DIOSA/XTP V3.1

API リファレンス

輸出する際の注意事項

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

その場合、日本国外へ輸出する場合には日本国政府の輸出許可が必要です。

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

はしがき

本書は、業務システムの構築を支援するDIOSA/XTP製品群のAPIリファレンスです。

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。

2019年11月 初版
2022年 9月 3版

本書の関連説明書としては次のものがあります。

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

目次

第Ⅰ編　　利用者インタフェース

第1章　利用者インタフェースの説明形式

1.1　概要

利用者インタフェースは、DIOSA/XTP の各種機能から利用者が作成したアプリケーションを呼び出す際に、アプリケーションとして実装する必要がある形式を規定するものである。

1.2　利用者インタフェースの説明形式

利用者インタフェースは、以下の形式で説明を記述する。

書式

関数の形式を記述する。
なお、特に説明がない限り diosa.h ヘッダをインクルードする前に、stdio.h ヘッダをインクルードする必要がある。

説明

書式に記載したアプリケーションの関数形式、および、関数パラメータについて説明する。
入力型のパラメータは利用者出口に渡されるパラメータ、出力型のパラメータは利用者出口から出力するパラメータであることを示す。

戻り値

アプリケーションが実装する必要がある関数の戻り値について説明する。

注意

アプリケーションの実装における注意事項について説明する。

関連

関連する機能について説明する。

第2章　利用者インタフェース

DIOSA/XTP がアプリケーションを呼び出す際の利用者インタフェースについて、以下のように機能に分けて説明する。

・アプリケーション実行制御
・通信制御
・メモリキャッシュ
・データストア基盤

2.1　アプリケーション実行制御

2.1.1　関数一覧

(1)　CO 制御

ＣＯ／利用者出口	ＣＯ制御環境定義(COCENV 節)に定義されたＣＯ／利用者関数の形式
受信電文解析出口	トランザクション開始前に受信電文を解析・編集を行う。

2.1.2　　CO・利用者出口(受信電文解析出口を除く)

書式

```
#include <diosa.h>
void  プログラム名( t_diosa_uca   *DiosaUca );
```

説明

CO 制御上およびバッチ制御で動作する CO および利用者出口である。
プログラム名は CO および利用者出口ごとに CO 制御環境定義(COCENV 節)に定義される。
本出口は、以下のパラメータが与えられる。

DiosaUca (入出力型)

t_diosa_uca を参照

本出口は、関数の戻り値を持たないが、DiosaUca の状態コード(Status)によって、CO 制御に状態を通知することができる。

2.1.3　　CO 制御 受信電文解析出口

書式

```
#include <diosa.h>
void  プログラム名( t_diosa_uca  *DiosaUca,  t_diosa_analyze  *Analyze );
```

説明

CO 制御において、トランザクション開始前に受信電文を解析・編集するために呼び出される利用者出口である。
プログラム名は、CO 制御環境定義 COCENV-EXIT-ANALYZE に定義される。
本出口は、以下の２つのパラメータが与えられる。

DiosaUca(入出力型)

t_diosa_uca を参照

Analyze(入出力型)

t_diosa_analyze を参照

本出口は、関数の戻り値を持たないが、DiosaUca の状態コード(Status)によって、CO 制御に状態を通知することができる。

2.2　通信制御

2.2.1　関数一覧

(1)　DB 監視機能

データベース接続出口	データベース接続時の利用者関数の形式

(2)　パス管理機能

リスナ電文送受信時出口	受信電文を処理するトランザクション ID の決定、および送信電文の編集を行う。
リスナ初期化出口	リスナ電文送受信時出口で必要なリソースの確保などを行う
リスナ終了出口	リスナ電文送受信時出口で使用したリソースの解放などを行う
パス接続切断同期通知 CO	ノード間、論理システム間のパスが接続/切断された際呼び出される
送受信エラー出口	外部システムへの都度接続送信でエラーが発生した際に呼び出される
送受信エラー初期化出口	送受信エラー出口で必要なリソースの確保などを行う
送受信エラー終了出口	送受信エラー初期化出口、送受信エラー終了出口で確保したリソースの解放などを行う
t_diosa_otcmsgerr	送受信エラー出口との UCA

(3)　電文保証

保証電文送信有無判定出口	保証電文送信前に、送信を行うか否かを判定する。
電文保証 TPP プロセス初期化出口	保証電文送信有無判定出口で必要なリソースの確保などを行う
電文保証 TPP プロセス終了出口	電文保証 TPP プロセス初期化出口、保証電文送信有無判定出口で確保したリソースの解放などを行う。

2.2.2　　リスナ電文送受信時出口

書式

#include <diosa.h>
int リスナ電文送受信時出口(void *RfuUca, t_diosa_lsnrexituca *LsnrExitUca)

説明

論理システム間電文の送受信時に呼び出される。
受信時は入力された電文を解析し、起動するトランザクション ID と電文長を決定する。
送信時は送信電文の編集を行う。

void *RfuUca（入出力型）
　将来拡張領域。現在 RfuUca には NULL が格納される。

t_diosa_lsnrexituca *LsnrExitUca（入出力型）
　t_diosa_lsnrexituca 構造体により情報の授受を行う。構造体は以下のメンバを含んでいる。

int　ExitId	出口の呼び出しタイミング。(入力)
	DIOSA_LSNREXIT_RECV_HEADER(0)　受信時(ヘッダ受信)
	DIOSA_LSNREXIT_RECV_ALL(1)　　受信時(全電文受信)
	DIOSA_LSNREXIT_SEND(2)　　送信時
void *MsgPtr	電文格納領域へのポインタ（入力）
int　LdDtLen	電文格納領域の長さ（入力）
int　MsgLen	電文長(入出力)
char TxId[7]	出口が実行されるノードで起動するトランザクション ID(出力)
char LsName[15+1]	送受信相手の論理システム名(入力)
char AcsPoint[15+1]	送受信相手のアクセスポイント名(入力)
char Protocol[31+1]	プロトコル名(入力)
char Term[32]	電文を送受信する端末名(入力)
char UserData[64]	TPBASE 端末定義ファイル中の USERDATA に設定された値(入力)

戻り値

出口の処理結果を返却する。
出口が異常終了した場合、戻り値をメッセージ出力するが、異常終了の詳細な原因を知ることができるよう、ユーザ独自の戻り値を返却することができる。

DIOSA_DONE(0)	正常終了。受信時は TPP へ電文を渡す。 送信時は宛先アクセスポイントへ電文を送信する。
DIOSA_RETRY(5)	受信時、MsgLen で指定された電文長まで電文を読み込み、リスナ電文送受信時出口を再実行する
DIOSA_BREAK(11)	電文を破棄し、端末を切断する（正常終了）
DIOSA_ERROR(-1)	電文を破棄し、リスナを終了する（異常終了）
その他	DIOSA_ERROR と同じ

電文受信時の動作

下位プロトコルが TCP/IP の場合、または都度接続の場合、利用者出口では電文長と電文を処理するトランザクション ID を決定する。出口の呼び出しは 1 回、2 回の 2 パターンがある。

- 2 回の場合の利用者出口での処理（電文ヘッダでトランザクション ID が決定できない場合）
  - 1 回目 受信する電文長を決定して MsgLen に設定する。出口の戻り値には DIOSA_RETRY を指定する。
  - 2 回目 電文を処理するトランザクション ID 決定して TxId に設定する。出口の戻り値には DIOSA_DONE を指定する。
- 1 回の場合の利用者出口での処理（電文ヘッダでトランザクション ID が決定できる場合）
  - 受信する電文長とトランザクション ID を決定して、それぞれ MsgLen と TxId に設定する。出口の戻り値には DIOSA_DONE を指定する。

下位プロトコルが OLF-TP/UT の場合、利用者出口はトランザクション ID を決定する。電文長は、OLF-TP プロトコルにより、呼び出された時点で決定されている。

- 利用者出口での処理
  - 受信した電文を処理するトランザクション ID 決定して TxId に設定する。

項目	設定値			設定者
	下位プロトコル TCP/IP		下位プロトコル OLF-TP/UT	
	1 回目	2 回目		
ExitId	DIOSA_LSNREXIT_RECV_HEADER	DIOSA_LSNREXIT_RECV_ALL		DIOSA
MsgPtr	受信電文が格納された領域のアドレス			DIOSA
LdDtLen	受信電文が格納された領域のサイズ(現バージョンでは電文長と同じ)			DIOSA
MsgLen	DIOSA がヘッダ長を設定して出口を呼び出す ユーザがヘッダを含む電文長を設定して出口を終了する	DIOSA が電文長を設定して出口を呼び出す		DIOSA ユーザ
TxId	電文を処理するトランザクション ID 出口を DIOSA_DONE で終了する場合に設定する	電文を処理するトランザクション ID		ユーザ
LsName	送信元の論理システム名			DIOSA
AcsPoint	送信元のアクセスポイント名			DIOSA
Protocol	プロトコル名			DIOSA
Term	電文を受信した端末名			DIOSA
UserData	TPBASE 端末定義ファイル中の USERDATA に設定された値			DIOSA

電文受信時の注意事項

- 1 回目の呼び出しで TxId 指定し、DIOSA_RETRY で終了した場合、指定した TxId は無視される。
- 1 回目の呼び出しで MsgLen に指定した値が LdDtLen と同じで、出口の戻り値に DIOSA_RETRY を指定した場合、読み込む電文が無いものとして、2 回目の呼び出しは行わない
- 2 回目の呼び出しで、戻り値に DIOSA_RETRY 指定しても、DIOSA_DONE として扱われる。
- MsgLen に LdDtLen よりも小さい値を指定した場合、LdDtLen が指定されたものとして扱われる。
- LsName、AcsPoint は以下の場合に設定される。
  - 常時接続プロトコル
  - 都度接続プロトコル かつ 自システムからの発呼に対する応答
- Term、UserData は以下の場合に設定される。
  - 常時接続プロトコル
  - 都度接続プロトコル かつ 自システムへの着呼

電文送信時の動作

送信電文の編集を行う。

項目	設定値	設定者
ExitId	DIOSA_LSNREXIT_SEND	DIOSA
MsgPtr	送信電文が格納された領域のアドレス	DIOSA
LdDtLen	送信が格納された領域のサイズ(現バージョンでは電文長と同じ)	DIOSA
MsgLen	DIOSA がアプリケーションにより指定された電文長を設定して出口を呼び出す ユーザが実際に送信する電文長を設定して出口を終了する	DIOSA ユーザ
LsName	送信先の論理システム名	DIOSA
AcsPoint	送信先のアクセスポイント名	DIOSA
Protocol	プロトコル名	DIOSA
Term	送信先の端末名	DIOSA
UserData	TPBASE 端末定義ファイル中の USERDATA に設定された値	DIOSA

電文送信時の注意事項

- 編集後の電文長が呼び出し時の電文長を超過した場合、超過分のデータは切り捨てられる。
- LsName、AcsPoint は以下の場合に設定される。
  - 常時接続プロトコル
  - 都度接続プロトコル かつ 自システムから発呼
- Term、UserData は以下の場合に設定される。
  - 常時接続プロトコル
  - 都度接続プロトコル かつ 自システムへの着呼に対する応答

電文送受信共通の注意事項

- 本出口は以下のタイミングで呼び出される
  - TPBASE 通信リスナの TPSEXIT 処理ルーチン実行時(下位プロトコルが TCP/IP および OLF/TP-UT の場合)
  - TPBASE 通信リスナの TPSEXIT2 処理ルーチン実行時（下位プロトコルが TCP/IP の場合）
  - 都度接続送信デーモンの応答電文送受信時
- MsgLen に 0 以下を設定した場合、電文を破棄し、端末を切断する。
- (常時接続プロトコルのみ)DIOSA/XTP が設定する LdDtLen および MsgLen の上限は、TPBASE の環境設定 tpbase.cnf のパラメータ DATABLKSIZE で指定された値から 0.5K 程度引かれたものとなります。なお、0.5K は TPBASE の制御領域になります。
- 送受信時に編集可能な電文の領域は、先頭から LdDtLen バイト目の範囲に限られます。

関連

リスナ初期化出口，リスナ終了出口

2.2.3　　リスナ初期化出口

書式

#include <diosa.h>

int リスナ初期化出口名(void *RfuUca)

説明

リスナ電文送受信時出口で必要なリソースの確保などを行う利用者出口。

void *RfuUca（入出力型）

　将来拡張領域。現在 RfuUca には NULL が格納される。

戻り値

利用者出口の処理結果を返却する。

DIOSA_DONE(0)	正常終了
DIOSA_ERROR(-1)	異常終了

注意

- 本出口は通信リスナ出口ルーチンのうち、TPSEXIT 初期化ルーチン上で実行される。
- 都度接続プロトコルを使用する場合は都度接続送信デーモン上でも実行される。

関連

リスナ電文送受信時出口,リスナ終了出口

2.2.4　　リスナ終了出口

書式

#include <diosa.h>
void リスナ終了出口名(void *RfuUca)

説明

リスナ初期化出口、リスナ電文送受信時出口で確保したリソースの解放などを行う利用者出口。

void *RfuUca（入出力型）
　将来拡張領域。現在 RfuUca には NULL が格納される。

戻り値

なし

注意

- 本出口は通信リスナ出口ルーチンのうち、TPSEXIT 終了ルーチン上で実行される。
- 都度接続プロトコルを使用する場合は都度接続送信デーモン上でも実行される。

関連

リスナ電文送受信時出口，リスナ初期化出口

2.2.5　　データベース接続出口

書式

#include <diosa.h>
int データベース接続出口名(t_diosa_dbconnect *DbConnect)

説明

データベース接続関数によるデータベース接続時に、接続先データベースの USERID、PASSWORD が環境定義 DBCTRL 節-INSTANCE 項に定義されていない場合に呼び出される利用者出口。

t_diosa_dbconnect *DbConnect（入出力型）

t_diosa_dbconnect 構造体により情報の授受を行う。
t_diosa_dbconnect 構造体は以下のメンバを含む。
char DbName[137] （入力型、1～136 バイト＋NULL 文字）
　環境定義 DBCTRL 節-INSTANCE 項の DBNAME に定義された接続先（ネット・サービス名）が格納される。
int　DbType（出力型）
　接続先 DB 種別が格納される。
　DIOSA_DBTYPE_ORACLE　　Oracle
　DIOSA_DBTYPE_POSTGRES　PostgreSQL
void *SqlCtx（出力型）
　SQL コンテキストを利用者が格納する。

戻り値

利用者出口の処理結果を返却する。

DIOSA_DONE(0)　　正常終了
その他　　異常終了

注意

- データベース・インスタンス毎にアカウントを変更したい場合、本 EXIT の接続先（ネット・サービス名）を元に、アカウントを切り替えること。

2.2.6　　パス接続切断同期通知 CO

書式

#include <diosa.h>
void パス接続切断同期通知 CO 名(t_diosa_uca *DiosaUca)

説明

論理システム(アクセスポイント)とのパスが接続または切断された際に呼び出される CO。
呼び出されるタイミングは以下の通り

1. アクセスポイントと接続された。
2. アクセスポイントの接続に失敗した。
3. アクセスポイントと正常に切断された。
4. アクセスポイントと何らかの異常により切断され、再接続のリトライが失敗した。
   異常により切断されたタイミングでは呼び出されない。

アクセスポイントとのパスが多重化されている場合、少なくとも一つパスが接続された際に接続が通知され、すべてのパスが切断された際に切断が通知される。

DiosaUca（入出力型）

t_diosa_uca を参照

電文形式

CO には接続先情報を以下の電文形式で渡される。接続先情報は CO 中で電文受信 API(diosarecvtx)を使用して取得する。

```
t_diosa_path_notify {
  int     Event;                  発生した事象
                                  DIOSA_PATH_CONNECTED     アクセスポイントと接続された
                                  DIOSA_PATH_ECONNECT      アクセスポイントとの接続に失敗した。
                                  DIOSA_PATH_DISCONNECTED アクセスポイントと正常に切断された。
                                  DIOSA_PATH_RETRYOVER     アクセスポイントとの再接続に失敗した。
  char    LsName[15+1];           論理システム名
  char    AcsPoint[15+1];         アクセスポイント名
  char    Term[31+1];             端末名
  int TermType;                   端末種別
                                  DIOSA_TERM_ALL           送受信用端末
                                  DIOSA_TERM_SEND          送信用端末
                                  DIOSA_TERM_RECV          受信用端末
  char    TpbaseMsg[384];         TPBASE からの通知内容
};
```

TPBASE からの通知内容については「TPBASE プログラミングの手引き 第 9 章 同期通知」を参照してください。

注意事項

- 環境定義 SYSMAP 節 LOGSYSTEM 項のパラメータ NOTIFY_CO で指定した名前で CO を作成すること。

関連

環境定義 SYSMAP 節 LOGSYSTEM 項のパラメータ NOTIFY_CO、NOTIFY_TXID

2.2.7　　送受信エラー出口

書式

```
#include <diosa.h>
int プログラム名( t_diosa_otcmsgerr *MsgErr )
```

説明

都度接続で外部システムへ送信(発呼)した際に、都度接続送信デーモンで外部システムへの送信に失敗した場合および応答受信に失敗した場合の対処を行う。**MsgErr** の設定によって、CO 制御 TPP にエラー応答電文を送信することができる。設定しない場合、送信電文は破棄される。

MsgErr　(入出力)
　送受信エラー電文情報
　送信電文情報やエラー応答情報を格納する

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_ERROR(-1)	異常終了

注意

- エラー応答電文の送信に失敗した場合、エラー応答電文は破棄される。この時エラーメッセージが出力される。
- DIOSA_ERROR を返却した場合、都度接続送信デーモンはエラーメッセージを出力した後にプロセス停止する。この時、デーモンが処理中の電文は全て破棄される。
- 本出口はマルチスレッドで呼び出されるため、スレッドセーフな処理として実装すること。
- 本出口処理のための初期化処理は送受信エラー初期化出口で実装すること。
- 本出口はアプリケーション動的置換機能によるライブラリの動的置換を行うことはできない。

関連

t_diosa_otcmsgerr, 送受信エラー初期化出口, 送受信エラー終了出口

2.2.8　　送受信エラー初期化出口

書式

```
#include <diosa.h>
int プログラム名(void *RfuUca)
```

説明

送受信エラー出口で必要なリソースの確保などを行う利用者出口。
都度接続送信デーモンのプロセス初期化時に呼び出される。

void *RfuUca（入出力型）
　将来拡張領域。NULL が格納される。

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_ERROR(-1)	異常終了

注意

- DIOSA_ERROR を返却した場合、都度接続送信デーモンはエラーメッセージ出力し、プロセスが異常終了する。詳細なエラー情報が必要ならば本出口内で diosamsgdisp()でメッセージ出力を行うこと。
- 送受信エラー出口の呼び出しは本出口とは別のスレッドで行われるため、確保するリソースはスレッド間で共有できるようにすること。
- 本出口はアプリケーション動的置換機能によるライブラリの動的置換を行うことはできない。

関連

送受信エラー出口,送受信エラー終了出口

2.2.9　　送受信エラー終了出口

書式

```
#include <diosa.h>
void プログラム名(void *RfuUca)
```

説明

送受信エラー初期化出口、送受信エラー出口で確保したリソースの解放などを行う利用者出口。
都度接続送信デーモンのプロセス終了時に呼び出される。

void *RfuUca（入出力型）

　将来拡張領域。NULL が格納される。

戻り値

なし

注意

- 送受信エラー初期化出口で異常終了を返却した場合も、本出口は呼び出される。
- 本出口で異常が発生した場合は、必要ならば出口内でメッセージ出力を行うこと。
- 本出口はアプリケーション動的置換機能によるライブラリの動的置換を行うことはできない。

関連

送受信エラー初期化出口，送受信エラー出口

2.2.10　保証電文送信有無判定出口

書式

```
#include <diosa.h>
void プログラム名(t_diosa_gntsendjdg *GntSendJdg)
```

説明

メッセージの送信可否を判定するための利用者出口である。
次のタイミングで本利用者出口が呼び出される。

・電文保証機能の制御 TPP が保証電文を再送する前。
・電文保証機能の制御 TPP が順序性保証の次保証電文を送信する前。
・電文保証機能の CO 制御 TPP 上の、diosagntdel()の延長で、順序性保証の次保証電文を送信する前。

t_diosa_gntsendjdg GntSendJdg（入出力型）

保証電文送信有無判定出口用 UCA
構造体の以下のメンバを使用する。

char LsName[16]
宛先論理システム名(入力)

char AcsPoint[16]
宛先アクセスポイント名(入力)

char APInfo[256]
diosasendtx で指定した AP 固有情報(入力)

int SendMax
最大送信回数(入力)
SG 値もしくは diosasendtx で指定した値

int SendCnt
現在の送信回数(入力)
今回の送信が何回目の送信かを表す。diosasendtx で初送している場合は初回再送時には 2 が、初送していない場合は初回再送時には 1 が設定される。

char MsgKey[32]
保証電文識別子(入力)

int Sequence
順序性保証指定有無(入力)
DIOSA_YES　：順序性保証あり
DIOSA_NO　：順序性保証なし

char SeqGroup[32]
順序性保証グループ識別子(入力)

char *Msg
保証電文アドレス(入力)
出口内で保証電文の内容を更新した場合、更新後の電文が送信される。ただし、電文サイズを変更することはできない。

sizt_t Size
保証電文サイズ(入力)

int Result
送信判定結果(出力)
以下のいずれかの値を設定する。
DIOSA_MSGGNT_AUTO　：送信回数に従う(初期値)
　最大送信回数≧送信回数の場合は送信される
　最大送信回数＜送信回数の場合は削除される

DIOSA_MSGGNT_STAY　：送信しない(送信回数はそのまま)
DIOSA_MSGGNT_SKIP　：送信しない(送信回数はカウントアップ)
DIOSA_MSGGNT_DEL　：削除する
DIOSA_MSGGNT_ERROR ：送信有無判定処理に失敗した

戻り値

なし

注意

- 本出口を指定しない場合、最大送信回数≧送信回数であれば送信が行われる。最大送信回数＜送信回数であれば電文は破棄される。
- 判定を実行するために必要なプロセス初期化(終了)処理は、電文保証 TPP プロセス初期化(終了)出口、および CO 制御のプロセス初期化(終了)出口内でおこなうこと。

関連

diosasendtx, t_diosa_dcuca

2.2.11　電文保証 TPP プロセス初期化出口

書式

#include <diosa.h>
int プログラム名 (void *RfuUca);

説明

電文保証機能の制御 TPP が、保証電文送信有無判定出口を呼ぶ前に呼び出す利用者出口。
保証電文送信有無判定出口で必要なリソースの確保などを行う。

void *RfuUca（入出力型）
　将来拡張領域。

戻り値

利用者出口の処理結果を返却する。
DIOSA_DONE(0)　　正常終了
DIOSA_ERROR(-1)　異常終了

注意

なし

関連

保証電文送信有無判定出口、電文保証 TPP プロセス終了出口

2.2.12　電文保証 TPP プロセス終了出口

書式

```
#include <diosa.h>
void プログラム名(void *RfuUca);
```

説明

電文保証 TPP プロセス初期化出口、保証電文送信有無判定出口で確保したリソースの解放などを行う利用者出口。

void *RfuUca（入出力型）
　将来拡張領域。

戻り値

なし

注意

なし

関連

保証電文送信有無判定出口、電文保証 TPP プロセス初期化出口

2.2.13　t_diosa_otcmsgerr(送受信エラー電文情報)

名前

t_diosa_otcmsgerr – 送受信エラー電文情報

書式

```
#include <diosa.h>
t_diosa_otcmsgerr MsgErr;
```

説明

t_diosa_otcmsgerr は送受信エラー出口で渡される送受信エラー電文情報である。

char LsName[15+1]	宛先論理システム名（入力）
char AcsPoint[15+1]	宛先アクセスポイント名（入力）
char Protocol[31+1]	プロトコル名（入力）
char TpMonitor[8+1]	送信元 TPBASE モニタ名（入力）
unsigned char SettingGroup	diosasendtx で指定した定義グループ（入力） 0 を指定していた場合は最も小さいグループ番号が設定される。
unsigned char Response	送信電文の応答要否（入力） DIOSA_RESP_NO　：応答なし DIOSA_RESP_YES ：応答あり
short　MsgTimeOut	送受信タイムアウト時間(秒)（入力） 0 を指定していた場合は環境定義の値が設定される。
char APInfo[256]	diosasendtx で指定した AP 固有情報（入力）
int　ErrorType	エラー種別（入力） DIOSA_OTC_PRE_TIMEOUT　：送信前タイムアウト DIOSA_OTC_CONN_ERROR　：接続失敗 DIOSA_OTC_CONN_TIMEOUT：接続タイムアウト DIOSA_OTC_SEND_ERROR　：送信失敗 DIOSA_OTC_SEND_TIMEOUT：送信タイムアウト DIOSA_OTC_RECV_ERROR　：応答受信失敗 DIOSA_OTC_RECV_TIMEOUT：応答受信タイムアウト
char *RespMsg	受信電文アドレス（入力） ErrorType が DIOSA_OTC_RECV_ERROR または DIOSA_OTC_RECV_TIMEOUT で応答電文を途中まで受信していた場合に設定される。 応答電文を受信していない場合は NULL が設定される。
int RespMsgSize	受信電文サイズ（入力） ErrorType が DIOSA_OTC_RECV_ERROR または DIOSA_OTC_RECV_TIMEOUT で応答電文を途中まで受信していた場合に設定される。 応答電文を受信していない場合は 0 が設定される。
char *SendMsg	送信電文アドレス（入力） 入力時は要求電文が格納されている。 TxId を設定した場合、ここに格納された電文をエラー応答電文として CO 制御 TPP に送信する。 電文の内容を編集することができる。
int SendMsgSize	送信電文サイズ（入出力） 入力時は要求電文サイズが格納されている。 TxId を設定した場合、SendMsg に格納したエラー応答電文サイズを設定する。ただし、入力時に格納されている要求電文サイズより大きいサイズを指定することはできない。
char TxId[6+1]	エラー応答電文の送信先トランザクション ID（出力）

	CO 制御 TPP にエラー応答電文を送信する場合に指定する。 設定しない場合、要求電文は破棄される。
char CoName[30+1]	CO 名（出力） TxId を設定した場合に設定することができる。 省略した場合、CO 名が未決定のまま CO 制御 TPP に電文が送信される。

2.3　メモリキャッシュ

2.3.1　関数一覧

(1)　インメモリサーバ所在管理

ハッシュ関数	ハッシュ値計算関数を呼び出す時の関数形式

2.3.2　　ハッシュ関数

書式

```
#include <diosa.h>
int プログラム名( char *MainKey,
                  unsigned int  *HashValue )
```

説明

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ある。
AP 層で送信先 OLTP 層を決定する時、および OLTP 層で接続先のインメモリサーバを決定する時に呼び出される。

MainKey（入力型）

AP 層の場合、ルーティング制御の利用者出口で返却したメインキーを格納した領域の先頭アドレス。
OLTP 層の場合、diosagetmap()で指定したメインキーを格納した領域の先頭アドレス。

HashValue（出力型）

ハッシュ値を設定する領域のアドレス。

戻り値

利用者出口の処理結果を返却する。

DIOSA_DONE(0)	正常終了。
DIOSA_ERROR(-1)	異常終了。

注意

- メインキー分散しない場合でも、固定のハッシュ値を返却するハッシュ関数を指定する必要がある。

2.4　データストア基盤

2.4.1　関数一覧

(1)　ログリーダ

プロセス初期化処理	プロセス起動時に必要な処理を行う利用者関数の形式
トランザクション初期化処理	トランザクション単位に必要な処理を行う利用者関数の形式
ログデータ実行メイン処理	渡されたログデータを処理する利用者関数の形式
トランザクション終了処理	トランザクション単位に必要な終了処理を行う利用者関数の形式
ユーザ用プロセス終了処理	プロセス停止時に必要な処理を行う利用者関数の形式

2.4.2　　ログリーダプロセス初期化処理

書式

```
#include <diosa.h>
int プログラム名( t_diosa_lrduca* LrdUca );
```

説明

ユーザ用ログデータ処理プロセスでプロセス起動時に必要な処理を行う利用者出口。

LrdUca（入出力型）

t_diosa_lrduca を参照

戻り値

利用者出口の処理結果を返却する。

DIOSA_DONE(0)	正常終了。
DIOSA_EABORT(-4)	アボート終了要求。

2.4.3　　ログリーダトランザクション初期化処理

書式

```
#include <diosa.h>
int プログラム名( t_diosa_lrduca* LrdUca );
```

説明

ユーザ用ログデータ処理プロセスでログデータ処理のトランザクション単位に必要な処理を行う利用者出口。

LrdUca（入出力型）

t_diosa_lrduca を参照

戻り値

利用者出口の処理結果を返却する。

DIOSA_DONE(0)	正常終了。
DIOSA_EABORT(-4)	アボート終了要求。

2.4.4　　ログリーダログデータ実行メイン処理

書式

```
#include <diosa.h>
int プログラム名( t_diosa_lrduca* LrdUca, char* Msg );
```

説明

渡されたログデータに対して必要な処理を行う利用者出口。

LrdUca（入出力型）

t_diosa_lrduca を参照

Msg（入力型）

ログデータ先頭ポインタ

戻り値

利用者出口の処理結果を返却する。

DIOSA_DONE(0)	正常終了。
DIOSA_RETRY(5)	リトライ要求(ロールバック後に処理をリトライ)
DIOSA_ETRAN(-19)	トランザクション異常終了(プロセス終了なし)
DIOSA_EFATAL(-30)	トランザクション異常終了(プロセス終了あり)

2.4.5　　ログリーダトランザクション終了処理

書式

```
#include <diosa.h>
int プログラム名( t_diosa_lrduca* LrdUca );
```

説明

ユーザ用ログデータ処理プロセスでログデータ処理のトランザクション単位に必要な後始末処理を行う利用者出口。

LrdUca（入出力型）

t_diosa_lrduca を参照

戻り値

利用者出口の処理結果を返却する。

DIOSA_DONE(0)	正常終了。
DIOSA_EABORT(-4)	アボート終了要求。

2.4.6　　ログリーダプロセス終了処理

書式

```
#include <diosa.h>
int プログラム名( t_diosa_lrduca* LrdUca );
```

説明

ユーザ用ログデータ処理プロセスでプロセス終了時に必要な処理を行う利用者出口。

LrdUca（入出力型）

t_diosa_lrduca を参照

戻り値

利用者出口の処理結果を返却する。

DIOSA_DONE(0)	正常終了。
DIOSA_EABORT(-4)	アボート終了要求。

2.4.7　　t_diosa_lrduca(ログデータ処理インタフェース構造体)

名前

t_diosa_lrduca - ログデータ処理用の構造体

書式

```
#include <diosa.h>
t_diosa_lrduca LrdUca;
```

説明

t_diosa_lrduca はユーザ用ログデータ処理プロセスとログリーダとのインタフェース情報を設定する構造体である。ユーザ用ログデータ処理機能使用時にパラメータとして使用する。メンバは以下の通りである。

int LsId	論理システム ID。
int LNodeType	論理ノード種別。
	DIOSA_LNODETYPE_AP　AP ノード
	DIOSA_LNODETYPE_DB　DB ノード
	DIOSA_LNODETYPE_OLTP　OLTP ノード
int pid	自プロセス ID。
char LsName[16]	論理システム名。最大文字数は 15 文字である。
char LNodeName[16]	論理ノード名。最大文字数は 15 文字である。
char SpstName[16]	スーパーストリーム名。最大文字数は 15 文字である。
char UnitName[16]	ユニット名。最大文字数は 15 文字である。
char StrmName[16]	ストリーム名。最大文字数は 15 文字である。
char ExitName[31]	ユーザ EXIT 名。呼び出す EXIT の情報。最大文字数は 30 文字である。
unsigned long DataLen	データ長。
unsigned char DataExist	ロット内残り電文有無。
	DIOSA_YES　残り電文あり
	DIOSA_NO　残り電文なし
short DataCount	ロット内電文処理回数。
int DivId	ディビジョン ID。
long UserDataNo	ディビジョン内通番。
short DBErrFlg	DB 障害フラグ。
	DIOSA_ON　障害あり
	DIOSA_OFF　障害なし
int RetryCount	リトライ回数。コミット後、0 に初期化される。
short RetryLim	リトライオーバー数。
void* PrcArea	ユーザ引継ぎエリア。プロセス初期化～プロセス終了までの引継ぎ用。
void* TrnArea	ユーザ引継ぎエリア。トランザクション初期化～トランザクション終了までの引継ぎ用。
int TrnEndStatus	トランザクション処理結果を設定する。
unsigned char PrcType	処理タイプ。コミット・ロールバック前か、コミット・ロールバック後を設定する。
	DIOSA_BEFORE　コミット・ロールバック前
	DIOSA_AFTER　コミット・ロールバック後
unsigned char TrnType	トランザクションタイプ。コミットかロールバックを設定する。
	DIOSA_COMMIT　コミット
	DIOSA_ROLLBACK ロールバック

利用者出口との対応

利用者出口ごとの入出力項目について以下に示す。

入出力項目 ＼ 利用者出口		ユーザ用プロセス初期化処理	ユーザ用トランザクション初期化処理	ユーザ用ログデータ実行メイン処理	ユーザ用トランザクション終了処理	ユーザ用プロセス終了処理
LsId	論理システム ID	I	I	I	I	I
LNodeType	論理ノード種別	I	I	I	I	I
pid	自プロセス ID	I	I	I	I	I
LsName[16]	論理システム名	I	I	I	I	I
LNodeName[16]	論理ノード名	I	I	I	I	I
SpstName[16]	スーパーストリーム名	I	I	I	I	I
UnitName[16]	ユニット名	I	I	I	I	I
StrmName[16]	ストリーム名	I	I	I	I	I
ExitName[31]	ユーザ EXIT 名	I	I	I	I	I
DataLen	データ長			I		
DataExist	ロット内残り電文有無			I		
DataCount	ロット内電文処理回数			I		
DivId	ディビジョン ID			I		
UserDataNo	ディビジョン内通番			I		
DBErrFlg	DB 障害フラグ	I	I	I	I	I
RetryCount	リトライ回数		I	I	I	
RetryLim	リトライオーバー数	I	I	I	I	I
PrcArea	ユーザ引継ぎエリア	O	I	I	I	I
TrnArea	ユーザ引継ぎエリア		O	I	I	I
TrnEndStatus	トランザクション処理結果			O	I	
PrcType	処理タイプ				I	
TrnType	トランザクションタイプ				I	

I：ログリーダから利用者出口へ渡す情報

O：利用者出口からログリーダへ渡す情報

2.5　データ変換・通信オプション

2.5.1　関数一覧

(1)　データ同期制御機能

ストリーム分割内通番決定出口　同一 MAPID 内のストリーム分割内通番を返却する。

(2)　セーブロード機能

構成変更情報取得出口　構成変更に対応したハッシュ値、ユーザデータを返却する

2.5.2　　ストリーム分割内通番決定利用者出口

書式

#include <diosa.h>
int ストリーム分割内通番決定利用者出口(char *MainKey, int *StrmSeqNo);

説明

ストリーム分割内通番**決定利用者出口**はメインキーを入力情報として、MAPID 内のストリーム分割内通番(0～99 の整数)を決定するための利用者出口である。
MainKey は、メインキー(32 バイト)の先頭アドレスが設定されている。
メインキーからストリーム分割内通番を決定し、StrmSeqNo の領域に返却する。StrmSeqNo の領域は 0 で初期化するため、出口内で何も更新されなかった場合は 0 が返却された場合と同様の動作となる。

更新ログ出力先のストリーム名は以下のように決定される。
[ストリーム分割内通番が 0～99]　TAM{MAPID(0 埋め 7 桁)}_{分割内通番(0 埋め 2 桁)}_{拠点識別情報}
[ストリーム分割内通番が-1]　　TAM{MAPID(0 埋め 10 桁)}_{拠点識別情報}

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_ERROR(-1)	異常終了

注意

- 関数が異常終了した場合、更新ログの書き込みが異常終了する。
- 分割内通番に-1～99 の範囲外の値を指定した場合、-1 を指定した場合と同じ動作となる。
- 異常終了する場合、原因解析に必要な情報(メッセージ、トレース等)は関数内で出力すること。

2.5.3　　構成変更情報取得出口

書式

```
#include <diosa.h>
int プログラム名(t_diatc_fltamdata *pstFlTamData);
```

説明

入力されたユーザデータを解析し、メインキーに対応するハッシュ値を返却する利用者出口。
TAMの構成変更によりユーザデータレコードの定義に変更がある場合、ユーザデータ、ユーザデータサイズ、ユーザデータ格納領域サイズを返却する。
本出口はファイル入力TAMロードコマンドより呼ばれる。

pstFlTamData（入出力型）

t_diatc_fltamdata構造体により情報の授受を行う。

t_diatc_fltamdata構造体は以下のメンバを含む。

char	*LTableName	ロード先の論理表名を指定する。	(入力)
char	*UserData	ユーザデータレコードを指定する。	(入力)
size_t	UserDataSize	ユーザデータレコードのレコードサイズを格納する。 変更有無フラグ(ModFlg)がDIOSA_ONの場合、更新データレコード(ModBuff)のレコードサイズを返却する。	(入出力)
unsigned int	HashValue	ユーザデータレコードのメインキーに対応するハッシュ値を返却する。ハッシュ値は**diosagethash()**により取得する。	(出力)
int	ModFlg	ユーザデータの変更の有無を返却する。	(出力)
	DIOSA_ON	変更あり	
	DIOSA_OFF	変更なし	
char	*ModBuff	TAMの構成変更により変更されたユーザデータレコードを返却する。 領域は出口関数内で領域種別にDIOSA_THRNOALL(一括解放対象外スレッド内メモリ)を指定し、**diosamalloc()により取得する。** 本パラメータは、次回の呼び出し時に情報を引き継ぐ。 領域の解放は呼出元で行う。	(出力)
size_t	ModBuffSize	TAMの構成変更により変更されたユーザデータレコード格納領域(ModBuff)のサイズを返却する。	(出力)

戻り値

DIOSA_DONE(0)	正常終了。
DIOSA_ERROR(-1)	異常終了。

注意

- 本出口で確保した領域は呼び出し側で解放する。

第II編　C言語インタフェース

第1章　C言語インタフェースの構成と記述形式

1.1　概要

Ｃ言語インタフェースは、DIOSA/XTP がアプリケーションに提供する機能へアクセスするための関数群である。
本章では、Ｃ言語インタフェースの構成と説明形式、および、利用上の一般的な規則について説明する。

1.2　C言語インタフェースの構成

本項では、Ｃ言語インタフェースの構成および利用上の一般的な規則について説明する。

(1)　Ｃ言語インタフェースの形式

Ｃインタフェースは次の基本形式を持つ。

インクルードファイル
関数型　関数名(パラメータ１，パラメータ２，・・・)；

(2)　Ｃ言語インタフェースの構成文字

Ｃインタフェースを構成する文字は、英字、数字、特殊文字である。
次の特殊文字は、Ｃインタフェース構成上固有の意味を持つ。

関数型 … Ｃインタフェースの変数宣言
(… パラメータ列の開始
) … パラメータ列の終了(Ｃインタフェース呼び出しの終了)
, … パラメータどうしの区切り
; … 終了符

(3)　Ｃ言語インタフェースの構成項目

Ｃインタフェースは、一つの関数名と、いくつかのパラメータからなり、すべてのパラメータに先行して関数名を指定する。

(4)　関数名

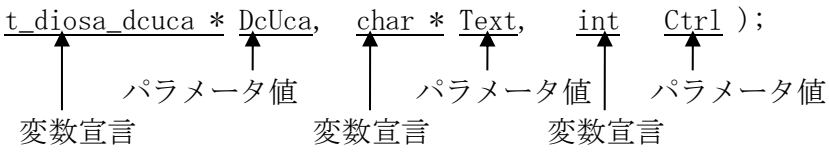
関数名は、先頭が"diosa"で始まり、32 バイト以内の英数字のみからなる。

(5)　パラメータ

パラメータは、Ｃ言語インタフェースの処理に必要な補助情報(パラメータ値)を与えるものである。
パラメータ値には指定の変数宣言が必要である。
パラメータ値は文字列であり、"__"、および互いに対応した"("と")"とを含むことができる。
ただし、空白、", "および"("と対応しない")"を含むことはできない。

例)

```
int  diosasendtx( t_diosa_dcuca * DcUca,   char * Text,    int   Ctrl );
```



(6)　Ｃ言語インタフェース利用時の一般規則

Ｃインタフェース利用時(呼び出し時)の一般的な規則について、主なものを説明する。

- 利用者が作成するＣ言語インタフェースでは、"diosa"で始まる関数名を用いてはならない。また、"diosa"で始まる外部変数を用いてはならない。
- Ｃ言語インタフェース呼び出しを含む行には、Ｃ言語インタフェース呼び出し以外のもの(コメントも含む)を記述してはならない。
- パラメータは",“で区切らなければならない。
- パラメータをすべて省略する場合には、パラメータ列の開始・終了を示す"("および")"を省略してはならない。

この規則は、個々のＣインタフェース説明の形式欄には記載していないので注意されたい。

1.3　C言語インタフェースの説明形式

各C言語インタフェースの詳細は「第2章C言語インタフェース」で説明する。ここでは、C言語インタフェースの説明形式と構文表記法について述べる。

名前

関数名、および、処理概要について説明する。

書式

C言語インタフェースを呼び出す際の形式およびインクルードファイルについて説明する。
なお、特に説明がない限り diosa.h ヘッダをインクルードする前に、stdio.h ヘッダをインクルードする必要がある。

説明

C言語インタフェースの動作、および、各パラメータに設定する値について説明する。

戻り値

C言語インタフェースの実行結果が返却される値について説明する。

注意

C言語インタフェースの呼び出し規則や注意点、呼び出し可能な環境について説明する。

関連

C言語インタフェースに関連する他のC言語インタフェースを列挙する。

第2章　C言語インタフェース

DIOSA/XTP が提供する API について、以下のように機能に分けて説明する。

・アプリケーション実行制御
・通信制御
・メモリキャッシュ
・データストア基盤

2.1　アプリケーション実行制御

2.1.1　関数一覧

(1)　ＣＯ制御

diosamsgbufalloc	電文バッファの確保
diosamsgbuffree	電文バッファの解放
diosarecvtx	トランザクション電文の受信
diosasendtx	トランザクション電文の送信
diosacommit	トランザクションのコミット
diosarollback	トランザクションのロールバック
diosaucaget	diosauca の取得
diosagetsrctx	トランザクション開始時電文の取得
diosagoback	CO 制御への強制リターン
t_diosa_uca	CO・利用者出口パラメータ（DIOSAUCA）
t_diosa_dbinfo	DB 情報
t_diosa_lsysaddr	論理システム間アドレス情報
t_diosa_lnodeaddr	論理システム内アドレス情報
t_diosa_dcuca	電文送受信パラメータ（DCUCA）
t_diosa_analyze	受信電文解析出口パラメータ
t_diosa_msgbuf	電文バッファ
t_diosa_otcinfo	都度接続電文送信情報
t_diosa_gntinfo	電文保証情報
t_diosa_tpstatus	TPBASE 電文送受信の実行結果

(2)　ＡＰ動的置換機能

diosavcall	AP 動的置換対象ライブラリの関数を呼び出す

(3)　タイマ制御

diosatmcactv	タイマ保留解除
diosatmccmdquery	コマンドタイマ照会
diosatmccmdset	コマンドタイマ登録
diosatmccoquery	CO タイマ照会
diosatmccoset	CO タイマ登録

diosatmchold	タイマ保留
diosatmcreset	タイマ削除
t_diosa_cmdqueryuca	コマンドタイマ照会用構造体
t_diosa_cmdsetuca	コマンドタイマ登録用構造体
t_diosa_coqueryuca	CO 制御タイマ照会用構造体
t_diosa_cosetuca	CO 制御タイマ登録用構造体
t_diosa_tmctime	タイマ時間設定用構造体
t_diosa_tmcuca	タイマ制御インタフェース構造体

(4) 稼動統計

diosaperfstart	稼動統計任意区間開始
diosaperfend	稼動統計任意区間終了
diosaopsusiput	稼動統計ユーザ情報登録

(5) メッセージ出力

diosamsgdisp	メッセージをメッセージログファイル、標準エラー出力へ出力する
diosamsgedit	メッセージを編集し、編集結果を返却する

(6) ＤＩＯＳＡ共通初期化終了

diosaprcinit	ユーザアプリケーションのためのプロセス初期処理を行う
diosaprcforkinit	ユーザアプリケーションのための fork 後のプロセス初期処理を行う
diosaprcterm	ユーザアプリケーションのためのプロセス終了処理を行う
diosaprcpreexecterm	ユーザアプリケーションのためのプロセス終了処理を行う
diosathrinit	ユーザアプリケーションのためのスレッド初期処理を行う
diosathrterm	ユーザアプリケーションのためのスレッド終了処理を行う
diosatrninit	ユーザアプリケーション上でトランザクション区間を開始する際に呼び出す
diosatrnterm	ユーザアプリケーション上でトランザクション区間を終了する際に呼び出す
diosatxstart	DB 更新対象を決定し、DB 更新を開始する際に呼び出す

