + 392$

- その他 API アクセス応答(diosaimread1, diosaimread, diosaimdeletex1 以外のアクセス応答)のサイズ

■計算式

- (O) アクセス応答のサイズ =  $40 + 392$

上記の各サイズと同時アクセス数により、インメモリサーバ側の IMS キューバッファのサイズを計算します。

■計算に必要な情報

- (P) 最大レコードサイズ = B, G, N, O で最大のもの
- (Q) アクセス数 = 該当 MAP に同時アクセスする可能性のあるアプリケーション数

■計算式

- IMS キューバッファのサイズ(バイト) =  $P \times Q$
- (注)SG ではキロバイト単位で設定する必要があります。
- (注) オーバーフローを抑止するには、IMQUEBUFUNIT に P の値を設定する必要があります。

実際に確保される IMS キューバッファは、MAP ごとのスレッド多重度分確保されるため、  
定義した IMS キューバッファサイズ  $\times$  ( %MAP-MULTI に指定した値 + 1 )  
分の領域が確保されます。

(3)            **ブリッジサーバの IMS キューバッファサイズ計算**

ブリッジサーバの特性として、アプリケーションに対してはアクセスサーバとして振る舞い、アクセスサーバに対しては、アプリケーションとして振舞うため、ブリッジサーバの IMS キューバッファは、両者の計算式を考慮する必要があります。そのため、ブリッジ経由のアクセスでは、(1)を元に計算したサイズ、(2)を元に計算したサイズの大きい方を基準にして、IMS キューバッファのサイズを計算します。

ブリッジサーバ経由での更新アクセス要求を行なった場合は、アプリケーションがコミットまたはロールバックするまで IMS キューバッファ上に保持され、コミットまたはロールバック処理が完了した時に解放されます。

また、エラー時の応答電文返却用領域として、512KB 余分に計上します。

通常は、上述した(1)と(2)の大きいサイズを採用することで十分ですが、ブリッジ経由アクセスが頻繁かつ大量に発生する場合は、(2)の「アプリケーション多重度」の部分を「アプリケーション多重度×ノード数」とする必要があります。本計算式は、他ノードのアプリケーションが全てブリッジ経由アクセスした場合を想定した計算式となりますので大量のメモリを必要とします。ブリッジ経由アクセス量を想定した上でサイズを設定してください。

■計算式

(A) IMS キューバッファのサイズ(バイト) =  
(ブリッジ経由アクセスに対して(1)を元に計算したサイズ、  
または、(2)を元に計算したサイズの大きい方) × 2 + 512  
(注)SG ではキロバイト単位で設定する必要があります。

4.1.4            **アクセスログ蓄積領域サイズ(%ACCESSLOG-BUFSIZE)の目安**

インメモリサーバの性能情報としてアクセスログを採取する場合、情報を格納するために共有メモリを確保します。アクセスログはアクセス要求の種別や更新方法によって、1 アクセス要求でのアクセスログ数が変わりますので、この計算方法は目安であり、格納されるアクセスログ数は前後する可能性があります。

例えば、可変長レコードの更新要求を行なった場合、更新後のレコードイメージがいくつの TAM レコードに分割されたかに応じて出力されるアクセスログの数が変わります。また、複数件読込要求時には、要求を発行したアプリケーションが既に取得しているロックの状態に応じて出力されるアクセスログの数が変わります。

そのため、より精緻に格納するアクセスログ数を調整したい場合は、実際にアプリケーションを動かして、アクセスログの格納状態を確認した後、下記計算式の保存したいアクセス要求数とログ係数(※で記す部分)を調整してください。

アクセスログ蓄積領域サイズ計算方法は下記の通りです。

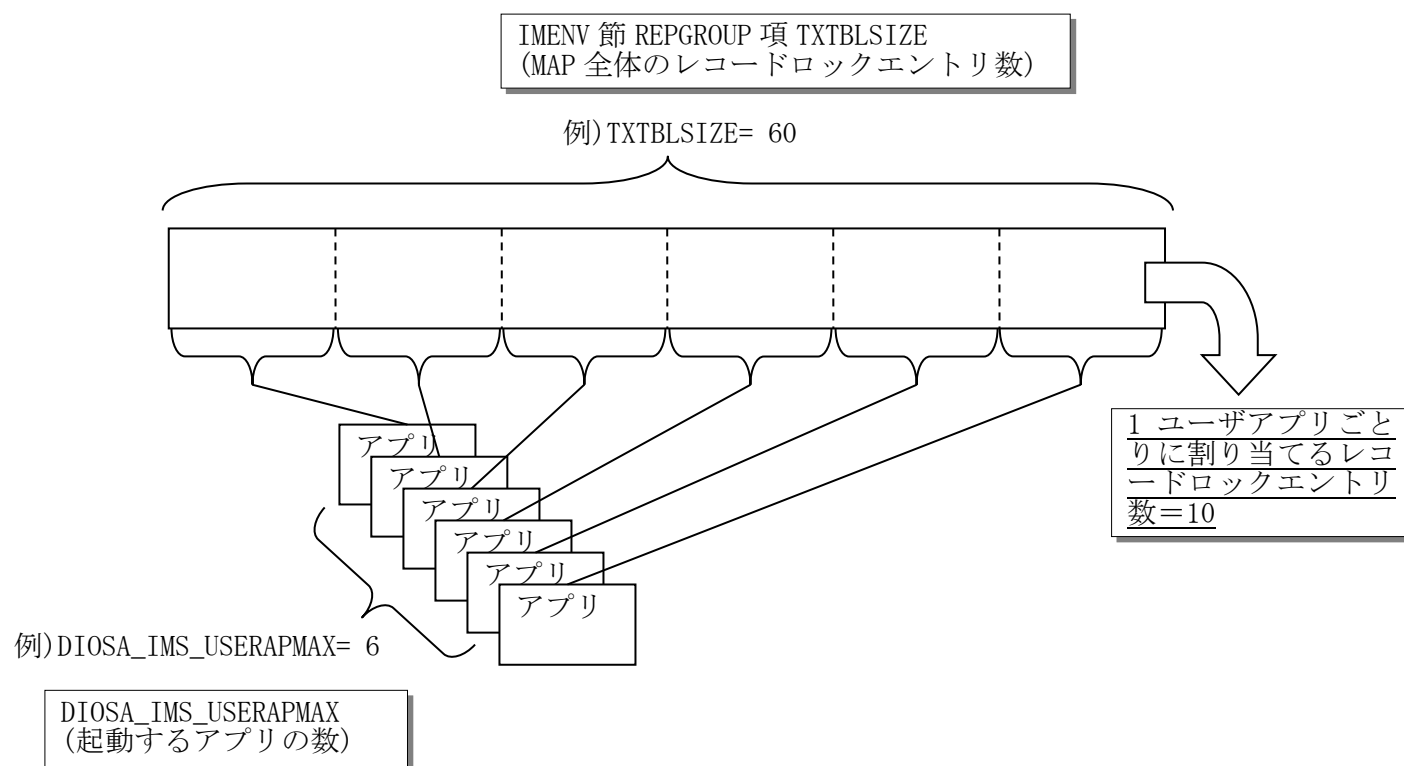
■計算式

(A) アクセスログ蓄積領域サイズ(バイト) = 112 + 64 × 保存したいアクセス要求数 × 5(※)  
※ ログ係数…1 アクセス要求で出力されるアクセスログ数を表す係数  
(注) SG ではキロバイト単位で設定する必要があります。

実際に確保されるアクセスログ蓄積領域サイズは、MAP ごとのスレッド多重度分確保されるため、定義したアクセスログ蓄積領域サイズ × ( %MAP-MULTI に指定した値 + 1 ) 分の領域が確保されます。

## 4.1.5 レコードロックエントリの割り当て数

インメモリサーバはマルチスレッドで効率よく動作するために、1 ユーザアプリが1 トランザクションでロックするレコード数を想定して、管理用領域(レコードロックエントリ)をユーザアプリごとに割り当てます。そのため、想定される1 トランザクション内でロックするレコード数を元に IMENV 節 REPGROUP 項 TXTBLSIZE(環境定義)、DIOSA\_IMS\_USERAPMAX(環境変数)を設定する必要があります。



1 トランザクションで実際にロックするレコード数が、割り当て数よりも大きくなった場合、インメモリサーバは他アプリに割り当てたレコードロックエントリのうち未使用のものを収集し、不足していたアプリに再割り当てします(DIIMS151 のメッセージを出力)。また、全てのエントリが使用中で再割り当て処理ができなかった場合、DIIMS152 のメッセージを出力します。DIIMS151 のメッセージが頻発する場合や、DIIMS152 のメッセージが出力された場合は、DIOSA\_IMS\_USERAPMAX、TXTBLSIZE の設定を見直してください。

## 4.2 環境定義

### 4.2.1 メモリキャッシュ関連 (IMENV)

メモリキャッシュ機能に関して、論理システム単位に定義します。  
各項のパラメータの詳細は、DIOSA\_XTP 環境定義リファレンスを参照してください。

(1) **COMMON 項**

- メモリキャッシュ機能の制御に関する以下の情報を定義します。
- ・ 障害検出時に自動的にマスタ切替を行うか、手動で行うかの指定  
省略又は、自動(AUTO)を指定した場合、MAP 項の TAM の更新ログを蓄積する(JOURNAL=YES)を指定できません。
  - ・ 制御電文再送間隔
  - ・ スレーブへのレプリケーション結果によって、コミットを正常終了する条件の指定
  - ・ マスタ切替時の AP 層の電文保留用トランザクション閉塞制御を自動的に行うか否かの指定  
(電文保留用トランザクション閉塞制御機能については「DIOSA/XTP 利用の手引き－電文保留機能」を参照してください)
- (定義例)

```
%COMMON
    SWITCH          = AUTO
    CTRLSENDINVL    = 10
    ALLOWCOMMIT     = REPGTHALF
    RESTTXIDBLK     = YES
;
```

(2) **FAULTDETECT 項**

- メモリキャッシュ機能の監視及び制御に関する情報を定義します。
- ・ NODEFAULT 項 でノード間通信の監視及び制御に関する情報を定義します。
  - ・ SERVERFAULT 項でアクセスサーバの監視及び制御に関する情報を定義します。
  - ・ BRIDGEFAULT 項でブリッジサーバの監視及び制御に関する情報を定義します。
  - ・ TAMFAULT 項で TAM インスタンスの障害監視及び制御に関する情報を定義します。
- (定義例)

```
%FAULTDETECT
%NODEFAULT
    HLTHCHKINVL     = 3
    HLTHCHKRETRY    = 3
;
%SERVERFAULT
    MSTRESTART      = 0
    SLVRESTART      = 3
    RESTARTTIME     = 60
;
%BRIDGEFAULT
    CHKINVL         = 30
```

```

        RESTART      = 3
        RESTARTTIME = 60
    ;
    %TAMFAULT
        CHKINVL      = 30
        SWITCHRETRY  = 15
        SWITCHINVL   = 2
    ;
;

```

(3)       **NODEFAULT 項**

ノード間通信の監視及び制御に関する以下の情報を定義します。

- AP 層-OLTP 層間で行うヘルスチェック処理間隔、ヘルスチェック処理失敗時のリトライ回数の指定  
OLTP 層ノード障害を自動検出させる場合は、本パラメータを定義してください。  
本パラメータは省略もしくは処理間隔に 0 を定義すると、AP 層-OLTP 層間のヘルスチェック処理は行なわれません。
- AP 層において T パス状態を参照し、T パス障害検出時にノード障害とみなすか否かの指定
- AP 層が検出した OLTP 層障害によって OLTP ノード障害とするかの条件と、条件を満たしノード障害とするまでの間隔の指定

但し、条件に関わらず、全 AP 層が障害検出した場合は、即時にノード障害と判定します。  
(定義例)

```

%NODEFAULT
    HLTHCHKINVL      = 3
    HLTHCHKRETRY     = 3
    TPATHCHK         = YES
    DOWNCNT          = ALL
    DOWNTIME         = 0
;

```

(4)       **SERVERFAULT 項**

アクセスサーバの監視及び制御に関する以下の情報を定義します。

- アクセスサーバの滞留電文数監視間隔とアクセスサーバ滞留電文数しきい値
- アクセスサーバ（マスタ／スレーブ）再起動回数
- アクセスサーバ再起動後に正常起動と判定する時間

(定義例)

```

%SERVERFAULT
    CHKINVL      = 50
    DELAYLIMIT   = 834
    MSTRESTART   = 0
    SLVRESTART   = 3
    RESTARTTIME  = 60
;

```

(5) **BRIDGEFAULT 項**

ブリッジサーバの監視及び制御に関する以下の情報を定義します。

- ・ブリッジサーバ監視間隔とブリッジサーバ再起動回数
- ・ブリッジサーバ再起動後に正常起動と判定する時間

(定義例)

```
%BRIDGEFAULT
      CHKINVL      = 30
      RESTART      = 3
      RESTARTTIME  = 60
;
```

(6) TAMFAULT 項

TAM インスタンスの監視及び制御に関する以下の情報を定義します。

- ・TAM インスタンス監視間隔

TAM コンフィグファイル(timer.conf)のコネクション切断検出までの最大時間

(link\_healthcheck\_timer×(link\_healthcheck\_retry\_count+2))より大きな値を指定することを推奨

※本パラメータは調整が必要です。

- ・マスタ切替失敗時のリトライ回数とリトライ間隔

※本パラメータは調整が必要です。

リトライ間隔×リトライ回数で算出される時間内(3 秒程度)に切替が完了する必要があります。

リトライ間隔は短くし、リトライ回数で調整することを推奨します。

又、インメモリサーバ所在管理プロセスが障害となりマスタ切替の制御が不可能な状況を考慮して、

死活監視機能監視間隔の環境変数を定義変更する (OLTP ノードのみで可) ことを推奨します。

(死活監視環境変数定義例)