(7) 経過時間監視

diosaetgreset	経過時間をリセットする
diosaetgstart	経過時間監視を再開する
diosaetgstop	経過時間監視を停止する
diosaetgusinfo	経過時間監視ユーザ情報を登録する

(8) メモリ管理

diosaaddrconv	割り当てた共有メモリのアドレスとオフセットを相互に変換する
diosaapptrget	DIOSA が保持している AP 共通領域のアドレスを返却する
diosaapptrset	ユーザ AP が確保した共通領域の先頭アドレスを DIOSA に通知する
diosamaddr	diosamalloc、diosarealloc で確保した領域のアドレスを取得する
diosamalloc	メモリ領域の割り当てを行う
diosamcopy	保護属性共有メモリにデータを書き込む
diosamfree	diosamalloc、diosarealloc で割り当てた領域の解放を行う
diosarealloc	メモリ領域の再割り当てを行う

(9) ＡＰＰトレース

diosaapptrcf	トレース情報をトレース情報ファイルに出力する
diosaapptrcm	トレース情報をトレース情報保存領域に格納する

diosaapptrcf_position	ファイル名、行番号、関数名が指定可能なトレース情報出力
diosaapptrcm_position	ファイル名、行番号、関数名が指定可能なトレース情報出力
diosaapptrcopen	トレース情報ファイルをオープンする
diosaapptrcread	トレース情報ファイルを読み込む
diosaapptrcclose	トレース情報ファイルをクローズする

(10) ロック制御

diosalock	指定されたモードに従い、ファイル型／DB 型のロックを取得する
diosaunlock	diosalock で獲得したロックを解放する

(11) コマンド配信

diosacmdconf	コマンド配信結果を確認する
diosacmdsend	指定した宛先にコマンドの配信を行う
t_diosa_cmdconfuca	コマンド配信結果情報構造体
t_diosa_cmdnodeconf	配信先ノード単位結果情報構造体
t_diosa_cmdresultinfo	配信結果確認情報構造体
t_diosa_cmdsendinfo	コマンド配信先情報構造体
t_diosa_cmdsenduca	コマンド配信情報構造体

(12) 起動／停止機能

diosagetlnodeinfo	ノード情報を取得する

(13) 閉塞機能

diosasetblockage	閉塞状態を設定する

(14) アプリケーション共通情報管理機能

diosaapegetitem	アイテム型の AP 共通情報を取得する
diosaapegettbl	テーブル型の AP 共通情報を取得する

2.1.2　　diosaaddrconv(メモリアドレス取得)

名前

diosaaddrconv - 割り当てた共有メモリのアドレスとオフセットを相互に変換する。

書式

```
#include <diosa.h>
int diosaaddrconv(int Mode, long *Offset, void **Ptr);
```

説明

diosaaddrconv() は割り当てた共有メモリのアドレスからオフセットへ変換、あるいはオフセットからメモリアドレスへ変換する。

int Mode(入力型)

変換モードを指定する。

DIOSA_OFFSETTOADDR	オフセットからアドレスに変換
DIOSA_ADDRTOOFFSET	アドレスからオフセットに変換

long *Offset(入力型又は出力型)

メモリオフセット領域ポインタ。

void **Ptr(入力型又は出力型)

メモリアドレス領域ポインタ。

戻り値

DIOSA_DONE(0)	処理が正常に終了した。
DIOSA_EPARAM(-3)	パラメータに誤りがある。
DIOSA_EFUNCNAV(-34)	機能が利用できない。

2.1.3　　diosaapegetitem（アイテム型 AP 共通情報取得）

名前

diosaapegetitem - アイテム型 AP 共通情報取得

書式

```
#include <diosa.h>
int diosaapegetitem(char *Id, void *Ptr, int Size);
```

説明

指定されたアイテム識別子に対応するアイテム値を、アイテム型 AP 共通情報（共有メモリ）から取得し返却する。

char *Id（入力型）

アイテム識別子を指定する。（最大：16 バイト+NULL 文字）

void *Ptr（出力型）

アイテム値を格納する領域のポインタを指定する。（最大：256 バイト+NULL 文字）

int Size（入力型）

アイテム値を格納する領域のサイズを指定する。

戻り値

DIOSA_DONE(0)	正常終了。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータに誤りがある。
DIOSA_ENOBUFS(-7)	アイテム値を格納する領域が不足している。
DIOSA_ENOENT(-8)	指定されたアイテム識別子は、定義されていない。

注意

- アイテム値格納領域は、NULL 文字終端で返却するため、NULL 文字まで格納可能な領域を呼出側で用意すること。
- SG 動的置換時、トランザクション内の値は一貫性を保証するため、SG 変更後、次のトランザクションまでは古い値を参照する。
- 初回の AP 共有情報の共有メモリへの展開（diapenv コマンド）は、本 API を実行するプロセス起動前に実行する必要がある。
- SG にアイテム識別子のみ設定しアイテム値を設定しない場合、該当アイテム識別子を指定して本 API を実行するとアイテム値格納領域に NULL 文字を返却する。

関連

diosaapegettbl

2.1.4　　diosaapegettbl（テーブル型 AP 共通情報取得）

名前

diosaapegettbl – テーブル型 AP 共通情報取得

書式

```
#include <diosa.h>
int diosaapegettbl(int Entnum, char *Id, void **Ptr, size_t *Size);
```

説明

指定されたテーブル識別子に対応するテーブルのポインタを、テーブル型 AP 共通情報（共有メモリ）から取得し返却する。

int Entnum（入力型）

テーブル型共通情報のエントリ番号を指定する。（範囲：1～64）

char *Id（入力型）

テーブル識別子を指定する。（最大 16 バイト+NULL 文字）

void **Ptr（出力型）

テーブル領域のポインタを返却する。

size_t *Size（出力型）

テーブル領域のサイズを返却する。

戻り値

DIOSA_DONE(0)	正常終了。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータに誤りがある。
DIOSA_ENOENT(-8)	指定されたアイテム識別子は、定義されていない。

注意

- 返却するテーブル領域の共有メモリのポインタ(Ptr)は利用者側では領域の更新、解放してはならない。更新した場合、シグナル(SIGSEGV)が発生し異常終了する。
- 動的 SG 変更時、トランザクション内の値は一貫性を保証するため、SG 変更後、次のトランザクションまでは古い値を参照する。
- 初回の AP 共有情報の共有メモリへの展開(diapenv コマンド)は、本 API を実行するプロセス起動前に実行する必要がある。
- SG にテーブル識別子のみ設定しファイルパスを省略、またはファイルが存在しない場合、該当テーブル識別子を指定して本 API を実行するとテーブル領域ポインタに NULL を返却する。

関連

diosaapegetitem

2.1.5　　diosaapptrcclose（トレース情報ファイルクローズ）

名前

diosaapptrcclose –トレース情報ファイルをクローズする。

書式

```
#include <diosa.h>
int diosaapptrcclose(t_diosa_apptrcinfo *appinfo);
```

説明

diosaapptrcclose()は **appinfo** に指定された情報を元に、トレース情報ファイルをクローズする。
本 API は直接出力トレース情報ファイル、メモリ経由出力トレース情報ファイルの両方で使用できる。
appinfo はトレース情報ファイル構造体のポインタを指定する。
トレース情報ファイル構造体については、**t_diosa_apptrcinfo** を参照すること。

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_ERROR(-1)	その他エラー
DIOSA_EPARAM(-3)	パラメータエラー
DIOSA_EFCLOSE(-50)	ファイルクローズエラー

注意

- **diosaapptrcopen**(トレース情報ファイルオープン)で指定した **appinfo** を指定すること。

関連

t_diosa_apptrcinfo, diosaapptrcopen, diosaapptrcread

2.1.6　diosaapptrcf(ファイル直接トレース情報出力)

2.1.7　diosaapptrcf_position(ファイル直接トレース情報出力)

名前

diosaapptrcf – トレース情報をトレース情報ファイル(直接出力)に出力する
diosaapptrcf_position – ファイル名、行番号、関数名が指定可能なトレース情報出力

書式

```
#include <diosa.h>
int diosaapptrcf(char *comment, void *data, int size, int errcode, unsigned char level);
int diosaapptrcf_position(const char* file, int line, const char* func, char *comment, void *data,
int size, int errcode, unsigned char level);
```

説明

diosaapptrcf()はトレース情報ファイル(直接出力)にトレース情報を出力する。トレース情報には**diosaapptrcf**()をコールした日時、ソースファイル名、関数名、行番号と、以下の各パラメータで指定した情報が含まれる。
comment は任意の文字列を指定する。文字列は最長 50 バイトで、null で終端している必要がある。不要な場合は null を指定する。
data はトレース情報に含めるユーザデータのポインタを指定する。ただし、現在の動作環境がユーザデータを出力しない設定になっている場合、トレース情報に含まれない。
size はユーザデータの有効バイト長を指定する。
errcode はエラーコードを指定する。
level はトレース情報の出力レベルを 1～9 の整数値で指定する。出力レベルは値が小さいほど重要度が高くなることを意味する。ただし、**level** が現在の動作環境の出力レベルより大きい場合、トレース情報そのものが出力されない。
diosaapptrcf_position()は出力情報として、任意のソースファイル名、行番号、関数名を指定可能である。

戻り値

DIOSA_DONE(0)	トレース情報の出力に成功した。
DIOSA_IGNORE(9)	**level** が現在の出力レベルより大きいためトレース情報が出力されなかった。
DIOSA_EPARAM(-3)	パラメータに不正な値を指定した。(**size** が負の値である等)
DIOSA_ENOINIT(-11)	**diapptrcinit** コマンドが未実行の状態である。
DIOSA_EOVERFLOW(-39)	トレース情報ファイル(直接出力)のサイズが最大サイズに達した時にローテーション先にトレース情報ファイル(直接出力)が存在した。(環境変数 **DIOSA_APPTRCFFILEOUTPUTMODE** の値が ROTATE かつ **DIOSA_APPTRCFOVERWRITE** の値が OFF の場合のみ)
DIOSA_ERROR(-1)	内部処理に失敗した。

2.1.8　diosaapptrcm(メモリ経由トレース情報出力)

2.1.9　diosaapptrcm_position(メモリ経由トレース情報出力)

名前

diosaapptrcm – トレース情報をトレース情報保存領域に格納する
diosaapptrcm_position – ファイル名、行番号、関数名が指定可能なトレース情報出力

書式

```
#include <diosa.h>
int diosaapptrcm(char *comment, void *data, int size, int errcode, unsigned char level, int trcid);
int diosaapptrcm_position(const char* file, int line, const char* func, char *comment, void *data, int size, int errcode, unsigned char level, int trcid);
```

説明

diosaapptrcm()はトレース情報保存領域にトレース情報を格納する。トレース情報には **diosaapptrcm**()をコールした日時、ソースファイル名、関数名、行番号と、以下の各パラメータで指定した情報が含まれる。
comment は任意の文字列を指定する。文字列は最長 50 バイトで、null で終端している必要がある。不要な場合は null を指定する。
data はトレース情報に含めるユーザデータのポインタを指定する。ただし、現在の動作環境がユーザデータを出力しない設定になっている場合、トレース情報に含まれない。
size はユーザデータの有効バイト長を指定する。
errcode はエラーコードを指定する。
level はトレース情報の出力レベルを 1～9 の整数値で指定する。出力レベルは値が小さいほど重要度が高くなることを意味する。ただし、**level** が現在の動作環境の出力レベルより大きい場合、トレース情報そのものが格納されない。
trcid はトレース識別 ID を 1～環境変数「DIOSA_APPTRCMIDNUM」での指定値の範囲(最大 3)で指定する。範囲外の値を指定した場合は、範囲エラーとなる。
diosaapptrcm_position()は出力情報として、任意のソースファイル名、行番号、関数名を指定可能である。

戻り値

DIOSA_DONE(0)	トレース情報の格納に成功した。
DIOSA_IGNORE(9)	**level** が現在の出力レベルより大きいためトレース情報が格納されなかった。
DIOSA_EPARAM(-3)	パラメータに不正な値を指定した。(**size** が負の値又は環境変数 **DIOSA_APPTRCMSIZE** の設定値－管理情報約 256byte 以上の値が指定された。等)
DIOSA_ENOINIT(-11)	トレース情報保存領域が存在しない。**diapptrcinit** コマンドが未実行の状態である。
DIOSA_EOVERFLOW(-39)	トレース情報保存領域がファイルへ未出力のトレース情報でいっぱいになり、トレース情報が格納されなかった。(環境変数 **DIOSA_APPTRCMOVERWRITE** の値が OFF の場合のみ)
DIOSA_ERROR(-1)	内部処理に失敗した。
DIOSA_ERANGE(-10)	トレース識別 ID に範囲外の値が指定された。

2.1.10　diosaapptrcopen(トレース情報ファイルオープン)

名前

diosaapptrcopen – トレース情報ファイルをオープンする。

書式

```
#include <diosa.h>
int diosaapptrcopen(const char* file, t_diosa_apptrcinfo *appinfo);
```

説明

diosaapptrcopen()は **file** で指定されたファイルパス/ファイル名を元にトレース情報ファイルをオープンする。
本 API は直接出力トレース情報ファイル、メモリ経由出力トレース情報ファイルの両方で使用できる。
オープン後の入力ファイルチェックとして、ファイルの先頭からヘッダーレコードサイズ分レコードを読み込む。ファイル形式に問題が無い場合は、**appinfo** の AppComHeader(共通ヘッダ)にヘッダレコード(レコードタイプ'H')を返却する。
本 API 処理完了後は、それぞれのトレース情報の形式に合わせてファイル終端まで **diosaapptrcread**(トレース情報ファイル読み込み)を実行し、ファイルの終端に達したら **diosaapptrcclose**(トレース情報ファイルクローズ)を実行する。
なお、**diosaapptrcopen** を実行後にファイルの途中まで **diosaapptrcread** を実施してから **diosaapptrcclose** を実行することもできる。
また、本 API は DIOSA/XTP が未起動でも使用可能である。

file の指定方法を以下に示す。
①絶対パスで指定した場合、そのパスでファイル検索する。(最大長:255 バイト)
②相対パス/ファイル名で指定した場合、API 呼出し時のカレントディレクトリ配下を検索する。
(最大長:ディレクトリパス＋ファイル名で 255 バイト)
カレントディレクトリにも存在しない場合は以下の条件でファイルをオープンする。
なお、ファイルの拡張子が「.log」であれば直接出力トレース情報ファイル、「.trc」であればメモリ経由出力トレース情報ファイルと判断する。
＜直接出力トレース情報ファイル＞
環境変数 **DIOSA_APPTRCFPATH** または **diapptrcfenv** コマンドの-P オプションで指定したディレクトリ配下のトレース情報ファイルをオープンする。
＜メモリ経由出力トレース情報ファイル＞
環境変数 **DIOSA_APPTRCMPATH** または **diapptrcmenv** コマンドの-P オプションで指定したディレクトリ配下のトレース情報ファイルをオープンする。

appinfo はトレース情報ファイル構造体のポインタを指定する。
トレース情報ファイル構造体については、**t_diosa_apptrcinfo** を参照すること。

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_ERROR(-1)	その他エラー
DIOSA_ESYS(-2)	システムコールエラー
DIOSA_EPARAM(-3)	パラメータエラー
DIOSA_ENOENT(-8)	指定されたファイルが存在しない
DIOSA_EINVAL(-9)	書式不正エラー
DIOSA_EFOPEN(-47)	ファイルオープンエラー

DIOSA_EFREAD(-48)　　ファイル読み込みエラー

注意

- **diosaapptrcopen**()に渡す前に **appinfo** を 0 で初期化すること。
- 本 API で使用した **appinfo** は **diosaapptrcread**(トレース情報ファイル読み込み)、**diosaapptrcclose** (トレース情報ファイルクローズ)でも使用する。

関連

t_diosa_apptrcinfo, diosaapptrcread, diosaapptrcclose

2.1.11　diosaapptrcread（トレース情報ファイル読み込み）

名前

diosaapptrcread –トレース情報ファイルを読み込む。

書式

```
#include <diosa.h>
int diosaapptrcread(t_diosa_apptrcinfo *appinfo);
```

説明

diosaapptrcread()は **appinfo** に指定された情報を元に、トレース情報ファイルを読み込む。
本 API は直接出力トレース情報ファイル、メモリ経由出力トレース情報ファイルの両方で使用できる。
appinfo はトレース情報ファイル構造体のポインタを指定する。
トレースデータ読み込み用バッファのアドレスとサイズを、**appinfo** の PtrBuff と BuffSize に指定する。
また、読み込んだトレースデータサイズは ReadSize に返却される。

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_DATALIM(4)	ファイルの終わりに達した
DIOSA_ERROR(-1)	その他エラー
DIOSA_EPARAM(-3)	パラメータエラー
DIOSA_ENOBUFS(-7)	バッファサイズ不足。BuffSize より ReadSize が大きいときに返却する。 必要なバッファサイズを呼び元で確保した後、再実行することで処理を継続することが可能。また、この時ヘッダ情報は返却される。
DIOSA_EINVAL(-9)	書式不正エラー
DIOSA_EFREAD(-48)	ファイル読み込みエラー

注意

- **diosaapptrcopen**(トレース情報ファイルオープン)で指定した **appinfo** を指定すること。

関連

t_diosa_apptrcinfo, diosaapptrcopen, diosaapptrcclose

2.1.12　diosaapptrget(共通領域取得)

名前

diosaapptrget - DIOSA が保持している AP 共通領域のアドレスを返却する。

書式

```
#include <diosa.h>
int diosaapptrget (void  **Comarea);
```

説明

diosaapptrget() は **diosaapptrset**()で設定した共有メモリのアドレスを返却する。

void **Comarea(出力型)

diosaapptrset で通知された共通領域のアドレスを返却する。

戻り値

DIOSA_DONE(0)	処理が正常に終了した。
DIOSA_EPARAM(-3)	パラメータに誤りがある。
DIOSA_EFUNCNAV(-34)	機能が利用できない。

関連

diosaapptrset

2.1.13　diosaapptrset(共通領域設定)

名前

diosaapptrset - ユーザ AP が確保した共有メモリの先頭アドレスを DIOSA に通知する。

書式

```
#include <diosa.h>
int diosaapptrset(void  *Comarea);
```

説明

diosaapptrset() はユーザ AP が diosamalloc()で確保した共有メモリの先頭アドレスを DIOSA に通知する。

void *Comarea(入力型)

AP が確保した共通領域アドレスを指定する。

戻り値

DIOSA_DONE(0)	処理が正常に終了した。
DIOSA_EPARAM(-3)	パラメータに誤りがある。
DIOSA_EFUNCNAV(-34)	機能が利用できない。

注意

- 共通領域ポインタの通知を複数回行った場合、最後に通知したポインタのみが **diosaapptrget**()で返却される。

関連

diosaapptrget

2.1.14　diosacmdconf(コマンド配信結果取得関数)

名前

diosacmdconf - コマンド配信結果取得関数

書式

```
#include <diosa.h>
int    diosacmdconf( int Socket, int APITimeOut, t_diosa_cmdconfuca **ConfUca );
```

説明

diosacmdconf()は、配信したコマンドの実行結果を取得する。

Socket	コマンド配信時に返却される配信結果確認用ソケット識別子を指定する。
APITimeOut	コマンド配信結果の応答待ち合わせ時間を指定する。(単位:秒、範囲:-1,1～86400) DIOSA_CMDSND_DEFAULT(-1)を指定した場合、環境変数、または環境定義の値が利用される。
ConfUca	コマンド配信結果格納領域のポインタが返却される。

戻り値

成功した場合には、0 が返される。エラー時は、負の値が返される。

DIOSA_DONE(0)	正常終了(コマンド配信結果が全て正常終了)。
DIOSA_ALMOST(10)	コマンド処理不完全(コマンド配信結果が一部異常終了)。
DIOSA_EFATAL(-30)	コマンド処理異常(コマンド配信結果が全部異常終了)。
DIOSA_EPARAM(-3)	パラメータエラー。
DIOSA_EMEM(-6)	メモリ関連エラー。
DIOSA_ENOENT(-8)	指定した宛先が見つからない。
DIOSA_EPERM(-12)	実行権エラー。
DIOSA_ERECV(-20)	受信エラー。
DIOSA_ETIMEOUT(-22)	一定時間以内に返信がない。
DIOSA_EFUNCNAV(-34)	DIOSA 未起動。
DIOSA_ECONNECT(-71)	接続エラー。
DIOSA_ESEQ(-94)	処理順序不正。
DIOSA_ERROR(-1)	その他エラー。

関連

diosacmdsend,t_diosa_cmdconfuca

2.1.15　diosacmdsend(コマンド配信関数)

名前

diosacmdsend - コマンド配信関数

書式

```
#include <diosa.h>
int    diosacmdsend( t_diosa_cmdsenduca *SendUca, t_diosa_cmdconfuca **ConfUca, int *Socket );
```

説明

diosacmdsend()は、指定した宛先にコマンドの配信を行う。

SendUca	コマンド配信情報構造体のポインタを指定する。
ConfUca	コマンド配信結果格納領域のポインタが返却される。
Socket	コマンド配信結果確認用ソケット識別子が返却される。

戻り値

成功した場合には、0 が返される。エラー時は、負の値が返される。

DIOSA_DONE(0)	正常終了(コマンド配信結果が全て正常終了)。
DIOSA_ALMOST(10)	コマンド処理不完全(コマンド配信結果が一部異常終了)。
DIOSA_EFATAL(-30)	コマンド処理異常(コマンド配信結果が全部異常終了)。
DIOSA_EPARAM(-3)	パラメータエラー。
DIOSA_EMEM(-6)	メモリ関連エラー。
DIOSA_ENOENT(-8)	指定した宛先が見つからない。
DIOSA_EPERM(-12)	実行権エラー。
DIOSA_ESEND(-15)	送信エラー。
DIOSA_ERECV(-20)	受信エラー。
DIOSA_ETIMEOUT(-22)	一定時間以内に返信がない。
DIOSA_EFUNCNAV(-34)	DIOSA 未起動。
DIOSA_ECONNECT(-71)	接続エラー。
DIOSA_ESEQ(-94)	処理順序不正。
DIOSA_ERROR(-1)	その他エラー。

関連

diosacmdconf,t_diosa_cmdsenduca,t_diosa_cmdconfuca

2.1.16　diosacommit(コミット)

名前

diosacommit – トランザクションのコミット

書式

```
#include <diosa.h>
int   diosacommit( int TimeReset );
```

説明

コミット処理を実行する。
本 API は CO 制御 TPP、バッチ AP 制御、ユーザアプリケーションプログラムで使用可能である。
CO 制御では、API 発行までに diosasendtx で VD 宛に遅延送信されたものは有効化(送信)される。
TimeReset は監視機能における経過時間と CPU 時間のリセットを行うフラグであり、DIOSA_YES が指定されていると、コミット時にリセットを行い、DIOSA_NO の場合はリセットを行わない。
本 API 発行後にデッドロック、ロールバックリトライが発生した場合は、diosacommit 発行前までのメモリデータ管理は確定されているので、AP の責任で再処理開始位置を確定する必要がある。diosauca の CommitNum(コミット API(diosacommit)実行回数)を参照してメモリデータ管理更新処理部分をスキップする等の考慮が必要である。
ロールバックリトライ処理は、CO 制御とバッチ AP 制御のみ機能提供される。

戻り値

DIOSA_DONE(0)	正常に実行が終了した
DIOSA_SWITCH(21)	計画マスタ切り替え中
DIOSA_EFUNCNAV(-34)	動作環境エラー
DIOSA_EPARAM(-3)	パラメータエラー
DIOSA_EACCES(-25)	メモリデータ管理アクセスエラー
DIOSA_EDEADLOCK(-36)	デッドロックを検出した(パッケージ破棄含む)
DIOSA_ECOND(-60)	同一トランザクション内で別 MAP の更新が行われている
DIOSA_EDB(-69)	コミットエラー(DB)
DIOSA_ETAM(-113)	IM の障害によりコミットが失敗した
DIOSA_ESWITCH(-115)	障害時マスタ切り替え中
DIOSA_EREADY(-118)	サーバが起動中(再起動)や環境定義変更中により、アクセスするための準備ができていない。
DIOSA_EBEFORE(-111)	コミット前エラー
DIOSA_EAFTER(-112)	コミット後エラー
DIOSA_ETPBASE(-120)	TPBASE のエラー
DIOSA_ERROR (-1)	その他 diosa エラー

注意

本 API をユーザアプリケーションプログラムで使用する場合

- **diosatxstart()**を呼び出している必要がある。呼び出しがなかった場合、
  戻り値に DIOSA_EFUNCNAV(-34)が設定される。
- 戻り値が DIOSA_DONE(0)でなければ **diosarollback**()を呼び出す必要がある。

関連

diosatxstart, diosarollback

2.1.17　diosaetgreset(経過時間リセット)

名前

diosaetgreset - 経過時間リセット

書式

```
#include <diosa.h>
int diosaetgreset(void);
```

説明

diosaetgreset()は、登録された経過時間情報の経過時間をリセットする。

戻り値

成功した場合には、0 が返される。エラー時は、負の値が返される。

DIOSA_DONE(0)	正常終了
DIOSA_ESYS(-2)	システムコールエラーが発生した
DIOSA_ERANGE(-10)	最大リセット回数を超えて呼び出されている
DIOSA_ENOINIT(-11)	経過時間が監視されていない
DIOSA_EELAPOV(-91)	経過時間情報が存在しない

注意

- **diosaetgreset**()を実行した時点で、既に経過監視時間を超過している場合、DIOSA_DONE が返却され、経過時間のリセットは行われない。

2.1.18　diosaetgstart(経過時間監視再開)

名前

diosaetgstart - 経過時間監視再開

書式

```
#include <diosa.h>
int diosaetgstart(void);
```

説明

diosaetgstart()は、一時停止されている自プロセスの経過時間監視を再開する。なお、再開時には経過時間をリセットする。

戻り値

成功した場合には、0 が返される。エラー時は、負の値が返される。

DIOSA_DONE(0)	正常終了
DIOSA_ENOINIT(-11)	経過時間が監視されていない
DIOSA_EELAPOV(-91)	経過時間情報が存在しない

関連

diosaetgstop

2.1.19　diosaetgstop(経過時間監視停止)

名前

diosaetgstop - 経過時間監視停止

書式

```
#include <diosa.h>
Int diosaetgstop(void);
```

説明

diosaetgstop()は、自プロセスの経過時間監視を一時停止する。

戻り値

成功した場合には、0 が返される。エラー時は、負の値が返される。

DIOSA_DONE(0)	正常終了
DIOSA_ENOINIT(-11)	経過時間が監視されていない
DIOSA_EELAPOV(-91)	経過時間情報が存在しない

関連

diosaetgstart

2.1.20　diosaetgusinfo(経過時間監視ユーザ情報登録)

名前

diosaetgusinfo - 経過時間監視ユーザ情報登録

書式

```
#include <diosa.h>
Int    diosaetgusinfo (char *UsInfo);
```

説明

diosaetgusinfo()は、経過時間超過時の CDO メッセージに表示されるユーザ情報を登録／更新する。

UsInfo	ユーザ情報が格納されている領域を指定する。(最大 51 バイト、NULL 終端あり) 格納領域は必ず 51 バイト確保する。

戻り値

成功した場合には、0 が返される。エラー時は、負の値が返される。

DIOSA_DONE(0)	正常終了
DIOSA_EPARAM(-3)	パラメータが不正である
DIOSA_ENOINIT(-11)	経過時間が監視されていない
DIOSA_EELAPOV(-91)	経過時間情報が存在しない

2.1.21　diosagetlnodeinfo(自ノード情報取得関数)

名前

diosagetlnodeinfo – 自ノード情報取得関数

書式

```
#include <diosa.h>
int   diosagetlnodeinfo( t_diosa_lnodeinfo *info );
```

説明

diosagetlnodeinfo()は、API を発行したノードの情報を取得する。
info に指定した領域に自ノード情報が設定される。

t_diosa_nodeinfo info (出力型)

自ノード情報に関連する情報が格納される。
構造体の以下のメンバを使用する。

int　LsId
論理システム ID(出力)

char　LsName[16]
論理システム名(出力)

int　LNodeId
論理ノード ID(出力)

char　LNodeName[16]
論理ノード名(出力)

int　LNodeStatus
論理ノードの閉塞状態 (出力)

DIOSA_BLOCKAGE_ACT	: 稼働中
DIOSA_BLOCKAGE_BLK	: 閉塞中
DIOSA_BLOCKAGE_NOTFOUND	: 未定義
DIOSA_BLOCKAGE_PBK	: 予閉塞中

short LNodeType
論理ノード種別(出力)

DIOSA_LNODETYPE_AP	: AP ノード
DIOSA_LNODETYPE_DB	: DB ノード
DIOSA_LNODETYPE_OLTP	: OLTP ノード

short DiosaStartMode
起動モード(出力)

DIOSA_STARTMODE_COLD	: コールド
DIOSA_STARTMODE_WARM	: ウォーム

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_EPARAM(-3)	パラメータが正しくない
DIOSA_ENOINIT(-11)	プロセス初期化が行われていない

2.1.22　diosagetsrctx(トランザクション開始時電文の取得)

名前

diosagetsrctx - トランザクション開始時電文の取得

書式

```
#include <diosa.h>
int   diosagetsrctx( t_diosa_dcuca *DcUca,  char **Msg );
```

説明

トランザクション開始時に受信した電文を返却する。連鎖により実行された CO およびアボート出口#1,#2 において、連鎖前の CO が diosarecvtx により取得したものと同じ電文を取得する。
トランザクションにおいて、複数回の連鎖が行われた場合でも最初の CO が取得したものと同じ電文が返される。
連鎖が実行されていない場合は、diosarecvtx と同じ結果が得られる。

返される DcUca と Msg については、diosarecvtx と同様である。
取得できる電文は、受信電文解析出口を呼びだす前の電文となる。

戻り値

DIOSA_DONE(0)	正常に実行が終了した
DIOSA_EPARAM(-3)	パラメータエラー
DIOSA_EFUNCNAV(-34)	本マクロを使用できないプロセス上のＡＰから要求された

注意

- 圧縮された電文は圧縮状態で返却される。

関連

t_diosa_uca, diosarecvtx

2.1.23　diosagoback(CO 制御への強制リターン)

名前

diosagoack - CO 制御への強制リターン

書式

```
#include <diosa.h>
int diosagoback(void);
```

説明

プログラムの実行位置を本 API 呼び出し位置から CO 制御内のメイン処理へ移動する。
CO や利用者出口から return 文を実行することと同等であり、本 API を実行する場合は、事前に diosauca 領域にリターンコードを設定しておく必要がある。

戻り値

正常終了の場合、本 API はリターンしない。

異常が検出された場合、下記のエラーをリターンする。

DIOSA_EFUNCNAV(-34)　本マクロを使用できないプロセス上のＡＰから要求された

関連

t_diosa_uca

2.1.24　diosalock(ロック取得)

名前

diosalock - 指定されたモードに従い、ファイル型／DB 型のロックを取得する

書式

```
#include <diosa.h>
int diosalock(int Id, int Mode);
```

説明

diosalock() は指定されたモードに従い、ファイル型／DB 型のロックを取得する。

int Id(入力型)

識別子を指定する。
ロック範囲が論理システム内の場合、1～4096 の範囲内で指定する。
ロック範囲が論理ノード内の場合、1～1024 の範囲内で指定する。

int Mode(入力型)

ロック範囲、ロックモード、ウェイトオプションの各設定項目の論理和を指定する。ロック範囲が「論理システム内」の場合は DB 型ロック制御、「論理ノード内」の場合はファイル型ロック制御となる。この値が「0」の場合は、「論理ノード内+占有+WAIT」の指定と同等となる。

ロック範囲

DIOSA_INSYSTEM	論理システム内
DIOSA_INNODE	論理ノード内(既定値)

ロックモード

DIOSA_EXCLUSIVE	占有ロック(既定値)
DIOSA_SHARE	共有ロック

ウェイトオプション

DIOSA_WAIT	資源がロックされていたら解放されるまで待つ(既定値)
DIOSA_NOWAIT	資源がロックされていたらエラーとする

本関数で取得したロックは以下の時点で解放される。

１．同一ロック範囲の同一識別子に対して diosaunlock を実行し、正常終了したとき
２．CO 制御によるトランザクションの実行の終了、及び diosacommit/diosarollback、diosatrninit/diosatrnterm を実行したとき
３．プロセスが終了したとき
４．プロセス例外および異常終了処理が実行されたとき
５．CO 制御によりリトライ(再実行)されたとき
６．OracleDB のコミット／ロールバックしたとき(論理システム内ロック制御時)
７．DB 接続が切断されたとき(論理システム内ロック制御時)

同一ロック範囲の同一識別子に対する同スレッド内複数回のロック(多重ロック)は以下の動作となる。

論理システム内ロック制御(DB)

共有 → 共有：可
共有 → 占有：可
占有 → 共有：不可
占有 → 占有：可

論理ノード内ロック制御(File)

共有 → 共有：可
共有 → 占有：不可
占有 → 共有：不可
占有 → 占有：不可

ロック要求時、指定したロック範囲の同一識別子に対して既に他のプロセスが共有モードでロックしていた場合は共有モードのロックのみ取得可能である(占有モードのロックはできない)。ただし、このときに、指定したロック範囲の同一識別子に対して占有モードのロックがウェイトしていた場合、ファイル型ロックでは、共有モードでのロックが取得可能であるが、DB 型ロックでは、占有モードのロックが取得･解放されるまで、共有モードでのロックは取得できない。
ロック要求時、指定したロック範囲の同一識別子に対して既に他のプロセスが占有モードでロックしていた場合は共有モード／占有モード共にロックは取得できない。
ロック要求時にロックを即時確保できない場合は、設定で WAIT 指定があれば解放待ちとなり、また NOWAIT 指定ではエラーとなり即時復帰する。
同スレッド内で複数回行ったロック(多重ロック)は、一回のアンロックで解除される。
ファイル型ロックでは、fcntl 動作中にシグナルを受信すると、エラーを返却するが、DB 型ロックでは DBMS_LOCK (OracleDB の場合)、または勧告的ロック(PostgreSQL の場合)動作中にシグナルを受信してもエラーを返却しない。

戻り値

DIOSA_DONE(0)	正常終了した。
DIOSA_EPARAM(-3)	パラメータに誤りがある。
DIOSA_ENOBUFS(-7)	システムのロックテーブルのエントリ数が諸元値を超えた。
DIOSA_ELOCK(-14)	ロック制御に失敗した。
DIOSA_ERECV(-20)	DIOSA_INNODE 指定の場合、資源待ち状態でシグナルを受信した。
DIOSA_EEXIST(-27)	指定された資源が既にロックされているためロックを獲得できない。(Mode が DIOSA_NOWAIT の場合)
DIOSA_EFUNCNAV(-34)	データベース機能は無効化されている。
DIOSA_EALREADY(-35)	自スレッド内において、指定のロック ID でのロックが確保されており、ロックモードを変更できない。
DIOSA_EDEADLOCK(-36)	デッドロックを検出した。
DIOSA_EDB(-69)	論理システム内ロック制御に失敗した。
DIOSA_ECLOSE(-75)	データベースとの接続が切断された。

注意

- 本機能を利用して取得したロックはホットスタンバイには対応していない。
- 論理システム内ロック制御を行う場合は、「DB 監視機能」によってデータベースと接続されていることが前提条件である。
- シグナルハンドラからの呼び出しは不可である。
- DB 型ロックを保持したまま、fork を行ってはならない。
- ロック範囲が論理システム内の場合 4065～4096 は予約ロック識別子で利用してはならない。また、ロック範囲が論理ノード内の場合 641～1024 は予約ロック識別子で利用してはならない。
- PostgreSQL の DB 型ロックで戻り値が DIOSA_EDEADLOCK、DIOSA_EDB の場合、ロック制御失敗後に DB アクセスするとエラーとなるため、トランザクションをロールバックする必要がある。
- DB 型ロックについて、論理システムが異なれば同一識別子が使用可能である。

関連

diosaunlock

2.1.25　diosamaddr(メモリアドレス取得)

名前

diosamaddr - diosamalloc、diosarealloc で確保した領域のアドレスを取得する。

書式

```
#include <diosa.h>
int diosamaddr(char *Id, int Type, void **Area);
```

説明

diosamaddr() は diosamalloc、diosarealloc で割り当てた領域のアドレスを取得する。

char *Id(入力型)

メモリ識別子領域ポインタを指定する。(1～16 バイト、必須)

int Type(入力型)

領域種別を指定する。(必須)

DIOSA_APNOPROT	更新可共有メモリ
DIOSA_APPROT	保護属性共有メモリ
DIOSA_PRCNOALL	一括解放対象外プロセスメモリ (一括解放対象外プロセス内メモリ)
DIOSA_THRNOALL	一括解放対象外プロセスメモリ (一括解放対象外スレッド内メモリ)
DIOSA_SVCALL	一括解放対象プロセスメモリ (一括解放対象サービス内メモリ)

void **Area(出力型)

識別子に対応する領域のアドレスが返却される。

戻り値

DIOSA_DONE(0)	処理が正常に終了した。
DIOSA_ERROR(-1)	アドレス取得に失敗した。
DIOSA_EPARAM(-3)	パラメータに誤りがある。
DIOSA_ENOENT(-8)	指定されたメモリ識別子に対応する領域は存在しない。
DIOSA_EFUNCNAV(-34)	メモリバッファを指定していないので、メモリ管理機能は使用できない。

注意

- メモリ割り当て時に識別子を指定していなかった場合は、本関数を使用できない。

関連

diosamalloc, diosarealloc

2.1.26　diosamalloc(メモリ割り当て)

名前

diosamalloc - メモリ領域の割り当てを行う。

書式

```
#include <diosa.h>
int diosamalloc(char *Id, char *Entry, int Type, int Size, void **Area);
```

説明

diosamalloc() は Type で指定されたメモリ領域を Size バイト割り当てる。更新可共有メモリ、保護属性共有メモリ、プロセスメモリの各領域間で同一のメモリ識別子を使用できる。指定されたメモリ識別子、サイズと重複するメモリがその領域に既に存在する場合は DIOSA_ALREADY を返す。
実際にメモリ管理が割り付けるサイズは 32 バイトの倍数である。

char *Id(入力型)

メモリ識別子領域ポインタ（1～16 バイト ）
省略したい場合は、NULL を指定する。

char *Entry(入力型)

エントリキー領域ポインタ（1～16 バイト ）
省略したい場合は、NULL を指定する。

int Type(入力型)

領域種別を指定する。(必須)

DIOSA_APNOPROT	更新可共有メモリ
DIOSA_APPROT	保護属性共有メモリ
DIOSA_PRCNOALL	一括解放対象外プロセスメモリ (一括解放対象外プロセス内メモリ)
DIOSA_THRNOALL	一括解放対象外プロセスメモリ (一括解放対象外スレッド内メモリ)
DIOSA_SVCALL	一括解放対象プロセスメモリ (一括解放対象サービス内メモリ)

int Size(入力型)

割り当て領域のサイズをバイト単位で指定する。(必須)

void **Area(出力型)

割り当てられた領域のアドレスが返却される。

戻り値

DIOSA_DONE(0)	処理が正常に終了した。
DIOSA_ALREADY(1)	指定されたメモリ識別子に対応する領域が既に存在する。
DIOSA_EPARAM(-3)	パラメータに誤りがある。
DIOSA_EMEM(-6)	領域確保に失敗した。
DIOSA_ENOBUFS(-7)	割り当て総サイズが定義した値を超えた。
DIOSA_EINVAL(-9)	サイズ指定が不正である。
DIOSA_ESIZE(-33)	指定されたメモリ識別子に対応する領域が既に存在するが、サイズが不一致である。
DIOSA_EFUNCNAV(-34)	メモリバッファを指定していないので、メモリ管理機能は使用できない。 または、排他制御に失敗した。

注意

- メモリ識別子を省略した場合、メモリ識別が必須な機能（diosarealloc、diosamaddr)を利用できない。

- エントリキーを省略した場合、メモリ情報表示コマンドでエントリキーによる絞り抽出ができない。
- 領域種別の一括解放対象プロセスメモリは、トランザクション初期化時に一括解放される。

関連

diosarealloc, diosamfree

2.1.27　diosamcopy(保護属性書き込み)

名前

diosamcopy - 保護属性共有メモリにデータを書き込む。

書式

```
#include <diosa.h>
int diosamcopy(void *From, void *To, int Size);
```

説明

diosamcopy() は保護属性共有メモリに From で指定されたデータを書き込む。指定サイズが複写するデータサイズより小さい場合、指定サイズまで複写する。

void *From(入力型)

複写元領域アドレスを指定する。(必須)

void *To(入力型)

複写先領域アドレスを指定する。(必須)
この領域は、diosamalloc 又は diosarealloc で保護属性で割り当てられた領域でなければならない。

int Size(入力型)

複写データサイズを指定する。(必須)

戻り値

DIOSA_DONE(0)	処理が正常に終了した。
DIOSA_ERROR(-1)	アドレス取得に失敗した。
DIOSA_EPARAM(-3)	パラメータに誤りがある。
DIOSA_ENOENT(-8)	指定された領域が存在しない。
DIOSA_ERANGE(-10)	割り当て領域の範囲外を指定している。
DIOSA_EFUNCNAV(-34)	機能利用不可。

関連

diosamalloc, diosarealloc

2.1.28　diosamfree(メモリ解放)

名前

diosamfree - diosamalloc、diosarealloc で割り当てた領域の解放を行う。

書式

```
#include <diosa.h>
int diosamfree(char *Id, int Type, void *Farea);
```

説明

diosamfree() は diosamalloc、diosarealloc で割り当てた領域の解放を行う。
Id と Farea に NULL、Type に DIOSA_SVCALL を指定した場合、全ての一括解放対象プロセスメモリを解放する。
Farea に NULL を指定し、Id に識別子を指定した場合、識別子で解放を行う。この時、Type の指定は必須である。
Id に NULL を指定し、Farea に解放する領域のアドレスを指定した場合、アドレスで解放を行う。この時、Type は無視される。
Id と Farea の両方が NULL 以外の場合は、パラメータエラー (DIOSA_EPARAM) になる。

char *Id(入力型)

メモリ識別子領域ポインタを指定する。(1～16 バイト)

int Type(入力型)

領域種別を指定する。

DIOSA_APNOPROT	更新可共有メモリ
DIOSA_APPROT	保護属性共有メモリ
DIOSA_PRCNOALL	一括解放対象外プロセスメモリ (一括解放対象外プロセス内メモリ)
DIOSA_THRNOALL	一括解放対象外プロセスメモリ (一括解放対象外スレッド内メモリ)
DIOSA_SVCALL	一括解放対象プロセスメモリ (一括解放対象サービス内メモリ)

void *Farea(入力型)

解放する領域のポインタを指定する。

戻り値

DIOSA_DONE(0)	処理が正常に終了した。
DIOSA_ERROR(-1)	メモリ解放が失敗した。
DIOSA_EPARAM(-3)	パラメータに誤りがある。
DIOSA_ENOENT(-8)	指定されたメモリ識別子に対応する領域は存在しない。
DIOSA_EFUNCNAV(-34)	メモリバッファを指定していないので、メモリ管理機能は使用できない。 または、排他制御に失敗した。

関連

diosamalloc, diosarealloc

2.1.29　diosamsgbufalloc(電文バッファ確保)

名前

diosamsgbufalloc - 電文バッファの確保

書式

```
#include <diosa.h>
int   diosamsgbufalloc( int ReqSize,  t_diosa_msgbuf **MsgBuf );
```

説明

本 API は **ReqSize** によって指定された大きさの電文領域を確保し、その制御ポインタを **MsgBuf** の指す領域に格納する。
電文領域のアドレスとサイズ（＝**ReqSize**）は(***MsgBuf**)の Addr と Size に格納される。
両パラメータ Addr と Size を変更してはならない。

本 API が返す電文領域は、電文の DIOSA の制御部を格納する領域を含まず、すべての領域を利用者が扱うことができる。

戻り値

DIOSA_DONE	正常
DIOSA_EPARAM(-3)	ReqSize が負の値、MsgBuf が NULL
DIOSA_EMEM(-6)	メモリ不足
DIOSA_EFUNCNAV(-34)	本マクロを使用できないプロセス上のＡＰから要求された

関連

t_diosa_msgbuf, diosamsgbuffree, diossendtx, t_diosa_analyze

2.1.30　diosamsgbuffree(電文バッファ解放)

名前

diosamsgbuffree - 電文バッファの解放

書式

```
#include <diosa.h>
int   diosamsgbuffree( t_diosa_msgbuf *MsgBuf );
```

説明

MsgBuf によって指定された電文バッファを解放する。

戻り値

DIOSA_DONE	正常
DIOSA_EPARAM(-3)	MsgBuf が NULL
DIOSA_ENOINIT(-11)	DIOSA が起動していない。
DIOSA_EFUNCNAV(-34)	本マクロを使用できないプロセス上のＡＰから要求された

関連

t_diosa_msgbuf, diosmsgbufalloc

2.1.31　diosamsgdisp(メッセージ出力関数)

名前

diosamsgdisp - メッセージをメッセージログファイル、標準エラー出力へ出力する

書式

```
#include <diosa.h>
int diosamsgdisp(char *MsgId, int ForceSend, int Output, char *Format, ...);
```

説明

MsgId で指定したメッセージをメッセージログファイル、標準エラー出力へ出力する。
本 API を呼び出す場合はプロセス初期化 (**diosaprcinit**)、スレッド初期化 (**diosathrinit**)を呼び出した後に行うことを推奨する。プロセス初期化が呼び出されていない状態で本 API を呼び出した場合、メッセージログファイルへは AP プロセス共通データと AP スレッド固有データには何も設定されていない状態で出力される。
パラメータの設定が正しくない場合、以下の設定でメッセージを出力する。

- メッセージ ID が NULL の場合は専用のメッセージ ID を設定する
- 置換テキスト数が 11 以上指定された場合は、11 以降を無視する
- 置換テキストが 1024 バイト('¥0'含む)を超えた場合、1024 バイト目を'¥0'とする
- 置換テキスト表示書式が NULL の場合は ″″ を設定する

char *MsgId(入力)　　出力するメッセージのメッセージ ID を指定する。
int ForceSend(入力)　　強制送信選択。コミット同期の指定を行う。

DIOSA_ON　: 強制送信を行う
DIOSA_OFF : 強制送信を行わない
ユーザアプリケーション(diosaprcinit を行うプロセス上のアプリケーション)での使用の場合、本パラメータの指定値に関係なく DIOSA_ON が指定されたものとして動作する。

int Output(入力) メッセージの出力先を指定する。

DIOSA_MSG_LOG(1) : メッセージログファイルへ出力する
DIOSA_MSG_ERR(2) : 標準エラー出力へ出力する
メッセージログファイルと標準エラー出力の両方へ出力する場合、OR(|)で設定する。

char *Format(入力)

置換テキストの表示書式を **sprintf** に準拠した形式で、置換テキスト数分指定する。
(例：置換テキストが、数値、文字列、数値のとき ″%d%s%d″)
以下の書式の設定が可能。

書式	説明	引数に指定する型
%c	文字	int
%[[.]桁数]s	文字列(桁数指定可)	char*
%[[0]桁数]d	符号あり 10 進整数(前 0 埋め、桁数指定可)	int
%[[0]桁数]u	符号なし 10 進整数(前 0 埋め、桁数指定可)	unsigned int
%[[0]桁数]x	符号なし 16 進整数(前 0 埋め、桁数指定可)	int
%[[0]桁数]ld	符号あり 10 進長整数(桁数指定なし)	long
%[[0]桁数]lu	符号なし 10 進長整数(桁数指定なし)	unsigned long
%[[0]桁数]lx	符号なし 16 進長整数(前 0 埋め、桁数指定可)	long

本パラメータは最大で 10 個まで繰り返し指定することができる。

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_ESYS(-2)	システムコールエラー
DIOSA_EPARAM(-3)	引数、もしくは環境変数設定内容不正
DIOSA_ENOENT(-8)	メッセージ ID が見つからない
DIOSA_EINVAL(-9)	書式不正エラー
DIOSA_ENOINIT(-11)	プロセス初期化に失敗した
DIOSA_ESEND(-15)	送信エラー
DIOSA_EWAIT(-16)	ウエイトエラー
DIOSA_ERECV(-20)	受信エラー
DIOSA_ETIMEOUT(-22)	タイムアウトエラー
DIOSA_EALREADY(-35)	二重処理エラー
DIOSA_EFILE(-46)	ファイルアクセスエラー
DIOSA_EFOPEN(-47)	ファイルオープンエラー
DIOSA_EFCLOSE(-50)	ファイルクローズエラー
DIOSA_EPTHREAD(-58)	スレッド生成エラー
DIOSA_ESOCKET(-70)	ソケット生成エラー
DIOSA_ECONNECT(-71)	コネクトエラー
DIOSA_EENV(-78)	環境変数の取得に失敗した

注意

- パラメータ不正の場合、メッセージ出力は行うが、本 API はパラメータ不正の警告をリターンコードで返却する。
- SIGPIPE はコール元で無視する設定をする必要がある。(本関数内では操作しない)
- 本関数はプロセス内共通テーブルに非同期シグナル区間の設定がある場合は非同期シグナルセーフとして動作する。ただし、一部機能が制限される。
- 表示書式と引数の型・個数が一致しない場合の動作は保証しない。

関連

diosaprcinit, diosathrinit

2.1.32　diosamsgedit(メッセージ編集関数)

名前

diosamsgedit - メッセージを編集し、編集結果を返却する。

書式

```
#include <diosa.h>
int diosamsgedit(char *MsgId, char *Msg, int MsgLen, char *Format, ...);
```

説明

MsgId で指定したメッセージを編集し、**Msg** に出力する。
本 API を呼び出す場合はプロセス初期化 (**diosaprcinit**)、スレッド初期化 (**diosathrinit**)を呼び出した後に行うことを推奨する。プロセス初期化が呼び出されていない状態で本 API を呼び出した場合、メッセージログファイルへは AP プロセス共通データと AP スレッド固有データには何も設定されていない状態で出力される。
パラメータの設定が正しくない場合、以下の設定でメッセージを出力する。

- メッセージ ID が NULL の場合は専用のメッセージ ID を設定する
- 置換テキスト数が 11 以上指定された場合は、11 以降を無視する
- 置換テキストが 1024 バイト('¥0'含む)を超えた場合、1024 バイト目を'¥0'とする
- 置換テキスト表示書式が NULL の場合は ″″ を設定する

char *MsgId(入力)　　出力するメッセージのメッセージ ID を指定する。
char *Msg(出力)　　編集後メッセージを受け取るために確保した領域の先頭アドレスを指定する。
int MsgLen(入力) 編集後メッセージを受け取るために確保した領域のサイズを指定する。
char *Format(入力)
　置換テキストの表示書式を **sprintf** に準拠した形式で、置換テキスト数分指定する。
　(例：置換テキストが、数値、文字列、数値のとき ″%d%s%d″)
　以下の書式の設定が可能である。

書式	説明	引数に指定する型
%c	文字	int
%[[.]桁数]s	文字列(桁数指定可)	char*
%[[0]桁数]d	符号あり 10 進整数(前 0 埋め、桁数指定可)	int
%[[0]桁数]u	符号なし 10 進整数(前 0 埋め、桁数指定可)	unsigned int
%[[0]桁数]x	符号なし 16 進整数(前 0 埋め、桁数指定可)	int
%[[0]桁数]ld	符号あり 10 進長整数(桁数指定なし)	long
%[[0]桁数]lu	符号なし 10 進長整数(桁数指定なし)	unsigned long
%[[0]桁数]lx	符号なし 16 進長整数(前 0 埋め、桁数指定可)	long

　本パラメータは最大で 10 個まで繰り返し指定することができる。

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_ESYS(-2)	システムコールエラー
DIOSA_EPARAM(-3)	引数、もしくは環境変数設定内容不正
DIOSA_ENOENT(-8)	メッセージ ID が見つからない
DIOSA_EINVAL(-9)	書式不正エラー
DIOSA_ENOINIT(-11)	プロセス初期化に失敗した
DIOSA_EFOPEN(-47)	ファイルオープンエラー

注意

- パラメータ不正の場合、メッセージ出力は行うが、本 API はパラメータ不正の警告をリターンコードで返却する。

- SIGPIPE はコール元で無視する設定をする必要がある。(本関数内では操作しない)
- 本関数はプロセス内共通テーブルに非同期シグナル区間の設定がある場合は非同期シグナルセーフとして動作する。ただし、一部機能が制限される。
- 表示書式と引数の型・個数が一致しない場合の動作は保証しない。

関連

diosaprcinit, diosathrinit, DIOSA_MSG_LOCALE

2.1.33　diosaopsusiput(稼動統計ユーザ情報登録)

名前

diosaopsusiput - 稼動統計ユーザ情報登録

書式

```
#include <diosa.h>
int    diosaopsusiput ( short    UsType,
                        int      UsInfoLen,
                        char    *UsInfo )
```

説明

diosaopsusiput()は、稼動統計情報として出力したいユーザ情報を登録する。

UsType　　ユーザ情報がバイナリ形式、文字列のいずれであるかを指定する。
　　DIOSA_USINFOTYPE_BIN　バイナリ形式データ
　　DIOSA_USINFOTYPE_CHAR　文字列形式データ

UsInfoLen　　出力するユーザ情報長を指定する。(最大 50、NULL 終端を含まない長さ)

UsInfo　　稼動統計情報として出力するユーザ情報が格納されている領域を指定する。

戻り値

成功した場合には、0 が返される。エラー時は、負の値が返される。

DIOSA_DONE(0)　　正常終了

DIOSA_EPARAM(-3)　　パラメータ不正

DIOSA_ENOINIT(-11)　　初期化処理未実施

注意

- CO 制御、バッチ AP 制御上においては、CO 内で実行することで CO の稼動統計情報にユーザ情報を登録することができる。ユーザ AP においては、diosatrninit()～diosatrnterm()区間内で実行することでその区間の稼動統計情報にユーザ情報を登録することができる。
- 同一区間内で登録可能なユーザ情報は１つであり、複数回実行された場合は最後に登録されたユーザ情報が有効となる。

2.1.34　diosaperfend(稼動統計任意区間終了)

名前

diosaperfend - 稼動統計任意区間終了

書式

```
#include <diosa.h>
int    diosaperfend ( t_diosa_perfuca   *PerfUca )
```

説明

diosaperfend()は、任意区間の稼動統計情報出力において、区間の終了を通知する。
diosaperfstart()から本 API までの区間について稼動統計情報を採取する。

t_diosa_perfuca *PerfUca

PerfUca は、区間終了時の情報を指定する。通常は、区間開始時に指定した領域をそのまま指定する。
（区間開始時とは別領域を利用する場合は、**DIOSA_PERFUCA_INIT** で初期化しておくこと。 ）
NULL を指定することで省略可。
省略した場合、Id、Status には 0 が指定されたことになる。

構造体の以下のメンバを使用する。

int　Id（入力）

対応する区間開始時に返却された区間識別子(1 以上)を指定する。
0 を指定した場合、直近に呼び出された区間終了が通知されていない **diosaperfstart()**に対応させる。
（初期値は 0）

int　Status（入力）

利用者任意区間の稼動統計情報に出力される区間終了ステータスを指定する。
（任意の値を指定可能、初期値は 0）

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_ENOINIT(-11)	初期化処理未実施
DIOSA_ENOENT(-8)	対応する区間開始情報が存在しない

注意

- 呼び出し元のプロセス毎に設定される稼動統計情報の採取有無に従って動作する。稼動統計情報を採取しない状態で呼び出された場合、何もせず DIOSA_DONE が返却される。
- トランザクション処理中にプロセス例外が発生した場合、またはトランザクション終了時に稼動統計デーモンが未起動の場合、そのトランザクション内で採取した稼動統計任意区間情報は全て破棄される。

2.1.35　diosaperfstart(稼動統計任意区間開始)

名前

diosaperfstart - 稼動統計任意区間開始

書式

```
#include <diosa.h>
t_diosa_perfuca PerfUca = DIOSA_PERFUCA_INIT;
int    diosaperfstart ( t_diosa_perfuca *PerfUca )
```

説明

diosaperfstart()は、任意区間の稼動統計情報出力において、区間の開始を通知する。
本 API から **diosaperfend()**までの区間について稼動統計情報を採取する。
区間は複数ネストして利用することも可能である。

t_diosa_perfuca *PerfUca

PerfUca は、区間開始時の情報を指定する。利用する場合は、**DIOSA_PERFUCA_INIT** で初期化しておくこと。
NULL を指定することで省略可。
省略した場合、以下のように動作する。
・Label には NULL が指定されたことになる
・Id は返却されないため、区間終了時は 0 を指定する必要がある

構造体の以下のメンバを使用する。

char　Label[31+1]（入力）
稼動統計情報に出力する区間名を指定する。
（最大 31 バイト、NULL 終端で終了すること）
（NULL を指定することで省略可、初期値は NULL）

int　Id（出力）
区間終了時に指定するための区間識別子が返却される。
返却された値を対応する区間終了の際に **diosaperfend()**に指定する。

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_ENOINIT(-11)	初期化処理未実施
DIOSA_EOVERFLOW(-39)	格納先エントリ不足

注意

- 呼び出し元のプロセス毎に設定される稼動統計情報の採取有無に従って動作する。稼動統計情報を採取しない状態で呼び出された場合、何もせず DIOSA_DONE が返却される。
- 稼動統計任意区間情報は、1 トランザクション分の情報がプロセス(スレッド)単位に確保されるメモリ領域に保持される。本領域サイズは環境変数 DIOSA_PERFENTRYNUM によって決定し、格納先が不足した場合は DIOSA_EOVERFLOW が返却され、利用者任意区間情報は採取されない。
- 本 API で区間開始を通知し、トランザクション区間内に diosaperfend()で区間終了が通知されなかった情報は破棄される。

2.1.36　diosaprcforkinit(fork 後プロセス初期化関数)

名前

diosaprcforkinit - ユーザアプリケーションのための fork 後のプロセス初期処理を行う。

書式

```
#include <diosa.h>
int diosaprcforkinit(t_diosa_comdata* ComData)
```

説明

ユーザアプリケーション(CO 制御、バッチ AP 制御上のアプリケーション、ログリーダは除く)のための **fork** 後のプロセス初期処理を行う。

t_diosa_comdata ComData (入力型)

プロセス初期化の詳細設定を行う。
NULL を指定することで省略可。
省略した場合、EtgFlag には DIOSA_OFF が指定されたことになる。

構造体の以下のメンバを使用する。

int　EtgFlag

経過時間監視機能の利用有無を指定する。(入力)

DIOSA_ON　: 経過時間監視機能を利用する。
DIOSA_OFF　: 経過時間監視機能を利用しない。

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_EPARAM(-3)	t_diosa_comdata 設定エラー
DIOSA_EFATAL(-30)	異常終了

注意

- 本関数を発行した場合、必ず **diosaprcpreexecterm** 関数を実行すること。(本関数が異常終了した場合も発行すること)
- 本関数はシングルスレッド用のプロセス初期化を行う。

関連

diosaprcpreexecterm

2.1.37　diosaprcinit(プロセス初期化関数)

名前

diosaprcinit - ユーザアプリケーションのためのプロセス初期処理を行う。

書式

```
#include <diosa.h>
int diosaprcinit(t_diosa_comdata* ComData)
```

説明

ユーザアプリケーション(CO 制御、バッチ AP 制御上のアプリケーション、ログリーダは除く)のためのプロセス初期処理を行う。

t_diosa_comdata ComData (入力型)

プロセス初期化の詳細設定を行う。
NULL を指定することで省略可。
省略した場合、
・ThreadType には DIOSA_APPTYPE_SINGLE が指定されたことになる。
・EtgFlag には DIOSA_OFF が指定されたことになる。

構造体の以下のメンバを使用する。

char　ThreadType
スレッド種別を指定する（入力）
DIOSA_APPTYPE_SINGLE ：シングルスレッド用のプロセス初期化を行う。
DIOSA_APPTYPE_MULTI　：マルチスレッド用のプロセス初期化を行う。

int　EtgFlag
経過時間監視機能の利用有無を指定する（入力）
DIOSA_ON　：経過時間監視機能を利用する。
DIOSA_OFF　：経過時間監視機能を利用しない。

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_EPARAM(-3)	t_diosa_comdata 設定エラー
DIOSA_EFATAL(-30)	異常終了

注意

- 本関数を発行した場合、必ず **diosaprcterm**(プロセス終了処理)を実行すること。(本関数が異常終了した場合も呼ぶこと)
- **diosaprcterm**(プロセス終了処理)実行後、再度 **diosaprcinit**(プロセス初期化処理)を実行することはできない。

関連

diosaprcterm

2.1.38　diosaprcpreexecterm(exec 前プロセス終了関数)

名前

diosaprcpreexecterm - ユーザアプリケーションのためのプロセス終了処理を行う。

書式

```
#include <diosa.h>
int diosaprcpreexecterm(void* Info)
```

説明

ユーザアプリケーション(CO 制御、バッチ AP 制御上のアプリケーション、ログリーダは除く)のための exec 前のプロセス終了処理を行う。

void* Info(入力) 将来の拡張の為の予約領域。NULL を指定する。

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_EFATAL(-30)	異常終了

注意

- **diosaprcforkinit** 関数を発行した場合、必ず本関数を実行すること。

関連

diosaprcforkinit

2.1.39　diosaprcterm(プロセス終了関数)

名前

diosaprcterm - ユーザアプリケーションのためのプロセス終了処理を行う。

書式

```
#include <diosa.h>
int diosaprcterm(void* Info)
```

説明

ユーザアプリケーション(CO 制御、バッチ AP 制御上のアプリケーション、ログリーダは除く)のためのプロセス終了処理を行う。

void* Info(入力) 将来の拡張の為の予約領域。NULL を指定する。

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_EFATAL(-30)	異常終了

注意

- **diosaprcinit**(プロセス初期化処理)を呼んだ場合、必ず本関数を実行すること。

関連

diosaprcinit

2.1.40　diosarealloc(メモリ再割り当て)

名前

diosarealloc - メモリ領域の再割り当てを行う。

書式

```
#include <diosa.h>
int diosarealloc(char *Id, char *Entry, int Type, int Size, void **Ptr);
```

説明

diosarealloc() は diosamalloc で割り当てた領域のサイズを変更し、メモリを再度割り当てる。Size に 0 を指定した場合は、diosamalloc で割り当てた領域を解放する。メモリの内容は引き継ぐが、Size で指定したサイズが前回より小さい場合は、そのサイズまでの内容を複写する。
実際にメモリ管理が割り付けるサイズは 32 バイトの倍数である。再割り当て後のメモリ属性は、再割り当て前と同じ属性となる。

char *Id(入力型)

メモリ識別子領域ポインタ（1～16 バイト)(必須)

char *Entry(入力型)

エントリキー領域ポインタ（1～16 バイト)
再割り当て後の領域に対するエントリキーを指定する。
省略した場合（NULL を指定)、元の領域のエントリキーを引き継ぐ。

int Type(入力型)

領域種別を指定する。(必須)

DIOSA_APNOPROT	更新可共有メモリ
DIOSA_APPROT	保護属性共有メモリ
DIOSA_PRCNOALL	一括解放対象外プロセスメモリ（ 一括解放対象外プロセス内メモリ ）
DIOSA_THRNOALL	一括解放対象外プロセスメモリ（ 一括解放対象外スレッド内メモリ ）
DIOSA_SVCALL	一括解放対象プロセスメモリ（ 一括解放対象サービス内メモリ ）

int Size(入力型)

割り当て領域のサイズをバイト単位で指定する。(必須)

void **Ptr(出力型)

割り当てられた領域のアドレスが返却される。

戻り値

DIOSA_DONE(0)	処理が正常に終了した。
DIOSA_EPARAM(-3)	パラメータに誤りがある。
DIOSA_EMEM(-6)	領域確保に失敗した。
DIOSA_ENOBUFS(-7)	割り当て総サイズが定義した値を超えた。
DIOSA_ENOENT(-8)	指定されたメモリ識別子に対応する領域は存在しない。
DIOSA_EINVAL(-9)	サイズ指定が不正である。
DIOSA_EFUNCNAV(-34)	メモリバッファを指定していないので、メモリ管理機能は使用できない。

注意

- メモリ割り当て時に識別子を指定していなかった場合は、本関数を使用できない。

関連

diosamalloc, diosamfree

2.1.41　diosarecvtx(電文受信)

名前

diosarecvtx - トランザクション電文の受信

書式

```
#include <diosa.h>
int   diosarecvtx( t_diosa_dcuca *DcUca,  char **Msg );
```

説明

CO およびアボート出口#1, #2 において、受信電文を取得する。連鎖要求後には連鎖を要求した電文が返却される。
受信電文は、受信電文バッファのポインタが **Msg** の指す領域に格納される。
電文長や送信元などの受信電文に関する情報は **DcUca** に格納される。

電文タイプごとに設定される **DcUca** のパラメータの対応表を欄外に示す。

戻り値

DIOSA_DONE(0)	正常に実行が終了した
DIOSA_EPARAM(-3)	パラメータエラー
DIOSA_EACCES(-25)	連鎖処理が終了していないため、読み込むデータがない
DIOSA_EFUNCNAV(-34)	本マクロを使用できないプロセス上のＡＰから要求された

関連

t_diosa_dcuca, diosagetsrctx

＜diosarecvtx＞

t_diosa_dcuca		電文タイプ(MsgType)						保留　注 8)
		論理システム間	論理システム内派生 (CoName 指定)　注 4)	論理システム内派生 (TxId 指定)　注 4)	論理システム内派生 (中継 Co 指定)　注 5)	論理システム内派生 (中継 TxId 指定) 注	連鎖　注 1)	
MsgType		●	●	●	●	●	●	－
LSys	LsName	□	□	□	□	□	－	－
	AcsPoint	□	□	□	□	□	－	－
	Protocol	□	□	□	□	□	－	－
	TermName	●	□	□	□	□	－	－
	TermType　注 7)	●	□	□	□	□	－	－
	ConnectTime	●	□	□	□	□	－	－
Entr	LNodeName	●	□	□	□	□	－	－
	TpMonitor	●	□	□	□	□	－	－
LNode	LNodeName	－	□	□	□	□	－	－
	TpMonitor	－	□	□	□	□	－	－
Db 注 3)	Type	－	□	□	□	□	－	－
	MainKey	－	□	□	□	□	－	－
	MapId	－	□	□	□	□	－	－
Source　注 6)		●	●	●	●	●	－	－
RecvType		●	●	●	●	●	－	－
TxId		●	●	●	●	●	－	●
VdName		－	□	□	□	□	－	－
CoName　注 2)		□	●	□	●	□	●	□
MsgSize		●	●	●	●	●	●	●
SendCnt		－	●	●	●	●	－	－
MsgGnt		□	□	□	□	□	－	－
APInfo		□	□	□	□	□	－	－

●：必ず設定される
□：設定される場合がある
文字属性で先頭 NULL 文字は指定なし。

注1)　MsgType、CO 名(CoName)と電文長(MsgSize)以外はトランザクション開始時に受信した電文情報が返却される。

注2)　電文送信(diosasendtx)時に指定された CO 名が格納される。

注3)　直近の diosasendtx で指定された Db 情報が返却され、論理システム内で引き継がれる情報となる。

注4)　論理システム間から受信した電文のトランザクション処理から論理システム内派生した場合、LSys 情報は論理システム間情報が引き継がれる。

注5)　中継指定された電文受信時は、中継要求の diosasendtx の LSys 情報が返却される。

注6)　DIOSA_SRC_AP は上記表の情報が返却される。DIOSA_SRC_LPATH、DIOSA_SRC_TMC が返却される場合、MsgType は DIOSA_MSG_TX となる。DIOSA_SRC_MSG_RETOV の場合、電文保証有の diosarecvtx の情報が返却される。DIOSA_SRC_OTC_RSP、DIOSA_SRC_OTC_ERR の場合、MsgType には DIOSA_MSG_LS が、また LSys 情報には AcsPoint、LsName、Protocol が返却される。(端末情報の TermName、TermType、ConnectTime は返却されない)

注7)　TermName が設定されている時のみ TermType を参照することが可能。

注8)　保留指定で diosasendtx(電文送信)した場合、基本的に保留要求をおこなったトランザクションが受信した電文情報と同じものが返却される。TxId、CoName、MsgSize は保留指定した diosasendtx の値で塗り替わる。MsgType に DIOSA_MSG_REST が返却されることはない。

注9)　＜diosarecvtx　電文保証あり＞

t_diosa_dcuca 内 **t_diosa_gntinfo** 注１	電文タイプ(MsgType)					
	論理システム間	論理システム内派生 (CoName 指定)	論理システム内派生 (TxId 指定)	論理システム内派生 (中継 Co 指定)	論理システム内派生 (中継 TxId 指定)	連鎖
MsgKey	－	□	□	□	□	－
SeqGroup	－	□	□	□	□	－
SeqNo	－	□	□	□	□	－
OrgCoName	－	□	□	□	□	－

●：必須
□：設定される場合がある

注 1)t_diosa_dcuca の MsgGnt=DIOSA_MSGGNT_YES、または DIOSA_MSGGNT_SEQ の時に有効となる。

2.1.42　diosarollback(ロールバック)

名前

diosarollback - トランザクションのロールバック

書式

```
#include <diosa.h>
int   diosarollback( int TimeReset );
```

説明

本 API は、CO 制御 TPP、バッチ AP 制御、アプリケーションプログラムで使用可能である。
本 API 実行前のメモリデータ管理アクセス、または DB 更新はキャンセルされ、CO 制御上では、diosasendtx によって VD 宛に遅延送信された電文は無効化される。
TimeReset は監視機能における経過時間と CPU 時間のリセットを行うフラグであり、DIOSA_YES が指定されていると、ロールバック時にリセットを行い、DIOSA_NO の場合はリセットを行わない。

戻り値

DIOSA_DONE(0)	正常に実行が終了した
DIOSA_EFUNCNAV(-34)	動作環境エラー
DIOSA_EPARAM(-3)	パラメータエラー
DIOSA_EACCES(-25)	メモリデータ管理アクセスエラー
DIOSA_EDB(-69)	ロールバックエラー(DB)
DIOSA_ETAM(-113)	IM の障害によりロールバックが失敗した。
DIOSA_EBEFORE(-111)	ロールバック前エラー
DIOSA_EAFTER(-112)	ロールバック後エラー
DIOSA_ETPBASE(-120)	TPBASE のエラー
DIOSA_ERROR(-1)	その他エラー

関連

なし

2.1.43　diosasendtx(電文送信)

名前

diosasendtx - トランザクション電文の送信

書式

```
#include <diosa.h>
int diosasendtx( t_diosa_dcuca *DcUca,  t_diosa_msgbuf *MsgBuf );
```

説明

本 API は、**MsgBuf** によって指定された電文を **DcUca** に設定された情報を元に送信する。
送信電文長は、**DcUca** の **MsgSize** に指定する。

DcUca について、電文タイプにより設定する必要のあるパラメータを欄外に示す。

戻り値

DIOSA_DONE(0)	正常に実行が終了した
DIOSA_ALMOST(10)	電文保証登録は成功したが送信はおこなわれていない。 電文送信は電文保証機能がおこなう。送信されなかった原因は、詳細ステータス(DetStatus)に設定される。
DIOSA_SWITCH(21)	計画マスタ切替中のため、処理継続不可
DIOSA_ERROR(-1)	その他 DIOSA エラー
DIOSA_EPARAM(-3)	パラメータエラー
DIOSA_EABORT(-4)	ハッシュ関数エラー
DIOSA_EMEM(-6)	メモリアクセスエラー
DIOSA_ENOENT(-8)	該当宛先なし
DIOSA_ENOINIT(-11)	電文保証機能未起動
DIOSA_ESEND(-15)	都度接続送信デーモンへの電文送信に失敗
DIOSA_EBLOCK(-21)	宛先の論理ノードが閉塞状態
DIOSA_ETIMEOUT(-22)	都度接続送信デーモンへの電文送信でタイムアウトが発生した
DIOSA_EEXIST(-27)	指定されたキーの保証電文が既に登録されている
DIOSA_ESIZE(-33)	電文長エラー
DIOSA_EFUNCNAV(-34)	動作環境エラー
DIOSA_EDEADLOCK(-36)	デッドロックが発生した
DIOSA_ENOMATCH(-40)	送信可能な端末が定義されていない
DIOSA_EDB(-69)	データベースアクセスエラー。詳細ステータスにエラー詳細が設定される。
DIOSA_ESOCKET(-70)	都度接続送信デーモンとのソケット生成に失敗した
DIOSA_ECONNECT(-71)	都度接続送信デーモンとのソケット接続に失敗した
DIOSA_ECONFLICT(-110)	t_diosa_dcuca パラメータエラー
DIOSA_ETAM(-113)	インメモリキャッシュへのアクセスエラー。詳細ステータスにエラー詳細が設定される。

DIOSA_ESTATE(-114)　更新先 DB がない。または RGSET がデフォルトインスタンスグループに属していない。
インメモリサーバアクセスしないメモリキャッシュの処理でエラーが発生した。
メモリキャッシュ処理でエラーの場合、詳細ステータスにエラー詳細が設定される。

DIOSA_ESWITCH(-115)　マスタ切替のためキュー登録に失敗した

DIOSA_EINACTIVE(-117)送信可能な端末が接続されていない。または送信可能な都度接続送信デーモンが存在しない。

DIOSA_EREADY(-118)　IMS サーバが起動中(再起動)や環境定義変更中により、アクセスするための準備ができていない。

DIOSA_ETPBASE(-120)　TP_open エラー

DIOSA_ETPSEND(-121)　TP_send エラー(詳細は t_diosa_dcuca の status_key,end_key を参照)

注意

- 論理システム間送信で電文保証ありを指定する場合、宛先端末名を指定してはならない。宛先端末名を指定した場合、保証電文宛先変更コマンド(digntchgdest)による宛先変更後、電文の再送が行われない。
- 電文保証ありを指定した場合、送信電文を DB に登録する。登録時の主キーには保証電文識別子と順序性保証グループ識別子を使用する。DB の更新順序によってはデッドロックが発生することがある。
- 更新先 DB が IM で、電文保証ありを指定する場合、本 API 呼び出し時点で更新先の MAPID が確定(受信電文解析出口で決定、または対象 MAPID に対して更新処理を実行)している必要がある。
- 都度接続プロトコルで着呼した端末に対して遅延送信を使用する場合、該当端末から電文を受信したトランザクションである必要がある。他の端末または VD から受信したトランザクションで都度接続プロトコルの着呼端末に電文送信する場合は、強制送信のみ使用可能である。

＜詳細ステータス(DIOSA_ALMOST)＞

詳細ステータス	説明
DIOSA_MSGQUE	順序性保証ありの保証電文送信が指定されたが、同一順序性保証グループ内に未完了電文があるため、送信を保留した。
DIOSA_ESEND	都度接続送信デーモンとの通信に失敗した。
DIOSA_ETPSEND	TPBASE の TP_Exsend に失敗した。
DIOSA_EBLOCK	宛先の論理ノードが閉塞状態。
DIOSA_EINACTIVE	送信可能端末が接続されていない。または送信可能な都度接続送信デーモンが存在しない。

＜詳細ステータス(DIOSA_ECONFLICT)＞

t_diosa_dcuca のパラメータエラーの原因を詳細ステータスへ返却する。

詳細ステータスの内訳は以下の通り

桁数	7	6	5	4	3	2	1
値	項目番号			原因番号			

項目番号

項目番号	項目名
1	t_diosa_dcuca-MsgType
2	t_diosa_dcuca-RcvType
3	t_diosa_dcuca-Source
4	t_diosa_dcuca-VdName
5	t_diosa_dcuca-TxId
6	t_diosa_dcuca-CoName
7	t_diosa_dcuca-SendMode
8	t_diosa_dcuca-IgnorePreBlk
9	t_diosa_dcuca-MsgSize

10	t_diosa_dcuca-NodeType
11	t_diosa_dcuca-Roundrobin
12	t_diosa_dcuca-SendCnt
13	t_diosa_dcuca-SimFlag
14	t_diosa_dcuca-MsgGnt
15	t_diosa_dcuca-APInfo
16	t_diosa_dcuca-DStatus
51	t_diosa_dcuca-LSys-LsName
52	t_diosa_dcuca-LSys-AcsPoint
53	t_diosa_dcuca-LSys-Protocol
54	t_diosa_dcuca-LSys-LNodeName
55	t_diosa_dcuca-LSys-TpMonitor
56	t_diosa_dcuca-LSys-TermType
57	t_diosa_dcuca-LSys-TermName
58	t_diosa_dcuca-LSys-ConnectTime
71	t_diosa_dcuca-LNode-LNodeName
72	t_diosa_dcuca-LNode-TpMonitor
81	t_diosa_dcuca-Db-Type
82	t_diosa_dcuca-Db-RgId
83	t_diosa_dcuca-Db-RgSet
84	t_diosa_dcuca-Db-MapId
85	t_diosa_dcuca-Db-Wait
86	t_diosa_dcuca-Db-MainKey
91	t_diosa_dcuca-gntinfo-MsgKey
92	t_diosa_dcuca-gntinfo-SendMax
93	t_diosa_dcuca-gntinfo-RetryInterval
94	t_diosa_dcuca-gntinfo-BkSyncType
95	t_diosa_dcuca-gntinfo-SendStay
96	t_diosa_dcuca-gntinfo-SeqGroup
97	t_diosa_dcuca-gntinfo-SeqNo
98	t_diosa_dcuca-gntinfo-OrgCoName
99	t_diosa_dcuca-gntinfo-BackUp
100	t_diosa_dcuca-gntinfo-BackUpStreamKey
111	t_diosa_dcuca-otcinfo-SettingGroup
112	t_diosa_dcuca-otcinfo-Response
113	t_diosa_dcuca-otcinfo-MsgTimeOut
121	t_diosa_dcuca-TpStatus-StatusKey
122	t_diosa_dcuca-TpStatus-EndKey

原因番号	原因
8	エントリやレコードが存在しない
9	値不正
10	範囲不正
11	未初期化
26	パス名長が最大値を超えている
40	一致しない
114	状態が不正
1000	整合性がない

関連

t_diosa_dcuca, diosarecvtx, t_diosa_msgbuf, diosamsgbufalloc

＜diosasendtx CO 制御 TPP＞

t_diosa_dcuca		電文タイプ(MsgType)						
		論理システム間	論理システム内派生（Co 指定） 注 5)	論理システム内派生（TxId 指定）	論理システム内派生（中継 Co 指定）	論理システム内派生（中継 TxId 指定）	連鎖 注 6)	保留
MsgType		●	●	●	●	●	●	●
LSys	AcsPoint	★注 1)	－	－	★注 1)	★注 1）	－	－
	TermName	★注 1)	－	－	★注 1)	★注 1）	－	－
	ConnectTime	★注 1)	－	－	★注 1)	★注 1）	－	－
LNode	LNodeName	－	★注 2)	★注 2)	★注 2)	★注 2)	－	－
	TpMonitor	－	★注 2)	★注 2)	★注 2)	★注 2)	－	－
Db	Type	－	★注 2)	★注 2)	★注 2)	★注 2)	－	－
	MainKey	－	★注 2)	★注 2)	★注 2)	★注 2)	－	－
	MapId	－	★注 2)	★注 2)	★注 2)	★注 2)	－	－
TxId		－	□	●	□	●	－	●
CoName		－	●	□	●	□	●	□
MsgSize		●	●	●	●	●	●	●
SendMode 注 7)		●	●	●	●	●	－	●
IgnorePreBlk		－	●	●	●	●	－	－
Roundrobin 注 4)		－	★注 3)	★注 3)	★注 3)	★注 3)	－	－
NodeType		－	●	●	●	●	－	－
MsgGnt		●	●	●	●	●	－	－
APInfo		●	●	●	●	●	－	－
DStatus		○	○	○	○	○	－	－
TpSt	StatusKey	○	○	○	○	○	○	○
	EndKey	○	○	○	○	○	○	○

●：必須
□：任意指定可
文字属性で先頭 NULL 文字は指定なし、数字属性で 0 は無効値とみなす
★：選択指定
○：返却される情報

注1)　AcsPoint と TermName はどちらか指定必須、両方が指定された場合は TermName が優先される。
TermName は、外部論理システムから電文を受信した端末宛に応答を返却したい場合に指定する。指定する TermName、および ConnectTime(都度接続端末の場合のみ必須)は、diosarecvtx()の返却パラメータをそのまま指定する。

注2)　IM のマスタノード宛に送信する場合は Db を指定し、ノードと TP モニタを指定して送信する場合は LNode を指定する。Db と LNode は同時に指定できず、どちらか一方を選択する。Db を指定しない場合は Db の Type に DIOSA_DB_NO を指定する。
DB として IM を使用する場合、Db の情報から IM のマスタノードを決定して、そのノードに存在する TPBASE モニタ宛てに電文送信を行うことができる。メインキーからマスタノードを決定する場合は、Type に DIOSA_DB_MAINKEY、MainKey にメインキーを指定する。MAPID からマスタノードを決定する場合は、Type に DIOSA_DB_MAPID、MapId に MAPID を指定する。
DB として DB アクセス制御(DAC)を使用する場合、Db の情報から IM のマスタノードを決定して、そのノードに存在する TPBASE モニタ宛てに電文送信を行うことができる。メインキーからマスタノードを決定する場合は、Type に DIOSA_DB_DACMAINKEY、MainKey にメインキーを指定する。MAPID からマスタノードを決定する場合は、Type に DIOSA_DB_DACMAPID、MapId に MAPID を指定する。
DB 固定モードの場合、Type に DIOSA_DB_MAINKEY、DIOSA_DB_MAPID、DIOSA_DB_DACMAINKEY、DIOSA_DB_DACMAPID が指定されている場合は Type=DIOSA_DB_NO とみなす。

注3)　送信先の論理ノード名と TPBASE モニタ名が一意に確定された場合、Ronudrobin 指定は無視される。

注4)　DIOSA_NO を指定した場合､優先度(自 TPBASE>自ノード別 TPBASE＞他ノード)順に送信先を選択する。宛先が複数となる場合は DIOSA_NO が指定されていてもラウンドロビン送信となる。

注5)　CO 指定で、指定された CO の宛先がない(環境定義の%COGROUP-%CO に該当する CO 名が定義されていない)場合、自 TPBASE の自トランザクション ID の別プロセスとして CO 呼び出しをおこなう。ただし、LNode 情報、TxId 等宛先を絞り込む情報が指定されている場合は宛先不正とみなしエラーとなる。また、電文保証をすることもできない。

注6)　連鎖指定された CO は、呼出元プロセスのトランザクションの延長で呼び出される。これに対して派生指定は自 TPBASE モニタや他 TPBASE モニタの別プロセスとして呼び出されるといった違いがある。
なお、連鎖 CO 名は%COGROUP-%CO に定義する必要はない。

注7)　アボート#1 出口からの電文送信は、DIOSA_SEND_FORCE のみ可能である。

＜diosasendtx　電文保証あり＞

t_diosa_dcuca 内 **t_diosa_gntinfo** 注１）	電文タイプ(MsgType)						
	論理システム間	論理システム内派生 (Co 指定)	論理システム内派生 (TxId 指定)	論理システム内派生 (中継 Co 指定)	論理システム内派生 (中継 TxId 指定)	連鎖	保留
MsgKey	●	－	－	●	●	－	－
SendMax	□	□	□	□	□	－	－
RetryInterval	□	□	□	□	□	－	－
SendStay	□	□	□	□	□	－	－
SeqGroup 注 3)	□	□	□	□	□	－	－
MainKey 注 4)	□	□	□	□	□	－	－
BkSyncType	●	●	●	●	●	－	－

●：必須
□：任意指定可
O：返却される情報

注1)　t_diosa_dcuca の MsgGnt=DIOSA_MSGGNT_YES、または DIOSA_MSGGNT_SEQ の時に有効となる。
注2)　AP ノード上での要求はエラー。
注3)　順序性保証を使用する(MsgGnt=DIOSA_MSGGNT_SEQ)場合に指定する
注4)　DB として DAC を使用する、受信電文解析出口で DbInfo.Type に DIOSA_DB_DAC を指定する場合に指定する。

＜diosasendtx　都度接続送信あり＞

t_diosa_dcuca 内 **t_diosa_otcinfo** 注１）	電文タイプ(MsgType)						
	論理システム間	論理システム内派生 (Co 指定)	論理システム内派生 (TxId 指定)	論理システム内派生 (中継 Co 指定)	論理システム内派生 (中継 TxId 指定)	連鎖	保留
SettingGroup	□	－	－	－	－	－	－
Response	●	－	－	－	－	－	－
MsgTimeOut	□	－	－	－	－	－	－

●：必須
□：任意指定可
O：返却される情報

注1)　送信先システムのアクセスポイントが都度接続送信デーモン経由の場合に有効となる。

＜diosasendtx　ユーザアプリケーションプログラム＞

t_diosa_dcuca		電文タイプ (MsgType) 論理システム内派生 (TxId 指定)	電文タイプ (MsgType) 論理システム内派生 (Co 指定)
MsgType		●	●
LNode	LnodeName 注 1)	－	－
	TpMonitor 注 1)	□	□注 2
TxId		●	□
CoName		□	● 注 2)
MsgSize		●	●
IgnorePreBlk		★	★
DStatus		○	○
TpSt	StatusKey	○	○
	EndKey	○	○

●：必須
□：任意指定可
　文字属性で先頭 NULL 文字
　は指定なし、数字属性で
　0 は無効値とみなす
★：選択指定
○：返却される情報

注1)　ユーザアプリケーションプログラムでは、同一論理ノード内のみ電文の送信が可能である
注2)　CO 所在管理にない共通 CO を指定する場合は、絞り込み条件として TpMonitor、TxId を指定してはいけない
注3)　ユーザアプリケーションプログラムでの電文送信は、強制送信のみ
注4)　メインスレッドからのみ使用可能

2.1.44　diosasetblockage（閉塞状態設定）

名前

diosasetblockage –閉塞状態を設定する。

書式

```
#include <diosa.h>
int diosasetblockage (t_diosa_blockageinfo *blockageinfo)
```

説明

論理ノードと CO の閉塞状態を設定する。
設定の為、デーモンとの通信を行う。

t_diosa_blockageinfo blockageinfo（入力型）

設定対象の情報を設定する。
構造体の以下のメンバを使用する。

int　Status（入力）

設定する閉塞状態

DIOSA_BLOCKAGE_BLK　　：閉塞中
DIOSA_BLOCKAGE_PBK　　：予閉塞中
DIOSA_BLOCKAGE_ACT　　：稼働中
DIOSA_BLOCKAGE_ACT_ALL ：全稼働中

int　Kind（入力）

設定する閉塞情報

DIOSA_BLOCKAGE_NOD　　：ノード閉塞状態設定
DIOSA_BLOCKAGE_COC　　：CO 閉塞状態設定

char Name[31]（入力）

論理ノード名（最大 15 文字）
CO 名（最大 30 文字）

Status に DIOSA_BLOCKAGE_ACT_ALL 以外が設定された場合、**Name** は必須。
Status に DIOSA_BLOCKAGE_ACT_ALL が設定された場合、**Name** は無視される。

Status に DIOSA_BLOCKAGE_ACT_ALL が設定された場合は **Kind** によって 2 パターンの動作をする。
1）**Kind** に DIOSA_BLOCKAGE_NOD が設定された場合は自論理システムの全論理ノードの状態を稼働中にする。
2）**Kind** に DIOSA_BLOCKAGE_COC が設定された場合は全 CO の状態を稼働中にする。

Kind に DIOSA_BLOCKAGE_NOD が設定された場合のみ、DIOSA_BLOCKAGE_PBK を指定できる。

Kind に DIOSA_BLOCKAGE_COC が設定され、**Name** に”all”が設定された場合は CO 制御で管理する情報の全体閉塞状態を変更する。
Kind に DIOSA_BLOCKAGE_NOD が設定され、**Name** に”all”が設定された場合は DIOSA_EPARAM となる。

戻り値

DIOSA_DONE(0)	正常	正常終了
DIOSA_ERROR(-1)	異常	異常終了
DIOSA_EPARAM(-3)	異常	パラメータエラー
DIOSA_ENOENT(-8)	異常	指定された情報が存在しなかった。

DIOSA_EBUSY (-31)	異常	デーモンが他要求による処理中で設定できない。
DIOSA_ESOCKET(-70)	異常	デーモンとの通信で障害が発生した。