DIOSA\_DAM\_BASEINVL = 1 #デーモン死活監視ベースタイム値

DIOSA\_DAM\_MONINVL = 1 (or 2) #デーモン死活監視間隔

詳細は DIOSA\_XTP 環境定義リファレンスを参照してください。

(定義例)

%TAMFAULT		
CHKINVL	=	30
SWITCHRETRY	=	15
SWITCHINVL	=	2
;		

(7) **DEF\_MAP 項**

MAP 項の省略可能パラメータの既定値を定義します。

- ・トランザクション管理テーブルサイズ
- ・IMS キューバッファサイズ
- ・TAM の更新ログを蓄積するか否かを指定

YES にした場合、TAM の更新ログ出力先識別子(txlog.conf の txlog\_id)を diosa\_ims\_jnl\_XXXXX

(xxxxxx: 10 桁未満の場合は 0 で埋めた MAPID を指定) と定義する必要があります。

例 MAPID=1 → txlog\_id = diosa\_ims\_jnl\_0000000001



- ・アクセス統計情報を蓄積するか否かを指定
- ・コミット方式を指定する。

(定義例)

```
%DEF_MAP
    TXTBLSIZE      = 1024
    IMQUEBUFSIZE   = 2448
    JOURNAL        = NO
    STATS          = YES
    COMMITMODE     = NORMAL
    %GROUPCOMMIT   REQNUM = 5,   TMOUT = 0;
    %ACCESSLOG     OUTPUT = YES, BUFSIZE = 1024, BUFCYCL = YES;

;
```

(8)      **REPGROUP 項**

メモリキャッシュ機能のレプリケーショングループに関する以下の情報を定義します。

- ・レプリケーショングループ I D
- ・TAMインスタンス名

TAM コンフィグファイル(tammng.conf)の instance\_name に定義された名称を指定します。

(定義例)

```
%REPGROUP
    ID              = 1
    INSTANCE        = TAMINS001
    %MAP
        ID          = 1
        %HASHRANGE  START = 100, END = 199;
    ;
    %MASTER LNODE = LNOL01, TAMVNODE = TAMLN001MASTER;
    %SLAVE  LNODE = LNOL02, TAMVNODE = TAMLN001SLAVE01;
    %SLAVE  LNODE = LNOL03, TAMVNODE = TAMLN001SLAVE02;
    ;
%REPGROUP
    ID              = 2
    INSTANCE        = TAMINS002
    %MAP
        ID          = 2
        %HASHRANGE  START = 200, END = 299;
    ;
    %MASTER LNODE = LNOL01, TAMVNODE = TAMLN002MASTER;
    %SLAVE  LNODE = LNOL02, TAMVNODE = TAMLN002SLAVE01;
    %SLAVE  LNODE = LNOL03, TAMVNODE = TAMLN002SLAVE02;
    ;
%REPGROUP
    ID              = 11
    INSTANCE        = TAMINS011
    %MAP
```

```

        ID          = 11
        %HASHRANGE START = 300, END = 399;
    ;
    %MASTER LNODE = LNOL02, TAMVNODE = TAMLN011MASTER;
    %SLAVE  LNODE = LNOL03, TAMVNODE = TAMLN011SLAVE01;
    %SLAVE  LNODE = LNOL01, TAMVNODE = TAMLN011SLAVE02;
;
%REPGROUP
    ID          = 12
    INSTANCE    = TAMINS012
    %MAP
        ID          = 12
        %HASHRANGE START = 400, END = 499;
    ;
    %MASTER LNODE = LNOL02, TAMVNODE = TAMLN012MASTER;
    %SLAVE  LNODE = LNOL03, TAMVNODE = TAMLN012SLAVE01;
    %SLAVE  LNODE = LNOL01, TAMVNODE = TAMLN012SLAVE02;
;
%REPGROUP
    ID          = 21
    INSTANCE    = INS021
    %MAP
        ID          = 21
        %HASHRANGE START = 500, END = 599;
    ;
    %MASTER LNODE = LNOL03, TAMVNODE = TAMLN021MASTER;
    %SLAVE  LNODE = LNOL01, TAMVNODE = TAMLN021SLAVE01;
    %SLAVE  LNODE = LNOL02, TAMVNODE = TAMLN021SLAVE02;
;
%REPGROUP
    ID          = 22
    INSTANCE    = TAMINS022
    %MAP
        ID          = 22
        %HASHRANGE START = 600, END = 699;
    ;
    %MASTER LNODE = LNOL03, TAMVNODE = TAMLN022MASTER;
    %SLAVE  LNODE = LNOL01, TAMVNODE = TAMLN022SLAVE01;
    %SLAVE  LNODE = LNOL02, TAMVNODE = TAMLN022SLAVE02;
;

```

(9)           **MAP 項**

MAP に関する以下の情報を定義します。

MAP 項の省略可能なパラメータを省略した際は、DEF\_MAP 項の定義が適用されます。

- MAPID
- トランザクション管理テーブルサイズ
- IMS キューバッファサイズ
- TAM の更新ログを蓄積するか否かを指定

YES にした場合、TAM の更新ログ出力先識別子(txlog.conf の txlog\_id)を diosa\_ims\_jnl\_xxxxx (xxxxxx : 10 桁未満の場合は 0 で埋めた MAPID を指定) と定義する必要があります。

例 MAPID=1 → txlog\_id = diosa\_ims\_jnl\_0000000001

- アクセス統計情報を蓄積するか否かを指定
- コミット方式を指定する。

(定義例)

```
%MAP
      ID              = 1
      TXTBLSIZE       = 1024
      IMQUEBUFSIZE    = 2448
      JOURNAL         = NO
      STATS           = YES
      COMMITMODE      = NORMAL
      %GROUPCOMMIT    REQNUM = 5,   TMOUT = 0;
      %HASHRANGE      START = 100, END = 199;
      %ACCESSLOG      OUTPUT = YES, BUFSIZE = 1024, BUFCYCL = YES;
;
```

(10)           **HASHRANGE 項**

MAP 毎に割り当てるハッシュ値の範囲に関する情報を定義します。

- 開始／終了ハッシュ値

各 MAP に割り当てたハッシュ値の範囲が重複することのないように定義する必要があります。

(定義例)

```
%HASHRANGE
      START = 100
      END   = 199
;
```

(11)           **MASTER 項**

マスタ TAM インスタンスに関する情報を定義します。

- 論理ノード名

マスタ TAM を配置する論理ノード名を指定します。

DIOSAMAP 節 自論理ノードシステム配下の属性が OLTP (%LNODE-TYPE=OLTP) である論理ノード名 (%LNODE-NAME) を指定。 REPGROUP 項内での重複は不可。

- TAM 論理ノード名

TAM インスタンスを識別するための論理ノード名を指定します。

TAM のコンフィグパラメータファイル(tammng.conf)の virtual\_node\_name に該当する名称を指定。  
(定義例)

```
%MASTER
    LNODE = LNOL01
    TAMVNODE = TAMLN001MASTER
;
```

(12) **SLAVE 項**

スレーブ TAM インスタンスに関する情報を定義します。

- ・ 論理ノード名  
スレーブ TAM を配置する論理ノード名を指定します。  
DIOSAMAP 節 自論理ノードシステム配下の属性が OLTP (%LNODE-TYPE=OLTP) である論理ノード名 (%LNODE-NAME) を指定。 REPGROUP 項内での重複は不可。

- ・ TAM 論理ノード名  
TAM インスタンスを識別するための論理ノード名を指定します。

TAM のコンフィグパラメータファイル(tammng.conf)の virtual\_node\_name に該当する名称を指定。  
(定義例)

```
%SLAVE
    LNODE = LNOL02
    TAMVNODE = TAMLN001SLAVE01
;
```

(13) **BRIDGE 項**

ブリッジサーバに関する情報を定義します。

- ・ 多重度
- ・ 再接続間隔
- ・ IMS キューバッファサイズ
- ・ アクセス統計情報を蓄積するか否かを指定

(定義例)

```
%BRIDGE
    MULTI          = 3
    CONNECTINVL    = 10
    IMQUEBUFSIZE   = 48959
    STATS          = YES
;
```

(14) **USERAP 項**

利用者アプリケーションに関わる情報を定義します。

- ・ ハッシュ関数名
- ・ 応答監視タイマ値  
0 を指定した場合は、無限扱いとされ、アクセス要求を行ってからアクセス結果を受信するまで、利用者アプリケーションに制御は戻りません。
- ・ IMS キューバッファサイズ

- ・利用者情報領域長
  - ・利用者情報初期値設定関数名
- (定義例)

```
%USERAP
    HASHEXIT      = HashExit
    REQTIMEOUT    = 30
    IMQUEBUFSIZE  = 6068
    MAXNUM        = 30
    USERAREASIZE  = 16
    USERAREAEXIT  = InitDataExit
;

```

(15)      **GROUPCOMMIT 項**

グループコミットに関する情報を定義します。

REPGROUP 項の MAP 項に GROUPCOMMIT 項が存在しない場合、また、同 GROUPCOMMIT 項のパラメータを省略した際は DEF\_MAP 節で定義した GROUPCOMMIT 項の定義が適用されます。

- ・グループ化するコミット要求数
- ・タイムアウト
- ・コミット要求数の合計がグループ化するコミット要求数 (REQNUM) に満たない場合、IMS キューに対し滞留電文の有無チェックを行うか否かを指定。

(定義例)

```
%GROUPCOMMIT
    REQNUM      = 5
    TMOUT       = 0
    IMQUECHK     = YES
;

```

(16)      **ACCESSLOG 項**

アクセス情報ログの蓄積に関して定義します。

- ・アクセス情報ログを蓄積するか否かを指定
- ・蓄積領域サイズ

アクセス情報の蓄積を指定した場合、本パラメータは 1 以上を指定する必要がある

- ・アクセス情報ログの蓄積領域をサイクリックに使用するか否かを指定

(定義例)

```
%ACCESSLOG
    OUTPUT      = YES
    BUFSIZE     = 1024
    BUFCYCL     = YES
;

```

## 4.2.2 メモリキャッシュ表関連 (IMTABLECONF)

メモリキャッシュ機能が扱う論理表と TAM の物理表に関して定義します。  
各項のパラメータの詳細は DIOSA\_XTP 環境定義リファレンスを参照してください。

(1) **LTABLE 項**

論理表に関する情報を定義します

- ・ 表種別
- ・ 論理表名  
TAM コンフィグファイル (table.conf) の tam\_table\_name を指定
- ・ 論理表の識別 ID

定義例)

```
%LTABLE
    TYPE          = 0
    NAME          = DIOSA_DELAYED_STRM
    ID            = 5
    %RECORDCONF
    :
    :
    ;
```

(2) **RECORDCONF 項**

レコードに関する情報を定義します。

- ・ レコード形式
- ・ プライマリキーに関する情報 (PRIMARYKEY 項)
- ・ セカンダリキーに関する情報 (SECONDARYKEY 項)

(3) **PRIMARYKEY 項**

プライマリキーに関する情報を定義します。

- ・ プライマリキー名  
TAM コンフィグファイル (table.conf) の index\_name1 の名称を指定
- ・ キー位置に関する情報 (KEYDEF 項)

(4) **SECONDARYKEY 項**

セカンダリキーに関する情報を定義する。

- ・ セカンダリキー名  
TAM コンフィグファイル (table.conf) の index\_nam2 の名称を指定
- ・ キー位置に関する情報 (KEYDEF 項)

(5) **KEYDEF 項**

キー位置に関する情報を定義します。

連結キーの場合、TAM コンフィグファイル (table.conf) の定義順と同じ順番で指定

- ・ オフセット
  - ・ キー長
- ((2)-(5)の定義例)

```
%LTABLE
    TYPE      = 0
    NAME      = DIOSA_DELAYED_STRM
    ID        = 5
%RECORDCONF
    TYPE  = FIXED
%PRIMARYKEY
    NAME = DIOSA_DELAYED_STRM_00_IDX
    %KEYDEF  OFFSET = 0, LENGTH = 16 ;
    %KEYDEF  OFFSET = 16, LENGTH = 16 ;
    %KEYDEF  OFFSET = 32, LENGTH = 4 ;
;
%SECONDARYKEY
    NAME = DIOSA_DELAYED_STRM_01_IDX
    %KEYDEF  OFFSET = 0 LENGTH = 16 ;
;
%SECONDARYKEY
    NAME = DIOSA_DELAYED_STRM_04_IDX
    %KEYDEF  OFFSET = 0, LENGTH = 16 ;
    %KEYDEF  OFFSET = 68, LENGTH = 4 ;
    %KEYDEF  OFFSET = 32, LENGTH = 4 ;
;
;
%PTABLE
:
```

(6) **PTABLE 項**

物理表に関する情報を定義します。

- ・物理表名

TAM コンフィグファイル (table.conf) の tam\_table\_name の名称を指定

- ・MAPID

IMENV 節内で定義されている必要があります。

(定義例)

```
%PTABLE
    NAME      = DIOSA_DELAYED_STRM_0000000001
    MAPID     = 1
;
%PTABLE
    NAME      = DIOSA_DELAYED_STRM_0000000002
    MAPID     = 2
;
%PTABLE
    NAME      = DIOSA_DELAYED_STRM_0000000011
    MAPID     = 11
;
```