注意

- 本関数を使用するためには diosaprcinit が呼び出されている必要がある。
- マルチスレッド環境での使用可。
- 論理ノード名、CO 名として”all”は使用できない。
- 全体閉塞状態は CO のみにある。
- 個別の閉塞状態よりも全体閉塞状態が優先される。
- 本 API 同士または本 API と変更コマンドが同時実行されると排他制御が働き、DIOSA_EBUSY となる。

関連

diosaprcinit, diosaprcterm

2.1.45　diosathrinit(スレッド初期化関数)

名前

diosathrinit - ユーザアプリケーションのためのスレッド初期処理を行う。

書式

```
#include <diosa.h>
int diosathrinit(void* Info)
```

説明

ユーザアプリケーション(CO 制御、バッチ AP 制御上のアプリケーション、ログリーダは除く)のためのスレッド初期処理を行う。

void* Info(入力) 将来の拡張の為の予約領域。NULL を指定する。

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_EFATAL(-30)	異常終了

注意

- 本関数を呼ぶ場合は事前に **diosaprcinit**(プロセス初期化処理)が呼ばれている必要がある。
- 本関数は **diosaprcinit**(プロセス初期化処理)を呼んだスレッド(メインスレッド)では呼ぶことはできない。
- 本関数を呼んだ場合、必ず **diosathrterm**(スレッド終了処理)を実行すること。(本関数が異常終了した場合も呼ぶこと)
- トランザクション区間内(**diosatrninit** と **diosatrnterm** の呼び出しの間)では、スレッドを起動してはいけない。

関連

diosaprcinit, diosaprcterm, diosathrterm , diosatrninit, diosatrnterm

2.1.46　diosathrterm(スレッド終了関数)

名前

diosathrterm - ユーザアプリケーションのためのスレッド終了処理を行う。

書式

```
#include <diosa.h>
int diosathrterm(void* Info)
```

説明

ユーザアプリケーション(CO 制御、バッチ AP 制御上のアプリケーション、ログリーダは除く)のためのスレッド終了処理を行う。

void* Info(入力) 将来の拡張の為の予約領域。NULL を指定する。

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_EFATAL(-30)	異常終了

注意

- **diosathrinit**(スレッド初期化処理)を呼んだ場合、必ず本関数を実行すること。
- 本関数は **diosaprcterm**(プロセス終了処理)を呼んだスレッド(メインスレッド)では呼ぶことはできない。
- **diosaprcterm**(プロセス終了処理)を呼ぶ前に本関数を実行すること。

関連

diosathrinit

2.1.47　diosatmcactv(タイマ保留解除)

名前

diosatmcactv - タイマ保留解除

書式

```
#include <diosa.h>
int diosatmcactv(t_diosa_tmcuca  *TmcUca)
```

説明

diosatmcactv()は、**TmcUca** に指定された情報を元に、実行が保留されているタイマ情報の保留解除を行う。

TmcUca	タイマ制御インタフェース構造体のポインタを指定する。 **t_diosa_tmcuca** 構造体については、**t_diosa_tmcuca** を参照すること。

戻り値

成功した場合には、0 が返される。エラー時は、負の値が返される。

DIOSA_DONE(0)	正常終了。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータが正しくない。
DIOSA_ENOENT(-8)	指定したタイマ情報が未登録である。
DIOSA_ELOCK(-14)	ロックエラーが発生した。
DIOSA_EUNLOCK(-29)	アンロックエラーが発生した。

注意

- 本 API 実行後、コミット処理によりトランザクションを確定した時点で保留状態が解除される。
- 本 API はシングルスレッド上での動作を保証する。

関連

t_diosa_tmcuca, diosatmchold

2.1.48　diosatmccmdquery(コマンドタイマ照会)

名前

diosatmccmdquery - コマンドタイマ照会

書式

```
#include <diosa.h>
int diosatmccmdquery(t_diosa_tmcuca *TmcUca, t_diosa_cmdqueryuca *CmdqueryUca, char *Command)
```

説明

diosatmccmdquery()は、**TmcUca**、**CmdqueryUca** に指定された情報を元に、コマンドタイマ登録情報の照会を行う。
照会結果は、**TmcUca**、**CmdqueryUca**、**Command** に返却する。

TmcUca	タイマ制御インタフェース構造体のポインタを指定する。 **t_diosa_tmcuca** 構造体については、**t_diosa_tmcuca** を参照すること。
CmdqueryUca	コマンドタイマ照会インタフェース構造体のポインタを指定する。 **t_diosa_cmdqueryuca** 構造体については、**t_diosa_cmdqueryuca** を参照すること。
Command	タイマ ID に対応したコマンド名の格納領域が返却される。 ユーザは領域として 200 バイト確保する必要がある。

戻り値

成功した場合には、0 が返される。エラー時は、負の値が返される。

DIOSA_DONE(0)	正常終了。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータが正しくない。
DIOSA_EMEM(-6)	メモリエラーが発生した。
DIOSA_ENOENT(-8)	指定したタイマ情報が未登録である。
DIOSA_ELOCK(-14)	ロックエラーが発生した。
DIOSA_EUNLOCK(-29)	アンロックエラーが発生した。
DIOSA_EOVERFLOW(-39)	エントリをオーバした。

注意

- コマンド名の返却領域の取得に失敗した場合、DIOSA_EMEM となる。
- タイマ情報を DIOSA_FIRST 指定で照会した場合、全件検索となるためタイマ ID を指定しなくても照会可能である。
- 本 API はシングルスレッド上での動作を保証する。

関連

t_diosa_tmcuca, t_diosa_tmctime, t_diosa_cmdqueryuca, diosatmccmdset

2.1.49　diosatmccmdset(コマンドタイマ登録)

名前

diosatmccmdset - コマンドタイマ登録

書式

```
#include <diosa.h>
int diosatmccmdset(t_diosa_tmcuca *TmcUca, t_diosa_cmdsetuca *CmdsetUca, char *Command)
```

説明

diosatmccmdset ()は、**TmcUca**、**CmdsetUca**、**Command** で指定された情報を元に、UNIX コマンドのタイマ登録をする。

TmcUca	タイマ制御インタフェース構造体のポインタを指定する。 **t_diosa_tmcuca** 構造体については、**t_diosa_tmcuca** を参照すること。
CmdsetUca	コマンドタイマ登録インタフェース構造体のポインタを指定する。 **t_diosa_cmdsetuca** 構造体については、**t_diosa_cmdsetuca** を参照すること。
Command	タイマに登録するコマンド名を英数字 200 文字以内で指定する。 コマンドのオプションをスペース区切りで設定することができるが改行コードなどの制御文字(ASCII 形式の 32 未満の文字コードおよび削除文字(127))は設定できない。

戻り値

成功した場合には、0 または正の値が返される。エラー時は、負の値が返される。

DIOSA_DONE(0)	正常終了。
DIOSA_ALREADY(1)	既に同じタイマ ID でタイマ情報が登録されている。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータが正しくない。
DIOSA_EMEM(-6)	メモリエラーが発生した。
DIOSA_ELOCK(-14)	ロックエラーが発生した。
DIOSA_EUNLOCK(-29)	アンロックエラーが発生した。
DIOSA_ESIZE(-33)	送信メッセージの長さが不正である。
DIOSA_EOVERFLOW(-39)	エントリをオーバした。
DIOSA_ETIMEFORM(-92)	指定された時刻(時間)が不正である。

注意

- 本 API 実行後、コミット処理によりトランザクションを確定した時点でタイマ情報が登録される。
- 実行対象コマンドの終了ステータスが、126,127 となる場合、コマンド実行エラーとして扱う。
- 本 API はシングルスレッド上での動作を保証する。

関連

t_diosa_tmcuca, t_diosa_tmctime, t_diosa_cmdsetuca, diosatmcreset, diosatmccmdquery

2.1.50　diosatmccoquery(CO タイマ照会)

名前

diosatmccoquery - CO タイマ照会

書式

```
#include <diosa.h>
int diosatmccoquery(t_diosa_tmcuca *TmcUca, t_diosa_coqueryuca *CoqueryUca, char *Data)
```

説明

diosatmccoquery()は、**TmcUca**、**CoqueryUca** に指定された情報を元に、CO タイマ登録情報の照会を行う。
照会結果は、**TmcUca**、**CoqueryUca**、**Data** に返却する。

TmcUca	タイマ制御インタフェース構造体のポインタを指定する。 **t_diosa_tmcuca** 構造体については、**t_diosa_tmcuca** を参照すること。
CoqueryUca	CO タイマ照会インタフェース構造体のポインタを指定する。 **t_diosa_coqueryuca** 構造体については、**t_diosa_coqueryuca** を参照すること。
Data	タイマ ID に対応した送信メッセージ領域が返却される。 ユーザは領域として 200 バイト確保する必要がある。

戻り値

成功した場合には、0 が返される。エラー時は、負の値が返される。

DIOSA_DONE(0)	正常終了。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータが正しくない。
DIOSA_EMEM(-6)	メモリエラーが発生した。
DIOSA_ENOENT(-8)	指定したタイマ情報が未登録である。
DIOSA_ELOCK(-14)	ロックエラーが発生した。
DIOSA_EUNLOCK(-29)	アンロックエラーが発生した。
DIOSA_EOVERFLOW(-39)	エントリをオーバした。

注意

- CO 制御通信情報の返却領域の取得に失敗した場合、DIOSA_EMEM となる。
- タイマ情報を DIOSA_FIRST 指定で照会した場合全件検索となるので、タイマ ID を指定しなくても照会可能である。
- 本 API はシングルスレッド上での動作を保証する。

関連

t_diosa_tmcuca, t_diosa_tmctime, t_diosa_coqueryuca, diosatmccoset

2.1.51　diosatmccoset(CO タイマ登録)

名前

diosatmccoset - CO タイマ登録

書式

```
#include <diosa.h>
int diosatmccoset(t_diosa_tmcuca *TmcUca, t_diosa_cosetuca *CosetUca, char *Data)
```

説明

diosatmccoset()は、**TmcUca**、**CosetUca**、**Data** で指定された情報を元に、CO タイマの登録を行う。

TmcUca	タイマ制御インタフェース構造体のポインタを指定する。 **t_diosa_tmcuca** 構造体については、**t_diosa_tmcuca** を参照すること。
CosetUca	CO タイマ登録インタフェース構造体のポインタを指定する。 **t_diosa_cosetuca** 構造体については、**t_diosa_cosetuca** を参照すること。
Data	タイマに登録する送信メッセージを 200 文字以内で指定する。 (バイナリデータ設定可)

戻り値

成功した場合には、0 または正の値が返される。エラー時は、負の値が返される。

DIOSA_DONE(0)	正常終了。
DIOSA_ALREADY(1)	既に同じタイマ ID でタイマ情報が登録されている。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータが正しくない。
DIOSA_EMEM(-6)	メモリエラーが発生した。
DIOSA_ELOCK(-14)	ロックエラーが発生した。
DIOSA_EUNLOCK(-29)	アンロックエラーが発生した。
DIOSA_ESIZE(-33)	送信メッセージの長さが不正である。
DIOSA_EOVERFLOW(-39)	エントリをオーバーした。
DIOSA_ETIMEFORM(-92)	指定された時刻(時間)が不正である。

注意

- 本 API 実行後、コミット処理によりトランザクションを確定した時点でタイマ情報が登録される。
- 本 API はシングルスレッド上での動作を保証する。

関連

t_diosa_tmcuca, t_diosa_tmctime, t_diosa_cosetuca, diosatmcreset, diosatmccoquery

2.1.52　diosatmchold(タイマ保留)

名前

diosatmchold - タイマ保留

書式

```
#include <diosa.h>
int diosatmchold(t_diosa_tmcuca  *TmcUca)
```

説明

diosatmchold()は、**TmcUca** に指定された情報を元に、登録タイマ情報の実行を保留する。

TmcUca	タイマ制御インタフェース構造体のポインタを指定する。 **t_diosa_tmcuca** 構造体については、**t_diosa_tmcuca** を参照すること。

戻り値

成功した場合には、0 が返される。エラー時は、負の値が返される。

DIOSA_DONE(0)	正常終了。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータが正しくない。
DIOSA_ENOENT(-8)	指定したタイマ情報が未登録である。
DIOSA_ELOCK(-14)	ロックエラーが発生した。
DIOSA_EUNLOCK(-29)	アンロックエラーが発生した。

注意

- 本 API 実行後、コミット処理によりトランザクションを確定した時点でタイマが保留状態となる。
- 本 API はシングルスレッド上での動作を保証する。

関連

t_diosa_tmcuca, diosatmcactv

2.1.53　diosatmcreset(タイマ削除)

名前

diosatmcreset - タイマ削除

書式

```
#include <diosa.h>
int diosatmcreset(t_diosa_tmcuca  *TmcUca)
```

説明

diosatmcreset()は、**TmcUca** で指定された情報を元に、対応するタイマ情報を削除する。

TmcUca	タイマ制御インタフェース構造体のポインタを指定する。 **t_diosa_tmcuca** 構造体については、**t_diosa_tmcuca** を参照すること。

戻り値

成功した場合には、0 が返される。エラー時は、負の値が返される。

DIOSA_DONE(0)	正常終了。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータが正しくない。
DIOSA_ENOENT(-8)	指定したタイマ情報が未登録である。
DIOSA_ELOCK(-14)	ロックエラーが発生した。
DIOSA_EUNLOCK(-29)	アンロックエラーが発生した。

注意

- 本 API 実行後、コミット処理によりトランザクションを確定した時点でタイマ情報が削除される。
- 本 API はシングルスレッド上での動作を保証する。

関連

t_diosa_tmcuca, diosatmccoset, diosatmccmdset

2.1.54　diosatrninit(トランザクション初期化関数)

名前

diosatrninit - ユーザアプリケーション上でトランザクション区間を開始する際に呼び出す。

書式

```
#include <diosa.h>
int diosatrninit(t_diosa_comdata* ComData)
```

説明

ユーザアプリケーション(CO 制御、バッチ AP 制御上のアプリケーション、ログリーダは除く)上でトランザクション区間を開始する際に呼び出す。
バッチアプリケーションに対し、DIOSA/XTP の変更コマンド等から環境定義置換や設定変更を通知するシグナルが通知されることがあるため、トランザクション区間など、シグナルにより中断されてはいけない箇所の前後で、それぞれ本関数 **diosatrninit**(トランザクション開始処理)と **diosatrnterm**(トランザクション終了処理)を呼ぶ必要がある。

t_diosa_comdata ComData (入力型)

トランザクション初期化の詳細設定を行う。
NULL を指定することで省略可。
省略した場合、
・OpsPerf には DIOSA_OFF が指定されたことになる。
・ElpsTime、CpuTime、MaxReset には 0 が指定されたことになる。
・AbortFlag には DIOSA_OFF が指定されたことになる。

構造体の以下のメンバを使用する。

int　OpsPerf

稼動統計情報の採取有無を指定する。(入力)

DIOSA_ON　: 稼動統計情報を採取する。
DIOSA_OFF　: 稼動統計情報を採取しない。

char　SvcName[31+1]

トランザクション区間を識別するためのサービス名を指定する。(入力)
(最大 31 バイト、省略可)
本項目は稼動統計情報を採取する場合、または経過時間監視機能を有効とした場合に利用される。

t_diosa_etginfo EtgInfo

トランザクション区間における経過時間監視情報を指定する。
以下の情報は全て、経過時間監視機能を有効とした場合のみ利用される。

int　　ElpsTime

経過時間制限値を秒単位で指定する。(入力)
0 を指定した場合、無制限とする。

int　　CpuTime

CPU 時間制限値を秒単位で指定する。(入力)
0 を指定した場合、無制限とする。

int　　MaxReset

経過時間のリセットを許可する回数を指定する。(入力)
0 を指定した場合、無制限とする。

int　　AbortFlag

経過時間が制限値を超過した際にプロセスを停止するかどうかを指定する。(入力)

DIOSA_ON　: プロセスを停止する。
DIOSA_OFF　: 警告メッセージのみ表示しプロセスの停止は行わない。

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_EPARAM(-3)	t_diosa_comdata 設定エラー
DIOSA_EFATAL(-30)	異常終了

注意

- バッチアプリケーションでは **diosaprcinit**(プロセス初期化処理)の後に実行すること。
- 本関数が異常終了した場合でも、次のトランザクション区間を開始する場合は **diosatrnterm**(トランザクション終了処理)を呼ぶ必要がある。

関連

diosatrnterm

2.1.55　diosatrnterm(トランザクション終了関数)

名前

diosatrnterm - ユーザアプリケーション上でトランザクション区間を終了する際に呼び出す。

書式

```
#include <diosa.h>
int diosatrnterm(void* Info)
```

説明

ユーザアプリケーション(CO 制御、バッチ AP 制御上のアプリケーション、ログリーダは除く)上でトランザクション区間を終了する際に呼び出す。
バッチアプリケーションに対し、DIOSA/XTP の変更コマンド等から環境定義置換や設定変更を通知するシグナルが通知されることがあるため、トランザクション区間など、シグナルにより中断されてはいけない箇所の前後で、それぞれ、**diosatrninit**(トランザクション開始処理)と本関数 **diosatrnterm**(トランザクション終了処理)を呼ぶ必要がある。
void* Info(入力) 将来の拡張の為の予約領域。NULL を指定する。
diosatrninit 未実施時、または **diosatxstart** 実施後、**diosacommit** もしくは **diosarollback** を実施しないで **diosatrnterm** を呼び出した場合は異常終了 (DIOSA_ESTATE) となる。

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_ESTATE(-114)	状態不正 (異常終了)
DIOSA_EFATAL(-30)	異常終了

注意

- バッチアプリケーションでは **diosaprcterm**(プロセス終了処理)の直前に実行すること。

関連

diosatrninit

2.1.56　diosatxstart(DB 更新開始)

名前

diosatxstart – DB 更新開始

書式

```
#include <diosa.h>
int diosatxstart( t_diosa_dbinfo *DBInfo );
```

説明

ユーザアプリケーションプログラム(CO 制御、バッチ AP 制御、ログリーダは除く)で DB 更新対象を決定し、DB 更新を開始する。
本 API を呼び出した場合は、DB 更新後に diosacommit()または diosarollback()を呼び出す必要がある。
DBInfo で設定したアクセス先は、diosacommit(),diosarollback()の処理対象として扱われる。
メモリキャッシュが更新対象で本 API を呼び出した場合、diosaimtxstart()は呼び出し不要である。

t_diosa_dbinfo DBInfo (入力型)

構造体の以下のメンバを使用する。

short　Type

DB タイプ(入力)

DIOSA_DB_NO	:DB 更新しない
DIOSA_DB_IM	:メモリキャッシュを対象とする場合に指定する
DIOSA_DB_RGSET	:Oracle もしくは PostgreSQL を対象とする場合に指定する
DIOSA_DB_DAC	:対象 DB を DB アクセス API で決定する場合に指定する

char　RgSet[31+1]

リソースグループセット(31 バイト) (入力)

Type に DIOSA_DB_RGSET、RgSet に NULL を指定した場合は、デフォルト RGSET を指定したものとして扱う。

int　Wait

クラスタ再構成待ち合わせ時間 (秒) (入力)

Oracle を対象(Type に DIOSA_DB_RGSET を指定した**場合**)の場合、Wait にデータベース障害の際のクラスタ再構成待ち時間を指定する。

-1	クラスタ再構成が完了するまで待ち合わせる。
0	待ち合わせを行わず、即時リターンする。
正の数値	指定された秒数待ち合わせる。

戻り値

DIOSA_DONE(0)	正常終了した。
DIOSA_ERROR(-1)	その他エラーが発生した。
DIOSA_ESTATE(-114)	**diosatrninit()** が未実施である。

注意

- DB タイプに DIOSA_DB_DAC を指定した場合、メモリキャッシュ API による更新はコミットされない。

関連

diosacommit, diosarollback, diosaimtxstart, diosaimcommit, diosaimrollback

2.1.57　diosaucaget(DIOSAUCA アドレス取得)

名前

diosaucaget - diosauca の取得

書式

```
#include <diosa.h>
int    diosaucaget( t_diosa_uca    **DiosaUca );
```

説明

利用者出口・CO において、現在の diosauca 領域のアドレスを取得する。
diosauca 領域のポインタは、DiosaUca が指す領域に格納される。

利用者出口・CO から呼び出す利用者のサブルーチンにおいて、引数などにより diosauca 領域を受け渡す代わりに本 API によって取得することができる。

戻り値

DIOSA_DONE(0)	正常に実行が終了した
DIOSA_EPARAM(-3)	diosauca ポインタが NULL
DIOSA_EFUNCNAV(-34)	本マクロを使用できないプロセス上の AP から要求された

関連

t_diosa_uca, diosagoback

2.1.58　diosaunlock(ロック解放)

名前

diosaunlock - diosalock で獲得したロックを解放する

書式

```
#include <diosa.h>
int diosaunlock(int Id, int Mode);
```

説明

diosaunlock() は diosalock で獲得したロックを解放する。

int Id(入力型)

識別子を指定する。
ロック範囲が論理システム内の場合、1～4096 の範囲内で指定する。
ロック範囲が論理ノード内の場合、1～1024 の範囲内で指定する。

int Mode(入力型)

ロック範囲を指定する。

DIOSA_INSYSTEM	論理システム内
DIOSA_INNODE	論理ノード内(既定値)

戻り値

DIOSA_DONE(0)	正常終了した。
DIOSA_EPARAM(-3)	パラメータに誤りがある。
DIOSA_EPERM(-12)	指定された識別子に対するロックの解除権がない。
DIOSA_ELOCK(-14)	ロック制御に失敗した。
DIOSA_EUNLOCK(-29)	指定されたロック範囲の識別子はロックされていない。
DIOSA_EFUNCNAV(-34)	データベース機能は無効化されている。
DIOSA_EDB(-69)	論理システム内ロック制御に失敗した。
DIOSA_ECLOSE(-75)	データベースとの接続が切断された。

注意

- 論理システム内ロック制御を行う場合は、「DB 監視機能」によってデータベースと接続されていることが前提条件である。
- 他のスレッドによって生じたロックは解放することができない。
- シグナルハンドラからの呼び出しは不可である。
- ロック範囲が論理システム内の場合 4065～4096 は予約ロック識別子で利用してはならない。また、ロック範囲が論理ノード内の場合 641～1024 は予約ロック識別子で利用してはならない。
- PostgreSQL の DB 型ロックで戻り値が DIOSA_EDB の場合、ロック制御失敗後に DB へアクセスするとエラーとなるため、トランザクションをロールバックする必要がある。

関連

diosalock

2.1.59　diosavcall(AP 動的置換対象関数呼び出し)

名前

diosavcall - AP 動的置換対象ライブラリの関数を呼び出す。

書式

```
#include <diosa.h>
int diosavcall(char *funcname, long *status, int num, char **args)
```

説明

funcname に指定した名前の関数を呼び出す。関数名は 30 バイト以内で指定すること。
関数呼び出しのパラメータは、**num** でパラメータ数(0～20)、**args** でパラメータの配列を格納した領域のアドレスを指定する。パラメータ数が 0 の場合、**args** には NULL を指定してもよい。
呼び出した関数の戻り値は **status** に返却する。戻り値がない場合や返却不要な場合、**status** には NULL を指定してもよい。

戻り値

DIOSA_DONE(0)	正常正常終了
DIOSA_EPARAM(-3)	異常パラメータエラー
DIOSA_ENOENT(-8)	異常指定された関数が見つからなかった
DIOSA_ENOINIT(-11)	異常プロセス初期化/スレッド初期化がおこなわれていない
DIOSA_EMEM(-6)	異常メモリ確保エラー
DIOSA_EMLOCK(-65)	異常mutex ロックエラー
DIOSA_EMUNLOCK(-66)	異常mutex ロック解除エラー

関連

diosaprcinit(), diosaprcterm(), diosatrninit(), diosatrnterm()

2.1.60　t_diosa_analyze(受信電文解析出口構造体)

名前

t_diosa_analyze - 受信電文解析出口構造体

書式

```
#include <diosa.h>
t_diosa_analyze   diosaanalyze 領域名;
```

説明

```
char  *RecvMsg;           受信電文のポインタ
int    RecvSize;          受信電文長
int    EditFlag;          編集フラグ： DIOSA_ON または DIOSA_OFF
t_diosa_msgbuf  *EditBuf; 編集電文バッファ
int    EditSize;          編集電文サイズ
short  MsgType;           受信電文タイプ
short  Source;            電文送信元識別子
char   CoName[30+1];      CO 名（最大 30 文字）
t_diosa_lsysaddr  LSys;   論理システム間アドレス情報
t_diosa_dbinfo    DbInfo; DB 情報
```

受信電文解析出口に与えられる diosaanalyze 領域の内容を以下に示す。

項目名	入力	出力	説明
RecvMsg	○	－	受信電文が設定される。
RecvSize	○	－	受信電文長が設定される。
EditFlag	○	○	出口呼び出し時は常に DIOSA_OFF が設定される。 EditFlag を DIOSA_ON に変更して出口を終了することにより、EditBuf に指定された編集電文バッファの電文が有効となる。
EditBuf	○	○	利用者が確保した電文バッファを指定して出口を終了することで、以降の出口呼び出し時に、その電文バッファが引き継がれる（EditFlag を DIOSA_ON に変更しない場合も、本設定値の変更は有効となる）。 プロセス起動直後の出口呼び出し時は NULL が設定される。
EditSize	－	○	EditFlag を DIOSA_ON に変更した場合、編集電文の電文長を指定しなければならない。
MsgType	○	－	受信電文タイプ(t_duisa_dcuca の MsgType と同じ) DIOSA_MSG_LS　：論理システム間電文 DIOSA_MSG_CO　：論理システム内派生(CoName 指定) DIOSA_MSG_TX　：論理システム内派生(TxId 指定) DIOSA_MSG_RELAYCO　：論理システム内派生(中継 Co 指定) DIOSA_MSG_RELAYTX　：論理システム内派生(中継 TxId 指定)
Source	□	－	電文の送信元識別子 DIOSA_SRC_AP　：業務 DIOSA_SRC_TMC　：タイマ DIOSA_SRC_LPATH　：端末接続/切断 DIOSA_SRC_GNT_RETOV　：電文保証(リトライオーバ) DIOSA_SRC_OTC_RSP　：都度接続応答電文 DIOSA_SRC_OTC_ERR　：都度接続エラー電文

CoName	□	○	受信電文が、実行する CO が指定された電文であった場合、その CO 名が設定される。そうでない場合は NULL が設定される。 利用者は、任意の CO 名に書き換えて出口を終了することにより、実行する CO を変更することができる。
t_diosa_lsysaddr(MsgType が DIOSA_MSG_LS の時に有効)			
LsName	□	－	外部論理システム名
AcsPoint	□	－	外部アクセスポイント名
Protocol	□	－	プロトコル名
TermName	□	－	入口(受信)端末名
ConnectTime	□	－	入口(受信)端末接続時間
TermType	□	－	入口(受信)端末タイプ DIOSA_PATH_FULLTIME　：常時接続端末 DIOSA_PATH_ONETIME　：都度接続端末
t_diosa_dbinfo			
Type	○	○	更新対象とする DB を選択する。入力値と変更可能な値は注意を参照する。 環境定義で指定した更新対象 DB 情報は t_diosa_uca に格納されている。 DIOSA_DB_NO　：DB 更新は行わない DIOSA_DB_RGSET　：RgSet に指定された RGSET に対応した DB インスタンスで更新を行う DIOSA_DB_MAINKEY：メインキー指定で IM 更新を行う DIOSA_DB_MAPID　：MAPID 指定で IM 更新を行う DIOSA_DB_IM　：IM の更新を行う DIOSA_DB_RGID　：リソースグループ ID に対応する DB インスタンスで更新を行う DIOSA_DB_DAC　：DB アクセス制御により更新先 DB を決定する DIOSA_DB_DACMAINKEY　:MAINKEY 指定で更新を行う。更新先 DB は DB アクセス制御により決定する DIOSA_DB_DACMAPID　:MAPID 指定で更新を行う。更新先 DB は DB アクセス制御により決定する
RgSet	□	○	出口呼び出し時、Type に DIOSA_DB_RGSET が設定されている場合、NULL(DBCTRL 節の DEFAULTRGSET と同意)が格納される DEFAULTRGSET を変更したい場合は、変更する RGSET 名を格納する。
MainKey	□	○	出口呼び出し時、Type に DIOSA_DB_MAINKEY が設定されている場合、メインキーが格納される。 メインキーを変更したい場合は、変更するメインキーを格納する。
MapId	□	○	出口呼び出し時、Type で DIOSA_DB_MAPID が設定されている場合、MAPID が格納される。 MAPID を変更したい場合は、変更する MAPID を格納する。
RgId	－	○	出口呼び出し時、Type に DIOSA_DB_RGSET、RgSet に NULL が設定されている。 リソースグループ ID から DB インスタンスを決定する場合、リソースグループ ID を格納する。

“○”：入力(設定されている)　出力(設定可能)

“□”：設定される可能性がある項目である

“－”：無効項目

説明

○受信電文解析出口の定義が必要となる条件

受信電文解析出口の定義が必要となる条件を以下に記述する。

・利用者が電文圧縮を行っており、トランザクション開始前に解凍が必要となる場合。

・出口で CO 名を決定したい場合。

・更新対象 DB として DIOSA_DB_ALL が指定され、かつ電文受信に応じて更新 DB を切り替えたい場合。
更新対象 DB は、環境定義で TPBASE のクラス ID 毎に定義できるが、電文毎に更新 DB を切り替える必要がある。

・DB を更新対象とするとき、電文毎に更新 RgSet(インスタンス)を変更したい、またはリソースグループ ID(RgId)に対応するインスタンスに変更したい場合。
RgSet 未指定(NULL 文字)は default の RgSet を採用する。(環境定義$DBCTRL を参照)

・IM を更新対象とするとき、電文毎に MainKey、または MapId を指定、変更したい場合。
受信電文に MainKey、MapId 情報が格納されている場合は、その MainKey、MapId を格納して出口が呼び出される。利用者は、この MainKey、MapId を変更することができる。

○受信電文と環境定義から更新 DB を決定する

CO 制御は、受信電文と環境定義から更新対象となる DB を決定する。利用者は受信電文解析出口を定義することにより、この更新 DB を変更することができる。

受信電文にメインキーや MAPID 情報が格納されている場合は、DIOSA_DB_MAINKEY、DIOSA_DB_MAPID が更新 DB として採用される。受信電文にメインキーや MAPID の情報が無い場合は、環境定義から更新 DB を決定する。環境定義が DIOSA_DB_RGSET と DIOSA_DB_ALL の場合は DIOSA_DB_RGSET(RgSet 名は NULL の DEFAULT-RGSET)が採用される。本更新 DB 情報は t_diosa_dbinfo に設定され、受信電文解析出口に渡され、利用者はこの情報を変更することができる。

出口が定義されない場合は CO 制御が採用した DB が更新 DB とみなされる。

t_diosa_uca の DbType(環境定義値)	CO 制御が採用する更新 DB 受信電文解析出口には t_diosa_dbinfo の Type で通知する				
	DIOSA_DB_NO	DIOSA_DB_RGSET	DIOSA_DB_MAINKEY	DIOSA_DB_MAPID	DIOSA_DB_IM
DIOSA_DB_NO	●		○	○	
DIOSA_DB_RGSET		●	○	○	
DIOSA_DB_IM			●	●	●
DIOSA_DB_ALL		●	●	●	

● ：採用される Type 情報(更新 DB タイプとして有効情報)

○ ：採用される Type 情報(環境定義と不整合)、受信電文解析出口で更新 DB を変更しない、または出口が未定義で更新 DB を変更できない場合、エラーCO が呼び出される。

○受信電文解析出口定義時の変更可能な更新対象 DB 情報(t_diosa_dbinfo)

更新対象 DB の環境定義値、受信電文解析出口呼び出し時の初期値、および受信電文解析出口の出力項目を以下の表で示す。

t_diosa_uca の DbType	t_diosa_dbinfo の Type への設定可否					t_diosa_dbinfo のメンバへの設定可否			
	NO	RGSET	MAIN KEY	MAP ID	RGID	RgSet	Main Key	Map Id	RgId
DIOSA_DB_NO	－	－	－	－	－	－	－	－	－
DIOSA_DB_RGSET	△	○	－	－	○	○	－	－	○
DIOSA_DB_IM	△	－	○	○	－	－	○	○	－
DIOSA_DB_ALL	△	○	○	○	○	○	○	○	○

○：設定、または変更可
△：DB を使わない指定
－：設定不可

○更新 DB のチェック処理

受信電文解析出口が定義してある場合、出口呼出後の t_diosa_dbinfo を出口未定義の場合 DB の既定値と環境定義の更新 DB との整合性チェックを行う。エラー検出時は、エラーメッセージの出力とエラーCO の呼び出しを行う。エラーCO の呼出条件は以下が設定される。

・更新 DB タイプエラー(DbInfo.Type)
・設定値エラー(RgId、または MapId が指定値範囲外)

○受信電文解析出口が未定義の時の動作

受信電文解析出口の定義がされていない場合の動作を以下に記述する。

・CO は送信元で指定された CO か、環境定義のトランザクション毎の既定 CO を呼び出す。(送信元で指定された CO が優先される)
・更新対象 DB は上の表「受信電文と環境定義から更新 DB を決定する」を参照する。

関連

t_diosa_msgbuf, diosamsgbufalloc, diosamsgbuffree

2.1.61　t_diosa_apptrcinfo（トレース情報ファイル構造体）

名前

t_diosa_apptrcinfo -トレース情報ファイル構造体

書式

#include <diosa.h>
t_diosa_apptrcinfo トレース情報ファイル構造体名;

説明

t_diosa_apptrcinfo 構造体は、トレース情報ファイルのレコードに関する情報が格納される。

t_diosa_appcomheader AppComHeader;共通ヘッダ(出力)

unsigned char	Rcdtype;	レコードタイプ ‘H’：ヘッダレコード ‘F’：トレース情報(直接出力) ‘M’：トレース情報(メモリ経由出力)
unsigned char	Year;	年(西暦年の下 2 桁)
unsigned char	Month;	月(2 桁)
unsigned char	Day;	日(2 桁)
unsigned char	Hour;	時(2 桁)
unsigned char	Minute;	分(2 桁)
unsigned char	Second;	秒(2 桁)
char	Procname[32];	プロセス名(トレース情報を出力したプロセスのプロセス名、NULL 終端)
unsigned int	Millisec;	ミリ秒(3 桁)
pid_t	Procid;	プロセス ID(トレース情報を出力したプロセスのプロセス ID)
int	Thrid;	スレッド ID(トレース情報を出力したプロセスのスレッド ID)

t_diosa_apptrcheader AppTrcHeder;トレース情報ヘッダ(出力)

char	Sectname[32];	関数名(NULL 終端)
char	Srcname[32];	ソースファイル名(NULL 終端)
int	Line;	行番号
int	Errcode;	エラーコード
unsigned char	Level;	トレース情報出力レベル　1～9 の整数
char	Comment[50];	コメント(NULL 終端)

int　BuffSize;　バッファサイズ(入力)
int　ReadSize;　読み込みサイズ(出力)
char* PtrBuff;　バッファのポインタ(入力)

※上記構造体とトレース情報ファイルオープン/読み込み/クローズ関連を下表に示す。

API	共通ヘッダ			トレース情報ヘッダ
	レコードタイプ＝H	レコードタイプ＝F	レコードタイプ＝M	
diosaapptrcopen	○	－	－	－
diosaapptrcread	－（注）	○	○	○
diosaapptrcclose	－	－	－	－

返却あり：○ 返却なし：－

（注）スワップした場合にトレース情報ファイルの途中にヘッダレコードが格納される場合があるが diosaapptrcread では返却しない。

関連

diosaapptrcopen, diosaapptrcread, diosaapptrcclose

2.1.62　t_diosa_blockageinfo(閉塞情報照会用構造体)

名前

t_diosa_blockageinfo – 閉塞情報照会用構造体

書式

```
#include <diosa.h>
t_diosa_blockageinfo Blockageinfo;
```

説明

t_diosa_blockageinfo 構造体は、閉塞情報照会処理のインタフェースに関連する情報が格納される。

int Status	閉塞状態を設定する。	
	DIOSA_BLOCKAGE_BLK	：閉塞中
	DIOSA_BLOCKAGE_PBK	：予閉塞中
	DIOSA_BLOCKAGE_ACT	：稼働中
	DIOSA_BLOCKAGE_ACT_ALL	：全稼働中
int Kind	閉塞状態を設定する対象を設定する。	
	DIOSA_BLOCKAGE_NOD	：ノード閉塞情報
	DIOSA_BLOCKAGE_COC	：CO 閉塞情報
char Name[31]	閉塞状態を設定する論理ノード名もしくは CO 名を設定する。	

関連

diosasetblockage

2.1.63　t_diosa_cmdconfuca(コマンド配信結果情報構造体)

名前

t_diosa_cmdconfuca - コマンド配信結果情報構造体

書式

```
#include <diosa.h>
t_diosa_cmdconfuca ConfUca;
```

説明

t_diosa_cmdconfuca 構造体は、コマンド配信結果に関する情報が格納される。

int NodeCnt;	実行結果返却ノード数
int RemNodeCnt;	未返却ノード数
t_diosa_cmdnodeconf NodeConf[NodeCnt];	ノード単位コマンド実行結果返却領域

関連

diosacmdsend,diosacmdconf,t_diosa_cmdnodeconf

2.1.64　t_diosa_cmdnodeconf(配信先ノード単位結果情報構造体)

名前

t_diosa_cmdnodeconf - 配信先ノード単位結果情報構造体

書式

```
#include <diosa.h>
t_diosa_cmdnodeconf NodeConf;
```

説明

t_diosa_cmdnodeconf 構造体は、配信先ノード単位のコマンド配信結果情報が格納される。

char LsName[16];	コマンド配信先の論理システム名。(最大 15 文字)
char LnodeName[16];	コマンド配信先の論理ノード名。(最大 15 文字)
int SendStatus;	コマンド配信結果ステータス。
	DIOSA_DONE(0)　コマンド配信成功
	DIOSA_ESEND(-15)　配信先ノードに接続、送信できない
	DIOSA_EBLOCK(-21)　配信先ノードが閉塞中である
	DIOSA_EFATAL(-30)　コマンド実行に失敗した
	DIOSA_ETIMEOUT(-22)　一定時間以内に返信がない
	DIOSA_ESETDATA(-18)　環境定義(DIOSAMAP)の設定に誤りがある
	DIOSA_EFUNCNAV(-34)　その他のエラー
int ExecStatus;	コマンド実行結果ステータス。(配信先で実行したコマンドの戻り値)
char StdoutFilePath[256];	stdout 用ファイルの絶対パス名。(最大 255 文字) データがない場合 NULL が設定される。
char StderrFilePath[256];	stderr 用ファイルの絶対パス名。(最大 255 文字) データがない場合 NULL が設定される。

注意

- コマンド実行結果ファイル名は下記の形式で作成する。
  stdout：
  diosa_cdd_result_要求元ﾉｰﾄﾞ_要求ﾌﾟﾛｾｽ ID_要求ｽﾚｯﾄﾞ ID_日時_配信先ﾉｰﾄﾞ名.std
  stderr：
  diosa_cdd_result_要求元ﾉｰﾄﾞ_要求ﾌﾟﾛｾｽ ID_要求ｽﾚｯﾄﾞ ID_日時_配信先ﾉｰﾄﾞ名.err
  (※日時は YYYYMMDDHHMMSSssssss)
- ファイル格納先のディレクトリは環境変数(DIOSA_CDDTMPDIR)、または環境定義(CMDSENDINFO 項 TMPDIR)で設定される。
- 自プロセスが取得したコマンド実行結果ファイルは、プロセス終了時に一括削除する。継続して利用する場合、必要なファイルは退避すること。

関連

t_diosa_cmdconfuca

2.1.65　t_diosa_cmdqueryuca(コマンドタイマ照会用構造体)

名前

t_diosa_cmdqueryuca - コマンドタイマ照会用構造体

書式

```
#include <diosa.h>
t_diosa_cmdqueryuca CmdqueryUca;
```

説明

t_diosa_cmdqueryuca 構造体は、コマンドタイマ照会処理のインタフェースに関連する情報が格納される。

char Hold[5]	タイマエントリ状態が返却される。	
	HOLD	保留状態
	ACTV	活性状態
short Seq	照会したいタイマ要求エントリを指定する。	
	DIOSA_FIRST	先頭タイマエントリを照会する。
	DIOSA_NEXT	前回照会した次のタイマエントリをタイマ ID 順に照会する。
	DIOSA_DIRECT	指定したタイマ ID のタイマエントリを照会する。
short Textalen	送信メッセージ長が返却される。	

関連

diosatmccmdquery

2.1.66　t_diosa_cmdresultinfo(配信結果確認情報構造体)

名前

t_diosa_cmdresultinfo - 配信結果確認情報構造体

書式

```
#include <diosa.h>
t_diosa_cmdresultinfo ResultInfo;
```

説明

t_diosa_cmdresultinfo 構造体は、コマンド配信結果の確認方法に関する情報が格納される。

char ResultMode;　　配信結果確認方法を指定する。

DIOSA_CMDSND_RESULT_NO	実行結果の確認は行わない
DIOSA_CMDSND_RESULT_CONF	実行結果を後で確認する
DIOSA_CMDSND_RESULT_WAIT	実行結果を待ち合わせる

char ResultFileFlg;　　コマンド実行結果ファイル(stdout/stderr)の返却有無を指定する。

DIOSA_ON	実行結果ファイルを返却する
DIOSA_OFF	実行結果ファイルを返却しない

配信結果確認方法が DIOSA_CMDSND_RESULT_NO の場 合、実行結果ファイルは返却されない。

int ResultCmdLen;　　未使用

char *ResultCmd;　　未使用

関連

diosacmdsend, t_diosa_cmdsenduca

2.1.67　t_diosa_cmdsendinfo(コマンド配信先情報構造体)

名前

t_diosa_cmdsendinfo - コマンド配信先情報構造体

書式

```
#include <diosa.h>
t_diosa_cmdsendinfo SendInfo;
```

説明

t_diosa_cmdsendinfo 構造体は、コマンド配信先に関する情報が格納される。

char DstName[16];　配信宛先名を指定する。(最大 15 文字)

論理ノード名、論理システム名、サーバグループ名での指定が可能である。コマンドルーティングを利用する場合、省略値として NULL を指定する。

char LsType;　配信宛先名が論理システム指定の場合、コマンド配信対象として論理システム配下のノード属性を指定することができる。

DIOSA_CMDSND_DST_LSALL	指定論理システム配下の全ノード宛
DIOSA_CMDSND_DST_LSAP	指定論理システム配下の AP ノード宛
DIOSA_CMDSND_DST_LSOLTP	指定論理システム配下の OLTP ノード宛
DIOSA_CMDSND_DST_LSDB	指定論理システム配下の DB ノード宛

コマンドルーティングを利用する場合、または配信宛先名に論理ノード名、サーバグループ名を指定した場合 DIOSA_CMDSND_DEFAULT(-1)を指定する。

char Target;　配信宛先名に論理システム名またはサーバグループ名を指定した場合、配信対象ノードを指定する。

DIOSA_CMDSND_TARGET_ALL	指定した配信先のうち全ノードが対象
DIOSA_CMDSND_TARGET_ANY	指定した配信先のうち任意の 1 ノードが対象

コマンドルーティングを利用する場合、または配信宛先名に論理ノード名を指定した場合 DIOSA_CMDSND_DEFAULT(-1)を指定する。

char SelfFlg;　複数ノードへの配信で配信元が配信対象に含まれる場合、配信元ノードにコマンドを配信する否かを指定する。

DIOSA_ON	配信元にもコマンドを配信する
DIOSA_OFF	配信元にはコマンドを配信しない

配信宛先名に論理ノード名を指定した場合、あるいは配信元が配信対象に含まれない場合、この値は無視される。

コマンドルーティングを利用する場合、DIOSA_CMDSND_DEFAULT(-1)を指定する。

char ForceFlg;　配信先ノードが閉塞中の場合、強制的にコマンド配信するか否かを指定する。

また、閉塞中のノードへの配信を抑止する場合、配信先結果として閉塞中のノードを含めるかどうかを指定する。

DIOSA_ON	強制的に配信する
DIOSA_OFF | DIOSA_CMDSND_BLKEXCL_OFF	閉塞中のノードへは配信しない/ 閉塞中のノードを配信結果に含める

	DIOSA_OFF | DIOSA_CMDSND_BLKEXCL_ON	閉塞中のノードへは配信しない/ 閉塞中のノードを配信結果に含めない

(DIOSA_OFF のみ指定した場合、DIOSA_OFF \| DIOSA_CMDSND_BLKEXCL_OFF 指定と同意となる)

int APITimeOut; コマンド配信結果が返却されるまでの応答待ち合わせ時間を指定する。(単位：秒、範囲：-1,1～86400)