### 4. 2. 3 DIOSA 関連 (DIOSAMAP)

メモリキャッシュ機能が認識可能な論理ノード、論理システムの構成を定義します。

メモリキャッシュが通信で使用するポート番号を指定する必要があります。

インメモリサーバ所在管理(IIC)のポートは2つ連番が必要です。

又、インメモリサーバ(IMS)のポートは連番で16必要です。

詳細は DIOSA/XTP 利用の手引き、DIOSA/XTP 環境定義リファレンスの記述を参照してください。

(DIOSAMAP 定義例)

```
$DIOSAMAP
%DEFAULT
    PORT_BCM      = 1100
    PORT_CDD      = 1101
    PORT_DLT      = 1102
    PORT_IIC      = 1106
    PORT_IMS      = 1108
;

%LOGSYSTEM
    NAME = LS11
%LNODE
    NAME      = LNAP01
    ID        = 101
    INITSTS   = BLK
    TYPE      = AP
    IPADDR    = 192.168.1.11
    IPCKEY    = 0001
    %TPM TPMNAME = TPMAXX00;
;
%LNODE
    NAME      = LNAP02
    ID        = 102
    INITSTS   = BLK
    TYPE      = AP
    IPADDR    = 192.168.1.12
    IPCKEY    = 0001
    %TPM TPMNAME = TPMAXX00;
;
%LNODE
    NAME      = LNOL01
    ID        = 201
    INITSTS   = BLK
    TYPE      = OLTP
    IPADDR    = 192.168.1.21
    IPCKEY    = 0001
    %TPM TPMNAME = TPMOXX00;
;
%LNODE
    NAME      = LNOL02
```



```

        ID          = 202
        INITSTS     = BLK
        TYPE        = OLTP
        IPADDR      = 192. 168. 1. 22
        IPCKEY      = 0001
        %TPM TPMNAME = TPMOXX00;
;
%LNODE
        NAME        = LNOL03
        ID          = 203
        INITSTS     = ACT
        TYPE        = OLTP
        IPADDR      = 192. 168. 1. 23
        IPCKEY      = 0001
        %TPM TPMNAME = TPMOXX00;
;
%LNODE
        NAME        = LNDB01
        ID          = 301
        INITSTS     = ACT
        TYPE        = DB
        IPADDR      = 192. 168. 1. 31
        IPCKEY      = 0001
;
;
;
```

(1) **DIOSA と TAM のネットワークを分離する場合**

DIOSA（メモリキャッシュ機能）が各ノード間で通信する業務用ネットワークと、TAM がマスタ・スレーブ間のレプリケーション等で利用するネットワークを分離する場合、次のように OLTP ノードの定義に IPADDR2 を追加し、TAM の通信用 IP アドレスを指定してください。これにより、DIOSA 側が通常利用する業務用ネットワークが障害となった際に、TAM 側のネットワークを緊急利用してマスタ切替のための制御用通信を行います。

(DIOSAMAP 定義例)

```
%LNODE
    NAME      = LNOL01
    TYPE      = OLTP
    IPADDR    = 192.168.1.21
    IPADDR2   = 192.168.2.21    ←tammng.conf の ip_memory_backup で指定する値
    ～～省略～～
;
%LNODE
    NAME      = LNOL02
    TYPE      = OLTP
    IPADDR    = 192.168.1.22
    IPADDR2   = 192.168.2.22    ←tammng.conf の ip_memory_backup で指定する値
    ～～省略～～
;
%LNODE
    NAME      = LNOL03
    TYPE      = OLTP
    IPADDR    = 192.168.1.23
    IPADDR2   = 192.168.2.23    ←tammng.conf の ip_memory_backup で指定する値
    ～～省略～～
;
```

# 4. 2. 4 TAM の環境定義

メモリキャッシュ機能は、TAM コンフィグファイル(table.conf)に必ず指定すべき設定値があります。  
下表に挙げました TAM パラメータに関して、設定値に示す値を指定してください。

TAM パラメータ	設定値	備考
record_duplicate_check	no	
dirty_read	no	レプリケーションありの場合のみ必須
backup	yes	レプリケーションありの場合のみ必須
share	yes	
sequence	no	
primary_index_duplicate	no	

詳細は TAM 利用の手引きを参照してください。

# 4. 2. 5      メモリキャッシュ表と TAM 表の定義

メモリキャッシュ表と TAM 表は密接に関係しています。そのため、お互いの定義の認識ずれが無いように定義する必要があります。また、レコードが固定長か可変長かによって設定方法が変わります。

## (1)      固定長レコードのレコードイメージと定義方法

固定長のレコードは、メモリキャッシュ表の定義と TAM 表の定義にレコードイメージの差はありません。アプリケーションのレコードイメージと TAM 表に格納されるレコードイメージは下記のようになります。

アプリケーションが入出力を行うレコードイメージ

PK		SK01		固定長項目	SK02	固定長項目
6	4 Byte	2 Byte	2 Byte	15 Byte	5 Byte	200 Byte

Byte

TAM 表のレコードイメージ

PK		SK01		固定長項目	SK02	固定長項目
6	4 Byte	2 Byte	2 Byte	15 Byte	5 Byte	200 Byte

Byte

このレコードイメージのレコードを格納するメモリキャッシュ表と TAM 表のキー定義は下記のようになります。

メモリキャッシュ表の定義

```
%TABLE
NAME = TABLE01
ID = 1
%RECORDCONF
TYPE = FIXED
%PRIMARYKEY
NAME = PRIMARY_KEY
%KEYDEF      ... ①
    OFFSET = 0
    LENGTH = 6
;
%KEYDEF      ... ②
    OFFSET = 6
    LENGTH = 4
;
;
%SECONDARYKEY
NAME = SECONDARY_KEY01
%KEYDEF      ... ③
    OFFSET = 10
    LENGTH = 2
;
%KEYDEF      ... ④
    OFFSET = 12
    LENGTH = 2
;
;
%SECONDARYKEY
NAME = SECONDARY_KEY02
%KEYDEF      ... ⑤
    OFFSET = 29
    LENGTH = 5
;
;
;
;
```

TAM 表の定義

```
tam_table_name      = TABLE01_0000000001_01

index_name1          = PRIMARY_KEY
keys1                = 0:6, 6:4
                     ①  ②

index_name2          = SECONDARY_KEY01
keys2                = 10:2, 12:2, 0:6, 6:4
                     ③  ④  ↑
                               ※プライマリキーを
                               後ろに付加する

index_name3          = SECONDARY_KEY02
keys3                = 29:5, 0:6, 6:4
                     ⑤  ↑
                               ※プライマリキーを
                               後ろに付加する
```

(2) 可変長レコードのレコードイメージと定義方法

可変長のレコードは、メモリキャッシュ表の定義と TAM 表の定義でレコードイメージが異なります。

アプリケーションのレコードイメージと TAM 表に格納されるレコードイメージは下記のようになります。

アプリケーションが入出力を行うレコードイメージ

PK		SK01		固定長項目	SK02	固定長項目
6	4 Byte	2 Byte	2 Byte	15 Byte	5 Byte	200 Byte

TAM 表のレコードイメージ

制御情報 1	分割通番	制御情報 2	PK	SK01	SK02	PK		SK01		固定長項目	SK02	固定長項目
6 Byte	2 Byte	16 Byte	10 Byte	4 Byte	5 Byte	6 Byte	4 Byte	2 Byte	2 Byte	15 Byte	5 Byte	64 Byte

制御情報 1	分割通番	制御情報 2	PK	SK01	SK02	固定長項目						
6 Byte	2 Byte	16 Byte	10 Byte	4 Byte	5 Byte	96 Byte						

制御情報 1	分割通番	制御情報 2	PK	SK01	SK02	固定長項目				
6 Byte	2 Byte	16 Byte	10 Byte	4 Byte	5 Byte	40 Byte				

このレコードイメージのレコードを格納するメモリキャッシュ表と TAM 表のキー定義は下記のようになります。

メモリキャッシュ表の定義

```
%LTABLE
NAME = TABLE01
ID = 1
%RECORDCONF
TYPE = FIXED
%PRIMARYKEY
NAME = PRIMARY_KEY
%KEYDEF ... ①
    OFFSET = 0
    LENGTH = 6
;
%KEYDEF ... ②
    OFFSET = 6
    LENGTH = 4
;
;
%SECONDARYKEY
NAME = SECONDARY_KEY01
%KEYDEF ... ③
    OFFSET = 10
    LENGTH = 2
;
%KEYDEF ... ④
    OFFSET = 12
    LENGTH = 2
;
;
%SECONDARYKEY
NAME = SECONDARY_KEY02
%KEYDEF ... ⑤
    OFFSET = 29
    LENGTH = 5
;
;
;
```

TAM 表の定義

```
tam_table_name = TABLE01_0000000001_01

index_name1 = PRIMARY_KEY
keys1 = 24:10, 6:2
    ①② ↑
    |   ※可変長レコード時は
    |   最後に制御情報の
    |   分割通番を付加する
    |   ※まとまったキーとして
    |   定義する

index_name2 = SECONDARY_KEY01
keys2 = 34:4, 24:10, 6:2
    ③④ ↑
    |   ※プライマリキーを
    |   後ろに付加する
    |   (分割通番を含む)
    |   ※まとまったキーとして
    |   定義する

index_name3 = SECONDARY_KEY02
keys3 = 38:5, 24:10, 6:2
    ⑤ ↑
    |   ※プライマリキーを
    |   後ろに付加する
    |   (分割通番を含む)
```

## 4.3 ノード間ヘルスチェックの信頼性向上

ノード間ヘルスチェックの間隔を短くする場合、高負荷時に OLTP ノードのインメモリサーバ所在管理デーモンに CPU が割り当てられない事態が発生すると、AP ノードのインメモリサーバ所在管理デーモンへの応答が返却できずに OLTP ノード障害と誤検出される可能性があります。

OLTP ノード障害の誤検出が発生し、ヘルスチェック間隔を長くできない場合には、以下のいずれかで対処することが可能です。ただし、他プロセスへの影響がありますので、この対処を行う場合には影響確認を行ってください。

- pset コマンドにより、diiicdmn プロセスを特定のプロセッサグループへ割り当てる。
- nice コマンドまたは renice コマンドで diiicdmn プロセスの優先度を上げる。

## 第5章 システムの運用

### 5.1 起動・停止

#### 5.1.1 メモリキャッシュの起動

メモリキャッシュの起動は、DIOSA/XTP の基本機能を起動後に行います。

基本機能の起動については、DIOSA/XTP 導入の手引きを参照してください。

##### (1) TAM の準備

利用開始に先立ち、TAM のコンフィグファイルなどを準備する必要があります。

詳細は、InfoFrame Table Access Method 利用の手引きを参照してください。

##### (2) 環境定義オブジェクト作成

環境構築で作成した、DIOSA/XTP の環境定義ファイルは、環境定義オブジェクトに変換することで使用可能となります。各ノードで最初のオブジェクト作成時のみ `diirmadd` コマンド、それ以降のオブジェクト更新は `diirmrep` コマンドを実行して、環境定義オブジェクトを作成してください。

```
> diirmadd -E 環境定義ファイル名
> diirmrep -E 環境定義ファイル名
```

コマンドはメモリキャッシュ起動をおこなう全てのノードで実行する必要があります。

1 回オブジェクトを作成したら、定義内容を変更するまでは DIOSA/XTP を停止したりした場合でも、再作成する必要はありません。

##### (3) メモリキャッシュ起動

メモリキャッシュの起動は、cold 起動時は環境定義オブジェクトから共有メモリを作成、warm 起動時は前回引継ぎ情報から共有メモリを作成し、IMS 所在管理デーモンを起動します。warm 起動時で前回引継ぎ情報が存在しない場合は、自動的に cold モードに切り替えて起動します。

インメモリサーバ所在管理デーモンは、他の起動済みノードから稼動情報を取得した後、OLTP ノードの場合は、マスタの TAM とインメモリサーバを起動します。

TAM 更新ログを SAN に採取する環境では、`-j off` オプションを指定して TAM 更新ログ開始を抑止することにより、ディスクのマウント前にメモリキャッシュを起動することができます。

```
> diiminit -c          (環境定義オブジェクトを使用して起動)
> diiminit             (前回の動作変更コマンドによる変更を引き継いで起動)
> diiminit -c -j off   (環境定義オブジェクトを使用して起動。TAM 更新ログ開始を行わない。)
> diiminit -j off      (環境定義オブジェクトを使用して起動。TAM 更新ログ開始を行わない。)
```

なお、TAM の更新ログを利用する環境における初回起動は、`-j off` を指定した `diiminit` により起動後、

更新ログ保存先識別子の cold 起動、-j on を指定した diimctrl の順で起動を行ってください。

更新ログ保存先識別子起動の詳細は、InfoFrame Table Access Method 更新ログ保存／適用機能利用の手引きを参照してください。

```
> diiminit -c -j off          (TAM 更新ログ開始を抑止したメモリキャッシュ起動)
> 更新ログ保存先識別子の cold 起動
> diimctrl -b -A              (スレーブ起動、TAM 更新ログ開始)
```

#### (4) スレーブ起動、TAM 更新ログ開始

全 OLTP ノードのメモリキャッシュを起動後に、スレーブの TAM およびインメモリサーバを起動します。

また、メモリキャッシュ起動時に TAM 更新ログを抑止していた場合は、ディスクのマウント後に、TAM 更新ログの開始を行います。

スレーブ起動と TAM 更新ログ開始を同時に行う場合や、メモリキャッシュ起動時に TAM 更新ログ開始の抑止を行っていない場合は、-A 指定でスレーブ起動と必要に応じた TAM 更新ログ開始を全レプリケーショングループに対して実行することが可能です。

メモリキャッシュ起動時に TAM 更新ログ開始の抑止を行っている場合で、スレーブ起動と TAM 更新ログ開始のタイミングを分けたい場合には、レプリケーショングループ ID 指定で diimctrl を実行する必要があります。

```
> diimctrl -b -A              (スレーブ起動、TAM 更新ログを抑止しているマスタの TAM 更新ログ開始)
> diimctrl -b -r レプリケーショングループ ID
                               (対象のみのスレーブ起動、または TAM 更新ログ開始)
```

#### (5) マスタの移動

障害または計画停止により 1 ノードだけ停止していた状態から復旧した場合、復旧したノードにはマスタが配置されていない状態になります。必要に応じて、マスタの配置を均等化するため、マスタ切替コマンドによりマスタを移動します。復旧したノード以外の全 OLTP ノードでマスタ切替コマンドを-s 指定で実行することを推奨します。

TAM 更新ログを SAN に採取する環境では、-j off オプションを指定して切替先の TAM 更新ログ開始を抑止する必要があります。この場合、切替先でディスクのマウント後に、diimctrl -b コマンドで TAM 更新ログを開始します。

```
> ditamswap -s                (環境定義でマスタとして定義されたノードへマスタを移動)
> ditamswap -r レプリケーショングループ ID -n 論理ノード名 (指定したノードへマスタを移動)
```



## 5.1.2 メモリキャッシュの停止

メモリキャッシュの停止は、DIOSA/XTP の基本機能の停止前に行ないます。

基本機能の停止については、DIOSA/XTP 導入の手引きを参照してください。

なお、メモリキャッシュの停止を行った場合は、必ず基本機能の停止も行ってください。

### (1) マスタの移動

マスタが存在する OLTP ノードを停止する場合、マスタを移動して停止対象のノードにマスタが存在しない状態にする必要があります。

```
> ditamswap (DIOSA が切替先を自動決定して全マスタを移動)
> ditamswap -r レプリケーショングループ ID -n 論理ノード名 (指定したノードへマスタを移
```

TAM 更新ログを稼動系のみにマウントされている共有ディスクへ出力している場合は、以下の手順でマスタを移動する必要があります。

```
停止ノード > diimctrl -e -m MAPID (MAP を停止。停止ノードがマスタの全 MAP に対して実行)
停止ノード > tamtxlogidstop (更新ログ停止。パラメータは TAM のマニュアル参照)
停止ノード > 更新ログ出力先をアンマウント
切替先ノード> 更新ログ出力先をマウント
停止ノード > ditamswap (全マスタを移動)
切替先ノード> diimctrl -b -m MAPID (MAP を起動。停止ノードがマスタの全 MAP に対して実行)
```

### (2) メモリキャッシュの停止

OLTP ノードの場合、メモリキャッシュの停止は、スレーブのインメモリサーバと TAM を停止後に IMS 所在管理デーモンの停止と共有メモリの解放を行ないます。

AP ノードの場合、メモリキャッシュの停止は、IMS 所在管理デーモンの停止と共有メモリの解放を行ないます。

停止対象ノードにマスタが存在する場合は、通常モードの停止は行なえません。強制モードの場合は、マスタのインメモリサーバは停止しますが、マスタの TAM は停止しません。

```
> diimterm (通常停止)
> diimterm -f (強制停止)
```

## 5.2 環境変更

### 5.2.1 環境定義の変更

環境定義を変更し環境定義置換コマンドを実行することにより、メモリキャッシュ機能の環境を動的に変更することができます。

環境定義パラメータの一覧と、環境を動的に変更可能かどうかについて以下に記します。変更(追加削除含)可否列が不可の場合は、環境定義変更後に起動停止手順に従って DIOSA を再起動(COLD 起動)する必要があります。

#### (1) メモリキャッシュ関連 (IMENV)

節名／項名	パラメータ名／項名	変更(追加削除含)可否	備考
\$IMENV	%COMMON	可	
	%FAULTDETECT	可	
	%DEF_MAP	可	
	%REPGROUP	可	
	%BRIDGE	不可	
	%USERAP	不可	
%COMMON	SWITCH	可	
	CTRLSENDINVL	可	
	ALLOWCOMMIT	可	
	RESTTXIDBLK	可	
%FAULTDETECT	%NODEFAULT	可	
	%SERVERFAULT	可	
	%BRIDGEFAULT	可	
	%TAMFAULT	可	
%NODEFAULT	HLTHCHKINVL	可	
	HLTHCHKRETRY	可	
	TPATHCHK	可	
	DOWNCNT	可	
	DOWNTIME	可	
%SERVERFAULT	CHKINVL	可	
	DELAYLIMIT	可	
	MSTRESTART	可	
	SLVRESTART	可	
	RESTARTTIME	可	
%BRIDGEFAULT	CHKINVL	可	
	RESTART	可	
	RESTARTTIME	可	
%TAMFAULT	CHKINVL	可	
	SWITCHRETRY	可	
	SWITCHINVL	可	
%DEF_MAP	—	—	MAP 項に同じ
%REPGROUP	ID	不可	
	INSTANCE	不可	
	%MAP	可	他の REPGROUP への移動はできない

	%MASTER	不可	
	%SLAVE	可	
%MAP	ID	不可	
	MULTI	不可	
	TXTBLSIZE	不可	
	IMQUEBUFSIZE	不可	
	IMQUEBUFUNIT	不可	
	JOURNAL	不可	
	STATS	可	
	COMMITMODE	可	
	%GROUPCOMMIT	可	
	%HASHRANGE	可	
	%ACCESSLOG	可	
%HASHRANGE	START	可	
	END	可	
%MASTER	LNODE	不可	
	TAMVNODE	不可	
%SLAVE	LNODE	可	
	TAMVNODE	不可	
%BRIDGE	MULTI	可	
	CONNECTINVL	可	
	IMQUEBUFSIZE	不可	
	IMQUEBUFUNIT	不可	
	STATS	可	
	%ACCESSLOG	可	
%USERAP	HASHEXIT	可	
	REQTIMEOUT	可	
	IMQUEBUFSIZE	不可	
	IMQUEBUFUNIT	不可	
	USERAREASIZE	可	
	USERAREAEXIT	可	
%GROUPCOMMIT	REQNUM	可	
	TMOUT	可	
	IMQUECHK	可	
%ACCESSLOG	OUTPUT	可	
	BUFSIZE	不可	
	BUFCYCL	可	

(2)           メモリキャッシュ表関連 (IMTABLECONF)

節名／項名	パラメータ名／項名	変更(追加削除含) 可否	備考
\$IMTABLECONF	%LTABLE	可	
%LTABLE	TYPE	可	
	NAME	不可	
	ID	不可	
	%RECORDCONF	不可	
	%PTABLE	可	

%RECORDCONF	TYPE	可	
	%PRIMARYKEY	不可	
	%SECONDARYKEY	可	
%PRIMARYKEY	NAME	可	
	%KEYDEF	可	
%SECONDARYKEY	NAME	可	
	%KEYDEF	可	
%KEYDEF	OFFSET	可	
	LENGTH	可	
%PTABLE	NAME	可	
	MAPID	不可	

(3)        **DIOSA 関連 (DIOSAMAP)**

節名／項名 (※1)	パラメータ名／項名	変更(追加削除含) 可否	備考
\$DIOSAMAP	%DEFAULT	可	
	%LOGSYSTEM	可	
%DEFAULT	PORT_IIC	不可	
	PORT_IMS	不可	
%LOGSYSTEM	%LNODE	可	
%LNODE	PORT_IIC	不可	
	PORT_IMS	不可	

(※1) メモリキャッシュ機能関連の節／項／パラメータのみ記載

(4)        **TAM の環境定義**

可変長レコードの場合、TAM コンフィグファイル(table.conf)を変更することで、メモリキャッシュ機能の環境定義を変更しなくても環境を変更することができます。

下記の TAM パラメータを変更し 5.2.3 の手順を実施することで、パラメータの変更をメモリキャッシュ機能が認識し、環境が変更されます。

カテゴリ名	パラメータ名
table	record_size
	keysN

## 5.2.2 表の追加／削除

表の追加または削除を行なう場合の環境定義変更は、以下の手順で行ないます。

### (1) 環境定義ファイル修正

IMTABLECONF 節の定義内容を記述したファイルを、変更後の内容に修正します。

変更後、全 OLTP ノードで環境定義オブジェクトファイルを更新します。

```
> diirmrep -E 環境定義ファイル名
```

### (2) TAM 定義変更

表を追加する場合は、環境定義置換コマンドを行なう前に TAM の定義変更を行ないます。

table.conf 修正後、全 OLTP ノードでコンフィグ中間ファイルを置換します。

```
> tamcfgmaint -update table
```

置換後、全 OLTP ノードで TAM ファイルを作成します。

```
> tamcreate 物理表名 [入力ファイル名]
```

作成後、全 OLTP ノードで TAM ファイルをロードします。

```
> tamload -instance TAM インスタンス名 物理表名
```

### (3) TAM 定義反映確認

定義が動作環境に反映されていること(追加した物理表が表示されること)を確認します。

```
> tamstatus -instance TAM インスタンス名 -cat
```

### (4) 表クローズ

定義を削除する表を表制御コマンドによりクローズします。

```
> diimtblopenclose -m MAPID -c 論理表名
```

### (5) 環境定義置換

全ノードで環境定義置換コマンドを通常モードで実行します。

```
> diimchg
```

### (6) 表オープン

定義を追加する表を表制御コマンドによりオープンします。

```
> diimtblopenclose -m MAPID -o 論理表名
```

(7) **TAM 定義変更**

表を削除する場合は、環境定義置換コマンドを行った後に TAM の定義変更を行ないます。  
全 OLTP ノードでロード済みのメモリテーブルを削除します。

```
> tamdrop -instance TAM インスタンス名 物理表名
```

table.conf 修正後、全 OLTP ノードでコンフィグ中間ファイルを置換します。

```
> tamcfgmaint -update table
```

(8) **TAM 定義反映確認**

定義が動作環境に反映されていること (削除した物理表が表示されないこと)を確認します。

```
> tamstatus -instance TAM インスタンス名 -cat
```

## 5.2.3 表の追加削除以外

IMENV 節および IMTABLECONF 節、TAM の環境定義変更は、以下の手順で行ないます。

ノードの追加／削除時は、合わせて DIOSAMAP 節の環境定義変更を行います。

### (1) 環境定義ファイル修正

IMENV 節、IMTABLECONF 節の定義内容を記述したファイルを、変更後の内容に修正します。

必要に応じて、DIOSAMAP 節の定義内容を記述したファイルも、変更後の内容に修正します。

変更後、全ノードで環境定義オブジェクトファイルを更新します。

```
> diirmrep -E 環境定義ファイル名
```

### (2) DIOSAMAP 節の動的変更準備

DIOSAMAP 節を変更した場合は、DIOSAMAP/SYSMAP 動的変更コマンドを動的変更準備モードで実行します。

```
> distesgchg -p
```

### (3) TAM 定義変更

MAP やレプリケーショングループを追加する場合は、環境定義置換コマンドを行なう前に TAM の定義変更を行ないます。

table.conf 修正後、全 OLTP ノードでコンフィグ中間ファイルを置換します。

```
> tamcfgmaint -update table
```

### (4) レプリケーショングループまたは MAP の停止

レプリケーショングループの削除を行なう場合は、対象レプリケーショングループを停止します。

MAP の削除またはハッシュ値変更、IMTABLECONF 節の変更を行なう場合は、対象 MAP または対象 MAP が含まれるレプリケーショングループを停止します。

停止コマンドは、全 OLTP ノードで実行します。

MAP 停止を行なう場合は、マスタのノードを先に実行してください。

```
> diimctrl -e -r レプリケーショングループ ID      (レプリケーショングループの停止)
> diimctrl -e -m MAPID                             (MAP の停止)
```

### (5) 環境定義置換

全ノードで環境定義置換コマンドを通常モードで実行します。

```
> diimchg
```

### (6) ハッシュ値切替

MAP の追加／削除またはハッシュ値変更を行なう場合は、全ノードでの環境定義置換コマンドの実行が完

了後、全 OLTP ノードでハッシュ値切替コマンドを実行します。必須ではありませんが、全ノードでの環境定義置換が完了し、同じ構成になっていることを確認するために、整合性チェックコマンドの実行を行うことを推奨します。整合性チェックコマンドは、任意の 1 ノードで実行します。

```
> diimcheck (整合性チェック)
> diimchghash (ハッシュ値切替)
```

(7) **TAM のロード**

MAP を追加する場合は、MAP 起動前に tamload コマンドによりデータをロードします。  
全 OLTP ノードで TAM ファイルを作成します。

```
> tamcreate 物理表名 [入力ファイル名]
```

作成後、全 OLTP ノードで TAM ファイルをロードします。

```
> tamload -instance TAM インスタンス名 物理表名
```

(8) **TAM 定義反映確認**

定義が動作環境に反映されていることを確認します。MAP、レプリケーショングループを追加した場合は、追加した MAP に対応する物理表が表示されることを確認します。

```
> tamstatus -instance TAM インスタンス名 -cat
```