DIOSA_CMDSND_DEFAULT(-1)を指定した場合、環境変数、または環境定義の値となる。配信結果確認方法に DIOSA_CMDSND_RESULT_WAIT を指定した場合のみ有効な値となる。

int ExecTimeOut; コマンドの実行応答待ち合わせ時間を指定する。(単位:秒、範囲:-1,1～86400)

DIOSA_CMDSND_DEFAULT(-1)を指定した場合、環境変数、または環境定義の値となる。

int RtryCnt; 接続処理に失敗した場合の接続リトライ回数を指定する。(-1～100)

0 を指定した場合、リトライしない。

DIOSA_CMDSND_DEFAULT(-1)を指定した場合、環境変数、または環境定義の値となる。

int RtryIntvl; 接続リトライ処理の待合せ時間を指定する。(単位：秒、範囲：-1,1～3600)

DIOSA_CMDSND_DEFAULT(-1)を指定した場合、環境変数、または環境定義の値となる。

char InFileOption[3]; 実行コマンドのパラメータに指定する入力ファイル名のオプションを指定する。(最大 2 文字)

入力ファイルのオプションが不要な場合、NULL を設定する。

char InFileDelMode; コマンド実行後の入力ファイルの削除有無について指定する。

DIOSA_CMDSND_DELMODE_NO	配信元、配信先の入力ファイルを削除しない
DIOSA_CMDSND_DELMODE_ORG	配信元の入力ファイルを削除する
DIOSA_CMDSND_DELMODE_DST	配信先の入力ファイルを削除する
DIOSA_CMDSND_DELMODE_BOTH	配信元、配信先の入力ファイルを削除する

char InFilePath[256]; コマンド配信時、配信元から配信先に転送したい入力ファイルを絶対パス名で指定する。(最大 255 文字)

入力ファイルの転送が不要な場合、NULL を設定する。

注意

- ForceFlg に DIOSA_OFF \| DIOSA_CMDSND_BLKEXCL_ON を指定した場合、閉塞中のノードは配信結果に含めない。配信対象となるノードが全て閉塞中であった場合、DIOSA_ENOENT が返却される。

関連

diosacmdsend,t_diosa_cmdsenduca

2.1.68　t_diosa_cmdsenduca(コマンド配信情報構造体)

名前

t_diosa_cmdsenduca - コマンド配信情報構造体

書式

```
#include <diosa.h>
t_diosa_cmdsenduca SendUca;
```

説明

t_diosa_cmdsenduca 構造体は、コマンド配信先情報、配信結果確認情報、配信コマンドが格納される。

t_diosa_cmdsendinfo SendInfo;	コマンド配信先情報
t_diosa_cmdresultinfo ResultInfo;	配信結果確認情報
int CmdTextLen;	コマンドの長さ(NULL を含まない)を指定する(1～1023)
char *CmdText;	コマンドの格納領域アドレスを指定する

関連

diosacmdsend,t_diosa_cmdsendinfo,t_diosa_cmdresultinfo

2.1.69　t_diosa_cmdsetuca(コマンドタイマ登録用構造体)

名前

t_diosa_cmdsetuca - コマンドタイマ登録用構造体

書式

```
#include <diosa.h>
t_diosa_cmdsetuca CmdsetUca;
```

説明

t_diosa_cmdsetuca 構造体は、コマンドタイマ登録処理のインタフェースに関連する情報が格納される。

short Textalen　　送信メッセージ長を指定する。

関連

diosatmccmdset

2.1.70　t_diosa_coqueryuca(CO 制御タイマ照会用構造体)

名前

t_diosa_coqueryuca - CO 制御タイマ照会用構造体

書式

```
#include <diosa.h>
t_diosa_coqueryuca CoqueryUca;
```

説明

t_diosa_coqueryuca 構造体は、CO 制御タイマ照会処理のインタフェースに関連する情報が格納される。

char CoName[31]	宛先の CO 名(英数字 30 文字以内)が返却される。	
char TxId[7]	トランザクション ID(英数字 6 文字以内)が返却される。	
char Hold[5]	タイマエントリ状態が返却される。	
	HOLD	保留状態
	ACTV	活性状態
short Seq	照会したいタイマ要求エントリを指定する。	
	DIOSA_FIRST	先頭タイマエントリを照会する。
	DIOSA_NEXT	前回照会した次のタイマエントリをタイマ ID 順に照会する。
	DIOSA_DIRECT	指定したタイマ ID のタイマエントリを照会する。
short Textalen	送信メッセージ長が返却される。	

関連

diosatmccoquery

2.1.71　t_diosa_cosetuca(CO 制御タイマ登録用構造体)

名前

t_diosa_cosetuca - CO 制御タイマ登録用構造体

書式

```
#include <diosa.h>
t_diosa_cosetuca CosetUca;
```

説明

t_diosa_cosetuca 構造体は、CO 制御タイマ登録処理のインタフェースに関連する情報が格納される。

char CoName[31]	宛先の CO 名を示す。 英数字 30 文字以内(NULL 終端で終了すること)で指定すること。
char TxId[7]	トランザクション ID を示す。 英数字 6 文字以内(NULL 終端で終了すること)で指定すること。
short Textalen	送信メッセージ長を指定する。

関連

diosatmccoset

2.1.72　t_diosa_dbinfo(DB 情報構造体)

名前

t_diosa_dbinfo - DB 情報

書式

```
#include <diosa.h>
t_diosa_dbinfo   DB 情報構造体名;
```

説明

```
short  Type;            DB タイプ
                          DIOSA_DB_NO          :DB 更新しない
                          DIOSA_DB_MAPID       :メモリキャッシュ対象で MAPID 指定
                          DIOSA_DB_MAINKEY     :メモリキャッシュ対象でメインキー指定
                          DIOSA_DB_IM          :メモリキャッシュ対象
                          DIOSA_DB_RGSET       :DB 対象でリソースグループセット指定
                          DIOSA_DB_RGID        :DB 対象でリソースグループ ID 指定
                          DIOSA_DB_DAC         :対象 DB を DB アクセス API で決定する
                          DIOSA_DB_DACMAINKEY  :対象 DB を DB アクセス API で決定し、MAINKEY 指定
                          DIOSA_DB_DACMAPID    :対象 DB を DB アクセス API で決定し、MAPID 指定
char   RgSet[31+1];     リソースグループセット(31 バイト)
char   MainKey[32];     メインキー(32 バイト)
int    MapId;           MAPID
short  RgId;            リソースグループ ID
int    Wait;            クラスタ再構成待ち合わせ時間 (秒)
```

関連

diosatxstart, t_diosa_analyze, t_diosa_dcuca, t_diosa_uca

2.1.73　t_diosa_dcuca(電文送受信構造体)

名前

t_diosa_dcuca - 電文送受信構造体 (DCUCA)

書式

```
#include <diosa.h>
t_diosa_dcuca    電文送受信構造体名;
```

説明

電文送受信時にCO 制御とAP 間の情報受け渡しを行うの領域(t_diosa_dcuca 領域)である。

```
  short    MsgType;              電文タイプ
                                     DIOSA_MSG_LS          : 論理システム間
                                     DIOSA_MSG_CO          : 論理システム内派生(CoName 指定)
                                     DIOSA_MSG_TX          : 論理システム内派生(TxId 指定)
                                     DIOSA_MSG_RELAYCO     : 論理システム内派生(中継 CO 指定)
                                     DIOSA_MSG_RELAYTX     : 論理システム内派生(中継 TxId 指定)
                                     DIOSA_MSG_CHAIN       : 連鎖
                                     DIOSA_MSG_REST        : 保留
  t_diosa_lsysaddr  LSys;        論理システム間アドレス情報

  t_diosa_lnodeaddr  Entry;      論理システム間電文登録アドレス情報
  t_diosa_lnodeaddr  LNode;      論理システム内送信先情報
  t_diosa_dbinfo     Db;         DB 情報

  char     VdName[24+1];         VD 名 (最大 24 文字)

  char     TxId[6+1];            TxID (最大 6 文字)
  char     CoName[30+1];         CO 名 (最大 30 文字)

  int      MsgSize;              受信電文長、または、送信電文長

(受信情報)
  short    RecvType;             電文を受信したリソース種別
                                     DIOSA_RECVFROM_TERM    (端末からの受信)
                                     DIOSA_RECVFROM_TPP     (VD からの受信)
  short    SendCnt;              電文タイプを論理システム内派生として送信し続けている回数
  short    Source;               論理システム内電文送信元識別子
                                     DIOSA_SRC_AP           : 業務
                                     DIOSA_SRC_TMC          : タイマ
                                     DIOSA_SRC_LPATH        : 端末接続/切断
                                     DIOSA_SRC_GNT_RETOV    : 電文保証(リトライオーバ)
                                     DIOSA_SRC_OTC_RSP      : 都度接続応答電文
                                     DIOSA_SRC_OTC_ERR      : 都度接続エラー電文
                                     DIOSA_SRC_COCCL        : CO 制御サーバ以外のクライアント、
                                                              バッチ AP

(送信用パラメータ)
  short    SendMode;             送信モード
                                     DIOSA_SEND_FORCE: 強制送信
```

DIOSA_SEND_DELAY: 遅延送信

short IgnorePreBlk; 宛先論理ノードが予閉塞中に送信するか

DIOSA_NO : 送信しない

DIOSA_YES : 送信する

short Roundrobin; ラウンドロビン

DIOSA_NO : 呼び出し優先度に従う

優先度	選択する宛先
高	同じ TPBASE モニタ
中	同じ論理ノードの TPBASE モニタ 複数ある場合はラウンドロビン
低	他論理ノードの TPBASE モニタ 複数ある場合はラウンドロビン

DIOSA_YES : ラウンドロビン呼び出しを行う

short NodeType; 送信先論理ノード種別

DIOSA_LNODETYPE_UNAVAIL：ノードタイプ無効

DIOSA_LNODETYPE_AP : AP ノード

DIOSA_LNODETYPE_OLTP : OLTP ノード

(送受信用パラメータ)

short MsgGnt; 電文保証フラグ

DIOSA_MSGGNT_NO : 電文保証しない

DIOSA_MSGGNT_YES : 電文保証を行う

DIOSA_MSGGNT_SEQ : 順序性保証あり

char APInfo[256]; AP 固有情報

電文保証、または都度接続送信を選択した場合は指定可

int DStatus; 電文送信時の実行結果(詳細)

t_diosa_tpstatus TpStatus; 電文送受信時の実行結果

(送受信用パラメータ、電文保証用)

t_diosa_gntinfo Gnt; 電文保証情報構造体

(送受信用パラメータ、都度接続電文送信デーモン用)

t_diosa_otcinfo Otc; 都度接続情報構造体

2.1.74　t_diosa_gntinfo(電文保証用構造体)

名前

t_diosa_gntinfo – 電文保証用構造体(t_diosa_gntinfo)

書式

```
#include <diosa.h>
t_diosa_gntinfo    電文保証用構造体名;
```

説明

char	MsgKey[32];	電文を一意に識別する識別子 論理システム間の電文保証をする場合、送信時に指定必須。 宛先システムに関わらず論理システム内で扱う保証電文内で一意に決まる識別子にする必要がある。 また、1 バイト目は'0xfe'以外を使用する。 論理システム内派生での電文保証の場合は指定不要。
int	SendMax;	最大送信回数(1～524287) 0 の場合は SG 値に従う。
short	RetryInterval;	再送時間間隔(秒)(1～9999) 0 の場合は SG 値に従う。
char	SendStay;	送信はせずに保留するかどうか DIOSA_NO　：　保留しない(電文送信する) DIOSA_YES　：　保留する(電文送信しない)
char	SeqGroup[32];	順序性保証グループ識別子 MsgGnt に順序性保証を指定した場合のみ有効になる。 同一 DB の範囲で、同じグループを指定した電文間の順序性が保証される。(IM の場合は MAPID が一致。DB の場合は RGSET が一致。)
int	SeqNo;	順序性保証通番 同一順序性保証グループ内での通番が設定される。
char	OrgCoName[30+1];	保証電文送信時に指定した CO 名 保証電文の再送リトライオーバー時に呼び出される CO 上で電文受信した場合のみ設定される。
char	MainKey[32];	受信電文解析出口で更新先 DB として DAC を指定した場合に、送信電文や電文保証機能の管理レコードを保存する MAP を決定するメインキーを指定する。
char	BkSyncType;	バックアップシステムに送信電文や管理レコードを同期する方法を指定する。 DIOSA_MSGGNT_SYNCNO :同期しない DIOSA_MSGGNT_SYNCCMD: 同期コマンド(digntbksync)で同期する リアルタイムでの同期は行わない

注意

- diosarecvtx、diosasendtx で使用する t_diosa_dcuca 構造体の内部情報である。
- 電文送信時は、MsgGnt に DIOSA_MSGGNT_YES、DIOSA_MSGGNT_SEQ が指定された場合、本構造体の情報が必要となる。
- 電文受信時は、MsgGnt に DIOSA_MSGGNT_YES、DIOSA_MSGGNT_SEQ が返却された場合、本構造体に情報が設定される。

2.1.75　t_diosa_lnodeaddr(論理ノードアドレス構造体)

名前

t_diosa_lnodeaddr – DIOSA 論理システム内部のアドレス情報

書式

```
#include <diosa.h>
t_diosa_lnodeaddr   DIOSA 論理システム内部のアドレス情報;
```

説明

```
  char    LNodeName[15+1];       論理ノード名 (最大 15 文字)
  char    TpMonitor[8+1];        TPBASE モニタ名 (最大 8 文字)
```

関連

t_diosa_dcuca、diosarecvtx、diosagetsrctx、diosasendtx

2.1.76　t_diosa_lsysaddr(論理システム間アドレス情報構造体)

名前

t_diosa_lsysaddr – 論理システム間アドレス情報構造体

書式

```
#include <diosa.h>
```

説明

```
char    LsName[15+1];           外部論理システム名 (最大 15 文字)
char    AcsPoint[15+1];         外部アクセスポイント名(最大 15 文字)
char    Protocol[31+1];         プロトコル名(最大 31 文字)
char    LNodeName[15+1];        外部論理ノード名 (最大 15 文字)
char    TpMonitor[8+1];         外部 TPBASE モニタ名 (最大 8 文字)
char    TermName[31+1];         入口(受信)端末名(最大 31 文字)
char    ConnectTime[14+1];      入口(受信)端末接続時間
short   TermType;               入口(受信)端末タイプ
                                  DIOSA_PATH_FULLTIME    : 常時接続端末
                                  DIOSA_PATH_ONETIME     : 都度接続端末
```

関連

t_diosa_dcuca、diosarecvtx、diosagetsrctx、diosasendtx

2.1.77　t_diosa_msgbuf(電文バッファ構造体)

名前

t_diosa_msgbuf - 電文バッファ構造体

書式

```
#include <diosa.h>
t_diosa_msgbuf　電文バッファ名;
```

説明

diosamsgbufalloc により確保される電文バッファの型。

メンバ変数

char	*Addr;	バッファアドレス
int	Size;	バッファサイズ

関連

diosamsgbufalloc, diosamsgbuffree, diosasendtx, t_diosa_analyze

2.1.78　t_diosa_otcinfo(都度接続電文送信情報構造体)

名前

t_diosa_otcinfo – 都度接続電文送信情報構造体(t_diosa_otcinfo)

書式

```
#include <diosa.h>
t_diosa_otcinfo　都度接続電文送信情報構造体名;
```

説明

unsigned char	SettingGroup;	都度接続送受信の定義グループ指定 (1～255) 0 を指定すると、最も小さいグループ番号の定義に従う。
unsigned char	Response;	応答受信を同一コネクションでもらうか否か DIOSA_RESP_NO　: 同一コネクションで応答をもらわない DIOSA_RESP_YES　: 同一コネクションで応答をもらう
short	MsgTimeOut;	都度接続送受信のタイムアウト値(秒) (1～32767) 0 を指定すると環境定義の値が使用される。

2.1.79　t_diosa_tmctime(タイマ時間設定用構造体)

名前

t_diosa_tmctime - タイマ時間設定用構造体

書式

```
#include <diosa.h>
t_diosa_tmctime Timer;
```

説明

t_diosa_tmctime 構造体はタイマ制御で扱う日時のインタフェースに関連する情報が格納される。

short Second	秒を示す。[0-59]の範囲で設定する。
short Minute	分を示す。[0-59]の範囲で設定する。
short Hour	時を示す。[0-23]の範囲で設定する。
short Day	日を示す。0、または[1-31]の範囲で設定する。
short Month	月を示す。0、または[1-12]の範囲で設定する。
short Year	年を示す。0、または[1990-2089]の範囲で設定する。

※上記構造体の各項目と、タイマ制御インタフェース構造体の時刻形式との関連を下表に示す。

項目	時刻形式			説明
	インターバル	即時	時刻	
short Second	○	○	○	[0-59]の範囲で設定
short Minute	○	○	○	[0-59]の範囲で設定
short Hour	○	○	○	[0-23]の範囲で設定
short Day	－	－	○	0、または[1-31]の範囲で設定
short Month	－	－	○	0、または[1-12]の範囲で設定
short Year	－	－	○	0、または[1990-2089]の範囲で設定

○：使用　－：未使用

注意

- タイマ制御インタフェース構造体において、Mode(時刻形式)が DIOSA_CLOCK (時刻指定)の時、タイマ値を設定する場合は、Year に年、Month に月、Day に日、Hour に時、Minute に分、Second に秒を設定する。Year、Month、Day の全てに 0 を指定した場合、Hour、Minute、Second で指定した時刻にタイマ通知を毎日行う。Year、Month、Day の一部だけに 0 を指定することはできない。
- Mode が DIOSA_INTER(インターバル指定)、DIOSA_IMMEDIATE(即時指定)の時、タイマ値を設定する場合は、Hour に時、Minute に分、Second に秒を設定し、他の項目には 0 を設定する。ただし、Hour、Minute、Second 全てに 0 を指定することはできない。

関連

t_diosa_tmcuca, diosatmccoset, diosatmccmdset, diosatmccoquery, diosatmccmdquery

2.1.80　t_diosa_tmcuca(タイマ制御インタフェース構造体)

名前

t_diosa_tmcuca - タイマ制御インタフェース構造体

書式

```
#include <diosa.h>
t_diosa_tmcuca Tmcuca;
```

説明

t_diosa_tmcuca 構造体は、利用者が設定するタイマ制御機能のインタフェースに関連する情報が格納される。

char TimerId[17]	タイマ ID を示す。(NULL 終端を含む英数字)	
int　Mode	時刻形式を示す。	
	DIOSA_INTER	インターバル指定
	DIOSA_IMMEDIATE	即時指定
	DIOSA_CLOCK	時刻指定
int　Count	通知回数(0～999)を示す。	
	DIOSA_NOLIM(0)	無限回の通知
t_diosa_tmctime Time	タイマ値を示す。 **Mode** が DIOSA_INTER、DIOSA_IMMEDIATE の場合は、タイマ実行間隔を設定する。 **Mode** が DIOSA_CLOCK の場合は、タイマ通知日時を設定する。 **t_diosa_tmctime** 構造体については、**t_diosa_tmctime** を参照すること。	
short KeyCheck	タイマ ID 重複チェック有無を示す。	
	DIOSA_ON	タイマ ID の重複チェックを行い、タイマ情報を上書きして警告終了する。
	DIOSA_OFF	タイマ ID の重複チェックは行なわず、タイマ情報を上書きして正常終了する。

※上記した t_diosa_tmcuca 構造体の各項目と、タイマ制御関数入出力情報との関連を下表に示す。

項目	登録関数	削除関数	照会関数	保留関数	解除関数	説明
char　TimerId[17]	I	I	I/O	I	I	タイマ ID
int　Mode	I	-	O	-	-	時刻形式
int　Count	I	-	O	-	-	通知回数
t_diosa_tmctime Timer	I	-	O	-	-	タイマ値
short KeyCheck	I	-	-	-	-	タイマ ID 重複チェック有無

I：入力型　O：出力型　I/O：入出力型　-：未使用

注意

- Mode(時刻形式)に時刻指定(DIOSA_CLOCK)を指定し Timer(タイマ値)に年月日を指定した場合、Count(通知回数)を指定することはできない。また、Timer に過去の日時、または存在しない日時を設定した場合、DIOSA_ETIMEFORM となる。

関連

diosatmccoset, diosatmccmdset, diosatmcreset, diosatmccoquery, diosatmccmdquery, diosatmchold, diosatmcactv, t_diosa_tmctime

2.1.81　t_diosa_tpstatus(TPBASE リターン情報構造体)

名前

t_diosa_tpstatus – TPBASE リターン情報

書式

```
#include <diosa.h>
t_diosa_tpstatus  TPBASE リターン情報;
```

説明

```
struct t_diosa_tpstatus {
  char    StatusKey[2];  TPBASE 送受信命令の実行結果
  char    EndKey;        TPBASE システムからの状態通知
};
```

StatusKey[2]および EndKey は、TPBASE の CS_STRUCT_EX の status_key[2]および end_key に相当する。

関連

t_diosa_dcuca, diosasendtx

2.1.82　t_diosa_uca(CO、利用者出口呼び出し構造体)

名前

t_diosa_uca - CO・利用者出口呼び出し構造体 (DIOSAUCA)

書式

```
#include <diosa.h>
t_diosa_uca    diosauca 領域名;
```

説明

CO 制御と AP 間、およびバッチ AP 制御と AP 間で受け渡す情報の格納領域(t_diosa_uca 領域)を宣言する。
CO／利用者出口の呼び出し時に t_diosa_uca 領域が第一パラメータとして渡される。

```
  short    UcaLen;              diosauca の長さ
  short    UcaRev;              diosauca のリビジョン
  short    ExitId;              利用者出口識別子
                                    DIOSA_EXIT_CO        : CO
                                    DIOSA_EXIT_PRCINIT   : プロセス初期化出口
                                    DIOSA_EXIT_PRCTERM   : プロセス終了出口
                                    DIOSA_EXIT_ANALYZE   : 受信電文解析出口
                                    DIOSA_EXIT_TRNSINIT  : トランザクション初期化出口
                                    DIOSA_EXIT_TRNSTERM  : トランザクション終了出口
                                    DIOSA_EXIT_ABORT1    : アボート出口#1
                                    DIOSA_EXIT_ABORT2    : アボート出口#2
  char     LsName[15+1];        論理システム名 (最大 15 文字)
  char     LNodeName[15+1];     論理ノード名 (最大 15 文字)
  short    LNodeType;           論理ノード種別
                                    DIOSA_LNODETYPE_AP      : AP ノード
                                    DIOSA_LNODETYPE_OLTP    : OLTP ノード
  short    DiosaStartMode;      DIOSA 起動モード
                                    DIOSA_STARTMODE_COLD    : コールド
                                    DIOSA_STARTMODE_WARM    : ウォーム
  char     TpMonitor[8+1];      TPBASE モニタ名 (最大８文字)
  char     TpClass[32+1];       TPBASE クラス名
  void    *TpsUcaPtr;           TPSUCA ポインタ
  char     TxId[6+1];           TPBASE の TxID (最大６文字)
  char     CoName[30+1];        CO 名(最大 30 文字)
  short    DbErr;               更新先の DB に障害が発生しているか
                                    DIOSA_OFF        : DB 障害なし
                                    DIOSA_ON         : DB 障害あり
  unsigned short  DbType;       トランザクション ID に対応する DB タイプ
                                環境定義($COCENV)の定義値
                                    DIOSA_DB_NO      : DB 未使用
                                    DIOSA_DB_RGSET   : DB のみ使用
                                    DIOSA_DB_IM      : IM のみ使用
                                    DIOSA_DB_ALL     : DB,IM 使用
  t_diosa_dbinfo  DbInfo;       トランザクション開始時の DB 情報
  short    ExitKey;             トランザクション処理結果 (後述)
  int      UserExitKey;         利用者コード (参照用) (後述)
```

```
  short   RetryLim;          トランザクション再実行回数上限値
  short   RetryCount;        トランザクション再実行回数
  short   ChainNum;          連鎖回数
  short   CommitNum;         コミット API(diosacommit)実行回数
  short   SendCnt;           電文タイプを論理システム内派生として送信し続けている回数
  int     MsgSizeMax;        クラスで処理可能な電文の最大サイズ
  char   *ApComarea;         AP 共通領域アドレス
  char   *ApAreaPtr;         利用者任意領域ポインタ
  char   *BatchApInfo;       バッチ AP 制御時の利用者任意領域のポインタ
  int     Status;            状態コード（CO・利用者出口の終了ステータス）（後述）
  int     UserStatus;        利用者コード（設定用）（後述）
};
```

AP 共通領域アドレス(ApComarea)

CO 制御の TPP において、プロセス初期化出口内で diosaapptrset により通知された AP 共通領域を、他の利用者出口・CO で ApComarea により参照することができる。

プロセス初期化出口では ApComarea は参照不可である。

利用者任意領域ポインタ(ApAreaPtr)

CO 制御の TPP において、各利用者出口・CO によって任意のポインタを指定することができる。

設定された ApAreaPtr は、以降の他の利用者出口・CO によって参照することができる。

プロセス初期化出口呼び出し時は、NULL が設定される。

トランザクション毎にプロセス初期化出口で設定された ApAreaPtr でリセットされる。

バッチ AP 制御時の利用者任意領域ポインタ(BatchApInfo)

バッチ AP 制御において、各利用者出口・CO によって任意のポインタを指定することができる。

バッチ AP 制御起動時のパラメータに「-- USER-PARAMETER」を指定した場合、このパラメータ群のアドレスが設定される。

設定された BatchApPtr は、以降の他の利用者出口・CO によって参照することができる。

トランザクション処理結果＜ExitKey＞

ExitKey にはトランザクションの状態が格納される。
特定の利用者出口において、トランザクション処理結果の値に応じて、利用者コード（UserExitKey）の値が有効となる。

ー：対象外 ○：対象 ●：利用者コード(UserExitKey)参照可 利用者出口 トランザクション処理結果	プロセス初期化	プロセス終了	受信電文解析	トランザクション初期化	トランザクション終了	CO	エラーCO	アボート＃1	アボート＃2
正常 DIOSA_EK_NORMAL	○	○	○	○	○	○	ー	ー	ー
AP からの異常終了要求 DIOSA_EK_APREQ	ー	ー	ー	ー	ー	ー	ー	●	●
リトライオーバー DIOSA_EK_RETRYOV	ー	ー	ー	ー	ー	ー	ー	●	●
エラーCO 呼び出しエラー DIOSA_EK_ERRCO	ー	ー	ー	ー	ー	ー	ー	○	○
継続不可能な障害発生 DIOSA_EK_COCREQ	ー	ー	ー	ー	ー	ー	ー	○	○
IM マスタ計画切替 DIOSA_EK_SWITCH	ー	ー	ー	ー	ー	ー	ー	○	○
IM マスタ障害切替 DIOSA_EK_ESWITCH	ー	ー	ー	ー	ー	ー	ー	○	○
IM サーバ障害 DIOSA_EK_EREADY	ー	ー	ー	ー	ー	ー	ー	○	○
Oracle 障害 DIOSA_EK_EDB	ー	ー	ー	ー	ー	ー	ー	○	○
IM 障害 DIOSA_EK_IMERR	ー	ー	ー	ー	ー	ー	ー	○	○
デッドロック発生 DIOSA_EK_COCDEADLOCK	-	-	-	-	-	-	-	○	○
例外発生（シグナル発生） DIOSA_EK_EXCP	ー	ー	ー	ー	ー	ー	ー	○	ー
CPU 時間超過 DIOSA_EK_CPUOV	ー	ー	ー	ー	ー	ー	ー	○	ー
ロールバックリトライ DIOSA_EK_ROLLBACK	ー	ー	ー	●	●	●	ー	ー	ー
デッドロックリトライ DIOSA_EK_DEADLOCK	ー	ー	ー	●	●	●	ー	ー	ー
CO 閉塞 DIOSA_EK_COBLOCK	ー	ー	ー	○	○	ー	○	ー	ー
CO 呼び出しエラー DIOSA_EK_CALLERR	ー	ー	ー	ー	○	ー	○	ー	ー

CO が決定できない DIOSA_EK_ECODEC	－	－	－	○	○	－	○	－	－
更新 DB タイプエラー DIOSA_EK_EDBTYPE	－	－	－	○	○	－	○	－	－
RG セットエラー DIOSA_EK_ERGSET	－	－	－	○	○	－	○	－	－
(MAPID,RGID)指定範囲エラー DIOSA_EK_EDBRANGE	－	－	－	○	○	－	○	－	－

状態コード＜CO 制御 TPP 管理下＞

各利用者出口にて状態コード(Status)に指定可能な値と、利用者コード(UserStatus)の設定が可能な組み合わせは以下のとおりである。

○：指定可 ×：指定不可 －：指定は無効 ●：利用者コード(UserStatus)指定可 状態コード ＼ 利用者出口	プロセス初期化（※1）	プロセス終了	受信電文解析	トランザクション初期化	トランザクション終了	CO	アボート#1	アボート#2
正常終了 DIOSA_ST_DONE	○	×	○	○	○	○	×	○
異常終了 DIOSA_ST_ABORT(※2)	○	×	○	○	●	●	×	○
オプション付き異常終了 DIOSA_ST_ABORTCONT：プロセス継続 DIOSA_ST_ABORTSTOP：プロセス終了	○	×	○	○	●	●	×	－
ロールバックリトライ DIOSA_ST_ROLLBACK	×	×	×	×	×	●	×	×
デッドロックリトライ DIOSA_ST_DEADLOCK	×	×	×	×	●	●	×	×
ロールバック連鎖 DIOSA_ST_RLBKCHAIN	×	×	×	×	×	○	×	×

※1 プロセス初期化の異常終了要求はどの異常終了要求を返却してもプロセス終了となる。

※2 環境定義 COCENV 節の COMMON 項 ABORTREQ の値に従ってプロセスの停止か継続を決定する。

利用者コード（UserStatus, UserExitKey）

各利用者出口を特定の状態コード（Status）を持って終了する際に UserStatus に指定された任意の値は、以降の利用者出口において、トランザクション処理結果に従って UserExitKey として参照することができる。

ただし、0 は未指定（無効値）を意味する予約値である。

状態コード＜バッチ AP 制御管理下＞

利用者は、状態コード(Status)、詳細コード(UserStatus)に以下の値を設定し、バッチ AP 制御に対して処理の指示を行う。

状態コード	詳細コード	説明
DIOSA_ST_DONE	0000～9999	正常終了。 バッチ AP 制御の戻り値は０
DIOSA_ST_ABORT	0000～9999	異常終了要求 バッチ AP 制御の戻り値は１
DIOSA_ST_ABORTCONT	0000～9999	
DIOSA_ST_ABORTSTOP	0000～9999	
DIOSA_ST_ROLLBACK	0000～9999	ロールバックリトライ要求(ロールバック後にＣＯを再度呼び出す) ＣＯからのリターン時のみ有効

利用者出口との対応

CO および各利用者出口の入出力項目について以下に示す。

項目名		CO 制御 TPP								バッチ AP 制御			
		プロセス初期化出口	受信電文解析出口	トランザクション初期化出口	CO	トランザクション終了出口	アボート出口＃1	アボート出口＃2	プロセス終了出口	初期化出口	CO	正常終了	アボート出口
UcaLen		I								I			
UcaRev		I								I			
ExitId		I								I			
LsName		I								I			
LNodeName		I								I			
LNodeType		I								I			
DiosaStartMode		I								I			
TpMonitor		I								—			
TpClass		I								—			
TpsUcaPtr		I								—			
TxId		—		I					—	—			
CoName		—		I					—	—	I		
DbErr		—		I					—	—	I		
DbType		I								—			
DbInfo	Type	—		I					—	I			
	MainKey	—		I					—	—			
	MapId	—		I					—	I			
	RgSet	—		I					—	I			
Exitkey		—		I					—	—		I	
UserExitKey		—		I					—	—			
RetryLim		—		I					—	—	I		
RetryCount		—		I					—	—	I		
ChainNum		—		I					—	—			
CommitNum		—		I					—	—			
SendCnt		—	I						—	—			
MsgSizeMax		I								—			
ApComarea		—	I							—			
ApAreaPtr		0	I/0		I	I/0	I			—			
BatchApInfo		—								I/0			
Status		0					—	0	—	0	I/0		I
UserStatus		—	0				—			0	I/0		I

2.2　通信制御

2.2.1　関数一覧

(1)　DB監視機能

diosadbchangeconnect	使用するデータベース・インスタンスの決定、および接続確認の実施
diosadbconnect	指定されたデータベース・インスタンスに接続する
diosadbdisconnect	現在接続されているデータベース・インスタンスを切断する
diosadbfaultnotification	DB障害検出時に障害通知を行う
diosadbmulticonnect	指定されたデータベース・インスタンスに接続する
diosadbreconnect	活性と判断されたデータベース・インスタンスに再接続する
diosagetdbctx	データベースコンテキストを返却する
t_diosa_dbuca	データベース接続先切り替え関数で使用する構造体

(2)　パス管理機能

diosagetacspinfo	アクセスポイントのプロトコル、端末情報を取得する
diosagetnodepathstatus	自ノードの論理ノード間パス状態を照会する
diosalspathctrl	論理システム間パスを接続・切断する
t_diosa_lspathctrl	論理システム間パスを接続切断関数で使用する構造体
t_diosa_nodepathstatus	論理ノード間パス状態照会関数で使用する構造体

(3)　電文保証

diosagntcheck	受信した電文の二重処理チェックを行う
diosagntcheck_mk	受信した電文の二重処理チェックを行う(更新先DBがDACの場合)
diosagntdel	保証電文を削除する
diosagntdel_mk	保証電文を削除する(更新先DBがDACの場合)
diosagntfin	受信した保証電文の処理が完了したことを送信元へ通知する
diosagntshiftnextsend	次回の送信タイミングを変更する
diosagntshiftnextsend_mk	次回の送信タイミングを変更する(更新先DBがDACの場合)

2.2.2　　diosadbchangeconnect(DB 接続先切り替え関数)

名前

diosadbchangeconnect -　使用するデータベース・インスタンスの決定、および接続確認の実施。

書式

```
#include <diosa.h>
int diosadbchangeconnect(t_diosa_dbuca *DbUca, long *Dstatus);
```

説明

指定されたリソースグループセットが使用する DB が RAC 構成の場合に、RAC を構成する DB インスタンスのどちらを使用するかをトランザクション単位に決定する。DB インスタンスへ未接続の場合は本関数内で接続する。

t_diosa_dbuca *DbUca(入出力型)

　リソースグループ ID／リソースグループセット名を指定する。

long *Dstatus(出力)

　データベース接続処理の詳細ステータスを返却する。

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_ERROR(-1)	異常終了
DIOSA_EPARAM(-3)	パラメータエラー
DIOSA_EMEM(-6)	メモリ確保エラー
DIOSA_ENOINIT(-11)	未初期化エラー
DIOSA_ELOCK(-14)	ロックエラー
DIOSA_EUNLOCK(-29)	アンロックエラー
DIOSA_EFATAL(-30)	全データベース・インスタンス使用不可能
DIOSA_EFUNCNAV(-34)	機能利用不可
DIOSA_EATTACH(-43)	アタッチエラー
DIOSA_EFLOCK(-54)	ファイルロックエラー

注意

- 本関数を実行するためには、DB ヘルスチェック機能が動作していなければならない。
- 本関数にて、データベース・インスタンスへの接続確認および再接続処理を行うため、インスタンスグループ内に１つのリソースグループセットしか定義されていない場合でも、常駐アプリケーションにおいては、トランザクション単位に本関数を実施する必要がある。
- 本関数内でデータベース・インスタンスに接続する際、接続の多重度制御(高負荷時に接続処理を行わない制御)を行わず、即時に接続を試みる。
- 本関数を呼び出した際は、トランザクション終了時またはスレッド終了時にデータベース切断関数を呼び出し、データベース・インスタンスを切断しておくこと。

関連

diosagetdbctx

2.2.3　　diosadbconnect(DB 接続関数[シングルコネクション])

名前

diosadbconnect - 指定されたデータベース・インスタンスに接続する。

書式

```
#include <diosa.h>
int diosadbconnect(t_diosa_dbuca *DbUca, long *Dstatus);
```

説明

DbUca で指定されたリソースグループセットに対応するデータベース・インスタンスに接続する。

t_diosa_dbuca *DbUca(入出力型)

　リソースグループ ID／リソースグループセット名を指定する。

long *Dstatus(出力)

　データベース接続処理の詳細ステータスを返却する。

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_ALREADY(1)	接続済み
DIOSA_ERROR(-1)	異常終了
DIOSA_EPARAM(-3)	パラメータエラー
DIOSA_EMEM(-6)	メモリ確保エラー
DIOSA_ELOCK(-14)	ロックエラー
DIOSA_EUNLOCK(-29)	アンロックエラー
DIOSA_EFATAL(-30)	接続失敗
DIOSA_EFUNCNAV(-34)	機能利用不可
DIOSA_EATTACH(-43)	アタッチエラー

注意

- 本関数を実行するためには、DB ヘルスチェック機能が動作していなければならない。
- 本関数を使用してデータベース・インスタンスに接続した場合は、データベース接続先切り替え関数を使用しなくても DB アクセスを行うことができる。
- 本関数では、指定されたリソースグループ ID もしくは、リソースグループセットに対応するデータベース・インスタンスにのみ接続する。
- 本関数は接続の多重度制御(高負荷時に接続処理を行わない制御)を行わず、即時に接続を試みる。
- マルチスレッドにおいてはスレッド単位に本関数を実行する必要がある。

関連

diosadbdisconnect

2.2.4　　diosadbdisconnect(DB 切断関数)

名前

diosadbdisconnect - 現在接続されているデータベース・インスタンスを切断する。

書式

```
#include <diosa.h>
int diosadbdisconnect(t_diosa_dbuca *DbUca, long *Dstatus);
```

説明

DbUca で指定されたリソースグループセットに対応したインスタンスグループの全データベース・インスタンスを切断する。

t_diosa_dbuca *DbUca(入出力型)

　リソースグループ ID／リソースグループセット名を指定する。

long *Dstatus(出力)

　データベース接続処理の詳細ステータスを返却する。

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_ERROR(-1)	異常終了
DIOSA_EPARAM(-3)	パラメータエラー
DIOSA_ENOINIT(-11)	未初期化エラー
DIOSA_ELOCK(-14)	ロックエラー
DIOSA_EUNLOCK(-29)	アンロックエラー
DIOSA_EFUNCNAV(-34)	機能利用不可
DIOSA_EDETACH(-44)	デタッチエラー

注意

- マルチスレッドにおいてはスレッド単位に本関数を実行する必要がある。

関連

diosadbmulticonnect, diosadbconnect

2.2.5　diosadbfaultnotification(DB障害通知関数)

名前

diosadbfaultnotification - DB障害検出時に障害通知を行う。

書式

```
#include <diosa.h>
int diosadbfaultnotification(void **SqlCtx, char *DbStatus, long *Dstatus);
```

説明

プロセスがDBの障害を検出した際に呼び出すことにより、自プロセスと障害データベース・インスタンスとのコネクションを切断する。

void **SqlCtx(入力型)

障害を検出したDBのコンテキストを指定する。

char *DbStatus (出力)

データベース・インスタンスの状態を返却する。

DIOSA_DBSTATUS_DBACTIVE (0x00)	:データベース・インスタンス使用可能
DIOSA_DBSTATUS_DBDOWN (0x01)	:全データベース・インスタンス障害
DIOSA_DBSTATUS_CLUSTERRECONFIG (0x02)	:クラスタ再構成中

long *Dstatus(出力)

データベース接続処理の詳細ステータスを返却する。

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_ERROR(-1)	異常終了
DIOSA_EPARAM(-3)	パラメータエラー
DIOSA_ENOINIT(-11)	未初期化エラー
DIOSA_EFUNCNAV(-34)	機能利用不可

2.2.6　　diosadbmulticonnect(DB 接続関数[マルチコネクション])

名前

diosadbmulticonnect -指定されたデータベース・インスタンスに接続する。

書式

```
#include <diosa.h>
int diosadbmulticonnect(t_diosa_dbuca *DbUca, long *Dstatus)
```

説明

DbUca で指定されたリソースグループセットに対応したインスタンスグループの全データベース・インスタンスに接続する。

t_diosa_dbuca *DbUca(入出力型)

　リソースグループ ID／リソースグループセット名を指定する。

long *Dstatus(出力)

　データベース接続処理の詳細ステータスを返却する。

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_ALREADY(1)	接続済み
DIOSA_ERROR(-1)	異常終了
DIOSA_EPARAM(-3)	パラメータエラー
DIOSA_EMEM(-6)	メモリ確保エラー
DIOSA_ELOCK(-14)	ロックエラー
DIOSA_EUNLOCK(-29)	アンロックエラー
DIOSA_EFATAL(-30)	全インスタンス接続失敗
DIOSA_EFUNCNAV(-34)	機能利用不可
DIOSA_EATTACH(-43)	アタッチエラー

注意

- 本関数を実行するためには、DB ヘルスチェック機能が動作していなければならない。
- 本関数は接続の多重度制御(高負荷時に接続処理を行わない制御)を行わず、即時に接続を試みる。
- マルチスレッドにおいてはスレッド単位に本関数を実行する必要がある。

関連

diosadbreconnect, diosadbchangeconnect, diosadbdisconnect

2.2.7　　diosadbreconnect(DB 接続関数[再接続])

名前

diosadbreconnect - 活性と判断されたデータベース・インスタンスに再接続する。

書式

```
#include <diosa.h>
int diosadbreconnect(long *Dstatus);
```

説明

DB ヘルスチェックの結果、データベース・インスタンスが活性であることが確認されているが、プロセスがデータベース・インスタンスに未接続である場合に再接続を行う。
本関数はトランザクションの最後に実行することを想定している。

long *Dstatus(出力)
　データベース接続処理の詳細ステータスを返却する。

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_ERROR(-1)	異常終了
DIOSA_EPARAM(-3)	パラメータエラー
DIOSA_ENOINIT(-11)	未初期化エラー
DIOSA_EFATAL(-30)	全ノード接続失敗
DIOSA_EFUNCNAV(-34)	機能利用不可

注意

- 本関数を実行するためには、DB ヘルスチェック機能が動作していなければならない。
- 本関数は接続の多重度制御(高負荷時に接続処理を行わない制御)を行い、同時に接続を行う最大数を制限する。(環境定義値)

関連

diosadbchangeconnect

2.2.8　diosagetacspinfo(アクセスポイント情報取得関数)

名前

diosagetacspinfo - アクセスポイントのプロトコル、端末情報を取得する

形式

```
#include <dios.h>
int diosagetacspinfo(char *AcsPoint, int Mode, void *AcsPInfo);
```

説明

char *AcsPoint (入力)

情報を取得するアクセスポイント名を指定する。

unsigned int Mode(入力)

取得する情報を選択する

DIOSA_GETACSP_PROTOCOL	プロトコル情報を取得する
DIOSA_GETACSP_SENDTERM	送信用端末名を取得する
DIOSA_GETACSP_RECVTERM	受信用端末名を取得する

取得する端末の状態を指定する。指定しない場合はすべての状態の端末を取得する。

DIOSA_GETACSP_OPEN	接続状態の端末を取得する
DIOSA_GETACSP_CLOSE	未接続状態の端末を取得する(障害状態を含む)

接続状態の送信用端末を取得する場合には、以下のように指定する。

Mode = DIOSA_GETACSP_SENDTERM | DIOSA_GETACSP_OPEN

void *AcsPInfo(入出力型)

アクセスポイント情報の返却する領域のアドレスを指定する。

Mode により以下の形式の領域を指定する。

DIOSA_GETACSP_PROTOCOL を指定した場合

```
t_diosa_protocol {
    char    Protocol[31+1];プロトコル名
    int     ConnectTiming; 接続タイミング
                           DIOSA_PATH_FULLTIME    常時接続
                           DIOSA_PATH_ONETIME     都度接続
    int     Direction;     送受信パス種別
                           常時接続のアクセスポイントの時のみ格納される。
                           DIOSA_PATH_COMBINE     1 つのパスで送受信を行う
                           DIOSA_PATH_SEPARATE    送受信を異なるパスで行う
    int     HeaderLength;  電文ヘッダ長
};
```

DIOSA_GETACSP_SENDTERM、DIOSA_GETACSP_RECVTERM を指定した場合

```
t_diosa_acsp_term {
    int     num;           返却する端末数
    char    term[32][31+1];端末名。
                           先頭から num で指定された数分、格納される。
};
```