既存の物理表の定義を変更した場合は、変更後の内容が表示されることを確認します。

```
> tamstatus -instance TAM インスタンス名 -table 物理表名 -cfg
```

(9) **レプリケーショングループまたは MAP の起動**

レプリケーショングループの追加を行なう場合は、対象レプリケーショングループを起動します。  
MAP の追加を行なう場合は、対象 MAP を起動します。

またはハッシュ値変更や IMTABLECONF 節の変更を行なう場合は、SG 動的置換前に停止した対象 MAP またはレプリケーショングループを起動します。

起動コマンドは、全 OLTP ノードで実行します。

MAP 起動を行なう場合は、マスタのノードを先に実行してください。

```
> diimctrl -b -r レプリケーショングループ ID (レプリケーショングループの起動)
> diimctrl -b -m MAPID (MAP の起動)
```

(10) **TAM 定義変更**

MAP やレプリケーショングループを削除する場合は、環境定義置換コマンドを行なった後に TAM の定義変更を行ないます。

全 OLTP ノードでロード済みのメモリテーブルを削除します。

```
> tamdrop -instance TAM インスタンス名 物理表名
```



table.conf 修正後、全 OLTP ノードでコンフィグ中間ファイルを置換します。

```
> tamcfgmaint -update table
```

#### (11) TAM 定義反映確認

定義が動作環境に反映されていることを確認します。MAP、レプリケーショングループを削除した場合は、削除した MAP に対応する物理表が表示されないことを確認します。

```
> tamstatus -instance TAM インスタンス名 -cat
```

#### (12) IMS キュー解放

プロセスの異常停止などにより未使用の状態となった IMS キューの解放を行ないます。

```
> diimquegc
```

#### (13) ブリッジサーバの起動／停止

ブリッジサーバの多重度を変更した場合は、インメモリサーバ起動停止コマンドにより、増やしたブリッジサーバの起動、または減らしたブリッジサーバの停止を行ないます。

```
> diimctrl -B (ブリッジサーバ多重度に合わせてブリッジサーバを起動／停止)
```

#### (14) 削除モードの環境定義置換

全ノードで環境定義置換コマンドを削除モードで実行します。

```
> diimchg -d
```

#### (15) DIOSAMAP 節の動的変更

DIOSAMAP 節を変更した場合は、DIOSAMAP/SYSMAP 動的変更コマンドを動的変更モードで実行します。

```
> distesgchg
```

## 5.3 障害対応

### 5.3.1 ノード障害

#### (1) OLTP ノード障害

##### (a) ノード間ヘルスチェックでの自動検出

AP ノードが存在し、IMENV 節の NODEFAULT 項で HLTHCHKINVL パラメータに 1 以上の値を指定している場合は、OLTP ノード障害を自動的に検出します。

##### (b) 利用者側での検出

利用者側で OLTP ノードの障害監視を行う場合、検出した障害をメモリキャッシュ機能へと通知する必要があります。

OLTP ノード障害を検出後、正常稼働している OLTP ノードでメモリキャッシュ通知コマンドを実行してください。

```
> diimnotify -M lnode_fault -n 障害 OLTP ノード名 (メモリキャッシュ通知コマンド)
```

##### (c) 障害検出後の動作

上記のいずれかの方法によって障害検出が行われた場合、該当 OLTP ノードに配置されているマスタ、及びスレーブは障害状態となります。

IMENV 節の COMMON 項で SWITCH=AUTO の定義を行っている場合は、自動的にマスタ切替が行なわれます。

IMENV 節の COMMON 項で SWITCH=MANUAL の定義を行っている場合は、マスタとしたい OLTP ノードでマスタ昇格コマンドを実行することにより、マスタ切替を行ってください。

```
> ditamswitch -r レプリケーショングループ ID (マスタ昇格コマンド)
```

障害からの復旧時の起動方法は、5.1.1 メモリキャッシュの起動を参照してください。

#### (2) AP ノード障害

AP ノード障害が発生した場合は、残りの正常稼働している AP ノードで OLTP ノード障害を検出します。

AP ノード障害からの復旧時の起動方法は、5.1.1 メモリキャッシュの起動を参照してください。

## 5.3.2 通信パス障害

AP ノードが存在し、IMENV 節の NODEFAULT 項で TPATHCHK パラメータに YES を指定している場合は、通信パス障害を自動的に検出します。

IMENV 節の COMMON 項で SWITCH=AUTO の定義を行っている場合は、自動的にマスタ切替が行なわれます。

IMENV 節の COMMON 項で SWITCH=MANUAL の定義を行い、通信パス障害検出のエラーメッセージが出力された場合は、マスタとしたいノードでマスタ昇格コマンドを実行することにより、マスタ切替を行ってください。

```
> ditamswitch -r レプリケーショングループ ID (マスタ昇格コマンド)
```

障害からの復旧には、まずエラーメッセージから通信パス障害の原因を取り除きます。

次に通信パスの復旧を行い、通信パスが接続されたことを確認します。

通信パスの復旧については、通信制御のマニュアルを参照してください。

最後にスレーブのアクセスサーバを起動します。

```
> diimctrl -b -r レプリケーショングループ ID
```

### 5.3.3 TAM 障害

IMENV 節の TAMFAULT 項で CHKINVL パラメータに 1 以上の値を指定している場合は、TAM 障害を自動的に検出します。

IMENV 節の COMMON 項で SWITCH=AUTO の定義を行っている場合は、自動的にマスタ切替が行なわれます。

IMENV 節の COMMON 項で SWITCH=MANUAL の定義を行い、TAM 障害検出のエラーメッセージが出力された場合は、マスタとしたいノードでマスタ昇格コマンドを実行することにより、マスタ切替を行ってください。

```
> ditamswitch -r レプリケーショングループ ID (マスタ昇格コマンド)
```

障害からの復旧については、以下の手順でスレーブの TAM とアクセスサーバを起動します。

```
> tamstatus -instance TAM インスタンス名 (復旧対象スレーブの TAM 状態確認)
復旧対象スレーブの TAM が稼動状態で、マスタが INACTIVE と表示される場合は、TAM を強制停止
    > tamstop -instance TAM インスタンス名 -emergency (TAM 強制停止)
> diimctrl -b -r レプリケーショングループ ID (スレーブ起動)
```

## 5.3.4 DIOSA 障害

### (1) アクセスサーバ障害

IMS 所在管理デーモンはアクセスサーバ障害を自動的に検出します。

IMENV 節の COMMON 項で SWITCH=AUTO の定義を行っている場合は、自動的にマスタ切替が行なわれます。

IMENV 節の COMMON 項で SWITCH=MANUAL の定義を行い、アクセスサーバ障害検出のエラーメッセージが出力された場合は、マスタとしたいノードでマスタ昇格コマンドを実行することにより、マスタ切替を行ってください。

```
> ditamswitch -r レプリケーショングループ ID (マスタ昇格コマンド)
```

障害からの復旧については、スレーブのアクセスサーバを起動します。

```
> diimctrl -b -r レプリケーショングループ ID
```

### (2) ブリッジサーバ障害

IMENV 節の BRIDGEFAULT 項で CHKINVL パラメータに 1 以上の値を指定している場合は、ブリッジサーバ障害を自動的に検出します。

IMENV 節の COMMON 項で SWITCH=AUTO の定義を行っている場合は、自動的にマスタ切替が行なわれます。

IMENV 節の COMMON 項で SWITCH=MANUAL の定義を行い、ブリッジサーバ障害検出のエラーメッセージが出力された場合は、マスタとしたいノードでマスタ昇格コマンドを実行することにより、マスタ切替を行ってください。

```
> ditamswitch -r レプリケーショングループ ID (マスタ昇格コマンド)
```

障害からの復旧には、まずエラーメッセージからブリッジサーバ障害の原因を取り除きます。

次にブリッジサーバを起動し、正常稼動となったことを確認します。

最後にスレーブのアクセスサーバを起動します。

```
> diimctrl -b -B  
> diimctrl -b -r レプリケーショングループ ID
```

### (3) IMS 所在管理デーモン障害

AP ノードが存在し、IMENV 節の NODEFAULT 項で HLTHCHKINVL パラメータに 1 以上の値を指定している場合は、IMS 所在管理デーモン障害を OLTP ノード障害として自動的に検出します。

IMENV 節の COMMON 項で SWITCH=AUTO の定義を行っている場合は、自動的にマスタ切替が行なわれます。

IMENV 節の COMMON 項で SWITCH=MANUAL の定義を行い、ノード障害検出のエラーメッセージが出力された場合は、マスタとしたいノードでマスタ昇格コマンドを実行することにより、マスタ切替を行ってください。

```
> ditamswitch -r レプリケーショングループ ID (マスタ昇格コマンド)
```

障害からの復旧時の起動方法は、5.1.1 メモリキャッシュの起動を参照してください。  
ただし、5.1.1(4)スレーブ起動の前に diimctrl コマンドを実行して、IMS を停止します。

```
> diimctrl -e -A
```

### 5.3.5 TAM の更新ログを採取する場合の障害時マスタ切替

TAM の更新ログを採取する環境においてマスタ TAM やマスタ TAM が存在するノード障害が発生した場合は、マスタ切替を行った後に、tamsave によるメモリテーブルバックアップを取得し、更新ログ保存先識別子の運用開始をして下さい。更新ログ保存先識別子の運用開始後に、インメモリサーバを再起動してください。

> ditamswitch -r レプリケーショングループ ID	(マスタ昇格コマンド)
> tamsave	(パラメータは TAM のマニュアル参照)
> アーカイブファイル削除	
> tamtxlogidstart (cold モード)	(パラメータは TAM のマニュアル参照)
> diimctrl -b	(インメモリサーバ起動)

### 5.3.6 レプリケーション障害からの復旧

障害が発生したレプリケーショングループについて、切替先として適切なノードが存在しない場合、対象レプリケーショングループはレプリケーション障害となります。この場合、まずマスタノードで TAM の起動状態を確認します。

```
> diimref -T (照会コマンド)
```

#### (1) マスタ TAM が正常稼働の場合

マスタのアクセスサーバから順に起動を行ってください。

```
> diimctrl -b -r レプリケーショングループ ID
```

#### (2) マスタ TAM が正常稼働以外の場合、もしくはマスタがノード障害(照会できない)の場合

対象レプリケーショングループ配下の全 TAM とアクセスサーバを停止します。

```
> diimctrl -e -r レプリケーショングループ ID -s
```

停止対象のノードにおいてメモリキャッシュのコマンドが利用できない場合、手動でプロセス停止してください。TAM の停止方法に関しては、TAM のマニュアルを参照してください。

TAM とアクセスサーバ停止後、マスタの TAM とアクセスサーバから順に起動を行ってください。

```
> diimctrl -b -r レプリケーショングループ ID  
または、任意のスレーブで  
> diimctrl -b -r レプリケーショングループ ID -M
```



## 5.3.7 レプリケーション状態が切替中からの復旧

自動切替ありの場合でも、複数の障害が重なるなど自動切替の継続が難しい特殊な状況に陥り、レプリケーション状態が「切替中」の状態となることがあります。この場合、まず任意のノードで TAM の起動状態を確認します。

```
> diimref -T (照会コマンド)
```

### (1) 切替先が存在する場合

TAM インスタンス状態が「正常稼動」になっているノードが存在する場合は、昇格対象のノードで TAM コマンドを利用してマスタ TAM の状態を確認します。

```
> tamstatus -instance インスタンス名
```

確認の結果マスタが稼動している場合、手動でプロセス停止してください。TAM の停止方法に関しては、TAM のマニュアルを参照してください。

マスタ TAM の停止確認後、昇格対象のノードでマスタ昇格コマンドを実行します。

```
> ditamswitch -r レプリケーショングループ ID (マスタ昇格コマンド)
```

### (2) 切替先が存在しない場合、かつスレーブのノードが正常稼動の場合

任意のスレーブノードでマスタ昇格コマンドを実行してください。

```
> ditamswitch -r レプリケーショングループ ID (マスタ昇格コマンド)
```

実行の結果、レプリケーション障害となります。復旧方法は「5.3.6 レプリケーション障害からの復旧」を参照してください。

### (3) 切替先が存在しない場合、かつ全スレーブのノードが正常稼動以外の場合

対象レプリケーショングループ配下の全 TAM とアクセスサーバを停止します。

スレーブ TAM とアクセスサーバは、IIC デーモンを再起動した上で停止コマンドを実行します。

```
> diimctrl -e -r レプリケーショングループ ID -s
```

マスタ TAM は手動でプロセス停止してください。TAM の停止方法に関しては、TAM のマニュアルを参照してください。

なお、マスタのノードも障害状態となっている場合、マスタのアクセスサーバ停止も手動で行います。

TAM とアクセスサーバ停止後、マスタの TAM とアクセスサーバから順に起動を行ってください。

```
> diimctrl -b -r レプリケーショングループ ID  
または、任意のスレーブで  
> diimctrl -b -r レプリケーショングループ ID -M
```

なおこのとき、警告メッセージ「DIIC427」が出力されることがあります。対象ノードの TAM とアクセスサーバが停止済であることを確認してください。

## 5.3.8 ノード孤立からの復旧

DIOSA ネットワーク障害によってノード障害判定を受けた場合、IMS 所在管理デーモンが起動状態のまま他ノードから孤立するケースが存在します。この状態で DIOSA ネットワークだけが復旧しても、該当ノードの IMS 所在管理デーモンは利用可能となりません。

この場合、孤立したノードの IMS 所在管理デーモンを起動しなおすことにより復旧することができます。ただし IMS 所在管理のテーブルは新規に作成されるため、IMS 所在管理以外の DIOSA 機能がテーブルを参照している場合、該当機能も起動しなおす必要があります。(新規テーブルへのアタッチが必要)

### (1) 停止処理

IMS 所在管理デーモンを停止します。

> diimterm -f	(強制停止)
---------------	--------

孤立したノードにマスタ TAM が存在している場合、該当マスタ TAM を停止する必要があります。必要があれば、停止手順を実行する前にデータの退避等を済ませておいてください。

TAM の停止方法に関しては、TAM のマニュアルを参照してください。

### (2) 起動処理

DIOSA ネットワーク接続が復旧していることを確認したうえで IIC デーモンを起動します。

起動方法は、5.1.1 メモリキャッシュの起動を参照してください。

## 5.4 稼動－待機構成における TAM 監視について

稼動－待機構成においては、DIOSA の自動切替は使わずに、定期的に DIOSA の API で TAM 状態を確認して、異常検出時には TP モニタのフェールオーバーとセットでマスタ切替を行います。

### 5.4.1 環境定義について

DIOSA の自動切替を無効とするために、\$IMENV 節－%COMMON 項の SWITCH パラメータに MANUAL を指定してください。

TAM インスタンスの定期的な監視を行うために、\$IMENV 節－%FAULTDETECT 項－%TAMFAULT 項の CHKINVL パラメータに 1 以上の値を定義してください。

なお、インメモリサーバの障害、およびインメモリサーバ経由の TAM アクセス結果による TAM 障害検出は、環境定義に関わらず実行されます。

### 5.4.2 障害の検出方法について

diosagetmapstat 関数により取得可能です。

TAM 障害またはインメモリサーバ障害が検出され、マスタ切替が必要な場合には、t\_diosa\_getmapstatus 構造体の Status に DIOSA\_REPGRP\_SWITCH\_ABN が返却されます。

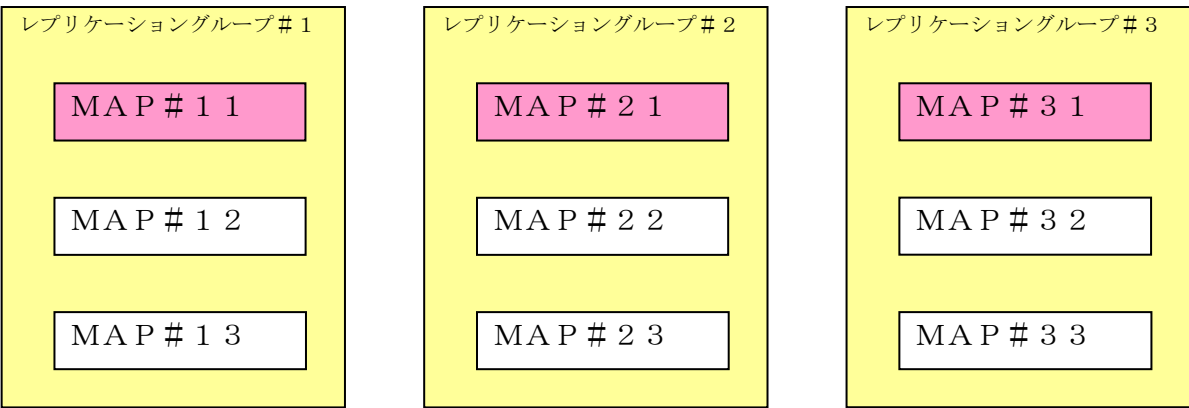
詳細は、DIOSA/XTP API リファレンスを参照してください。

### 5.4.3 補足

1 つのレプリケーショングループで複数の MAP に分割している場合は、どの MAP でも良いので、代表する 1 つの MAP を指定して diosagetmapstat 関数を実行すれば十分です。

レプリケーショングループを分割している場合は、それぞれのレプリケーショングループの代表 MAP に対して diosagetmapstat 関数を実行する必要があります。

例えば、下記のようなレプリケーショングループと MAP の構成では、MAP#11、MAP#21、MAP#31 を指定して diosagetmapstat 関数を実行することで、全ての TAM を監視することができます。



## 5.5 一時ファイルの削除

IMS 所在管理デーモンで実行された TAM コマンドの実行結果等は、\${環境変数 DIOSA\_TMP 指定ディレクトリ}/{論理ノード名}/log 配下のログファイル(diiicdmn.std、diiicdmn.err)として出力され、蓄積されていきます。

このログファイルは自動的に削除されないため、定期的に「:>file」でファイルの中身を削除することを推奨します。

付録A	リソース一覧
付録B	プロセス一覧
付録C	諸元一覧

「DIOSA/XTP 導入の手引き」を参照してください。

# 付録D 環境定義例

## D.1 DIOSA 環境定義例

DIOSA/XTP の環境定義例を記述します。

### メモリキャッシュ基本動作の設定

メモリキャッシュの基本動作の設定を行います。この定義は IMENV 節のみに記述します。

- マスタ切替動作 (SWITCH)  
各種障害検出時にマスタ切替を自動的に行うかどうかを指定します。
- 制御電文再送間隔 (CTRLSENDINVL)  
メモリキャッシュが内部で送信する制御電文の再送間隔を指定します。
- レプリケーション結果判定 (ALLOWCOMMIT)  
メモリ DB のスレーブへのレプリケーション結果によってコミットを正常と判断する基準を指定します。
- トランザクション閉塞 (RESTTXIDBLK)  
マスタ切替を行う場合に新たなトランザクションを処理しないように閉塞することを自動的に行うかどうかを指定します。

### 定義例 (IMENV 節)

```
$IMENV
%COMMON
    SWITCH          = AUTO
    CTRLSENDINVL    = 50
    ALLOWCOMMIT     = REPGTHALF
    RESTTXIDBLK     = NO
;
:
;
```

各種監視動作の設定

メモリキャッシュの各種監視動作の設定を行います。この定義は IMENV 節のみに記述します。

(a) ノード間通信の監視の設定

ノード間通信の監視に関する設定を行います。

- ヘルスチェック間隔 (HLTHCHKINVL)  
AP 層－OLTP 層のノード間で行うヘルスチェック処理の間隔を指定します。
- ヘルスチェックリトライ回数 (HLTHCHKRETRY)  
AP 層－OLTP 層のノード間で行うヘルスチェック処理が失敗した時のリトライする回数を指定します。
- T パス状態チェック有無 (TPATHCHK)  
T パス状態によりノード障害判定を行うかどうかを指定します。
- ノード障害の検出基準 (DOWNCNT)  
OLTP 層のノード障害を検出した AP 層の数によってノード障害と判断する基準を指定します。
- ノード障害と判断する間隔 (TPATHCHK)  
ノード障害の検出基準を満たした状態がそのまま続いた場合にノード障害と判定するまでの間隔を指定します。

定義例 (IMENV 節)

```
$IMENV
%FAULTDETECT
%NODEFAULT
    HLTHCHKINVL      = 20
    HLTHCHKRETRY     = 1
    TPATHCHK         = YES
    DOWNCNT          = GTHALF
    DOWNTIME         = 10
;
:
;
;
```



(b)    **アクセスサーバの監視の設定**

アクセスサーバの監視に関する設定を行います。

- **アクセスサーバ監視間隔 (CHKINVL)**  
アクセスサーバのプロセス状態と滞留電文数を監視する間隔を指定します。
- **アクセスサーバ滞留電文しきい値 (DELAYLIMIT)**  
アクセスサーバへ要求するキューに電文が滞留した場合にアクセスサーバを障害とみなす電文数を指定します。
- **マスタのアクセスサーバ再起動回数 (MSTRESTART)**  
マスタのアクセスサーバが異常終了した場合に再起動する回数を指定します。ここで指定した回数分の再起動を行ってもアクセスサーバが起動できなかった場合は、マスタ切替の対象になります。
- **スレーブのアクセスサーバ再起動回数 (SLVRESTART)**  
スレーブのアクセスサーバが異常終了した場合に再起動する回数を指定します。
- **再起動後に正常起動と判定する時間 (RESTARTTIME)**  
マスタやスレーブのアクセスサーバが異常終了し再起動した後、正常に起動したと判定する時間を指定します。

**定義例 (IMENV 節)**

```
$IMENV
%FAULTDETECT
  %SERVERFAULT
    CHKINVL      = 10
    DELAYLIMIT   = 100
    MSTRESTART   = 0
    SLVRESTART   = 5
    RESTARTTIME  = 3
  ;
;
;
;
```

(c)   ブリッジサーバの監視の設定

ブリッジサーバの監視に関する設定を行います。

- ブリッジサーバ監視間隔 (CHKINVL)  
ブリッジサーバのプロセス状態を監視する間隔を指定します。
- ブリッジサーバ再起動回数 (RESTART)  
ブリッジサーバが異常終了した場合に再起動する回数を指定します。ここで指定した回数分の再起動を行ってもブリッジサーバが起動できなかった場合で、全てのブリッジサーバが異常終了してしまった場合は、マスタ切替の対象になります。
- 再起動後に正常起動と判定する時間 (RESTARTTIME)  
ブリッジサーバが異常終了し再起動した後、正常に起動したと判定する時間を指定します。

定義例 (IMENV 節)

```
$IMENV
%FAULTDETECT
  %BRIDGEFAULT
    CHKINVL      = 10
    RESTART      = 5
    RESTARTTIME  = 3
  ;
;
;
;
```

(d)   メモリ DB 監視の設定

メモリ DB の監視に関する設定を行います。

- メモリ DB インスタンス監視間隔 (CHKINVL)  
メモリ DB のインスタンスの状態を監視する間隔を指定します。
- マスタ切替リトライ回数 (SWITCHRETRY)  
メモリ DB のマスタ切替処理失敗時にリトライする回数を指定します。
- マスタ切替リトライ間隔 (SWITCHINVL)  
メモリ DB のマスタ切替処理失敗時にリトライする間隔を指定します。

定義例 (IMENV 節)

```
$IMENV
%FAULTDETECT
  %TAMFAULT
    CHKINVL      = 10
    SWITCHRETRY  = 0
    SWITCHINVL   = 0
  ;
;
;
;
```

レプリケーショングループの設定

メモリ DB の内容を別ノードにレプリケーションして、同じ情報を保持するレプリケーショングループの設定を行います。この定義は IMENV 節のみに記述します。

- メモリ DB のインスタンス名 (INSTANCE)  
接続するメモリ DB のインスタンス名を指定します。
- 論理ノード名 (LNODE、TAMVNODE)  
マスタのメモリ DB とスレーブのメモリ DB を配置する論理ノードを指定します。  
論理ノードは DIOSA の論理ノード (LNODE) とメモリ DB の論理ノード (TAMVMODE) の両方を指定します。
- 配下の MAP 情報 (%MAP)  
レプリケーショングループ配下で管理する MAP の情報も指定します。(詳細は後述します。)

定義例 (IMENV 節)

```
$IMENV
  %REPGROUP
    ID          = 1
    INSTANCE    = instance01
    %MAP
      :
    ;
    %MASTER
      LNODE      = Node01
      TAMVNODE   = vNode01
    ;
    %SLAVE
      LNODE      = Node02
      TAMVNODE   = vNode02
    ;
    %SLAVE
      LNODE      = Node03
      TAMVNODE   = vNode03
    ;
  ;
;
```

ここで設定されている接続するメモリ DB のインスタンス名とメモリ DB の論理ノード (TAMVNODE) はメモリ DB 側の定義に記述されている必要があります。

また、論理ノード (LNODE) は DIOSAMAP 節に記述されている必要があります。

MAP の設定

メモリ DB へのアクセス管理を行う MAP の設定を行います。この定義は、IMENV 節と IMTABLECONF 節の両方に記述する必要があります。

- アクセススレッド多重度 (MULTI)  
該当 MAP にアクセスするためのアクセススレッド多重度を指定します。  
MAP ごとに ( MULTI に指定した値 + 1 ) 個のスレッドが起動します。
- MAP 内で管理可能なレコード数 (TXTBLSIZE)  
複数のアプリケーションが同時にアクセスした場合を考慮し、MAP 内で管理する最大レコード数を指定します。
- MAP 内でアプリケーションにアクセス結果の内容を返却するためのバッファサイズ (IMQUEBUFSIZE, IMQUEBUFUNIT)  
複数のアプリケーションが同時にアクセスした場合を考慮し、アクセス結果 (レコード内容も含む) を格納するためのバッファ領域のサイズを IMQUEBUFSIZE に指定します。  
「指定サイズ × ( MULTI パラメータ指定値 + 1 )」のバッファが確保されます。
- メモリ DB のジャーナル機能の有無 (JOURNAL)  
障害時の復旧を行うためのメモリ DB の更新ログを採取するかどうかを指定します。
- アクセス統計情報の蓄積有無 (STATS)  
アクセス種別毎の統計情報 (TAT) の蓄積を行うかどうかを指定します。
- コミット方式 (COMMITMODE)  
コミット要求毎にコミットを行うか複数のコミット要求を一つのコミットにまとめるかを指定します。
- ハッシュ値の範囲 (%HASHRANGE)  
該当 MAP に割り当てるハッシュ値の範囲を指定します。
- アクセスログ (%ACCESSLOG)  
各アクセスの詳細な性能情報と処理結果情報の蓄積有無、格納するバッファサイズ、バッファのサイクリック使用の有無を指定します。  
実際に確保されるバッファサイズは、「指定サイズ × ( MULTI パラメータ指定値 + 1 )」である。