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_EMPTY(15)	都度接続のアクセスポイントのため端末情報がない。
DIOSA_EPARAM(-3)	パラメータエラー
DIOSA_ENOENT(-8)	AcsPoint で指定されたアクセスポイントが見つからない
DIOSA_ENOINIT(-11)	未初期化エラー
DIOSA_ENOMATCH(-40)	指定された状態の端末がない。

注意

- アクセスポイントが 1 パスの場合、送信用端末名、受信用端末名は同じ値が返却される。

2.2.9　diosagetdbctx(DB コンテキスト取得関数)

名前

diosagetdbctx -データベースコンテキストを返却する。

書式

```
#include <diosa.h>
int diosagetdbctx(t_diosa_dbuca *DbUca, void **SqlCtx);
```

説明

データベースコンテキストを返却する。

t_diosa_dbuca *DbUca(入出力型)

　リソースグループ ID／リソースグループセット名を指定する。

void **SqlCtx(出力)

　データベースコンテキストを返却する。

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_ERROR(-1)	異常終了
DIOSA_EPARAM(-3)	パラメータエラー
DIOSA_ENOENT(-8)	カレントデータベースが未決定
DIOSA_ENOINIT(-11)	未初期化エラー
DIOSA_EFUNCNAV(-34)	機能利用不可

注意

- マルチコネクション接続の場合は、本関数を使用する前に、データベース接続先切り替え関数(**diosadbchangeconnect**)を実行して、使用するデータベース・インスタンスのコンテキストを決定する必要がある。（ただし、受信電文解析出口、または diosatxstart で指定した更新先リソースグループセットに対しては、DIOSA 内部でデータベース接続先切り替えを実行するため不要）

関連

diosadbchangeconnect

2.2.10　diosagetnodepathstatus(論理ノード間パス状態照会)

名前

diosagetnodepathstatus – 自 TPBASE の論理ノード間パス状態を照会する。

書式

```
#include <diosa.h>
int diosagetnodepathstatus(int Condition, t_diosa_nodepathstatus **NodePathStatus);
```

説明

int　　Condition(入力型)

照会する情報のレベルを以下から指定する。DIOSA_NODEPATH_TPMONITOR を指定する場合、パスを構成する端末が 1 つでも接続済の状態であれば、そのパスは接続済の状態と示される。

DIOSA_NODEPATH_TPMONITOR	相手 TP モニタ毎の論理ノード間パス状態
DIOSA_NODEPATH_TERMINAL	端末毎の接続状態

照会するパスの状態を指定する。指定しない場合は全ての状態のパスを照会する。

DIOSA_NODEPATH_OPEN	接続状態のパスを取得する
DIOSA_NODEPATH_CLOSE	未接続状態のパスを取得する(障害状態を含む)

照会する情報のレベル、パスの状態は論理和で組み合わせて指定できる。
例えば、接続状態の TP モニタ毎の情報を照会する場合には、以下のように指定する。

```
Condition = DIOSA_NODEPATH_TPMONITOR | DIOSA_NODEPATH_OPEN;
```

t_diosa_nodepathstatus　**NodePathStatus(入出力型)**

自ノードの論理ノード間パス状態を返却する。領域は本関数内で用意される。

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_ERROR(-1)	異常終了
DIOSA_EPARAM(-3)	パラメータエラー
DIOSA_EMEM(-6)	領域確保に失敗した
DIOSA_ENOINIT(-11)	未初期化エラー
DIOSA_EFUNCNAV(-34)	機能利用不可

注意

- 返却された領域は、本関数が再度実行された場合に無効になっている可能性があるため、最後に実行された本関数による返却領域だけを参照すること。

関連

t_diosa_nodepathstatus

2.2.11　diosagntcheck(保証電文二重処理検査関数)

2.2.12　diosagntcheck_mk(保証電文二重処理検査関数)

名前

diosagntcheck, diosagntcheckmk - 保証電文の二重処理検査を行う。

書式

```
#include <diosa.h>
int diosagntcheck(char *MsgKey, int *Status);
int diosagntcheck_mk(char *MainKey, char *MsgKey, int *Status);
```

説明

受信した保証電文が処理済みかどうかチェックする。
diosagntcheck()は更新先 DB が DB,IM の場合に使用する。
diosagntcheck_mk()は更新先 DB が DAC の場合に使用する。更新先 MAP を決定するメインキーを指定する。
更新先 DB は受信電文解析出口で指定する。

MsgKey　保証電文の一意な識別子を指定する(入力)
32 バイトの領域が確保されていること。
32 バイト以上の領域が確保されている場合、32 バイトまでが有効となる。
識別子は宛先論理システムによらず、論理システム内で一意にする必要がある。

論理システム内通信の電文保証の場合は、NULL を指定することができる。NULL を指定した場合、受信電文から自動的に解決される。
論理システム間通信の電文保証の場合は、指定必須である。

Status　エラー詳細 (出力)
NULL を指定することができる。

MainKey　更新先 DB が DAC の場合に、更新先 MAP を決定するためのメインキーを指定する。

戻り値

正常終了すると次の値を返却する。

DIOSA_DONE(0)　正常終了。

DIOSA_ALREADY(1)　二重受信を検出した。

DIOSA_IGNORE(9)　受信電文から保証電文識別子が解決できなかった。

異常終了すると次の値を返却する。

DIOSA_SWITCH(21)　計画マスタ切替中のため、処理継続不可。

DIOSA_ERROR(-1)　更新先 DB 情報の取得に失敗した。

DIOSA_ENOINIT(-11)　電文保証機能が起動していない。OLTP ノード以外で実行した。

DIOSA_EFUNCNAV(-34)　**diosagntcheck** の使用方法に誤りがある。
・ **diosagntcheck** が複数回実行された。または **diosagntfin** 実行後に **diosagntcheck** を実行した。
・ CO 制御 TPP 以外で実行した。

DIOSA_EDEADLOCK(-36) デッドロックエラーが発生した。

DIOSA_EDB(-69)　データベースアクセスエラー。**Status** にエラー詳細が設定される。

DIOSA_ETAM(-113)　インメモリキャッシュへのアクセスエラー。**Status** にエラー詳細が設定される。

DIOSA_ESTATE(-114)　更新先 DB がない。または RGSET がデフォルトインスタンスグループに属していない。データベースコンテキスト取得に失敗した。インメモリサーバアクセスしないメモリキャッシュの処理でエラーが発生した。メモリキャッシュ処理のエラーとデータベースコンテキスト取得失敗の場合、Status にエラー詳細が設定される。

DIOSA_ESWITCH(-115)　障害時マスタ切替中のため、処理継続不可。

DIOSA_EREADY(-118)　IMS サーバが起動中(再起動)や環境定義変更中により、アクセスするための準備ができていない。

注意

- 更新先 DB が IM の場合、本 API 呼び出し時点で更新先の MAPID が確定(受信電文解析出口で決定)している必要がある。
- OLTP 層の CO 制御上において、1 トランザクションで 1 回のみ使用可能。
- 電文保証対象のメッセージを受信した際に使用する。
- 同一の識別子で呼び出すときは、必ず前回と同一の更新先 DB を指定して CO を動作させること。

関連

diosagntfin,　diosarecvtx

2.2.13　diosagntdel(保証電文削除関数)

2.2.14　diosagntdel_mk(保証電文削除関数)

名前

diosagntdel, diosagntdel_mk - 保証電文をデータベースから削除する。

書式

```
#include <diosa.h>
int diosagntdel(char *MsgKey, int *Status);
int diosagntdel_mk(char *MainKey, char *MsgKey, int *Status);
```

説明

MsgKey に指定した識別子の保証電文について、処理が完了したことを通知し、再送処理対象から削除する。
論理システム間通信の保証電文に対する応答電文を受信した際に、再送を停止させるために呼び出す。また、再送リトライオーバー時に CO 呼び出しを指定した場合、呼び出された CO で該当保証電文に対して本 API を実行する必要がある。
diosagntdel()は更新先 DB が IM,DB の場合に使用する。
diosagntdel_mk()は更新先 DB が DAC の場合に使用する。更新先 MAP を決定するメインキーを指定する。
更新先 DB は受信電文解析出口で指定する。
順序性保証ありの保証電文の場合、同一順序性保証グループの電文が滞留していれば、本関数での削除を契機に次電文が送信される。

MsgKey　保証電文の識別子を指定する(入力)
リトライオーバーCO 上で論理システム内保証電文の再送を停止する際は NULL を指定することができる。NULL を指定した場合、受信電文から自動的に解決される。
論理システム間通信の電文保証の場合は、指定必須である。

Status　エラー詳細（出力）
NULL を指定することができる。

MainKey　更新先 DB が DAC の場合に、更新先 MAP を決定するためのメインキーを指定する。

戻り値

正常終了すると次の値を返却する。

DIOSA_DONE(0)　正常終了。

DIOSA_IGNORE(9)　受信電文から保証電文識別子が解決できなかった。

DIOSA_ALMOST(10)　電文削除は成功したが次電文送信はおこなわれていない。送信されなかった原因は、詳細ステータス(Status)に設定される。

異常終了すると次の値を返却する。

DIOSA_SWITCH(21)　計画マスタ切替中のため、処理継続不可。

DIOSA_ERROR(-1)　保証電文送信有無判定出口上での処理に失敗した。
または更新先 DB 情報の取得に失敗した。

DIOSA_EMEM(-6)　メモリアクセスエラー。

DIOSA_ENOINIT(-11)　電文保証機能が起動していない。OLTP ノード以外で実行した。

DIOSA_EFUNCNAV(-34)　**diosagntdel** の使用方法に誤りがある。
CO 制御 TPP 以外で実行した。

DIOSA_EDEADLOCK(-36) デッドロックエラーが発生した。

DIOSA_ECALLEXIT(-38) 保証電文送信有無判定出口の呼び出しに失敗した。

DIOSA_EDB(-69)　　データベースアクセスエラー。Status に SQL コードが設定される。

DIOSA_ETAM(-113)　　インメモリキャッシュへのアクセスエラー。Status にエラー詳細が設定される。

DIOSA_ESTATE(-114)　更新先 DB がない。または RGSET がデフォルトインスタンスグループに属していない。データベースコンテキスト取得に失敗した。インメモリサーバアクセスしないメモリキャッシュの処理でエラーが発生した。メモリキャッシュ処理のエラーとデータベースコンテキスト取得失敗の場合、Status にエラー詳細が設定される。

DIOSA_ESWITCH(-115)　障害時マスタ切替中のため、処理継続不可。

DIOSA_EREADY(-118)　IMS サーバが起動中(再起動)や環境定義変更中により、アクセスするための準備ができていない。

DIOSA_EMEM(-6)　　メモリアクセスエラー。

DIOSA_EINVAL(-9)　　電文保証機能の制御表が不整合な状態である。

＜詳細ステータス(DIOSA_ALMOST)＞

詳細ステータス	説明
DIOSA_ENOENT	保証電文の送信宛先へのパスが無い。
DIOSA_ESEND	保証電文の送信に失敗した。

注意

- 更新先 DB が IM の場合、本 API 呼び出し時点で更新先の MAPID が確定(受信電文解析出口で決定、または対象 MAPID に対して更新処理を実行)している必要がある。
- OLTP 層の CO 制御上で使用する。

関連

diosasendtx,　diosarecvtx

2.2.15　diosagntfin(保証電文処理確定関数)

名前

diosagntfin - 保証電文の送信元に処理完了を通知する。

書式

```
#include <diosa.h>
int diosagntfin(int *Status);
```

説明

受信した電文の業務処理が完了したことを、電文の送信元に通知する。
論理システム内の保証電文を受信し、その業務処理が完了した際に使用する。

Status	エラー詳細（出力） NULL を指定することができる。

戻り値

正常終了すると次の値を返却する。

DIOSA_DONE(0)	正常終了。
DIOSA_IGNORE(9)	受信電文が、DIOSA が生成した電文識別子の保証電文ではない。

異常終了すると次の値を返却する。

DIOSA_ENOINIT(-11)	電文保証機能が起動していない。OLTP ノード以外で実行した。
DIOSA_ESEND(-15)	到達完了の通知に失敗した。Status にエラー詳細が設定される。
DIOSA_EFUNCNAV(-34)	diosagntfin の使用方法誤り。 ・diosagntfin が複数回実行された。または diosagntcheck が未実行である。 ・CO 制御 TPP 以外で実行した。

＜詳細ステータス(DIOSA_ESEND)＞

上位 2 桁に TPBASE の TP_send が返却する STATUS KEY が返却される。

下位 2 桁に TPBASE の TP_send が返却する END KEY(※)が返却される。

(※)END KEY は 0～9 及び A～Z が返却されるため A～Z については、A は 10、B は 11...Z は 35 として扱う。

注意

- OLTP ノードの CO 制御上で使用する。

関連

diosagntcheck, diosarecvtx, diosasendtx, t_diosa_dcuca

2.2.16　diosagntshiftnextsend(保証電文送信タイミング変更関数)

2.2.17　diosagntshiftnextsend_mk(保証電文送信タイミング変更関数)

名前

diosagntshiftnextsend, diosagntshiftnextsend_mk - 保証電文の次回送信タイミングを変更する

書式

```
#include <diosa.h>
int diosagntshiftnextsend (char *MsgKey, int Time, int *Status);
int diosagntshiftnextsend_mk (char *MainKey, char *MsgKey, int Time, int *Status);
```

説明

保証電文の次回の送信タイミングを、本 API 実行時刻の **Time**(秒)後に変更する。次々回以降は従来の間隔で送信される。

diosagntshiftnextsend()は更新先 DB が IM,DB の場合に使用する。

diosagntshiftnextsend_mk()は更新先 DB が DAC の場合に使用する。更新先 MAP を決定するメインキーを指定する。

更新先 DB は受信電文解析出口で返却されたものを使用する。

MsgKey　変更対象保証電文の識別子を指定する(入力)
32 バイトの領域が確保されていること。
32 バイト以上の領域が確保されている場合、32 バイトまでが有効となる。

Time　次回の送信を何秒後に実施するかを 0～9999 秒の間で指定する(入力)
0 を指定した場合は即時送信を意味する。

Status　エラー詳細(出力)
NULL を指定することができる。

MainKey　更新先 DB が DAC の場合に、更新先 MAP を決定するためのメインキーを指定する。

戻り値

正常終了すると次の値を返却する。

DIOSA_DONE(0)　正常終了

異常終了すると次の値を返却する。

DIOSA_SWITCH(21)　計画マスタ切替中のため、処理継続不可。

DIOSA_ENOENT(-8)　指定された識別子の保証電文が見つからなかった。

DIOSA_ERROR(-1)　更新先 DB 情報の取得に失敗した。

DIOSA_EPARAM(-3)　パラメータエラー。

DIOSA_ENOINIT(-11)　電文保証機能が起動していない。OLTP ノード以外で実行した。

DIOSA_EFUNCNAV(-34)　CO 制御 TPP 以外で実行した。

DIOSA_EDEADLOCK(-35)　デッドロックを検出した。

DIOSA_EDB(-69)　データベースアクセスエラー。**Status** にエラー詳細が設定される。

DIOSA_ETAM(-113)　インメモリキャッシュへのアクセスエラー。**Status** にエラー詳細が設定される。

DIOSA_ESWITCH(-115)　障害時マスタ切替中のため、処理継続不可。

DIOSA_ESTATE(-114)　更新先 DB がない。または RGSET がデフォルトインスタンスグループに属していない。データベースコンテキスト取得に失敗した。インメモリサーバアクセスしないメモリキャッシュの処理でエラーが発生した。メモリキャッシュ処理のエラーとデータ

	ベースコンテキスト取得失敗の場合、Status にエラー詳細が設定される。
DIOSA_EREADY(-118)	サーバが起動中(再起動)や環境定義変更中により、アクセスするための準備ができていない。

注意

- OLTP ノードにおいて使用可能。
- 実際の送信タイミングは、最大で自動再送 TPP の呼び出し間隔(環境定義 APMGRNT 節-MSGGNT 項-RSNDCHKINVL)の分だけ、ずれる可能性がある。例えば自動再送 TPP 呼び出し間隔が 10 秒のとき、本 API で指定した次回送信タイミングが 1 秒後でも、再送 TPP が起き上がるタイミングによっては、最大で 10 秒後に電文を送信することが有り得る。
- 本 API 実行後からコミットするまでの間は、送信処理は行われない。

2.2.18　diosalspathctrl(L パス接続切断 API)

名前

diosalspathctrl － L パス(常時接続)の接続・切断を指示する。

形式

```
#include <diosa.h>
int diosalspathctrl(t_diosa_lspathctrl *LsPathCtrl)
```

説明

LsPathCtrl で指定されたアクセスポイントと接続・切断を行う。
1 論理ノードに複数の TPBASE が動作している場合、API を呼び出した CO 制御 TPP が動作する TPBASE が対象となる。

t_diosa_lspathctrl *LsPathCtrl;

接続・切断をするアクセスポイントの情報や操作を指定する領域を指定する。

戻り値

DIOSA_DONE(0)	正常終了
DIOSA_ALREADY(1)	LsPathCtrl-> Operation で指定された指示は実施済み
DIOSA_ALMOST(10)	一部の端末で成功
DIOSA_ERROR(-1)	TPBASE または内部エラー TPBASE のエラーの場合、LsPathCtrl->tu_fstatus に原因が設定される
DIOSA_EPARAM(-3)	パラメータエラー
DIOSA_ENOENT(-8)	AcsPoint で指定されたアクセスポイントが見つからない
DIOSA_ENOINIT(-11)	未初期化エラー
DIOSA_EBLOCK(-21)	AcsPoint で指定されたアクセスポイントは接続拒否状態
DIOSA_EFUNCNAV(-34)	TPP 以外のプロセスで実行された

関連

t_diosa_lspathctrl

2.2.19　t_diosa_dbuca(データベース関連 API 用構造体)

名前

t_diosa_dbuca - データベース関連 API で使用する構造体。

書式

```
#include <diosa.h>
t_diosa_dbuca DbUca
```

説明

データベース関連 API で使用する構造体。

short InputFlag(入力)

入力情報の種別を指定する。

DIOSA_DBCONNECT_RGID	リソースグループ ID 指定
DIOSA_DBCONNECT_RGSET	リソースグループセット指定
DIOSA_DBCONNECT_DEFAULTRGSET	デフォルトリソースグループセット指定
DIOSA_DBCONNECT_ALL	全リソースグループセット指定
DIOSA_DBCONNECT_UPDRGSET	受信電文解析出口、または diosatxstart で指定した更新先リソースグループセット指定

char RgSetName[32](入力)

リソースグループセット名を null で終了する文字列で指定する。
(**InputFlag** が DIOSA_DBCONNECT_RGSET の場合、指定必須)

short RgId(入力)

リソースグループ ID を指定する。
(**InputFlag** が DIOSA_DBCONNECT_RGID の場合、指定必須)

int Wait(入力)

データベース・インスタンスの状態がクラスタ再構成中(DIOSA_DBSTATUS_CLUSTERRECONFIG)の場合、本関数の中で待ち合わせを行うかどうかを指定する。

-1(DIOSA_DBNOTIMEOUT)	クラスタ再構成が完了するまで待ち合わせる。
0	待ち合わせを行わず、即時リターンする。
正の数値	指定された秒数待ち合わせる。

char DbStatus(出力)

データベース・インスタンスの状態を返却する。

DIOSA_DBSTATUS_DBACTIVE(0x00)	データベース・インスタンス使用可能
DIOSA_DBSTATUS_DBDOWN(0x01)	全データベース・インスタンス障害
DIOSA_DBSTATUS_CLUSTERRECONFIG(0x02)	クラスタ再構成中

int DbType(出力)

接続先 DB 種別を返却する。

DIOSA_DBTYPE_NONE	DB アクセスなし
DIOSA_DBTYPE_IM	インメモリサーバ
DIOSA_DBTYPE_ORACLE	Oracle
DIOSA_DBTYPE_POSTGRES	PostgreSQL
DIOSA_DBTYPE_DAC	DB アクセス制御で決定

データベース関連 API で指定が必要となる項目、および指定可能な項目について下表に示す。

項目	接続（シングル）	接続（マルチ）	切断	接続先切り替え	コンテキスト取得
InputFlag					
リソースグループ ID 指定	○	○	○	○	○
リソースグループセット指定	○	○	○	○	○
デフォルトリソースグループセット指定	○	○	○	○	○
全リソースグループセット指定	×	○	○	×	×
更新先リソースグループセット指定	×	×	×	×	○
RgSetName	○(*1)	○(*1)	○(*1)	○(*1)	○(*1)
RgId	○(*2)	○(*2)	○(*2)	○(*2)	○(*2)
Wait	－	－	－	○	－

○：指定項目(指定可)　×：指定不可　－：指定は無効

*1　InputFlag がリソースグループセット指定の場合、指定必須

*2　InputFlag がリソースグループ ID 指定の場合、指定必須

関連

diosadbchangeconnect, diosadbconnect, diosadbdisconnect, diosadbmulticonnect, diosagetdbctx

2.2.20　t_diosa_lspathctrl(L パス接続切断 API 用構造体)

名前

t_diosa_lspathctrl – L パス接続切断 API 用構造体

書式

```
#include <diosa.h>
t_diosa_lspathctrl LPathCtrl;
```

説明

接続・切断をするアクセスポイントの情報や操作を指定する領域を指定する。

int Operation (入力)

アクセスポイントに対する操作を指定する。

DIOSA_PATH_OPEN	アクセスポイントと接続する
DIOSA_PATH_CLOSE	アクセスポイントと切断する
DIOSA_PATH_ACCEPT	アクセスポイントとの接続を許可する
DIOSA_PATH_REJECT	アクセスポイントとの接続を拒否する。

char AcsPoint[15+1] (入力)

操作対象のアクセスポイント名を指定する。

char Term[31+1] (入力)

操作対象の端末名を指定する。
アクセスポイントに属するすべての端末を対象とする場合は空文字(NULL)を設定する。

int TermType (入力)

操作対象の端末種別を指定する。

DIOSA_TERM_ALL	すべての端末を対象とする。
DIOSA_TERM_SEND	送信用端末を対象とする
DIOSA_TERM_RECV	受信用端末を対象とする

Term で端末名が指定された場合、**TermType** は無視される。

char tu_fstatus[2] (出力)

操作結果の詳細を返却する。
値については「TPBASE プログラミングの手引き」の「7.10 TPP からのコマンド投入(TP_ctps)」を参照。

関連

diosalspathctrl

2.2.21　t_diosa_nodepathstatus(論理ノード間パス状態照会用構造体)

名前

t_diosa_nodepathstatus - 論理ノード間パス状態照会関数で使用する構造体。

書式

```
#include <diosa.h>
t_diosa_nodepathstatus NodePathStatus;
```

説明

論理ノード間パス状態照会関数で使用する構造体。

int　　InfoNum(出力)

返却されたパス状態情報の数。

t_diosa_nodepathinfo　　Info[](出力)

接続先とのパス状態情報を返却する。

int　LNodeType

接続先の論理ノードの種別を返却する。
DIOSA_LNODETYPE_AP　　AP ノード
DIOSA_LNODETYPE_OLTP　OLTP ノード

int　LNodeId

接続先の論理ノードの番号を返却する。

char　LNodeName[16]

接続先の論理ノード名を返却する。

int　TpMonitorId

接続先 TPBASE の TP モニタ番号を返却する。

char　TpMonitor[9]

接続先 TPBASE の TP モニタ名を返却する。

char　TermName[32]

接続先 TPBASE のリスナに対応する端末名を返却する。

short　Status

接続先とのパス状態を返却する。
DIOSA_NODEPATH_OPEN　　接続済
DIOSA_NODEPATH_CLOSE　未接続
DIOSA_NODEPATH_BLOCK　接続先の論理ノードが閉塞されている

注意

- 論理ノード間パス状態照会関数で、照会する情報のレベルに DIOSA_TPMONITOR を指定した場合、返却される端末名の領域は無効である。

関連

diosagetnodepathstatus

2.3　メモリキャッシュ

2.3.1　　関数一覧

(1)　　インメモリサーバ機能

MAP 単位に分散されたメモリテーブル(以下、表と記す)へのアクセス機能を提供する。

本アクセス機能は、利用者が MAP を意識してアクセスする必要がある。このため、事前に diosaimsetmap() でアクセス対象とする MAP を宣言してから、アクセス要求 API を実行する。但し、利用者は、MAP がどのノードに配置されているかについて意識する必要はない。

また、本アクセス機能では、diosaimtxstart() 実行後から diosaimcommit()、または diosaimrollback() 実行までを 1 トランザクションと定義し、このトランザクション区間でのみ、更新系のアクセス要求 API(※)を実行することができる。

なお、本アクセス機能は、CO 制御、バッチ AP 制御、それ以外の環境で使用可能である。但し、CO 制御とバッチ AP 制御以外の環境で本アクセス機能を使用する場合は、diosaprcinit()、または diosathrinit() を実行後、diosaimopen() でインメモリサーバ(以下、IM サーバと記す)との接続を確立し、diosatrninit() を実行してから各 API を実行する必要がある。

(※)更新系のアクセス要求 API とは、diosaimwrite()、diosaimrewrite()、diosaimdelete()、diosaimdeletex()、diosaimtruncate()、DIOSA_LOCK_WAIT、DIOSA_LOCK_NOWAIT のいずれかを指定した diosaimread1() である。

接続処理

diosaimopen	インメモリサーバとの接続を確立する。
diosaimclose	インメモリサーバとの接続を切断する。

アクセス要求

diosaimread1	１レコードを取得する(主キーの完全一致)。
diosaimread	複数レコードを取得する。
diosaimwrite	レコードを挿入する。
diosaimrewrite	レコードを更新する。
diosaimdelete	レコードを削除する(レコード情報で指定) 。
diosaimdeletex1	レコードを削除する(主キーの完全一致)。
diosaimtruncate	全レコードを削除する。

トランザクション制御

diosaimtxstart	トランザクションの開始を宣言する。
diosaimcommit	更新要求を確定する。
diosaimrollback	更新要求をロールバックする。

アクセス先 MAP

diosaimsetmap	アクセス先の MAP を設定する。
diosaimgetmap	設定されている MAPID を取得する。

検索条件

t_diosa_imcond	検索条件構造体

diosaimcondsetkey	検索条件の構築を行う(キー指定)。
diosaimcondsetrange	検索条件の構築を行う(範囲指定)。

検索コンテキスト

diosaimctxopen	検索コンテキストを生成する。
diosaimctxclose	検索コンテキストを破棄する。

レコード情報

t_diosa_recinfo	レコード情報構造体

照会

t_diosa_imrefctx	照会コンテキスト構造体
t_diosa_imreckeyinfo	レコードキー情報構造体
t_diosa_imtbllist	論理表一覧構造体
diosaimgetmaplist	分散先の MAP 一覧を照会する。
diosaimgetptblname	物理表名を照会する。
diosaimgetreckeyinfo	レコードのキー情報を照会する。
diosaimgetsvcstatus	サービス状態を紹介する。
diosaimgettblid	論理表 ID を照会する。
diosaimgettbllist	論理表名、論理表 ID の一覧を照会する。

性能情報

t_diosa_imperfuca	性能情報構造体
diosaimgetperf	性能情報を利用者に返却する。

(2) インメモリサーバ所在管理

diosagethash	メインキーに対応するハッシュ値取得関数
diosagetmaphash	MAP のハッシュ値範囲一覧取得関数
diosagetmap	MAP 情報取得関数
diosagetmaplist	MAP 一覧取得関数
diosagetmapstat	MAP のレプリケーション状態取得関数
diosagettamnode	TAM のノード配置情報取得関数
diosatamswitch	マスタ昇格関数
t_diosa_getmapuca	MAP 情報取得関数インタフェース構造体
t_diosa_getmapstatuca	MAP のレプリケーション状態取得関数インタフェース構造体
t_diosa_mapent	MAP 情報エントリ構造体
t_diosa_maphashent	MAP のハッシュ値範囲情報エントリ構造体
t_diosa_tamnodeent	TAM のノード配置情報エントリ構造体

2.3.2　　diosagethash(メインキーに対応するハッシュ値取得関数)

名前

diosagethash - メインキーに対応するハッシュ値取得関数

書式

```
#include <diosa.h>
int    diosagethash( char *MainKey, unsigned int *HashValue );
```

説明

diosagethash()は、指定されたメインキーに対応するハッシュ値を返却する。

MainKey	メインキーを格納した領域のポインタを指定する。
HashValue	ハッシュ値が返却される。

戻り値

成功した場合には、0 が返される。エラー時は、負の値が返される。

DIOSA_DONE(0)	正常終了。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータエラー。
DIOSA_EABORT(-4)	ハッシュ関数から異常終了要求が返却された。
DIOSA_ENOINIT(-11)	プロセス初期化処理が行われていない。 (環境変数 DIOSA_IIC_HASHEXIT が未指定)
DIOSA_ECALLEXIT(-38)	ハッシュ関数呼び出しに失敗した。

注意

- CO 制御、バッチ AP 制御以外の環境で diosagethash()を実行する場合は、diosaprcinit()、diosathrinit()を呼び出した後に実行する必要がある。
- メモリキャッシュ未起動の場合、環境変数 DIOSA_IIC_HASHEXIT にハッシュ関数名を指定することで、本 API を利用できる。

関連

diosaprcinit, diosaprcterm, diosathrinit, diosathrterm

2.3.3　　diosagetmap(MAP 情報取得関数)

名前

diosagetmap - MAP 情報取得関数

書式

```
#include <diosa.h>
int    diosagetmap( t_diosa_getmapuca *GetMapUca );
```

説明

diosagetmap()は、**GetMapUca** に指定された情報を元に該当する MAP を検索し、その MAP の状態を **GetMapUca** に返却する。

GetMapUca	MAP 情報取得関数インタフェース構造体のポインタを指定する。

戻り値

成功した場合には、0 が返される。エラー時は、負の値が返される。

DIOSA_DONE(0)	正常終了。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータエラー。
DIOSA_EABORT(-4)	ハッシュ関数から異常終了要求が返却された。
DIOSA_ENOENT(-8)	指定された MAP が存在しない。 または、指定されたメインキーに対応するハッシュ値が定義されていない。
DIOSA_ENOINIT(-11)	プロセス初期化処理が行われていない。
DIOSA_ESIZE(-33)	**GetMapUca** の UserAreaLen が利用者領域データ長より小さい。
DIOSA_ECALLEXIT(-38)	ハッシュ関数呼び出しに失敗した。

注意

- CO 制御、バッチ AP 制御以外の環境で diosagetmap ()を実行する場合は、diosaprcinit()、diosathrinit()を呼び出した後に実行する必要がある。

関連

diosaprcinit, diosaprcterm, diosathrinit, diosathrterm, t_diosa_getmapuca

2.3.4　　diosagetmaphash(MAP のハッシュ値範囲一覧取得関数)

名前

diosagetmaphash - MAP のハッシュ値範囲一覧取得関数

書式

```
#include <diosa.h>
int    diosagetmaphash( int *MapNum, t_diosa_mapent **MapEnt );
```

説明

diosagetmaphash()は、MAP 一覧を格納した領域を **MapEnt** に返却する。また、MAP に対応するハッシュ値範囲の情報を **MapEnt** 内の HashEnt に返却する。

MapNum	MAP 数が返却される。
MapEnt	MAP 情報エントリのポインタが返却される。

戻り値

成功した場合には、0 が返される。エラー時は、負の値が返される。

DIOSA_DONE(0)	正常終了。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータエラー。
DIOSA_EMEM(-6)	メモリ確保エラー。
DIOSA_ENOINIT(-11)	プロセス初期化処理が行われていない。

注意

- CO 制御、バッチ AP 制御以外の環境で diosagetmaphash()を実行する場合は、diosaprcinit()、diosathrinit()を呼び出した後に実行する必要がある。
- MapEnt の領域は、関数内部で確保し、ポインタを返却する。同一スレッド内で再実行された場合、同一領域が再利用される。当領域は利用者側で解放してはならない。

関連

diosaprcinit, diosaprcterm, diosathrinit, diosathrterm, t_diosa_maphashent, t_diosa_mapent

2.3.5　　diosagetmaplist(MAP 一覧取得関数)

名前

diosagetmaplist - MAP 一覧取得関数

書式

```
#include <diosa.h>
int    diosagetmaplist( int RepGrpId, int *MapNum, t_diosa_mapent **MapEnt );
```

説明

diosagetmaplist()は、MAP 一覧を格納した領域を **MapEnt** に返却し、**MapEnt** に返却した MAP 数を **MapNum** に返却する。

RepGrpId	検索対象のレプリケーショングループ ID を指定する。 全レプリケーショングループを対象とする場合は、0 を指定する。
MapNum	MAP 数が返却される。
MapEnt	MAP 情報エントリのポインタが返却される。

戻り値

成功した場合には、0 が返される。エラー時は、負の値が返される。

DIOSA_DONE(0)	正常終了。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータエラー。
DIOSA_EMEM(-6)	メモリ確保エラー。
DIOSA_ENOENT(-8)	指定されたレプリケーショングループが存在しない。
DIOSA_ENOINIT(-11)	プロセス初期化処理が行われていない。

注意

- CO 制御、バッチ AP 制御以外の環境で diosagetmaplist()を実行する場合は、diosaprcinit()、diosathrinit()を呼び出した後に実行する必要がある。
- MapEnt の領域は、関数内部で確保し、ポインタを返却する。同一スレッド内で再実行された場合、同一領域が再利用される。当領域は利用者側で解放してはならない。
- 全 MAP 指定の場合、同一レプリケーショングループの MAP は連続して返却される。

関連

diosaprcinit, diosaprcterm, diosathrinit, diosathrterm, t_diosa_mapent

2.3.6　　diosagetmapstat(MAP のレプリケーション状態取得関数)

名前

diosagetmapstat - MAP のレプリケーション状態取得関数

書式

```
#include <diosa.h>
int    diosagetmapstat( t_diosa_getmapstatuca *GetMapStatUca );
```

説明

diosagetmapstat()は、**GetMapStatUca** に指定された情報を元に該当する MAP を検索し、その MAP のレプリケーション状態とマスタ／スレーブ種別を **GetMapStatUca** に返却する。

GetMapStatUca　　MAP のレプリケーション状態取得関数インタフェース構造体のポインタを指定する。

戻り値

成功した場合には、0 が返される。エラー時は、負の値が返される。

DIOSA_DONE(0)	正常終了。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータエラー。
DIOSA_EABORT(-4)	ハッシュ関数から異常終了要求が返却された。
DIOSA_ENOENT(-8)	指定された MAP が存在しない。 または、指定されたメインキーに対応するハッシュ値が定義されていない。
DIOSA_ENOINIT(-11)	プロセス初期化処理が行われていない。
DIOSA_ECALLEXIT(-38)	ハッシュ関数呼び出しに失敗した。

注意

- CO 制御、バッチ AP 制御以外の環境で diosagetmapstat()を実行する場合は、diosaprcinit()、diosathrinit()を呼び出した後に実行する必要がある。

関連

diosaprcinit, diosaprcterm, diosathrinit, diosathrterm, t_diosa_getmapstatuca

2.3.7　　diosagettamnode(TAM のノード配置情報取得関数)

名前

diosagettamnode - TAM のノード配置情報取得関数

書式

```
#include <diosa.h>
int    diosagettamnode( int RepGrpId, int *NodeNum, t_diosa_tamnodeent **NodeEnt );
```

説明

diosagettamnode()は、**RepGrpId** に指定したレプリケーショングループ ID のマスタ TAM またはスレーブ TAM が存在するノードの一覧を **NodeEnt** に返却し、**NodeEnt** に返却したノード数を **NodeNum** に返却する。

RepGrpId	検索対象のレプリケーショングループ ID を指定する。
NodeNum	ノード数が返却される。
NodeEnt	TAM のノード配置情報エントリのポインタが返却される。

戻り値

成功した場合には、0 が返される。エラー時は、負の値が返される。

DIOSA_DONE(0)	正常終了。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータエラー。
DIOSA_EMEM(-6)	メモリ確保エラー。
DIOSA_ENOENT(-8)	指定されたレプリケーショングループが存在しない。
DIOSA_ENOINIT(-11)	プロセス初期化処理が行われていない。

注意

- CO 制御、バッチ AP 制御以外の環境で diosagettamnode()を実行する場合は、diosaprcinit()、diosathrinit()を呼び出した後に実行する必要がある。
- NodeEnt の領域は、関数内部で確保し、ポインタを返却する。同一スレッド内で再実行された場合、同一領域が再利用される。当領域は利用者側で解放してはならない。

関連

diosaprcinit, diosaprcterm, diosathrinit, diosathrterm, t_diosa_tamnodeent

2.3.8　diosaimclose(IM サーバクローズ関数)

名前

diosaimclose - IM サーバクローズ

書式

```
#include <diosa.h>
int diosaimclose(void)
```

説明

diosaimclose() は、IM サーバとの通信路の切断処理を行う。

戻り値

DIOSA_DONE(0)	正常終了した。
DIOSA_ERROR(-1)	その他エラーが発生した。
DIOSA_ESYS(-2)	システムコールエラーが発生した。
DIOSA_EMEM(-6)	メモリ操作に失敗した。
DIOSA_EMSGQ(-41)	メッセージキューの障害が発生した。
DIOSA_ESOCKET(-70)	ソケット送受信で障害が発生した。
DIOSA_ESTATE(-114)	**diosaprcinit()**、または **diosathrinit()** が未実施である。

関連

diosaimopen()

2.3.9　diosaimcommit(コミット関数)

名前

diosaimcommit - 更新要求の確定

書式

```
#include <diosa.h>
int diosaimcommit(void)
```

説明

diosaimcommit()は、**diosaimtxstart()**以降に実行された更新要求を確定し、ロックしていたレコードのロックを解除する。
更新系のアクセス要求 API を実行していない場合でも、**diosaimtxstart()**を実行した後は、**diosaimcommit()**、または**diosaimrollback()** を実行する必要がある。

戻り値

DIOSA_DONE(0)	正常終了した。
DIOSA_BLOCK(6)	該当 MAP が閉塞している。
DIOSA_SWITCH(21)	計画マスタ切替中である。
DIOSA_ERROR(-1)	その他エラーが発生した。
DIOSA_ESYS(-2)	システムコールエラーが発生した。
DIOSA_ETIMEOUT(-22)	要求がタイムアウトした。
DIOSA_EOVERFLOW(-39)	IMS キューバッファに空きがない。
DIOSA_EMSGQ(-41)	メッセージキューの障害が発生した。
DIOSA_ESOCKET(-70)	ソケット送受信で障害が発生した。
DIOSA_ECONNECT(-71)	通信パスが切断された。
DIOSA_ETAM(-113)	IM の障害により更新確定が失敗した。
DIOSA_ESTATE(-114)	**diosaimtxstart()** が未実施である。
DIOSA_ESWITCH(-115)	障害時マスタ切替中である。
DIOSA_EINACTIVE(-117)	レプリケーショングループが停止している。
DIOSA_EREADY(-118)	サーバが起動中(再起動)や環境定義変更中により、アクセスするための準備ができていない。

注意

- 異常終了(負値)が返却された場合は、必ず **diosaimrollback()**を実行すること。
- 異常終了(負値)が返却された場合でも、更新要求が確定している可能性がある。

関連

diosaimtxstart(), diosaimrollback()

2.3.10　diosaimcondsetkey(キー指定検索条件設定関数)

名前

diosaimcondsetkey - 検索条件の構築(キー指定)

書式

```
#include <diosa.h>
int diosaimcondsetkey(t_diosa_imcond* Cond, char* Value, int ValueLen)
```

説明

diosaimcondsetkey() は、キー指定でのレコード取得、またはレコード削除を行う時に、検索キーを設定するための関数である。
diosaimcondsetkey() は、**Value**、**ValueLen** により構築した検索条件を、**Cond** に設定する。
Value には検索キー、**ValueLen** には検索キー長を指定する。
但し、完全一致検索を行う場合は **ValueLen** に環境定義に指定したキー長と同じ長さを指定し、前方一致検索を行う場合は **ValueLen** に一致させたいところまでのキー長を指定する。

戻り値

DIOSA_DONE(0)	正常終了した。
DIOSA_ERROR(-1)	その他エラーが発生した。
DIOSA_EPARAM(-3)	パラメータが不正である。
DIOSA_ESTATE(-114)	**diosaprcinit()**, **diosathrinit()**(マルチスレッドの場合), **diosaimopen()**が未実施である。

注意

- 前方一致検索は、**diosaimread()**のみで行うことができる。

関連

diosaimread1(), diosaimread(), diosaimdeletex1, diosaimctxopen(), t_diosa_imcond

2.3.11　diosaimcondsetrange(範囲指定検索条件設定関数)

名前

diosaimcondsetrange - 検索条件の構築(範囲指定)

書式

```
#include <diosa.h>
int diosaimcondsetrange(t_diosa_imcond* Cond, int Operator1, char* MinValue, int MinValueLen,
int Operator2, char* MaxValue, int MaxValueLen)
```

説明

diosaimcondsetrange() は、範囲指定による検索でレコード取得を行う時に、検索条件を設定するための関数である。

diosaimcondsetrange() は、**Operator1**、**MinValue**、**MinValueLen**、**Operator2**、**MaxValue**、**MaxValueLen** により検索条件を構築して、**Cond** に設定する。**Operator1**、**MinValue**、**MinValueLen** には下限値を指定し、**Operator2**、**MaxValue**、**MaxValueLen** には上限値を指定する。

Operator1 には以下の演算子を指定できる。

DIOSA_COND_GT	>
DIOSA_COND_GE	>=

Operator2 には、以下の演算子を指定できる。

DIOSA_COND_LT	<
DIOSA_COND_LE	<=

MinValue、**MaxValue** には検索キーの値、**MinValueLen**、**MaxValueLen** には検索キーの長さを指定する。この検索キーの長さは、環境定義に指定したキー長と同じ長さである必要がある。

下限のみ(または、上限のみ)の検索条件を構築する場合は、**Operator2**(または、**Operator1**)に DIOSA_COND_NOP を指定する。

戻り値

DIOSA_DONE(0)	正常終了した。
DIOSA_ERROR(-1)	その他エラーが発生した。
DIOSA_EPARAM(-3)	パラメータが不正である。
DIOSA_ESTATE(-114)	**diosaprcinit()**, **diosathrinit()**(マルチスレッドの場合), **diosaimopen()**が未実施である。

関連

diosaimread(), diosaimctxopen(), t_diosa_imcond

2.3.12　diosaimctxclose(検索コンテキストクローズ関数)

名前

diosaimctxclose - 検索コンテキストの破棄

書式

```
#include <diosa.h>
int diosaimctxclose(int CtxId)
```

説明

diosaimctxclose() は **diosaimctxopen()** で生成された **CtxId** を無効の状態にする。

戻り値

DIOSA_DONE(0)	正常終了した。
DIOSA_ERROR(-1)	その他エラーが発生した。
DIOSA_EPARAM(-3)	パラメータが不正である。
DIOSA_ESTATE(-114)	**CtxId** で指定された検索コンテキストが **diosaimctxopen()** で生成されていない。

関連

diosaimctxopen()

2.3.13　diosaimctxopen(検索コンテキストオープン関数)

名前

diosaimctxopen - 検索コンテキストの生成

書式

#include <diosa.h>

int diosaimctxopen(int TableId, char* IndexName, t_diosa_imcond* Cond, int Order, int Lock, int *CtxId)

説明

diosaimctxopen() は、**diosaimread()** によるレコード取得を行う時に、検索コンテキストを生成するための関数である。

diosaimctxopen() は、現在設定されている MAP と、パラメータで指定された情報(検索に使用する)により、検索コンテキストを生成し、検索コンテキストの識別子を返却する。

TableId には検索対象の論理表 ID、**IndexName** には検索対象のインデックス名を指定する。

検索条件 **Cond** には **diosaimcondsetkey()**、または **diosaimcondsetrange()** を使って構築されたものを指定する。**IndexName** と **Cond** に NULL を指定することも可能であり、その場合は、主キー(以下、プライマリキーと記載)を対象とした全件検索となる。

Order には検索方向として、**DIOSA_ASCEND**(昇順)を指定する。

Lock には排他オプションとして、**DIOSA_NOLOCK**(排他なし)を指定する。

なお、不要になった検索コンテキストは、**diosaimctxclose()** を実行して破棄する。

戻り値

DIOSA_DONE(0)	正常終了した。
DIOSA_ERROR(-1)	その他エラーが発生した。
DIOSA_EPARAM(-3)	パラメータが不正である。
DIOSA_EMEM(-6)	メモリ操作に失敗した。
DIOSA_EINVAL(-9)	対象の表が存在しない。
DIOSA_EFUNCNAV(-34)	未サポート機能を指定した。
DIOSA_ESTATE(-114)	検索条件 **Cond** が **diosaimcondsetkey()**、**diosaimcondsetrange()** のいずれかで構築されていない。もしくは、**diosaimsetmap()** で検索対象の MAP が設定されていない。