定義例 (IMENV 節)

\$IMENV		
%REPGROUP		
:		
%MAP		
ID	=	1
MULTI	=	2
TXTBLSIZE	=	5000
IMQUEBUFSIZE	=	81920
IMQUEBUFUNIT	=	8
JOURNAL	=	YES
STATS	=	YES
COMMITMODE	=	GROUP
%GROUPCOMMIT		
REQNUM	=	5
TMOUT	=	4000

```
        IMQUECHK          = YES
    ;
    %HASHRANGE
        START    = 1000
        END      = 1999
    ;
    %ACCESSLOG
        OUTPUT    = YES
        BUFSIZE   = 10240
        BUFCYCL   = YES
    ;
    ;
    :
    ;
    ;
```

IMTABLECONF 節には、メモリ DB に定義した物理表をどの MAP で扱うかを指定します。

定義例(IMTABLECONF 節)

```
$IMTABLECONF
    %LTABLE
    :
    %PTABLE
        NAME          = PTABLE01
        MAPID         = 1
    ;
    ;
    ;
```

ブリッジサーバの設定

ノードを跨った時のメモリ DB へのアクセス管理を行うブリッジサーバの設定を行います。この定義は、IMENV 節に記述する必要があります。

- ブリッジサーバ数(MULTI)  
ノードを跨ったアクセスを中継するブリッジサーバのプロセス数を指定します。
- 再接続間隔(CONNECTINVL)  
別のノードとの接続を要求する間隔を指定します。
- ブリッジがアプリケーションにアクセス結果の内容を返却するためのバッファサイズ(IMQUEBUFSIZE, IMQUEBUFUNIT)  
複数のアプリケーションが同時にアクセスした場合を考慮し、アクセス結果(レコード内容も含む)を格納するためのバッファ領域のサイズを IMQUEBUFSIZE に指定します。
- アクセス統計情報の蓄積有無(STATS)  
ブリッジサーバにおいてアクセス処理を中継する処理の統計情報(TAT)の蓄積を行うかどうかを指定します。
- アクセスログ(%ACCESSLOG)  
ブリッジサーバにおいてアクセス処理を中継する処理の詳細な情報の蓄積有無、格納するバッファサイズ、バッファのサイクリック使用の有無を指定します。

定義例(IMENV 節)

```
$IMENV
%BRIDGE
    MULTI          = 3
    CONNECTINVL    = 60
    IMQUEBUFSIZE    = 81920
    IMQUEBUFUNIT    = 8
    STATS          = YES
%ACCESSLOG
    OUTPUT         = YES
    BUFSIZE        = 10240
    BUFCYCL        = YES
;
;
;
```

アプリケーションの設定

アプリケーション上のアクセス用 API の設定や MAP を振り分けるためのハッシュ関数の設定を行います。  
この定義は、IMENV 節に記述する必要があります。

- ハッシュ関数名 (HASHEXIT)  
メインキーをハッシュして、処理する MAP を振り分けるためのハッシュ関数の名前を指定します。
- 応答監視タイマ値 (REQTIMEOUT)  
アプリケーションがアクセス用 API を呼び出した際に、アクセスサーバ側から応答が来ない場合にタイムアウトする時間を指定します。
- アプリケーションがアクセス要求するためのバッファサイズ (IMQUEBUFSIZE, IMQUEBUFUNIT)  
アプリケーションがトランザクション内でアクセスするレコードのサイズを考慮してバッファのサイズを IMQUEBUFSIZE に指定します。
- 利用者情報領域の生成有無 (USERAREASIZE、USERAREAEXIT)  
利用者が格納したい情報領域を予め確保するかどうかを指定します。また、必要であれば、その領域を初期化する関数名を指定します。

定義例(IMENV 節)

```
$IMENV
%USERAP
    HASHEXIT      = hash_func
    REQTIMEOUT    = 1000
    IMQUEBUFSIZE  = 10240
    IMQUEBUFUNIT  = 8
    USERAREASIZE = 128
    USERAREAEXIT = user_area_init
;
;
```

表の設定

メモリ DB に用意する表の設定を行います。この定義は、IMTABLECONF 節に記述する必要があります。

(e) 論理表の設定

メモリ DB に用意する論理表の設定を行います。

- 表種別 (TYPE)  
表をグルーピングしたい場合に表種別を指定します。  
区別が必要なければ省略可能です。
- 論理表名 (NAME)  
論理表の名前を指定します。
- 論理表 ID (ID)  
論理表を識別するための ID を指定します。
- レコード情報 (%RECORDCONF)  
表に格納されるレコードのキー情報を指定します。

定義例 (IMTABLECONF 節)

```
$IMTABLECONF
%LTABLE
  TYPE      = 1
  NAME      = LTABLE01
  ID        = 1
%RECORDCONF
  :
  ;
;
;
```



(f)    **レコード情報の設定**

メモリ DB に格納されるレコードのキー情報の設定を行います。

- **レコード形式 (TYPE)**  
レコードの長さを固定とするか可変とするかを指定します。
- **プライマリキー名 (%PRIMARYKEY-NAME)**  
プライマリキーの名前を指定します。メモリ DB のキー情報と同じものを指定する必要があります。
- **セカンダリキー名 (%SECONDARYKEY-NAME)**  
セカンダリキーの名前を指定します。メモリ DB のキー情報と同じものを指定する必要があります。
- **キー位置、サイズ情報 (%KEYDEF)**  
表に格納されるレコードのキーの開始位置と長さを指定します。

※ メモリ DB のキー位置、サイズ情報に同様のものを指定する必要がありますが、可変長・固定長で設定する値が変わります。

**定義例 (IMTABLECONF 節 (固定長))**

```
$IMTABLECONF
%LTABLE
:
%RECORDCONF
TYPE  = FIXED
%PRIMARYKEY
NAME  = PrimaryKey01
%KEYDEF
      OFFSET      = 5
      LENGTH      = 10
;
;
%SECONDARYKEY
NAME  = SecondaryKey02
%KEYDEF
      OFFSET      = 16
      LENGTH      = 10
;
;
;
;
;
```

定義例(IMTABLECONF 節(可変長))

```
$IMTABLECONF
%LTABLE
:
%RECORDCONF
TYPE  = VARLEN
%PRIMARYKEY
NAME  = PrimaryKey01
%KEYDEF
      OFFSET      = 5
      LENGTH      = 10
;
;
%SECONDARYKEY
NAME  = SecondaryKey02
%KEYDEF
      OFFSET      = 16
      LENGTH      = 10
;
;
;
;
;
```

(g) 物理表の設定

メモリ DB に定義した表の設定を行います。

- 物理表名 (NAME)  
メモリ DB に定義した表名を指定します。
- MAPID (MAPID)  
MAP の設定で記述したようにどの MAP で物理表を管理するかを指定します。

定義例(IMTABLECONF 節)

```
$IMTABLECONF
%LTABLE
:
%PTABLE
NAME          = PTABLE01
MAPID         = 1
;
;
;
```

## D.2 TAM 環境定義例

メモリ DB の TAM の環境定義例を記述します。

### TAM のインスタンスの設定

メモリ DB の TAM のインスタンスに関する設定を行います。この定義は `tammng.conf` に記述します。

#### (a) クラスタの設定

TAM のクラスタ構成の設定を行います。

- マルチインスタンス機能使用有無(`multi_instance`)

TAM のインスタンスをマルチインスタンス化するかを指定します。

- クラスタ名(`cluster_name`)

TAM のクラスタの名前を指定します。

- インスタンスグループ(`instance_groupN`) (N:1～の数値)

グループ化する TAM のインスタンスを指定します。

- ノードグループ(`node_groupN`) (N:1～の数値)

グループ化する TAM のインスタンスが動作するノードを指定します。

#### 定義例(`tammng.conf`)

```
[cluster]
multi_instance = yes
cluster_name   = instance
instance_group1 = (instance01@tam01, instance01@tam02, instance01@tam03)
instance_group2 = (instance02@tam01, instance02@tam02, instance02@tam03)
instance_group3 = (instance03@tam01, instance03@tam02, instance03@tam03)
node_group1     = (Node01, Node02, Node03)
:
```

## (b) ノードの設定

TAM が動作するノードの設定を行います。

- ノード名 (node\_name)

TAM が動作するノードの名前を指定します。DIOSA 環境定義のレプリケーショングループ設定において、このノード名を設定する。

- インスタンス間データ転送用 IP アドレス (ip\_memory\_backup)

TAM のインスタンス間でデータ転送を行うための IP アドレスを指定します。

- バックアップ連携有無 (do\_backup)

バックアップ連携 (レプリケーション) を行うかどうかを指定します。

- メモリ DB のトランザクション制御用の共有メモリサイズ (tx\_shm\_table\_size)

メモリ DB のトランザクション制御に使用する共有メモリサイズを指定します。

### 定義例 (tammng.conf)

```
[node]
node_name          = Node01
ip_memory_backup   = (192.168.1.1)
do_backup          = yes
tx_shm_table_size  = 50
:
```

## (c) 論理ノードの設定

TAM が動作する論理ノードの設定を行います。

- ノード名 (virtual\_node\_name)

TAM が動作する論理ノードの名前を指定します。DIOSA 環境定義のレプリケーショングループ設定において、この論理ノード名を設定する。

- レプリケーション構成の役割 (backup\_role)

レプリケーション構成において、マスタとして動作するかスレーブとして動作するかを指定します。

### 定義例 (tammng.conf)

```
[vnode]
virtual_node_name   = vNode01
backup_role         = master
:
```

(d)    **インスタンスの設定**

TAM インスタンスの設定を行います。

- インスタンス名 (instance\_name)  
TAM インスタンスの名前を指定します。DIOSA 環境定義のレプリケーショングループ設定にこの名前を使用する。
- 動作物理ノード名 (base\_node)  
TAM インスタンスが動作する物理ノード名を指定します。
- メモリテーブル用共有メモリサイズ (i\_init\_shm\_size)  
TAM インスタンスが起動時に確保するメモリテーブル用の共有メモリ 1 つのサイズを指定します。
- メモリテーブル用共有メモリ個数 (i\_init\_shm\_num)  
TAM インスタンスが起動時に確保するメモリテーブル用の共有メモリの個数を指定します。
- メモリテーブル用共有メモリ最大個数 (i\_max\_shm\_num)  
TAM インスタンスが拡張時にメモリテーブル用の共有メモリの個数の最大値を指定します。
- メモリテーブル用共有メモリ拡張サイズ (i\_shm\_extend\_size)  
TAM インスタンスがメモリテーブル用の共有メモリを拡張する時のサイズを指定します。
- ノード間通信用ポート番号 (i\_port)  
TAM インスタンスがノード間で通信を行う際のポート番号を指定します。
- ノード間通信用ポート番号の範囲 (i\_port\_range)  
TAM インスタンスがノード間で通信を行う際のポート番号の開始からの範囲を指定します。
- TAM 起動チェック用ポート番号 (i\_start\_check\_port)  
TAM インスタンスが起動をチェックするポート番号を指定します。
- データ転送を行うデーモンの同時処理可能な転送要求数 (i\_max\_proc\_request\_num)  
TAM インスタンス内でデータ転送を行うデーモンの同時処理可能な転送要求数を指定します。
- データ転送を行う際にデータを分割するサイズ (i\_send\_piece\_size)  
TAM インスタンス内でデータ転送を行うデーモンがデータ転送を行う際にデータ分割するサイズを指定します。

**定義例 (tammng.conf)**

```
[instance]
instance_name      = instance01
base_node          = Node01
i_init_shm_size    = 200
i_init_shm_num     = 1
i_max_shm_num      = 500
i_shm_extend_size  = 1
i_port             = 16980
i_port_range       = 5
i_start_check_port = 16990
i_max_proc_request_num = 20
i_send_piece_size  = 8192
:
```

TAM のメモリテーブルの設定

メモリ DB の TAM のメモリテーブル(物理表)に関する設定を行います。この定義は table.conf に記述します。

(e)   メモリテーブル共通の設定

メモリ DB の TAM のメモリテーブルに共通する情報の設定を行います。

- 更新ログサイズ(update\_log\_size)  
メモリ DB の更新ログ情報を保持するメモリ領域のサイズを指定します。
- 更新ログ拡張サイズ(update\_log\_extend\_size)  
メモリ DB の更新ログ情報を保持するメモリ領域が不足した際に動的に拡張するサイズを指定します。
- 更新ログ最大サイズ(max\_update\_log\_size)  
メモリ DB の更新ログ情報を保持するメモリ領域の最大サイズを指定します。
- 性能情報トレースエントリ数(trace\_entry)  
メモリ DB の性能情報をトレースするエントリ数を指定します。
- レコード重複チェック有無(record\_duplicate\_check)  
レコードが重複するかどうかのチェック有無を指定します。DIOSA では必ず no を設定してください。
- ダーティリード可否(dirty\_read)  
更新中のレコードをダーティリード可能とするかを指定します。レプリケーションするメモリテーブルでは必ず no を設定してください。

定義例(table.conf)

[common]		
update_log_size	=	1
update_log_extend_size	=	1
max_update_log_size	=	50
trace_entry	=	0
record_duplicate_check	=	no
dirty_read	=	no
:		