注意

- **diosaimctxopen()**実行前に **diosaimsetmap()**で検索対象の MAP を設定しておく必要がある。

関連

diosaimsetmap(), diosaimread(), diosaimctxclose(), diosaimcondsetkey(), diosaimcondsetrange(), diosaimgettblid(), t_diosa_imcond

2.3.14　diosaimdelete(レコード削除関数)

名前

diosaimdelete - レコードの削除(レコード情報指定)

書式

```
#include <diosa.h>
int diosaimdelete(int TableId, t_diosa_recinfo* RecInfo, int RecInfoNum)
```

説明

diosaimdelete() は、**RecInfo** に指定されたレコードを削除する。
削除対象となる表は、**TableId** に指定された論理表 ID と **diosaimsetmap()** により設定された MAP から決定する。
RecInfo には、排他オプションに **DIOSA_LOCK_WAIT**、**DIOSA_LOCK_NOWAIT** のいずれかを指定して **diosaimread1()** で取得したレコード情報を指定する。
RecInfoNum には、1 を指定する。2 以上を指定した場合は **DIOSA_EFUNCNAV**、0 以下を指定した場合は **DIOSA_EPARAM** が返却される。
diosaimdelete() は、**diosaimtxstart()** と **diosaimcommit()**、または **diosaimrollback()** の区間内でのみ、実行できる。

戻り値

DIOSA_DONE(0)	正常終了した。
DIOSA_BLOCK(6)	該当 MAP が閉塞している。
DIOSA_NOENT(20)	該当レコードが削除対象の表に存在しなかった。
DIOSA_SWITCH(21)	計画マスタ切替中である。
DIOSA_ERROR(-1)	その他エラーが発生した。
DIOSA_ESYS(-2)	システムコールエラーが発生した。
DIOSA_EPARAM(-3)	パラメータが不正である。
DIOSA_EINVAL(-9)	削除対象の表が存在しない。
DIOSA_ETIMEOUT(-22)	要求がタイムアウトした。
DIOSA_EACCES(-25)	アクセス対象の表がオープンされていない。もしくは、更新ログ出力機能が準備できてないため、更新アクセスできない。
DIOSA_ESIZE(-33)	レコードサイズが不正である。
DIOSA_EFUNCNAV(-34)	未サポート機能を指定した。
DIOSA_EOVERFLOW(-39)	IMS キューバッファに空きがない。
DIOSA_EMSGQ(-41)	メッセージキューの障害が発生した。
DIOSA_ECOND(-60)	別 MAP への更新系のアクセス要求 API が実行されている。もしくは、同一トランザクション内で **diosaimtruncate()** が実行されている。
DIOSA_ESOCKET(-70)	ソケット送受信で障害が発生した。
DIOSA_ECONNECT(-71)	通信パスが切断された。
DIOSA_ETAM(-113)	IM の障害によりレコード削除に失敗した。
DIOSA_ESTATE(-114)	**diosaimtxstart()**、または **diosaimsetmap()** が未実施である。もしくは、レコードのロックを取得せずに実行した。
DIOSA_ESWITCH(-115)	障害時マスタ切替中である。

DIOSA_EINACTIVE(-117)　レプリケーショングループが停止している。

DIOSA_EREADY(-118)　サーバが起動中(再起動)や環境定義変更中により、アクセスするための準備ができていない。

注意

- DIOSA_SWITCH などのサービス抑止となる戻り値が返却された場合、diosaimrollback()、diosatrnterm()実行後、diosatrninit()から実行し直す必要がある。サービス抑止となる戻り値については、メモリキャッシュ利用の手引きの「3.1.8 戻り値について」を参照。

関連

diosaimsetmap(), diosaimgettblid(), diosaimread1(), diosaimtxstart, t_diosa_recinfo

2.3.15　diosaimdeletex1(キー指定レコード削除関数)

名前

diosaimdeletex1 - １レコードの削除(主キー指定)

書式

```
#include <diosa.h>
int diosaimdeletex1(int TableId, t_diosa_imcond *Cond, int Lock)
```

説明

diosaimdeletex1() は、**Cond** に指定されたキー値と一致するレコードを削除する。
Cond には、**diosaimcondsetkey()** を使用してプライマリキーを設定したものを指定する。
削除対象となる表は、**TableId** に指定された論理表 ID と **diosaimsetmap()** により設定された MAP から決定する。
Lock には、排他オプションとして以下の値のいずれかを指定する。

DIOSA_LOCK_WAIT	他トランザクションで既に削除対象レコードがロックされている場合は、ロックが解放されるまで待ち合わせる。但し、応答監視タイマ(環境定義 IMENV 節-USERAP 項-REQTIMEOUT で定義したもの)の時間を経過してもロックが解放されない場合は、**DIOSA_ETIMEOUT** が返却される。
DIOSA_LOCK_NOWAIT	他トランザクションで既に削除対象レコードがロックされている場合は、待ち合わせをせずにエラーとなる。

diosaimdeletex1() は、**diosaimtxstart()** と **diosaimcommit()**、または **diosaimrollback()** の区間内でのみ、実行できる。

戻り値

DIOSA_DONE(0)	正常終了した。
DIOSA_BLOCK(6)	該当 MAP が閉塞している。
DIOSA_NOENT(20)	該当レコードが削除対象の表に存在しなかった。
DIOSA_SWITCH(21)	計画マスタ切替中である。
DIOSA_ERROR(-1)	その他エラーが発生した。
DIOSA_ESYS(-2)	システムコールエラーが発生した。
DIOSA_EPARAM(-3)	パラメータが不正である。
DIOSA_EMEM(-6)	メモリ操作に失敗した。
DIOSA_EINVAL(-9)	削除対象の表が存在しない。
DIOSA_ELOCK(-14)	ロック取得に対しロックリストオーバーフローが発生した。
DIOSA_ETIMEOUT(-22)	要求がタイムアウトした。
DIOSA_EACCES(-25)	アクセス対象の表がオープンされていない。もしくは、更新ログ出力機能が準備できてないため、更新アクセスできない。
DIOSA_EBUSY(-31)	他の利用者により既に該当レコードがロックされている。**Lock** に **DIOSA_LOCK_NOWAIT** が指定されていた場合にのみ発生する。
DIOSA_EDEADLOCK(-36)	デッドロックが発生した。**Lock** に **DIOSA_LOCK_WAIT** を指定した場合のみ発生する。
DIOSA_EOVERFLOW(-39)	IMS キューバッファに空きがない。
DIOSA_EMSGQ(-41)	メッセージキューの障害が発生した。

DIOSA_ECOND(-60)	別 MAP への更新系のアクセス要求 API が実行されている。もしくは、同一トランザクション内で **diosaimtruncate()** が実行されている。
DIOSA_ESOCKET(-70)	ソケット送受信で障害が発生した。
DIOSA_ECONNECT(-71)	通信パスが切断された。
DIOSA_ETAM(-113)	IM の障害によりレコード削除に失敗した。
DIOSA_ESTATE(-114)	**diosaimtxstart()**、または **diosaimsetmap()** が未実施である。
DIOSA_ESWITCH(-115)	障害時マスタ切替中である。
DIOSA_EINACTIVE(-117)	レプリケーショングループが停止している。
DIOSA_EREADY(-118)	サーバが起動中(再起動)や環境定義変更中により、アクセスするための準備ができていない。
DIOSA_EEXTEND(-123)	管理テーブルの拡張に失敗した。

注意

- **DIOSA_SWITCH** などのサービス抑止となる戻り値が返却された場合、**diosaimrollback()**、**diosatrnterm()** 実行後、**diosatrninit()**から実行し直す必要がある。サービス抑止となる戻り値については、メモリキャッシュ利用の手引きの「3.1.8 戻り値について」を参照。

関連

diosaimsetmap(), diosaimgettblid(), diosaimtxstart()

2.3.16　diosaimgetmap(アクセス先 MAP 取得関数)

名前

diosaimgetmap – 設定されている MAPID の取得

書式

```
#include <diosa.h>
int diosaimgetmap(int* MAPId)
```

説明

diosaimgetmap() は、**diosaimsetmap()** で設定された、現在のアクセス先となっている MAPID を **MAPId** に返却する。

戻り値

DIOSA_DONE(0)	正常終了した。
DIOSA_ERROR(-1)	その他エラーが発生した。
DIOSA_EPARAM(-3)	パラメータが不正である。
DIOSA_ENOENT(-8)	MAP が設定されていない。
DIOSA_ESTATE(-114)	**diosaimopen()**、または **diosatrninit()** が未実施である。

関連

diosaimsetmap()

2.3.17　diosaimgetmaplist(分散先 MAP 一覧照会関数)

名前

diosaimgetmaplist - 論理表の分散先 MAP 一覧照会

書式

#include <diosa.h>

int diosaimgetmaplist(t_diosa_imrefctx* Refctx, char* LTableName, int ReqMapNum, int* MapNum, int* MAPId)

説明

diosaimgetmaplist() は、指定した論理表の分散先となる MAP の一覧を返却する。

LTableName には照会対象の論理表名、**MAPId** には分散先となる MAPID の返却領域として int 配列の先頭アドレス、**ReqMapNum** には **MAPId** に指定した int 配列の個数を指定する。

MAPId に分散先となる MAPID、**MapNum** に MAPID の個数が返却される。

但し、**ReqMapNum** に指定された個数より、分散先となる MAP の個数が多い場合は、**diosaimgetmaplist()** を続けて実行することで、前回の続きから MAPID を取得することができる。

初回実行時は、**Refctx** の各メンバに負値を指定して **diosaimgetmaplist()** を実行し、前回の続きから取得する場合は、前回返却した **Refctx** の値をそのまま指定して **diosaimgetmaplist()** を実行する。

なお、分散先となる全ての MAPID を返却した場合は、**MapNum** に 0、戻り値に **DIOSA_DATALIM** が返却される。

戻り値

DIOSA_DONE(0)	分散先となる MAPID を取得した。
DIOSA_DATALIM(4)	分散先となる MAPID を全て取得し終わった。
DIOSA_ERROR(-1)	その他エラーが発生した。
DIOSA_EPARAM(-3)	パラメータが不正である。
DIOSA_EINVAL(-9)	指定された論理表名が存在しない。
DIOSA_ESTATE(-114)	**diosaprcinit()**、または **diosathrinit()** が未実施である。

関連

t_diosa_imrefctx

2.3.18　diosaimgetperf(API 性能情報取得関数)

名前

diosaimgetperf - 性能情報の取得

書式

```
#include <diosa.h>
int diosaimgetperf(t_diosa_imperfuca* PerfUca)
```

説明

diosaimgetperf() は、前回の本 API 呼び出し以降に実行されたインメモリサーバ機能の API の性能情報を **PerfUca** に返却する。
diosatrninit() で **ComData** の **ImsPerf** に DIOSA_ON が設定されていた場合のみ性能情報は蓄積される。
ImsPerf に DIOSA_OFF が設定された状態で本 API を実行した場合、何もせずに正常終了する。
diosatrninit()を連続して実行した場合、1 回目の **diosatrninit()**の **ImsPerf** の設定にしたがって動作する。

戻り値

DIOSA_DONE(0)	正常終了した。
DIOSA_ERROR(-1)	その他エラーが発生した。
DIOSA_EPARAM(-3)	パラメータが不正である。
DIOSA_EFUNCNAV(-34)	本 API を使用できないプロセス上の AP から要求された。
DIOSA_ESTATE(-114)	**diosaprcinit()**, **diosathrinit()**(マルチスレッドの場合), **diosaimopen()**が未実施である。

関連

diosatrninit, t_diosa_imperfuca

2.3.19　diosaimgetptblname(物理表名照会関数)

名前

diosaimgetptblname - 物理表名照会

書式

```
#include <diosa.h>
int diosaimgetptblname(int MAPId, char* LTableName, char* PTableName)
```

説明

diosaimgetptblname() は、指定された MAP の論理表に対応する物理表名を返却する。
MAPId には MAPID、**LTableName** には論理表名を指定し、**PTableName** には 256 バイト以上の領域を指定する。

戻り値

DIOSA_DONE(0)	正常終了した。
DIOSA_ERROR(-1)	その他エラーが発生した。
DIOSA_EPARAM(-3)	パラメータが不正である。
DIOSA_EINVAL(-9)	指定された MAPID が存在しない、または指定された論理表名が存在しない。
DIOSA_ESTATE(-114)	**diosaprcinit()**、または **diosathrinit()** が未実施である。

2.3.20　diosaimgetreckeyinfo(レコードキー情報照会関数)

名前

diosaimgetreckeyinfo – レコードのキー情報照会

書式

#include <diosa.h>

int diosaimgetreckeyinfo(t_diosa_imrefctx* Refctx, char* LTableName, int ReqKeyNum, int* KeyNum, t_diosa_imreckeyinfo* KeyInfo)

説明

diosaimgetreckeyinfo() は、指定した論理表に関するレコードのキー情報を返却する。
LTableName には照会対象の論理表名、**KeyInfo** にはキー情報の返却領域として **t_diosa_imreckeyinfo** 配列の先頭アドレス、**ReqKeyNum** には **KeyInfo** に指定した **t_diosa_imreckeyinfo** 配列の個数を指定する。
KeyInfo に各キーの情報、**KeyNum** にキーの個数が返却される。
但し、**ReqKeyNum** に指定された個数より、キーの個数が多い場合は、**diosaimgetreckeyinfo()** を続けて実行することで、前回の続きからキー情報を取得することができる。
初回実行時は、**Refctx** の各メンバに負値を指定して **diosaimgetreckeyinfo()** を実行し、前回の続きから取得する場合は、前回返却した **Refctx** の値をそのまま指定して **diosaimgetreckeyinfo()** を実行する。
なお、初回実行時の先頭 **KeyInfo** には、必ずプライマリキーのキー情報が返却される。
全てのキーの情報を返却した場合は、**KeyNum** に 0、戻り値に **DIOSA_DATALIM** が返却される。

戻り値

DIOSA_DONE(0)	キー情報を取得できた。
DIOSA_DATALIM(4)	キー情報を全て取得し終わった。
DIOSA_ERROR(-1)	その他エラーが発生した。
DIOSA_EPARAM(-3)	パラメータが不正である。
DIOSA_EINVAL(-9)	指定された論理表名が存在しない。
DIOSA_ESTATE(-114)	**diosaprcinit()**、または **diosathrinit()** が未実施である。

関連

t_diosa_imreckeyinfo,　t_diosa_imrefctx

2.3.21　diosaimgetsvcstatus(サービス状態取得関数)

名前

diosaimgetsvcstatus – サービス状態の取得

書式

```
#include <diosa.h>
int diosaimgetsvcstatus(int *Status)
```

説明

diosaimgetsvcstatus () は、サービスの状態コードを Status に返却する。
DIOSA_DONE 以外のコードが返された場合はサービスが抑止されている状態を示し、抑止の原因を識別するコードが返却される。

DIOSA_DONE(0)	正常終了。
DIOSA_BLOCK(6)	MAP が閉塞している。
DIOSA_SWITCH(21)	計画マスタ切替中である。
DIOSA_ERROR(-1)	異常終了、その他エラー。
DIOSA_ESYS(-2)	システムコールエラー。
DIOSA_EMEM(-6)	メモリ操作エラー。
DIOSA_ETIMEOUT(-22)	タイムアウト発生。
DIOSA_EACCES(-25)	更新ログ出力機能が準備できてないため、更新アクセスできない。
DIOSA_EOVERFLOW(-39)	IMS キューバッファオーバフロー。
DIOSA_EMSGQ(-41)	メッセージキューエラー。
DIOSA_ESOCKET(-70)	ソケット送受信エラー。
DIOSA_ECONNECT(-71)	通信パス切断。
DIOSA_ETAM(-113)	IM 障害。
DIOSA_ESWITCH(-115)	障害時マスタ切替中である。
DIOSA_EINACTIVE(-117)	レプリケーショングループが停止している。
DIOSA_EREADY(-118)	サーバが起動中(再起動)や環境定義変更中により、アクセスするための準備ができていない。
DIOSA_EEXTEND(-123)	管理テーブル拡張エラー。

戻り値

DIOSA_DONE(0)	正常に IMS サーバ状態コードを取得した。
DIOSA_EPARAM(-3)	パラメータが不正である。
DIOSA_ESTATE(-114)	**diosaimopen()、diosatrninit()**が未実施である。

2.3.22　diosaimgettblid(論理表 ID 照会関数)

名前

diosaimgettblid - 論理表名に対応する論理表 ID 照会

書式

```
#include <diosa.h>
int diosaimgettblid(char* LTableName, int* TableId)
```

説明

diosaimgettblid() は、指定した論理表名に対応する、論理表 ID を返却する。
LTableName には論理表名を指定し、**TableId** には論理表 ID が返却される。

戻り値

DIOSA_DONE(0)	正常終了した。
DIOSA_ERROR(-1)	その他エラーが発生した。
DIOSA_EPARAM(-3)	パラメータが不正である。
DIOSA_EINVAL(-9)	指定された論理表名が存在しない。
DIOSA_ESTATE(-114)	**diosaprcinit()**、または **diosathrinit()** が未実施である。

関連

diosaimread1(), diosaimread(), diosaimwrite(), diosaimrewrite(), diosaimdelete(), diosaimdeletex1(), diosaimtruncate(), diosaimctxopen()

2.3.23　diosaimgettbllist(論理表一覧照会関数)

名前

diosaimgettbllist - 論理表一覧の照会

書式

#include <diosa.h>

int diosaimgettbllist(t_diosa_imrefctx* Refctx, int Type, int MAPId, int ReqTblNum, int* TblNum, t_diosa_imtbllist* TblList)

説明

diosaimgettbllist() は、指定された MAP と表種別に対応する、全論理表の論理表 ID と論理表名を返却する。

MAPId には照会対象の MAPID、**Type** には照会対象の表種別、**TblList** には論理表 ID と論理表名の返却領域として **t_diosa_imtbllist** 配列の先頭アドレス、**ReqTblNum** には **TblList** に指定した **t_diosa_imtbllist** 配列の個数を指定する。

TblNum には、**TblList** に返却した論理表の個数が返却される。

初回実行時は、**Refctx** の各メンバに負値を指定して **diosaimgettbllist()** を実行し、前回の続きから取得する場合は、前回返却した **Refctx** の値をそのまま指定して **diosaimgettbllist()** を実行する。

Type には以下を指定できる。

DIOSA_TABLE_ALL	DIOSA 制御表、ユーザ表を含む全て
1～9 の数値	ユーザ表（環境定義 IMTABLECONF 節-LTABLE 項-TYPE で定義したもの）

全ての論理表を返却した場合は、**TblNum** に 0、戻り値に **DIOSA_DATALIM** が返却される。

戻り値

DIOSA_DONE(0)	論理表を取得できた。
DIOSA_DATALIM(4)	論理表を全て取得し終わった。
DIOSA_ERROR(-1)	その他エラーが発生した。
DIOSA_EPARAM(-3)	パラメータが不正である。
DIOSA_EINVAL(-9)	指定された表種別が存在しない、または指定された MAPId が存在しない。
DIOSA_ESTATE(-114)	**diosaprcinit()**、または **diosathrinit()** が未実施である。

関連

t_diosa_imtbllist,　t_diosa_imrefctx

2.3.24　diosaimopen(IM サーバオープン関数)

名前

diosaimopen – IM サーバオープン

書式

```
#include <diosa.h>
int diosaimopen(int Mode)
```

説明

diosaimopen() は、IM サーバとの通信に必要な資源の確保を行う。
Mode には、接続オプションとして以下の値のいずれかを指定する。

DIOSA_CONN_INADVANCE 当 API 内で全ての MAP に対して接続処理を行う。

DIOSA_CONN_ONDEMAND 当 API 内では接続処理を行わず、MAP ごとに **diosaimsetmap()** 初回実行時に接続処理を行う。

戻り値

DIOSA_DONE(0)	正常終了した。
DIOSA_ERROR(-1)	その他エラーが発生した。
DIOSA_ESYS(-2)	システムコールエラーが発生した。
DIOSA_EPARAM(-3)	パラメータが不正である。
DIOSA_EMEM(-6)	メモリ操作に失敗した。
DIOSA_ENOINIT(-11)	メモリキャッシュ起動コマンドが未実施である。
DIOSA_EMSGQ(-41)	メッセージキューの障害が発生した。
DIOSA_ESOCKET(-70)	ソケット送受信で障害が発生した。
DIOSA_ESTATE(-114)	**diosaprcinit()**、または **diosathrinit()** が未実施である。

注意

スレッド単位に実行する必要がある。**diosaimopen()** を実行したスレッドは、**diosaimopen()** が正常終了したか否かに関わらず、スレッドを終了する前に **diosaimclose()** を実行しなくてはならない。

関連

diosaimclose()

2.3.25　diosaimread(複数レコード読込関数)

名前

diosaimread – レコードの複数取得

書式

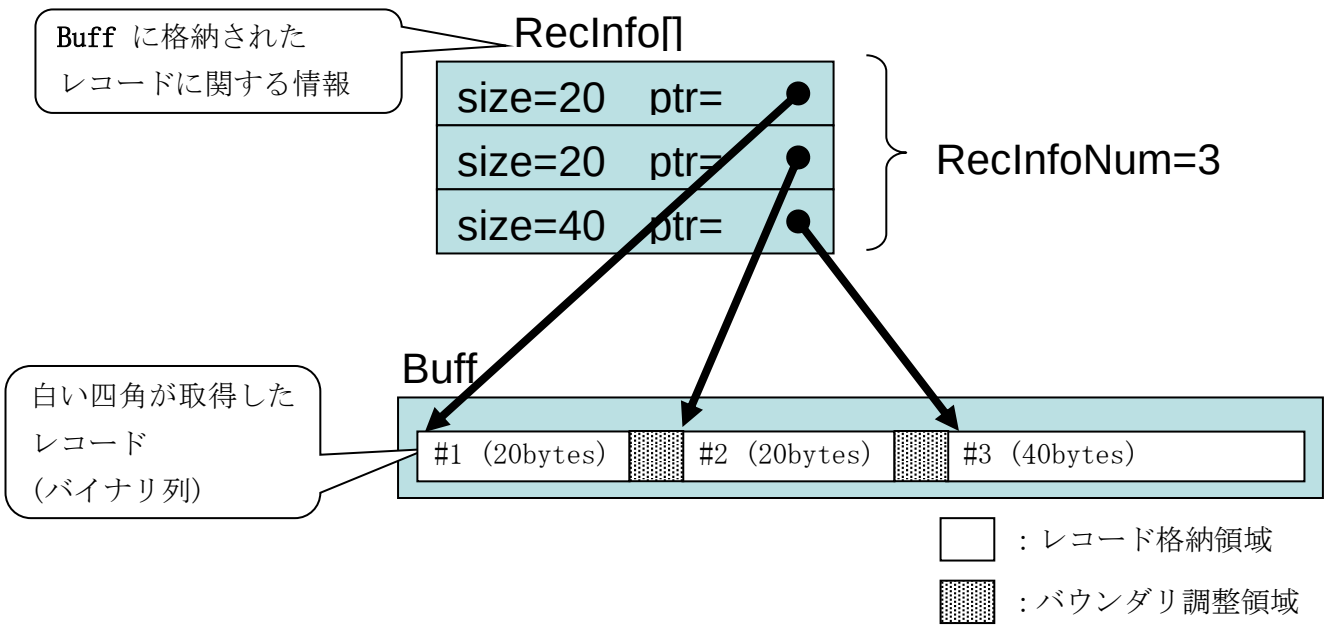
#include <diosa.h>

int diosaimread(int CtxId, t_diosa_recinfo* RecInfo, int RecInfoNum, int* FetchedNum, char* Buff, size_t BuffSize)

説明

diosaimread() は、**CtxId** で指定された検索条件に合致するレコードを、**RecInfoNum** に指定された件数単位に取得する。**Buff** にはレコードを格納する領域のアドレス、その格納する領域のサイズを **BuffSize** に指定する。**RecInfo** には、**t_diosa_recinfo** 配列の先頭アドレスを指定し、配列の数を **RecInfoNum** に指定する。なお、**RecInfoNum** に指定した件数が読込要求件数となる。

取得したレコードとレコードに関する情報は、指定したバッファ **Buff** とレコード情報構造体 **RecInfo** に返却する。このとき、取得したレコード1件目はバッファ **Buff** の先頭から格納し、2件目以降を格納する際は8バイトのバウンダリ調整を行う。



但し、バッファ **Buff** に、読込要求件数分のレコードを格納できない場合は、格納できる件数分のレコードを取得することになる。取得できたレコード件数は、**FetchedNum** に返却する。

読込対象となる表は、**TableId** に指定された論理表 ID と **diosaimsetmap()** により設定された MAP から決定する。

検索条件に合致するレコードの件数が一回の **diosaimread()** で取得できなかった場合は、**diosaimread()** を続けて実行することで、前回実行時の続きのレコードを取得することができる。検索条件に合致するレコードの取得が全て終わった時は、**FetchedNum** に0件、戻り値に **DIOSA_NOENT** を返却する。

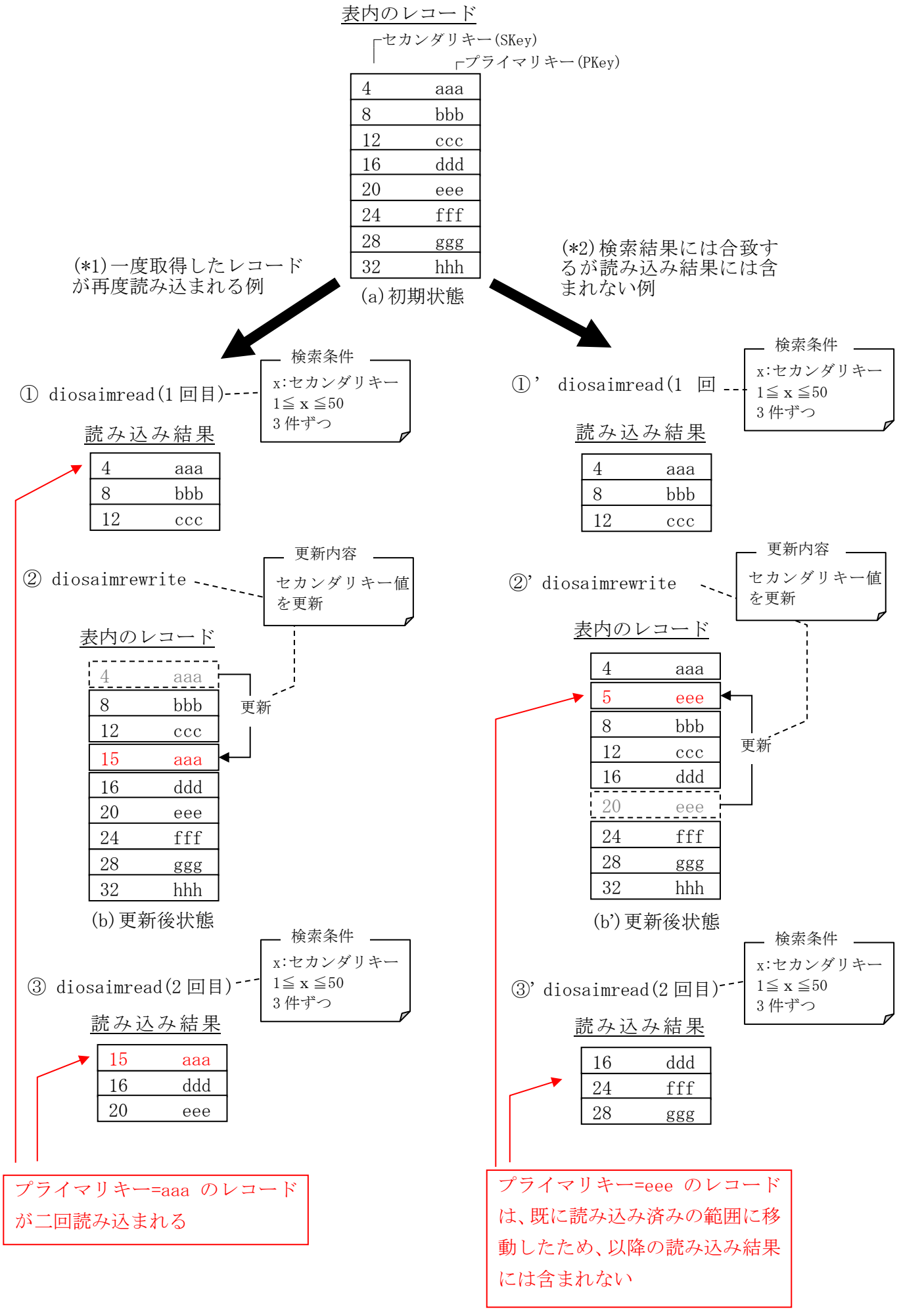
戻り値

DIOSA_DONE(0)	条件に合致するレコードを取得した。
DIOSA_BLOCK(6)	該当 MAP が閉塞している。
DIOSA_NOENT(20)	該当レコードが読込対象の表に存在しない。もしくは、条件に合致するレコードの取得が全て終わった。
DIOSA_SWITCH(21)	計画マスタ切替中である。
DIOSA_ERROR(-1)	その他エラーが発生した。
DIOSA_ESYS(-2)	システムコールエラーが発生した。

DIOSA_EPARAM(-3)	パラメータが不正である。
DIOSA_EMEM(-6)	メモリ操作に失敗した。
DIOSA_ENOBUFS(-7)	**Buff** に指定された領域に取得したレコードが 1 件も入らない
DIOSA_EINVAL(-9)	読込対象の表が存在しない。
DIOSA_ETIMEOUT(-22)	要求がタイムアウトした。
DIOSA_EACCES(-25)	アクセス対象の表がオープンされていない。
DIOSA_EOVERFLOW(-39)	IMS キューバッファに空きがない。
DIOSA_EMSGQ(-41)	メッセージキューの障害が発生した。
DIOSA_ESOCKET(-70)	ソケット送受信で障害が発生した。
DIOSA_ECONNECT(-71)	通信パスが切断された。
DIOSA_ETAM(-113)	IM の障害によりレコード読み込みに失敗した。
DIOSA_ESTATE(-114)	指定した **CtxId** は **diosaimctxopen()** 未実施である。もしくは、**diosaimsetmap()** が未実施である。
DIOSA_ESWITCH(-115)	障害時マスタ切替中である。
DIOSA_EINACTIVE(-117)	レプリケーショングループが停止している。
DIOSA_EREADY(-118)	サーバが起動中(再起動)や環境定義変更中により、アクセスするための準備ができていない。

注意

- **DIOSA_SWITCH** などのサービス抑止となる戻り値が返却された場合、**diosaimrollback()**(**diosaimtxstart()** を実行済みの場合のみ)、**diosatrnterm()**実行後、**diosatrninit()**から実行し直す必要がある。サービス抑止となる戻り値については、メモリキャッシュ利用の手引きの「3.1.8 戻り値について」を参照。
- **BuffSize** は、 IMENV 節-MAP 項-IMQUEBUFFSIZE に依存しているため、IMQUEBUFFSIZE に指定した領域に確保できるサイズを指定すること。IMQUEBUFFSIZE の詳細は、メモリキャッシュ利用の手引き「4.1.3 (2)アクセスサーバの IMS キューバッファサイズ計算」を参照。
- diosaimread 開始後(複数回に渡って diosaimread をする場合の初回呼び出しを行なった場合)に、検索対象キーの値を更新した場合、一度取得したレコードが再度読み込まれる(*1)、または、検索条件には合致するが読込結果には含まれない場合がある(*2)。



関連

diosaimctxopen()

補足

diosaimread 関数の使用方法を記述する。

1) 検索条件(t_diosa_imcond)構造体の宣言と領域の割当を行う。
2) レコード情報配列(t_diosa_recinfo[])の宣言と割当を行う。
   今回の例では 10 件ごとの読込みを想定しているため、10 の配列を用意する。
3) diosaimcondsetkey 関数、または diosaimcondsetrange 関数で、検索条件の構築を行う。

- 〈完全一致の場合〉

  diosaimcondsetkey 関数を使用する(定義長と同じ長さのキーを指定)。

- 〈前方一致の場合〉

  diosaimcondsetkey 関数を使用する(定義長よりも短い長さのキーを指定)。

- 〈全件検索の場合〉

  全件検索の場合は、検索条件構築は不要。下記 5)で IndexName と Cond に NULL を指定する。

- 〈範囲指定の場合〉

  diosaimcondsetrange 関数を使用する。

4) diosaimgettblid 関数で、論理表 ID を取得する。
5) diosaimctxopen 関数で、検索コンテキストを生成する。
6) 条件に合致するレコードが存在する間、diosaimread 関数を繰り返す。
7) diosaimctxclose 関数で、検索コンテキストを破棄する。

ソース中の番号は、上記説明文の番号に対応する。
(エラー処理は省略)

```
1)      t_diosa_imcond  Cond;
        int    CtxId;
2)      t_diosa_recinfo RecInfo[10];
        int    FetchedNum;
        char  *Buff;
        size_t BuffSize;
        int    Ret;
        int    TblId

        /*
         * SG 定義上、長さ 8 バイトのキー対する前方一致検索を行う
         * (YYYYMMDD が格納されているインデックスを YYYYMM の前方一致で検索)
         */
3)      diosaimcondsetkey(&Cond, "201106", 6);
4)      diosaimgettblid("SAMPLE_TABLE", &TblId);
5)      diosaimctxopen(TblId, "index2", &Cond, DIOSA_ASCEND, DIOSA_NOLOCK, &CtxId);

        /* レコード 10 件分が格納できる領域を確保する */
        BuffSize = レコードを格納する領域のサイズを設定;
        Buff = diosamalloc(BuffSize);

6)      while(true){
            /* 条件に合致するものを 10 件ずつ取得する */
            Ret = diosaimread(CtxId, &RecInfo, 10, &FetchedNum, Buff, BuffSize);
            if(DIOSA_NOENT == nRet){
                /* 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)に格納されている。
            レコードイメージ格納アドレスは、RecInfo[i].RecordPtr でアクセスする。
            レコードサイズは、RecInfo[i].RecordSize でアクセスする。
        }
    }
```

7)　**diosaimctxclose**(CtxId);

2.3.26　diosaimread1(キー指定レコード読込関数)

名前

diosaimread1 - 1 レコードの取得(主キーの完全一致)

書式

```
#include <diosa.h>
int diosaimread1(int TableId, t_diosa_imcond* Cond, t_diosa_recinfo* RecInfo, char* Buff, size_t BuffSize, int Lock)
```

説明

diosaimread1() は、**Cond** で指定されたキー値に一致するレコードを１件取得する。
Cond には、**diosaimcondsetkey()** を使用してプライマリキーを設定したものを指定する。
読込対象となる表は、**TableId** に指定された論理表 ID と **diosaimsetmap()** により設定された MAP から決定する。
取得したレコードとレコードに関する情報は、指定したバッファ Buff とレコード情報構造体 **RecInfo** に返却する。バッファのサイズは、**BuffSize** に指定する。
Lock には、排他オプションとして以下の値のいずれかを指定する。

DIOSA_NOLOCK	ロックせずに読込を行う。他トランザクションが既に読込対象レコードをロックしていても、待ち合わせすることなく読込を行う。なお、同時に他トランザクションが更新している場合は、更新前のレコードを読み込む。
DIOSA_LOCK_WAIT	ロックして読込を行う。他トランザクションで既に読込対象レコードがロックされている場合は、ロックが解放されるまで待ち合わせる。但し、応答監視タイマ(環境定義 IMENV 節-USERAP 項-REQTIMEOUT で定義したもの)の時間を経過してもロックが解放されない場合は、**DIOSA_ETIMEOUT** が返却される。
DIOSA_LOCK_NOWAIT	ロックして読込を行う。他トランザクションで既に読込対象レコードがロックされている場合は、待ち合わせをせずにエラーとなる。

Lock に **DIOSA_LOCK_WAIT**、または **DIOSA_LOCK_NOWAIT** を指定した **diosaimread1()** は、**diosaimtxstart()** と **diosaimcommit()**、または **diosaimrollback()** の区間内でのみ、実行できる。

戻り値

DIOSA_DONE(0)	正常終了した。
DIOSA_BLOCK(6)	該当 MAP が閉塞している。
DIOSA_NOENT(20)	該当レコードが読込対象の表に存在しなかった。
DIOSA_SWITCH(21)	計画マスタ切替中である。
DIOSA_ERROR(-1)	その他エラーが発生した。
DIOSA_ESYS(-2)	システムコールエラーが発生した。
DIOSA_EPARAM(-3)	パラメータが不正である。
DIOSA_ENOBUFS(-7)	Buff に指定された領域に取得したレコードが入らない。
DIOSA_EINVAL(-9)	読込対象の表が存在しない。
DIOSA_ELOCK(-14)	ロック取得に対しロックリストオーバーフローが発生した。
DIOSA_ETIMEOUT(-22)	要求がタイムアウトした。
DIOSA_EACCES(-25)	アクセス対象の表がオープンされていない。もしくは、更新ログ出力機能が準備できてないため、更新アクセスできない。

DIOSA_EBUSY(-31)	他の利用者により既に該当レコードがロックされている。Lock に DIOSA_LOCK_NOWAIT が指定されていた場合にのみ発生する。
DIOSA_EDEADLOCK(-36)	デッドロックが発生した。Lock に DIOSA_LOCK_WAIT を指定した場合のみ発生する。
DIOSA_EOVERFLOW(-39)	IMS キューバッファに空きがない。
DIOSA_EMSGQ(-41)	メッセージキューの障害が発生した。
DIOSA_ECOND(-60)	別 MAP への更新系のアクセス要求 API が実行されている。もしくは、同一トランザクション内で diosaimtruncate() が実行されている。Lock に DIOSA_LOCK_WAIT、DIOSA_LOCK_NOWAIT のいずれかを指定した場合のみ発生する。
DIOSA_ESOCKET(-70)	ソケット送受信で障害が発生した。
DIOSA_ECONNECT(-71)	通信パスが切断された。
DIOSA_ETAM(-113)	IM の障害によりレコード読み込みに失敗した。
DIOSA_ESTATE(-114)	diosaimtxstart()、または diosaimsetmap() が未実施である。もしくは、検索条件 Cond が diosaimcondsetkey() で構築されていない。
DIOSA_ESWITCH(-115)	障害時マスタ切替中である。
DIOSA_EINACTIVE(-117)	レプリケーショングループが停止している。
DIOSA_EREADY(-118)	サーバが起動中(再起動)や環境定義変更中により、アクセスするための準備ができていない。
DIOSA_EEXTEND(-123)	管理テーブルの拡張に失敗した。

注意

- Lock に DIOSA_NOLOCK を指定して diosaimread1() を実行する場合は、トランザクション区間内である必要はない。
- DIOSA_SWITCH などのサービス抑止となる戻り値が返却された場合、diosaimrollback()(diosaimtxstart()を実行済みの場合のみ)、diosatrnterm()実行後、diosatrninit()から実行し直す必要がある。サービス抑止となる戻り値については、メモリキャッシュ利用の手引きの「3.1.8 戻り値について」を参照。
- BuffSize は、 IMENV 節-MAP 項-IMQUEBUFFSIZE に依存しているため、IMQUEBUFFSIZE に指定した領域に確保できるサイズを指定すること。IMQUEBUFFSIZE の詳細は、メモリキャッシュ利用の手引き「4.1.3 (2)アクセスサーバの IMS キューバッファサイズ計算」を参照。

関連

diosaimsetmap(), diosaimtxstart(), diosaimgettblid(), t_diosa_imcond

2.3.27　diosaimrewrite(レコード更新関数)

名前

diosaimrewrite - １レコード更新

書式

```
#include <diosa.h>
int diosaimrewrite(int TableId, t_diosa_recinfo* RecInfo, int RecInfoNum)
```

説明

diosaimrewrite() は、**RecInfo** に指定されたレコードを更新する。
更新対象となる表は、**TableId** に指定された論理表 ID と **diosaimsetmap()** により設定された MAP から決定する。
RecInfo には、排他オプションに **DIOSA_LOCK_WAIT**、**DIOSA_LOCK_NOWAIT** のいずれかを指定して **diosaimread1()** で取得したレコード情報を指定する。
RecInfoNum には、1 を指定する。2 以上を指定した場合は **DIOSA_EFUNCNAV**、0 以下を指定した場合は **DIOSA_EPARAM** が返却される。
diosaimrewrite() は、**diosaimtxstart()** と **diosaimcommit()**、または **diosaimrollback()** の区間内でのみ、実行できる。

戻り値

DIOSA_DONE(0)	正常終了した。
DIOSA_BLOCK(6)	該当 MAP が閉塞している。
DIOSA_NOENT(20)	更新対象のレコードが存在しない。
DIOSA_SWITCH(21)	計画マスタ切替中である。
DIOSA_ERROR(-1)	その他エラーが発生した。
DIOSA_ESYS(-2)	システムコールエラーが発生した。
DIOSA_EPARAM(-3)	パラメータが不正である。
DIOSA_EINVAL(-9)	更新対象の表が存在しない。
DIOSA_ETIMEOUT(-22)	要求がタイムアウトした。
DIOSA_EACCES(-25)	アクセス対象の表がオープンされていない。もしくは、更新ログ出力機能が準備できてないため、更新アクセスできない。
DIOSA_ESIZE(-33)	レコードサイズが不正である。
DIOSA_EFUNCNAV(-34)	未サポート機能を指定した。
DIOSA_EOVERFLOW(-39)	IMS キューバッファに空きがない。
DIOSA_EMSGQ(-41)	メッセージキューの障害が発生した。
DIOSA_ECOND(-60)	別 MAP への更新系のアクセス要求 API が実行されている。もしくは、同一トランザクション内で **diosaimtruncate()** が実行されている。
DIOSA_ESOCKET(-70)	ソケット送受信で障害が発生した。
DIOSA_ECONNECT(-71)	通信パスが切断された。
DIOSA_ETAM(-113)	IM の障害によりレコード更新に失敗した。
DIOSA_ESTATE(-114)	**diosaimtxstart()**、または **diosaimsetmap()** が未実施である。もしくは、レコードのロックを取得せずに実行した。
DIOSA_ESWITCH(-115)	障害時マスタ切替中である。

DIOSA_EINACTIVE(-117)　レプリケーショングループが停止している。

DIOSA_EREADY(-118)　サーバが起動中(再起動)や環境定義変更中により、アクセスするための準備ができていない。

注意

- **DIOSA_SWITCH** などのサービス抑止となる戻り値が返却された場合、**diosaimrollback()**、**diosatrnterm()** 実行後、**diosatrninit()**から実行し直す必要がある。サービス抑止となる戻り値については、メモリキャッシュ利用の手引きの「3.1.8 戻り値について」を参照。
- diosaimread1()で取得したレコードのプライマリキーを変更して、diosaimrewrite()を実行することはできない。実行した場合 DIOSA_ESTATE で異常終了するが、変更後のプライマリキーのレコードも同一トランザクション内でロック済みの場合は、そのレコードを更新してしまうこととなる。

関連

diosaimsetmap(), diosaimtxstart(), diosaimgettblid(), diosaimread1(), t_diosa_recinfo

2.3.28　diosaimrollback(ロールバック関数)

名前

diosaimrollback - 更新要求のロールバック

書式

```
#include <diosa.h>
int diosaimrollback(void)
```

説明

diosaimrollback() は、**diosaimtxstart()** 以降に実行された更新要求をロールバックし、ロックしていたレコードのロックを解除する。
更新系 API が実行されていない状態で **diosaimrollback()** が実行された場合も正常終了する。

戻り値

DIOSA_DONE(0)	正常終了した。
DIOSA_ALMOST(10)	ロールバックは正常終了したが、処理の継続はできない。
DIOSA_ERROR(-1)	その他エラーが発生した。
DIOSA_ESYS(-2)	システムコールエラーが発生した。
DIOSA_EMSGQ(-41)	メッセージキューの障害が発生した。
DIOSA_ESOCKET(-70)	ソケット送受信で障害が発生した。
DIOSA_ETAM(-113)	IM の障害によりロールバックが失敗した。
DIOSA_ESTATE(-114)	**diosatrninit()** が未実施である。

注意

- **DIOSA_DONE** 以外が返された場合は、**diosatrnterm()** 実行後、**diosatrninit()**から実行し直す必要がある。

関連

diosaimtxstart(), diosaimcommit()

2.3.29　diosaimsetmap(アクセス先 MAP 宣言関数)