(f)    **メモリテーブル毎の設定**

メモリ DB の TAM のメモリテーブルに関する情報の設定を行います。

- **メモリテーブル名 (tam\_table\_name)**  
メモリテーブルの名前を指定します。DIOSA 環境定義の物理表の設定にこの名前を使用する。
- **TAM ファイル名 (tam\_table\_file)**  
tamcreate コマンド時に作成する TAM ファイル名を指定します。
- **メモリテーブルのバックアップ有無 (backup)**  
バックアップ対象のメモリテーブルかどうかを指定します。レプリケーションするメモリテーブルでは必ず yes を設定してください。
- **メモリテーブルの共有利用可否 (share)**  
メモリテーブルの共有利用を可能とするかどうかを指定します。DIOSA では必ず yes を設定してください。
- **メモリテーブルのレコードのシーケンス番号付与の有無 (sequence)**  
メモリテーブルに格納されるレコードにシーケンス番号を付与するかどうかを指定します。DIOSA では必ず no を設定してください。
- **プライマリキー重複の可否 (primary\_index\_duplicate)**  
プライマリキーの重複を許可するか否かを指定します。DIOSA では必ず no を設定してください。
- **インデックス領域の予備確保のサイズ (index\_reserve\_size)**  
レコードの追加に備えて、インデックス領域を余分に確保しておく際のサイズを指定します。
- **レコード領域の予備確保のサイズ (data\_reserve\_size)**  
レコードの追加に備えて、レコードデータ領域を余分に確保しておく際のサイズを指定します。
- **予備領域不足時に動的にメモリ追加の可否 (auto\_extend)**  
予備領域が不足した際に動的にメモリ追加を行うかどうかを指定します。
- **予備領域不足時に動的に追加するメモリサイズ (auto\_extend\_size)**  
予備領域が不足した際に動的にメモリ追加する際のサイズを指定します。
- **予備領域不足時に動的に追加する最大サイズ (max\_auto\_extend\_size)**  
予備領域が不足した際に動的にメモリ追加する最大サイズを指定します。
- **格納される 1 レコードのサイズ (record\_size)**  
メモリテーブルに格納される 1 レコードのサイズを指定します。
- **インデックス数 (index\_number)**  
メモリテーブルのインデックスの数を指定します。
- **1 番目のインデックス (index\_name1)**  
メモリテーブルの 1 番目のインデックスの名前を指定します。DIOSA 環境定義のレコード情報のプライマリキー名の設定にこの名前を使用する。
- **1 番目のインデックスとして用意するエントリ数 (index\_entry\_num1)**  
メモリテーブルの 1 番目のインデックスに使用するエントリ数を指定します。
- **1 番目のインデックスとして用意するキーの位置とサイズ (keys1)**  
メモリテーブルの 1 番目のインデックスのキーの開始位置とサイズを指定します。DIOSA 環境定義のレコード情報のプライマリキーのキー位置、サイズ情報設定にこの値を使用する。可変長時は、レコードの 24 バイト目以降からの連続領域になり、制御用のキーの記述が必要となります。

※固定長と可変長で記述方法が異なります。

- 1 番目のインデックス領域を動的拡張する際の動的拡張サイズ(auto\_extend\_key\_num1)  
メモリテーブルの 1 番目のインデックス領域を動的に拡張する際のサイズを指定します。
  - 1 番目のインデックスに関して不要になったキーの自動削除(auto\_release\_index1)  
メモリテーブルの 1 番目のインデックスにあるキーが、メモリテーブルの更新によって不要になった場合、コミットまたはロールバックのタイミングで削除するか否かを指定します。  
**DIOSA では yes に設定することを強く推奨します。**
  - N 番目のインデックス(index\_nameN) (N : 2～の数値)
  - N 番目のインデックスとして用意するエントリ数(index\_entry\_numN) (N : 2～の数値)
  - N 番目のインデックスとして用意するキーの位置とサイズ(keysN) (N : 2～の数値)
  - N 番目のインデックス領域を動的拡張する際の動的拡張サイズ(auto\_extend\_key\_numN) (N : 2～の数値)  
メモリテーブルの N 番目のインデックスに関する情報を指定します。DIOSA 環境定義のレコード情報のセカンダリキーに関する情報と同様です。キー位置とサイズ情報にはプライマリキーの情報も付加します。  
可変長時は、プライマリキー格納位置の直後からの連続領域になり、制御用のキーの記述が必要となります。
- ※キー位置とサイズ情報は、固定長と可変長で記述方法が異なります。
- N 番目のインデックスに関して不要になったキーの自動削除(auto\_release\_indexN)  
メモリテーブルの N 番目のインデックスにあるキーが、メモリテーブルの更新によって不要になった場合、コミットまたはロールバックのタイミングで削除するか否かを指定します。  
**DIOSA では yes に設定することを強く推奨します。**
  - メモリテーブルの更新ログのディスク保存有無(txlog)  
メモリテーブルの更新ログをディスクに保存するかどうかを指定します。DIOSA 環境定義の MAP 情報のジャーナル機能の有無に設定した値と同じ値を設定してください。

**定義例(table.conf(固定長))**

```
[table]
tam_table_name      = PTABLE01
tam_table_file      = /DIOSA/TAM/table/PTABLE01
backup              = yes
share               = yes
sequence            = no
primary_index_duplicate = no
index_reserve_size  = 1
data_reserve_size   = 1
auto_extend         = yes
auto_extend_size    = 1
max_auto_extend_size = 1
record_size         = 8192
index_number        = 2
index_name1         = PrimaryKey01
index_entry_num1    = 100
keys1               = 5:10
auto_extend_key_num1 = 100
auto_release_index1 = yes
index_name2         = SecondaryKey02
index_entry_num2    = 100
```



keys2	= 16:10, 5:10
auto_extend_key_num2	= 100
auto_release_index2	= yes
txlog	= yes
:	

定義例(table.conf(可変長))

```
[table]
tam_table_name      = PTABLE01
tam_table_file      = /DIOSA/TAM/table/PTABLE01
backup              = yes
share               = yes
sequence            = no
primary_index_duplicate = no
index_reserve_size  = 1
data_reserve_size   = 1
auto_extend         = yes
auto_extend_size    = 1
max_auto_extend_size = 1
record_size         = 8192
index_number        = 2
index_name1         = PrimaryKey01
index_entry_num1    = 100
keys1               = 24:10, 6:2
auto_extend_key_num1 = 100
auto_release_index1 = yes
index_name2         = SecondaryKey02
index_entry_num2    = 100
keys2               = 34:10, 24:10, 6:2
auto_extend_key_num2 = 100
auto_release_index2 = yes
txlog               = yes
:
```

固定長

アプリケーションが入出力を行うレコードイメージ

	PrimaryKey01 10 バイト	SecondaryKey02 10 バイト	
--	------------------------	--------------------------	--

TAM 表のレコードイメージ

	PrimaryKey01 10 バイト	SecondaryKey02 10 バイト	
--	------------------------	--------------------------	--

可変長

アプリケーションが入出力を行うレコードイメージ

	PrimaryKey01 10 バイト	SecondaryKey02 10 バイト	
--	------------------------	--------------------------	--

TAM 表のレコードイメージ

DIOSA 制御情報	PrimaryKey01 10 バイト	SecondaryKey02 10 バイト		PrimaryKey01 10 バイト	SecondaryKey02 10 バイト	
DIOSA 制御情報	PrimaryKey01 10 バイト	SecondaryKey02 10 バイト				

可変長時は、先頭に 24 バイトの DIOSA 制御情報が付加されます。  
DIOSA 制御情報以降に定義された全てのキー情報が集約されます。  
その後ろに実際の利用者のレコードが格納されます。

## 付録E TAM コマンド制限

## E.1 TAM コマンド制限

メモリキャッシュを導入している場合は、メモリキャッシュが自動的に TAM のコマンドを実行します。このため、TAM が提供しているコマンドの一部については、メモリキャッシュの機能経由で実行する必要があります。

各 TAM コマンドの制限有無一覧を以下に示します。（制限事項の欄が空欄のコマンドは、制限がありません。

TAM コマンド名	TAM コマンドの説明	TAM コマンドを使用する際の制限事項	代替コマンド
tamstart	起動	DIOSA(メモリキャッシュ)のコマンドを使用して行うこと	diiminit、 diimctrl -b
tamstop	停止	DIOSA(メモリキャッシュ)のコマンドを使用して行うこと	diimterm、 diimctrl -e
tamcfgmaint	コンフィグファイルの解析、反映および確認		
tamcreate	TAM ファイルの生成	TAM の各レコードには DIOSA および事業部 PP の制御情報が付加されるためアプリケーションデータだけを入力とした tamcreate は利用不可。 但し、Oracle へのレプリケーションを利用しない環境で、かつ DIOSA および事業部 PP の API でデータ登録した表に対して、tamexport で作成したファイルを利用して tamcreate することは可能。	
tamload	メモリテーブルのロード	アクセスサーバで表オープン中のテーブルは指定不可 (表オープン/クローズコマンドでクローズしてからコマンドを実行すること) Oracle へのレプリケーションを利用する環境では使用不可。事業部 PP が提供する Oracle からのロードコマンドを使用。 Oracle へのレプリケーションを利用しない環境では、tamsave で作成した TAM ファイルを使ったロードは可。(tamcreate で作成した TAM ファイルからのロードは不可)	daslutamsave、 daslutamload
tamdrop	メモリテーブル削除	アクセスサーバで表オープン中のテーブルは指定不可 (表オープン/クローズコマンドでクローズしてからコマンドを実行すること) ※DIOSA および事業部 PP の環境定義と整合を取る必要あり	
tambackupstart	レプリケーションの開始	DIOSA(メモリキャッシュ)のコマンドを使用して行うこと	diiminit、 diimctrl -b
tambackupstop	レプリケーションの停止	DIOSA(メモリキャッシュ)のコマンドを使用して行うこと	diimterm、 diimctrl -e
tambackup	レプリケーション再構成	使用不可	
tamcopy	メモリテーブルコピー	使用不可	
tammove	メモリテーブル移動	使用不可	
tambackupswap	マスター切替	DIOSA(メモリキャッシュ)のコマンドを使用して行うこと	ditamswap
tambackupswitch	マスター昇格	DIOSA(メモリキャッシュ)のコマンドを使用して行うこと	ditamswitch
tamstatus	状態表示		
tampertrace	TAM-API 性能トレースの編集		
tamlogedit	内部ログの編集		
tamtrcredit	トレースの編集		

TAM コマンド名	TAM コマンドの説明	TAM コマンドを使用する際の制限事項	代替コマンド
tamtrcset	トレース採取開始、停止		
tamaredit	統計情報の編集		
tamarset	統計情報採取開始、停止		
tamrebuild	メモリテーブルの再編成		
tamremake	メモリテーブルの再構築	IM アクセスサーバで表オープン中のテーブルは指定不可 (表オープン/クローズコマンドでクローズしてからコマンドを実行すること) ※DIOSA および事業部 PP の環境定義と整合を取る必要あり	
tamsave	メモリテーブルから TAM ファイルへの退避	Oracle へのレプリケーションを利用する環境では使用不可。事業部 PP が提供する Oracle からのセーブコマンドを使用。 Oracle へのレプリケーションを利用しない環境は使用可。	daslutamsave、 daslutamload
tamspace	メモリの使用状況表示		
tamrecordchk	メモリテーブルのメモリ破壊検証		
tamnetconfig	帯域制御の切替、動的インスタンス削除	※DIOSA および事業部 PP の環境定義と整合を取る必要あり	
tamlockcheck	制御ロック残留確認		
tamfsync	ファイル書き込み保証		
tamblockset	TAM 閉塞状態の設定	DIOSA(メモリキャッシュ)のコマンドを使用して行うこと tamblockset で閉塞をかけた場合、DIOSA が障害と認識し、マスタ切替が発生します。	diimtblopenclose
tammodechange	メモリテーブル属性変更/参照		
tamcheckstat	状態表示		
tamexport	レコードデータからの入力ファイル作成		
tamperinfo	メモリテーブルのチューニング情報の表示		
tamrecdump	レコードデータファイルのソート・ダンプ出力		
taminstancelist	起動済インスタンスの一覧出力		
tammemverify	メモリテーブル構造の妥当性検証		
tammemdump	メモリテーブル、TAM ファイル論理ダンプ表示		
tammemsave	メモリテーブルの退避		
tampageinfo	メモリテーブルのページ利用状況出力		
tamshmdump	共有メモリアメージをファイルに出力		
tamshmdumpedit	共有メモリアメージを表示		
tamtxdump	トランザクション制御用メモリ情報を表示		
tamtxlogdump	更新ログ情報表示		
tamcmdlocklist	排他制御機能の使用状況を表示		

TAM コマンド名	TAM コマンドの説明	TAM コマンドを使用する際の制限事項	代替コマンド
tamcmdlockps.sh	実行中等の確認		
tammemindex	メモリテーブル、TAM ファイルのインデックス情報表示		
tamkill.sh	デーモンの強制停止		
tarmm.sh	共有メモリ等の強制削除		

D I O S A / X T P V2.1  
メモリキャッシュ 利用の手引

2018 年 2 月 4 版

日本電気株式会社  
東京都港区芝五丁目 7 番 1 号  
TEL (03) 3 4 5 4 - 1 1 1 1 (大代表)

©NEC Corporation 2011, 2017, 2018

日本電気株式会社の許可なく複製・改変などを行うことはできません。  
本書の内容に関しては将来予告なしに変更することがあります。