名前

diosaimsetmap - アクセス先 MAP の設定

書式

```
#include <diosa.h>
int diosaimsetmap(int MAPId)
```

説明

diosaimsetmap() は、指定された **MAPId** の MAP をアクセス先として設定する。
CO 制御、およびバッチ AP 制御の環境の場合は、利用者がアクセス先を変更する必要がある時のみ、**diosaimsetmap()** を実行する必要がある。
上記以外の環境の場合は、**diosatrninit()** 実行後に、必ず **diosaimsetmap()** を実行してアクセス先を設定する必要がある。
なお、設定された MAP は、**diosaimcommit()**、または **diosaimrollback()** が実行された後も、次に **diosaimsetmap()** が実行されるまで有効である。

戻り値

DIOSA_DONE(0)	正常終了した。
DIOSA_BLOCK(6)	該当 MAP が閉塞している。
DIOSA_SWITCH(21)	計画マスタ切替中である。
DIOSA_ERROR(-1)	その他エラーが発生した。
DIOSA_ESYS(-2)	システムコールエラーが発生した。
DIOSA_EPARAM(-3)	パラメータが不正である。
DIOSA_EMEM(-6)	メモリ操作に失敗した。
DIOSA_EINVAL(-9)	対象の MAP が存在しない。
DIOSA_EMSGQ(-41)	メッセージキューの障害が発生した。
DIOSA_ESOCKET(-70)	ソケット送受信で障害が発生した。
DIOSA_ECONNECT(-71)	**MAPId** に指定されたサーバに接続できない。
DIOSA_ESTATE(-114)	**diosaimopen()**、または **diosatrninit()** が未実施である。
DIOSA_ESWITCH(-115)	障害時マスタ切替中である。
DIOSA_EINACTIVE(-117)	レプリケーショングループが停止している。
DIOSA_EREADY(-118)	サーバが起動中(再起動)や環境定義変更中により、アクセスするための準備ができていない。

注意

- **DIOSA_DONE**、**DIOSA_EPARAM**、**DIOSA_EINVAL** 以外が返された場合は、**diosatrnterm()**実行後、**diosatrninit()** から実行し直す必要がある。

関連

diosatrninit(), diosaimgetmap(), diosaimopen()

2.3.30　diosaimtruncate(全レコード削除関数)

名前

diosaimtruncate – 全レコード削除

書式

```
#include <diosa.h>
int diosaimtruncate(int TableId)
```

説明

diosaimtruncate() は、指定された表の全レコードを削除する。
削除対象となる表は、**TableId** に指定された論理表 ID と **diosaimsetmap()** により設定された MAP から決定する。
diosaimtruncate() は、**diosaimtxstart()** と **diosaimcommit()** の区間内でのみ、実行できる。

戻り値

DIOSA_DONE(0)	正常終了した。
DIOSA_BLOCK(6)	該当 MAP が閉塞している。
DIOSA_SWITCH(21)	計画マスタ切替中である。
DIOSA_ERROR(-1)	その他エラーが発生した。
DIOSA_ESYS(-2)	システムコールエラーが発生した。
DIOSA_EPARAM(-3)	パラメータが不正である。
DIOSA_EINVAL(-9)	削除対象の表が存在しない。
DIOSA_ETIMEOUT(-22)	要求がタイムアウトした。
DIOSA_EACCES(-25)	アクセス対象の表がオープンされていない。もしくは、更新ログ出力機能が準備できてないため、更新アクセスできない。
DIOSA_EBUSY(-31)	他トランザクションで削除対象の表のレコードをロックしている。
DIOSA_EOVERFLOW(-39)	IMS キューバッファに空きがない。
DIOSA_EMSGQ(-41)	メッセージキューの障害が発生した。
DIOSA_ECOND(-60)	同一トランザクション内で **diosaimread1(DIOSA_LOCK_WAIT、DIOSA_LOCK_NOWAIT** のいずれかが指定)、**diosaimwrite()**、**diosaimrewrite()**、**diosaimdelete()**、**diosaimdeletex1()** が既に実行されている。
DIOSA_ESOCKET(-70)	ソケット送受信で障害が発生した。
DIOSA_ECONNECT(-71)	通信パスが切断された。
DIOSA_ETAM(-113)	IM の障害によりレコード削除に失敗した。
DIOSA_ESTATE(-114)	**diosaimtxstart()**、または **diosaimsetmap()** が未実施である。
DIOSA_ESWITCH(-115)	障害時マスタ切替中である。
DIOSA_EINACTIVE(-117)	レプリケーショングループが停止している。
DIOSA_EREADY(-118)	サーバが起動中(再起動)や環境定義変更中により、アクセスするための準備ができていない。

注意

- 正常終了した場合は、 **diosaimrollback()** によるキャンセルは出来ない。

- 異常終了(負値)が返却された場合は、必ず diosaimrollback() を実行すること。但し、DIOSA_ETIMEOUT が返却された場合、全レコード削除要求が確定している可能性があり、その場合はキャンセル出来ない。
- 他トランザクションが削除対象の表のレコードをロックしている場合、DIOSA_EBUSY となる。
- 当 API を使用する場合、同一トランザクション内で diosaimread1(DIOSA_LOCK_WAIT、または DIOSA_LOCK_NOWAIT 指定)、diosaimwrite()、diosaimrewrite()、diosaimdelete()、diosaimdeletex1() を使用することはできない。

関連

diosaimsetmap(), diosaimtxstart(), diosaimgettblid()

2.3.31　diosaimtxstart(トランザクション開始関数)

名前

diosaimtxstart - トランザクションの開始宣言

書式

```
#include <diosa.h>
int diosaimtxstart(void)
```

説明

diosaimtxstart()はトランザクションの開始を宣言する。**diosaimtxstart()**実行後から **diosaimcommit()**、または **diosaimrollback()**実行までが 1 トランザクションとなる。
diosaimtxstart()を実行したスレッドは、トランザクションの最後に、**diosaimcommit()**、または **diosaimrollback()**でトランザクションを完了させなくてはならない。

戻り値

DIOSA_DONE(0)	正常終了した。
DIOSA_ERROR(-1)	その他エラーが発生した。
DIOSA_ESTATE(-114)	**diosaimopen()**、または **diosatrninit()** が未実施である。

関連

diosaimcommit(), diosaimrollback()

2.3.32　diosaimwrite(レコード追加関数)

名前

diosaimwrite – レコードの追加

書式

```
#include <diosa.h>
int diosaimwrite(int TableId, t_diosa_recinfo* RecInfo, int RecInfoNum, int Lock)
```

説明

diosaimwrite() は、**RecInfo** に指定されたレコードを追加する。
追加対象となる表は、**TableId** に指定された論理表 ID と **diosaimsetmap()** により設定された MAP から決定する。
Lock には、排他オプションとして以下の値のいずれかを指定する。

DIOSA_LOCK_WAIT	他トランザクションで同じプライマリキーのレコードを追加している場合は、そのトランザクションが終了するまで待ち合わせる。但し、応答監視タイマ(環境定義 IMENV 節-USERAP 項-REQTIMEOUT で定義したもの)の時間を経過してもロックが解放されない場合は、**DIOSA_ETIMEOUT** が返却される。
DIOSA_LOCK_NOWAIT	他トランザクションで同じプライマリキーのレコードを追加している場合は、待ち合わせをせずにエラーとなる。

RecInfoNum には、1 を指定する。2 以上を指定した場合は **DIOSA_EFUNCNAV**、0 以下を指定した場合は **DIOSA_EPARAM** が返却される。
diosaimwrite() は、**diosaimtxstart()** と **diosaimcommit()**、または **diosaimrollback()** の区間内でのみ、実行できる。

戻り値

DIOSA_DONE(0)	正常終了した。
DIOSA_BLOCK(6)	該当 MAP が閉塞している。
DIOSA_SWITCH(21)	計画マスタ切替中である。
DIOSA_ERROR(-1)	その他エラーが発生した。
DIOSA_ESYS(-2)	システムコールエラーが発生した。
DIOSA_EPARAM(-3)	パラメータが不正である。
DIOSA_EINVAL(-9)	追加対象の表が存在しない。
DIOSA_ELOCK(-14)	ロック取得に対しロックリストオーバーフローが発生した。
DIOSA_ETIMEOUT(-22)	要求がタイムアウトした。
DIOSA_EACCES(-25)	アクセス対象の表がオープンされていない。もしくは、更新ログ出力機能が準備できてないため、更新アクセスできない。
DIOSA_EEXIST(-27)	同じプライマリキーのレコードが、既に追加対象の表に存在する。
DIOSA_EBUSY(-31)	他の利用者により既に該当レコードがロックされている。**Lock** に **DIOSA_LOCK_NOWAIT** が指定されていた場合にのみ発生する。
DIOSA_ESIZE(-33)	レコードサイズが不正である。
DIOSA_EFUNCNAV(-34)	未サポート機能を指定した。
DIOSA_EDEADLOCK(-36)	デッドロックが発生した。**Lock** に **DIOSA_LOCK_WAIT** を指定した場合のみ発生する。

DIOSA_EOVERFLOW(-39)	IMS キューバッファに空きがない。
DIOSA_EMSGQ(-41)	メッセージキューの障害が発生した。
DIOSA_ECOND(-60)	別 MAP への更新系のアクセス要求 API が実行されている。もしくは、同一トランザクション内で **diosaimtruncate()** が実行されている。
DIOSA_ESOCKET(-70)	ソケット送受信で障害が発生した。
DIOSA_ECONNECT(-71)	通信パスが切断された。
DIOSA_ETAM(-113)	IM の障害によりレコード追加に失敗した。
DIOSA_ESTATE(-114)	**diosaimtxstart()**、または **diosaimsetmap()** が未実施である。
DIOSA_ESWITCH(-115)	障害時マスタ切替中である。
DIOSA_EINACTIVE(-117)	レプリケーショングループが停止している。
DIOSA_EREADY(-118)	サーバが起動中(再起動)や環境定義変更中により、アクセスするための準備ができていない。
DIOSA_EEXTEND(-123)	管理テーブルの拡張に失敗した。

注意

- **DIOSA_SWITCH** などのサービス抑止となる戻り値が返却された場合、**diosaimrollback()**、**diosatrnterm()** 実行後、**diosatrninit()**から実行し直す必要がある。サービス抑止となる戻り値については、メモリキャッシュ利用の手引きの「3.1.8 戻り値について」を参照。

関連

diosaimsetmap(), diosaimtxstart(), diosaimgettblid(), t_diosa_recinfo

2.3.33　diosatamswitch(マスタ昇格関数)

名前

diosatamswitch - IM のスレーブをマスタに昇格する

書式

```
#include <diosa.h>
int    diosatamswitch( int RepGrpId, int TimeOut );
```

説明

diosatamswitch()は、**RepGrpId** で指定されたレプリケーショングループのスレーブ IM をマスタに昇格する。本関数は、昇格させるスレーブの IM が存在するノードで実行する。

RepGrpId	自ノードにスレーブが存在する特定 IM をマスタに切り替える場合に、対象の IM が属するレプリケーショングループの ID を指定する。
TimeOut	タイムアウト時間を指定する。（秒単位：0～2147483647、0 は無制限）

戻り値

成功した場合には、0 が返される。エラー時は、負の値が返される。

DIOSA_DONE(0)	正常終了
DIOSA_ERROR(-1)	異常終了
DIOSA_EPARAM(-3)	パラメータエラー。
DIOSA_EMEM(-6)	メモリ確保エラー
DIOSA_ENOENT(-8)	指定されたレプリケーショングループが存在しない。 または、指定レプリケーショングループの IM が自ノードに配置されていない。
DIOSA_ENOINIT(-11)	プロセス初期化処理が行われていない。
DIOSA_ESEND(-15)	電文送信エラー
DIOSA_ERECV(-20)	電文受信エラー
DIOSA_ETIMEOUT(-22)	一定時間内に応答がない。
DIOSA_EBUSY(-31)	他要求による処理中で対応できない。
DIOSA_EFUNCNAV(-34)	動作不可能なノードで実行された。
DIOSA_ECONNECT(-71)	デーモンへの接続に失敗した。
DIOSA_ESTATE(-114)	対象が正常稼動のスレーブではない。

注意

- CO 制御、バッチ AP 制御以外の環境で diosatamswitch()を実行する場合は、diosaprcinit()、diosathrinit()を呼び出した後に実行する必要がある。

関連

diosaprcinit, diosaprcterm, diosathrinit, diosathrterm

2.3.34　t_diosa_getmapstatuca(MAP のレプリケーション状態取得関数インタフェース構造体)

名前

t_diosa_getmapstatuca - MAP のレプリケーション状態取得関数インタフェース構造体

書式

```
#include <diosa.h>
t_diosa_getmapstatuca GetMapStatUca;
```

説明

t_diosa_getmapstatuca 構造体は、ヘッダ<diosa.h>で定義されていて、その中に MAP のレプリケーション状態取得関数のインタフェースに関連する情報が格納される。

char *MainKey;	メインキーを格納した領域のポインタを指定する。	
	MAPID による検索を行う場合は、NULL を指定する。	
int RepGrpId;	レプリケーショングループ ID が返却される。	
int MapId;	MAPID を指定する。	
	メインキーによる検索を行う場合、0 を指定する。	
	メインキーによる検索の場合、MAPID が返却される。	
char Status;	レプリケーショングループの状態が返却される。	
	DIOSA_REPGRP_ACTIVE	正常稼動
	DIOSA_REPGRP_STOP	停止
	DIOSA_REPGRP_ABNORMAL	障害
	DIOSA_REPGRP_SWITCH_ABN	障害によるマスタ切替中
	DIOSA_REPGRP_SWITCH_PLN	計画マスタ切替中
char Type;	自ノードのマスタ／スレーブ種別が返却される。	
	DIOSA_MASTER	マスタ
	DIOSA_SLAVE	スレーブ
	DIOSA_NO	該当 MAP の IM が自ノードに定義されていない
char MapStatus;	MAP の運用状態が返却される。	
	DIOSA_MAP_ACTIVE	正常稼動
	DIOSA_MAP_STOP	停止状態
char BlockStatus;	MAP の閉塞状態が返却される。	
	DIOSA_MAP_ACTIVE	正常稼動
	DIOSA_MAP_BLOCK	閉塞状態

関連

diosagetmapstat

2.3.35　t_diosa_getmapuca(MAP 情報取得関数インタフェース構造体)

名前

t_diosa_getmapuca - MAP 情報取得関数インタフェース構造体

書式

```
#include <diosa.h>
t_diosa_getmapuca GetMapUca;
```

説明

t_diosa_getmapuca 構造体は、ヘッダ<diosa.h>で定義されていて、その中に MAP 情報取得関数のインタフェースに関連する情報が格納される。

char *MainKey;	メインキーを格納した領域のポインタを指定する。 MAPID による検索を行う場合は、NULL を指定する。
unsigned int HashValue;	メインキーによる検索を行った場合、ハッシュ値が返却される。
int RepGrpId;	レプリケーショングループ ID が返却される。
int MapId;	MAPID を指定する。 メインキーによる検索を行う場合、0 を指定する。 メインキーによる検索の場合、MAPID が返却される。
char Status;	レプリケーショングループの状態が返却される。 DIOSA_REPGRP_ACTIVE　正常稼動 DIOSA_REPGRP_STOP　停止 DIOSA_REPGRP_ABNORMAL　障害 DIOSA_REPGRP_SWITCH_ABN　障害によるマスタ切替中 DIOSA_REPGRP_SWITCH_PLN　計画マスタ切替中
unsigned int UserAreaLen;	DIOSA 内部でのみ利用する。
unsigned int UserDataLen;	DIOSA 内部でのみ利用する。
void *UserArea;	DIOSA 内部でのみ利用する。

関連

diosagetmap

2.3.36　t_diosa_imcond(検索条件構造体)

名前

t_diosa_imcond - 検索条件構造体

書式

```
#include <diosa.h>
t_diosa_imcond cond;
```

説明

t_diosa_imcond は、**diosaimcondsetkey()**、**diosaimcondsetrange()** を使用して、検索条件を設定する構造体である。

関連

diosaimcondsetkey(), diosaimcondsetrange(), diosaimctxopen(), diosaimread1(), diosaimdeletex1()

2.3.37　t_diosa_imperfuca（性能情報構造体）

名前

t_diosa_imperfuca - 性能情報構造体

書式

```
#include <diosa.h>
t_diosa_imperfuca PerfUca;
```

説明

t_diosa_imperfuca 構造体は、ヘッダ<diosa.h>で定義されていて、その中に利用者プロセスの性能情報が格納される。

long　SvcNo;	サービス通番が返却される。
int　MapId;	収集区間内で更新系のトランザクションを実行した場合、更新対象の MAPID が返却される。複数の MAPID に対して更新をおこなった場合、最後に更新対象となった MAPID が返却される。 収集区間内で更新系のトランザクションが実行されていない場合は、0 が返却される。
struct perfdata{	要求毎の性能統計情報が配列で格納される。 配列添え字は、以下を指定する。

要求	定義
レコード取得	DIOSA_IMPERF_READ
レコード追加	DIOSA_IMPERF_WRITE
レコード更新	DIOSA_IMPERF_REWRITE
レコード削除	DIOSA_IMPERF_DELETE
全レコード削除	DIOSA_IMPERF_TRUNCATE
コミット	DIOSA_IMPERF_COMMIT
ロールバック	DIOSA_IMPERF_ROLLBACK

long Avg;	TAT 平均(ナノ秒)が返却される。
int　Cnt;	実行回数が返却される。
int　BrgCnt;	ブリッジ回数が返却される。
} PerfData[DIOSA_IMPERF_DATANUM];	
struct lockinf{	最後に発生したロック待ちまたはデッドロックの情報が返却される。 ただし、両方発生している場合はデッドロックの情報が優先される。
int　LockCnt;	ロック待ち発生回数が返却される。
int　DeadLockCnt;	デッドロック発生回数が返却される。
int　NodeId;	相手ノード ID が返却される。
int　PrcId;	相手プロセス ID が返却される。
int　ThrId;	相手スレッド ID が返却される。
int　TblId;	テーブル ID が返却される。
char PKey[32];	プライマリキー(先頭 32byte)が返却される。
} LockInf;	
struct gcinf{	グループコミットの情報が返却される。 複数回コミットをおこなった場合、最後の情報が有効となる。

unsigned char	GcSgVal;	グループコミット SG 定義値が返却される。
unsigned char	GcCollectCnt;	グループコミット纏め数が返却される。
unsigned char	GcSeq;	グループコミット順序が返却される。
unsigned char	GcTrigger;	グループコミット発生契機が返却される。 返却される値が、DIOSA_IMPERF_GC_NONE の場合はグループコミットなし、DIOSA_IMPERF_GC_CNT の場合はコミット数に達したことによるグループコミット、DIOSA_IMPERF_GC_TIME の場合は制限時間に達したことによるグループコミット、DIOSA_IMPERF_GC_QUEUE の場合はキューなし検出によるグループコミットとなる。

} GcInf;

注意

- **diosaimtruncate()**後に **diosaimgetperf()**を実行し、さらに更新系のアクセス要求 API を実行せずに **diosaimcommit()**／**diosaimrollback()**を実行した場合、その後に **diosaimgetperf()**を実行し取得した PerfUca の MapId は 0 となる。

関連

diosaimgetperf, diosatrninit

2.3.38　t_diosa_imreckeyinfo(レコードキー情報構造体)

名前

t_diosa_imreckeyinfo – レコードのキー情報構造体

書式

```
#include <diosa.h>
t_diosa_imreckeyinfo  keyinfo;
```

説明

t_diosa_imreckeyinfo は、論理表のレコードに関するキー情報を格納する構造体である。

```
int    KeyType;      キー種別として、下記の値が返却される。
                        DIOSA_KEY_PRIMARY      プライマリキー
                        DIOSA_KEY_SECONDARY    セカンダリキー
char   KeyName[31];  キー名が返却される。
ulong  KeyDivNum;    キーを構成する項目(分割)数が返却される。
struct key{          キーを構成する項目数分、以下の情報が格納される(最大 100)。
    ulong Pos;       キーを構成する項目について、レコードの先頭からのオフセットが返却される。
    ulong Len;       キーを構成する項目の長さが返却される。
} Key[100];
```

関連

diosaimgetreckeyinfo()

2.3.39　t_diosa_imrefctx(照会コンテキスト構造体)

名前

t_diosa_imrefctx – 照会コンテキストの構造体

書式

```
#include <diosa.h>
t_diosa_imrefctx refctx;
```

説明

t_diosa_imrefctx は、照会のコンテキスト情報を格納する構造体である。
本コンテキストを使用して各照会 API を実行する際､初回実行前に下記項目に-1 を設定する必要があるが、以降は、照会 API から返却された値をそのまま設定した状態で使用する。

```
int Pos1;          ポジション 1
int Pos2;          ポジション 2
int Pos3;          ポジション 3
int Pos4;          ポジション 4
```

関連

diosaimgetmaplist() , diosaimgetreckeyinfo() , diosaimgettbllist

2.3.40　t_diosa_imtbllist(論理表一覧構造体)

名前

t_diosa_imtbllist – 論理表一覧の構造体

書式

```
#include <diosa.h>
t_diosa_imtbllist tbllist;
```

説明

t_diosa_imtbllist は、論理表に関する論理表名、論理表 ID を格納する構造体である。

char　LtableName[256];　論理表名が返却される。

int　TableId;　論理表 ID が返却される。

関連

diosaimgettbllist()

2.3.41　t_diosa_mapent(MAP 情報エントリ構造体)

名前

t_diosa_mapent - MAP 情報エントリ構造体

書式

```
#include <diosa.h>
t_diosa_mapent MapEnt;
```

説明

t_diosa_mapent 構造体は、ヘッダ<diosa.h>で定義されていて、その中に MAP に関連する情報が格納される。

int RepGrpId;	レプリケーショングループ ID を返却する。	
int MapId;	MAPID を返却する。	
char Status;	レプリケーショングループの状態を返却する。	
	DIOSA_REPGRP_ACTIVE	正常稼動
	DIOSA_REPGRP_STOP	停止
	DIOSA_REPGRP_ABNORMAL	障害
	DIOSA_REPGRP_SWITCH_ABN	障害によるマスタ切替中
	DIOSA_REPGRP_SWITCH_PLN	計画マスタ切替中
char SgChgFlag	最後に実行した diimchg コマンドで SG 変更されたか否かを返却する。	
	DIOSA_ENTRY_NOCHANGE	変更なし
	DIOSA_ENTRY_UPDATE	ハッシュ値変更あり
	DIOSA_ENTRY_ADD	エントリ追加
	DIOSA_ENTRY_DELETE	エントリ削除
char LNodeName[16];	マスタ IM が存在するノード名を返却する。	
int HashEntNum;	*HashEnt 領域のエントリ数を返却する。	
t_diosa_maphashent *HashEnt;	MAP に対応するハッシュ値範囲エントリのポインタを返却する。	

注意

- diosagetmaplist()実行時は、HashEntNum と HashEnt の内容は無効である。
- diosagetmaphash()実行時は、Status、SgChgFlag、LNodeName の内容は無効である。

関連

diosagetmaplist, diosagetmaphash, t_diosa_maphashent

2.3.42　t_diosa_maphashent(MAP のハッシュ値範囲情報エントリ構造体)

名前

t_diosa_maphashent - MAP のハッシュ値範囲情報エントリ構造体

書式

```
#include <diosa.h>
t_diosa_maphashent HashEnt;
```

説明

t_diosa_maphashent 構造体は、ヘッダ<diosa.h>で定義されていて、その中に MAP のハッシュ値範囲情報が格納される。

unsigned int From;　　ハッシュ値範囲開始値を返却する。

unsigned int To;　　ハッシュ値範囲終了値を返却する。

関連

diosagetmaphash, t_diosa_mapent

2.3.43　t_diosa_recinfo(レコード情報構造体)

名前

t_diosa_recinfo - レコード情報構造体

書式

```
#include <diosa.h>
t_diosa_recinfo recinfo;
```

説明

t_diosa_recinfo は、レコードに関する情報を格納する構造体である。

char*　RecordPtr;　　レコードの先頭アドレスが格納される。

size_t RecordSize;　レコードのサイズが格納される。

char　 CtlInfo[8];　インメモリキャッシュ機能のレコード制御情報である。
DIOSA 内部の制御情報のため、この値は更新しないこと。

関連

diosaimread1(), diosaimread(), diosaimwrite(), diosaimrewrite(), diosaimdelete(), diosaimdeletex1()

2.3.44　t_diosa_tamnodeent(TAMのノード配置情報エントリ構造体)

名前

t_diosa_tamnodeent - TAMのノード配置情報エントリ構造体

書式

```
#include <diosa.h>
t_diosa_tamnodeent NodeEnt;
```

説明

t_diosa_tamnodeent構造体は、ヘッダ<diosa.h>で定義されていて、その中にTAMのノード配置情報に関連する情報が格納される。

char LNodeName[16];	論理ノード名を返却する。
char VNodeName[41];	TAMの論理ノード名を返却する。
char Type;	マスタ／スレーブ種別を返却する。 DIOSA_MASTER　マスタ DIOSA_SLAVE　スレーブ

関連

diosagettamnode

2.4　データストア基盤

2.4.1　関数一覧

(1)　ログライタ

diosadgetlog	ログデータをプールファイルから読み込む
diosadputlog	ユーザデータをログデータとしてプールファイルに書き込む
t_diosa_dloggetuca	ログデータ読み込み用の UCA
t_diosa_dloguca	ログデータ書き込み用の UCA

(2)　ログリーダ

diosalrdcommit	ログデータ処理の途中であっても DB を一旦確定させる処理を行う。

2.4.2　　diosadgetlog(ログデータ読み込み)

名前

diosadgetlog - ログデータをプールファイルから読み込む

書式

```
#include <diosa.h>
int diosadgetlog(t_diosa_dloggetuca *DlogGetUca);
```

説明

diosadgetlog()はプールファイルから指定された通番のデータを読み込む。
対象のログデータ情報は、**DlogGetUca** パラメータで指定する t_diosa_dloggetuca 構造体に設定する。

戻り値

ログデータの読み込みに成功した場合には DIOSA_DONE が返却される。失敗した場合、原因に合わせて戻り値が返却される。

DIOSA_DONE(0)	正常終了
DIOSA_SWITCH(21)	計画マスタ切替中である。
DIOSA_ERROR(-1)	その他の異常が発生した。
DIOSA_ESYS(-2)	システムコールエラー
DIOSA_EPARAM(-3)	パラメータエラー
DIOSA_ENOBUFS(-7)	読み込みバッファ領域サイズ不足
DIOSA_ENOENT(-8)	処理対象が存在しない
DIOSA_ENOINIT(-11)	ディレード転送機能が起動されていない。
DIOSA_EBUSY(-31)	メンテナンス中
DIOSA_EFUNCNAV(-34)	解凍処理ライブラリなし
DIOSA_EDEADLOCK(-36)	デッドロックを検出した(パッケージ破棄)
DIOSA_EOVERFLOW(-39)	インメモリサーバオーバーフロー
DIOSA_EDB(-69)	DB 制御ファイルのアクセスに失敗した。
DIOSA_EPOOLFILE(-93)	プールファイル矛盾
DIOSA_ECTLFILE(-100)	制御 DB 矛盾
DIOSA_ETAM(-113)	IM 制御ファイルのアクセスに失敗した。
DIOSA_ESWITCH(-115)	障害時マスタ切替中である。
DIOSA_EREADY(-118)	サーバが起動中(再起動)や環境定義変更中により、アクセスするための準備ができていない。

注意

- **diosadgetlog**()の呼び出し時は、ディレード転送が起動されている必要がある。
- CO 制御上で呼び出す場合は、CO 制御の更新対象 DB に、読み込むログデータの格納先を指定しなければならない。
- バッチ AP 制御上で呼び出す場合は、パラメータで指定する DB に、読み込むログデータの格納先を指定しなければならない。
- ユーザ AP 上で呼び出す場合は、**diosatxstart**()で指定するアクセス先に、読み込むログデータの格納先を指定しなければならない。

関連

diosaprcinit, diosathrinit, diosatrninit, diosatxstart, t_diosa_dloggetuca, t_diosa_dbinfo

2.4.3　　diosadputlog(ログデータ書き込み)

名前

diosadputlog - ユーザデータをログデータとしてプールファイルに書き込む

書式

```
#include <diosa.h>
int diosadputlog(t_diosa_dloguca *DlogUca, char *Text);
```

説明

diosadputlog()は**Text** で指定されたユーザデータをログデータとして登録し、プールファイルに書き込む。
ユーザデータを登録する際に付加するユーザデータ属性情報は **DlogUca** パラメータで指定する t_diosa_dloguca 構造体に設定する。
書き込みが正常終了した場合のログデータのディビジョン ID や通番、異常終了した場合の詳細ステータスは、**DlogUca** 領域の出力パラメータとして返却する。

戻り値

ログデータの登録に成功した場合には DIOSA_DONE が返却される。ログデータの登録に失敗した場合は、失敗した原因に合わせて戻り値が返却される。

DIOSA_DONE(0)	ログデータの登録に成功した。
DIOSA_SWITCH(21)	計画マスタ切替中である。
DIOSA_ERROR(-1)	制御 DB と共有メモリの定義情報に不一致がある。 その他の異常が発生した。
DIOSA_EPARAM(-3)	指定したスーパーストリーム名が存在しない。
DIOSA_ENOINIT(-11)	ディレード転送機能が起動されていない。
DIOSA_ESIZE(-33)	指定したユーザデータのデータ長が登録可能範囲を超えている。または異常値である。
DIOSA_EFUNCNAV(-34)	メンテナンス中に書き込みを行おうとした。または、無効化中のスーパーストリームに対して書き込みを行おうとした。 POOLFILE=RECEIVER と定義されたスーパーストリームに対して書き込みを行おうとした。
DIOSA_EDEADLOCK(-36)	デッドロックを検出した(パッケージ破棄含む)。
DIOSA_EOVERFLOW(-39)	プールファイルがオーバーフローした。
DIOSA_ENOMATCH(-40)	アクセス先と指定されたスーパーストリームのログデータ格納先が一致しない。
DIOSA_EDB(-69)	DB 制御ファイルのアクセスに失敗した。
DIOSA_ETAM(-113)	IM 制御ファイルのアクセスに失敗した。
DIOSA_ESWITCH(-115)	障害時マスタ切替中である。
DIOSA_EREADY(-118)	サーバが起動中(再起動)や環境定義変更中により、アクセスするための準備ができていない。

注意

- **diosadputlog**()の呼び出し時は、ディレード転送が起動されている必要がある。
- CO 制御上で呼び出す場合は、CO 制御の更新対象 DB に、書き込むログデータの格納先を指定しなければならない。
- バッチ AP 制御上で呼び出す場合は、パラメータで指定する DB に、書き込むログデータの格納先を指定しなければならない。

- ユーザ AP 上で呼び出す場合は、**diosatxstart()**で指定するアクセス先に、書き込むログデータの格納先を指定しなければならない。

関連

diosaprcinit, diosathrinit, diosatrninit, diosatxstart, t_diosa_dloguca, t_diosa_dbinfo

2.4.4　　diosalrdcommit(ログリーダ強制コミット)

名前

diosalrdcommit - ログデータ処理の途中であってもDBまたはIMを一旦確定させる処理を行う。

書式

```
#include <diosa.h>
int diosalrdcommit(void);
```

説明

ログデータ処理の途中であってもDBまたはIMを一旦確定させる処理(強制コミット)を行う。

戻り値

強制コミットに成功した場合にはDIOSA_DONEが返却される。強制コミットに失敗した場合は、失敗した原因に合わせて戻り値が返却される。

DIOSA_DONE(0)	強制コミットに成功した。
DIOSA_ERROR(-1)	その他の異常が発生した。
DIOSA_EDB(-69)	DB制御ファイルのアクセスに失敗した。
DIOSA_ETAM(-113)	IM制御ファイルのアクセスに失敗した。

2.4.5　　t_diosa_dloggetuca(ログデータ読み込み UCA)

名前

t_diosa_dloggetuca - ログデータ読み込み用の構造体

書式

```
#include <diosa.h>
t_diosa_dloggetuca DlogGetUca;
```

説明

t_diosa_dloggetuca は、ログデータ読み込みに必要な入出力情報を設定する構造体である。diosadgetlog() 使用時にパラメータとして使用する。メンバは以下の通りである。

char	SpstName[16]	I	スーパーストリーム名を指定する。(最大 15 文字)
int	DivId	I	ログデータのディビジョン ID を指定する。
long	DataNo	I	ログデータの通番を指定する。
void*	GetBuf	I	読み込み領域先頭アドレスを指定する。
size_t	GetBufSize	I	読み込み領域サイズを指定する。
long	UserDataNo	O	読み込んだログデータのユーザ通番を返却する。
size_t	DataSize	O	読み込んだログデータのサイズを返却する。
size_t	ShortageSize	O	領域サイズ不足時の不足サイズを返却する。
int	Dstatus	O	読み込みに失敗した場合の詳細ステータスを返却する。

関連

diosadgetlog

2.4.6　t_diosa_dloguca(ログデータ書き込みUCA)

名前

t_diosa_dloguca - ログデータ登録用の構造体

書式

```
#include <diosa.h>
t_diosa_dloguca DlogUca;
```

説明

t_diosa_dloguca はユーザデータの属性情報を設定する構造体である。diosadputlog()使用時にパラメータとして使用する。メンバは以下の通りである。

char	SpstName[16]	I	ログデータ登録先スーパーストリーム名を指定する。 環境定義でエイリアス名を定義している場合は、エイリアス名の指定が可能である。(最大 15 文字)
int	TextLen	I	ユーザデータ長を指定する。(最大 2,147,483,640 バイト)
char	CompressFlg	I	ユーザデータをプールファイルに書き込む際にデータ圧縮を行うかどうかを設定する。 DIOSA_DATACOMP_ON　: 圧縮する DIOSA_DATACOMP_OFF　: 圧縮しない DIOSA_DATACOMP_SG　: 環境定義の指定に従う
char	CoName[31]	I	CO 名を指定する。(最大 30 文字、省略可)
int	DivId	O	登録したログデータのディビジョン ID を返却する。
long	UserDataNo	O	登録したログデータのディビジョン ID 内通番を返却する。
int	Dstatus	O	登録に失敗した場合の詳細ステータスを返却する。

関連

diosadputlog

2.5　データ変換・通信オプション

2.5.1　関数一覧

(1)　DB アクセス制御

DB アクセス API

diatcdbcondsetkey	検索条件を構築する（キー指定）
diatcdbcondsetrange	検索条件を構築する（範囲指定）
diatcdbctxclose	検索コンテキストを破棄する
diatcdbctxopen	検索コンテキストを生成する
diatcdbdelete	レコードを削除する
diatcdbgetrplmode	更新ログ登録要否モードを取得する
diatcdbgettblid	テーブル ID を取得する
diatcdbread	レコードを取得する（複数件）
diatcdbread1	レコードを取得する（１件ごと、主キーの完全一致）
diatcdbrewrite	レコードを更新する
diatcdbsetrplmode	更新ログ登録要否モードを設定する
diatcdbtruncate	レコードを全件削除する
diatcdbwrite	レコードを追加する

ユーザデータ状態管理 API

diatcmkadd	ユーザデータ状態管理表にメインキーを登録する
diatcmkdelete	ユーザデータ状態管理表からメインキーを削除する
diatcmkgettblid	ユーザデータ状態管理表のテーブル ID を取得する
diatcmkread	ユーザデータ状態管理表からデータ登録状態を取得する

構造体

t_diatc_dbcond	検索条件構造体
t_diatc_recinfo	レコード情報構造体
t_diatc_dbaccinfo	データベースアクセス情報構造体

2.5.2　　diatcdbcondsetkey(キー指定検索条件設定関数)

名前

diatcdbcondsetkey - 検索条件を構築する (キー指定)

書式

```
#include <diosa.h>
int diatcdbcondsetkey(t_diatc_dbcond *Cond, char *Value, int ValueLen);
```

説明

diatcdbcondsetkey() は、キー指定でのレコード取得、またはレコード削除を行う時に、検索キーを設定するための関数である。
diatcdbcondsetkey() は、**Value** と **ValueLen** により構築した検索条件を **Cond** に設定する。
Value には検索キー、**ValueLen** は検索キー長を指定する。
但し、完全一致検索を行う場合はプライマリキー長を **ValueLen** に指定し、前方一致検索を行う場合は一致させたいところまでのキー長を **ValueLen** に指定する。

戻り値

本 API は、以下の戻り値を返却する。

DIOSA_DONE(0)	正常終了。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータ誤り。
DIOSA_EINVAL(-9)	パラメータ値不正。

注意

- 本関数で構築した Cond で前方一致検索を行う場合は diatcdbread() を呼び出すこと。

関連

diatcdbread1(), diatcdbctxopen(), diatcdbread(), t_diatc_dbcond

2.5.3　　diatcdbcondsetrange(範囲指定検索条件設定関数)

名前

diatcdbcondsetrange - 検索条件を構築する (範囲指定)

書式

```
#include <diosa.h>
int diatcdbcondsetrange(t_diatc_dbcond *Cond, int Operator1, char *MinValue, int MinValueLen,
int Operator2, char *MaxValue, int MaxValueLen);
```

説明

diatcdbcondsetrange () は、範囲指定による検索でレコード取得を行う時に、検索条件を設定するための関数である。

diatcdbcondsetrange() は、**Operator1**、**MinValue**、**MinValueLen**、**Operator2**、**MaxValue**、**MaxValueLen** により検索条件を構築して、**Cond** に設定する。**Operator1**、**MinValue**、**MinValueLen** には下限値を指定し、**Operator2**、**MaxValue**、**MaxValueLen** には上限値を指定する。

Operator1 には以下の演算子を指定できる。

DIOSA_COND_GT	>
DIOSA_COND_GE	>=

Operator2 には、以下の演算子を指定できる。

DIOSA_COND_LT	<
DIOSA_COND_LE	<=

MinValue、**MaxValue** には検索キーの値、**MinValueLen**、**MaxValueLen** には検索キーの長さを指定する。この検索キーの長さは、環境定義に指定したキー長と同じ長さである必要がある。

下限のみ(または、上限のみ)の条件を構築する場合は、**Operator2** (または、**Operator1**) に DIOSA_COND_NOP を指定する。

戻り値

本 API は、以下の戻り値を返却する。

DIOSA_DONE(0)	正常終了。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータ誤り。
DIOSA_EINVAL(-9)	パラメータ値不正。

注意

- 検索キーはホストバイトオーダで作成すること。
- DB アクセスモードでは、DATE 型・バイナリ型の範囲指定検索を行うことはできない。

関連

t_diatc_dbcond,　diatcdbctxopen(),　diatcdbread()

2.5.4　　diatcdbctxclose(検索コンテキストクローズ関数)

名前

diatcdbctxclose - 検索コンテキストを破棄する

書式

```
#include <diosa.h>
int diatcdbctxclose(int CtxId);
```

説明

diatcdbctxclose() は、**diatcdbctxopen()** で生成された検索コンテキスト **CtxId** を無効の状態にする。

戻り値

本 API は、以下の戻り値を返却する。

DIOSA_DONE(0)	正常終了。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータ誤り。
DIOSA_EMEM(-6)	メモリ操作に失敗した。
DIOSA_ETIMEOUT(-22)	要求がタイムアウトした。
DIOSA_ESTATE(-114)	**CtxId** に指定された検索コンテキストが **diatcdbctxopen()** で生成されていない。

関連

diatcdbctxopen()

2.5.5　diatcdbctxopen(検索コンテキストオープン関数)

名前

diatcdbctxopen - 検索コンテキストを生成する

書式

```
#include <diosa.h>
int diatcdbctxopen(t_diatc_dbaccinfo *AccInfo, int TableId, char *IndexName, t_diatc_dbcond *Cond,
int Order, int Lock, int *CtxId)
```

説明

diatcdbctxopen() は、パラメータで指定された情報から検索コンテキストを生成し、その識別子 **CtxId** を返却する。生成した検索コンテキストを指定して **diatcdbread()** を呼び出すことによって、検索条件に合致するレコードを取得する。
AccInfo は、データベースアクセス情報構造体へのポインタを指定する。
TableId は検索対象のテーブル ID、**IndexName** は検索対象のインデックス名を指定する。
Cond は **diatcdbcondsetkey()**、または **diatcdbcondsetrange()** で構築した検索条件構造体を指定する。
また、**IndexName** と **Cond** に NULL を指定した場合は、全件検索となる。
Order は検索方向で、DIOSA_ASCEND（昇順）を指定する。
Lock は排他オプションで、DIOSA_NOLOCK（排他なし）を指定する。
不要になった検索コンテキストは、**diatcdbctxclose()** を実行して必ず破棄しなければならない。

戻り値

本 API は、以下の戻り値を返却する。

DIOSA_DONE(0)	正常終了。
DIOSA_SWITCH(21)	計画マスタ切替中である。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータ誤り。
DIOSA_EMEM(-6)	メモリ操作に失敗した。
DIOSA_EINVAL(-9)	パラメータ値不正。または、**TableId** で指定されたテーブルが存在しない。
DIOSA_ERANGE(-10)	範囲指定検索条件値が不正。
DIOSA_EFUNCNAV(-34)	未サポート機能を指定した。 更新 DB タイプと運用モードが一致しない。
DIOSA_ECONFLICT(-110)	キーサイズが定義情報と不一致。
DIOSA_ESTATE(-114)	検索条件 **Cond** が **diatcdbcondsetkey()**、**diatcdbcondsetrange()** のいずれかで構築されていない。または、**diosatrninit()** が未実施である。
DIOSA_ESWITCH(-115)	障害時マスタ切替中である。
DIOSA_EREADY(-118)	インメモリサーバの準備ができていない。

関連

diatcdbread(), diatcdbctxclose(), diatcdbcondsetkey(), diatcdbcondsetrange(), diatcdbgettblid(), t_diatc_dbcond

2.5.6　diatcdbdelete(レコード削除関数)

名前

diatcdbdelete - レコードを削除する（１件ごと）

書式

```
#include <diosa.h>
int diatcdbdelete(t_diatc_dbaccinfo *AccInfo, int TableId, t_diatc_recinfo *RecInfo);
```

説明

diatcdbdelete() は、**TableId** に指定されたテーブルから **RecInfo** で指定された１件のレコードを削除する。
事前に **diatcdbread1()** で削除対象のレコードをロックして読み込む必要がある。
RecInfo には、**diatcdbread1()** で取得したレコード情報構造体のポインタを指定する。
本 API の実行時、処理対象のレコードをトランザクションがコミットまたはロールバックされる時までロックする。
本 API の実行時、処理対象のレコードが既にロックされている場合は、ロックが解除されるまで待機し続ける。但し、インメモリサーバへのアクセスで応答監視タイマ(環境定義 IMENV 節-USERAP 項-REQTIMEOUT で定義したもの)の時間を経過してもロックが解放されない場合は、DIOSA_ETIMEOUT を返却する。

t_diatc_dbaccinfo *AccInfo(入力)　　データベースアクセス情報構造体へのポインタ。**AccInfo** 構造体のメンバー **Target** により、IM と DB から削除、もしくは IM からのみ削除を選択する。
int TableId(入力)　　**diatcdbgettblid()** で取得するテーブル ID。
t_diatc_recinfo *RecInfo(入力)　　レコード情報構造体へのポインタ。

戻り値

本 API は、以下の戻り値を返却する。

DIOSA_DONE(0)	正常終了。
DIOSA_SWITCH(21)	計画マスタ切替中である。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータ誤り。
DIOSA_EMEM(-6)	メモリ操作に失敗した。
DIOSA_EINVAL(-9)	パラメータ値不正。または、**TableId** で指定されたテーブルが存在しない。または、**AccInfo** で指定したメインキーがレコード追加時に指定したメインキーと異なる。
DIOSA_ELOCK(-14)	ロック制御に失敗した。
DIOSA_ETIMEOUT(-22)	要求がタイムアウトした。
DIOSA_EACCES(-25)	事前に **diatcdbread1()** が排他指定で呼ばれていない。または、アクセス対象の表がオープンされていない。または、IM の更新ログ出力機能が準備できていないため更新アクセスできない。
DIOSA_ESIZE(-33)	レコードサイズが不正である。
DIOSA_EFUNCNAV(-34)	更新 DB タイプと運用モードが一致しない。
DIOSA_ECOND(-60)	同一トランザクション内で **diatcdbtruncate()**が実行されている。
DIOSA_EDB(-69)	IM または DB へのアクセス失敗。
DIOSA_ESG(-77)	環境定義に矛盾がある。

DIOSA_ECONFLICT(-110)	レコードサイズが定義情報と不一致。
DIOSA_ESTATE(-114)	同一トランザクション内で別 MAP への更新系のアクセス要求 API が実行されている。
DIOSA_ESWITCH(-115)	障害時マスタ切替中である。
DIOSA_EREADY(-118)	インメモリサーバの準備ができていない。

関連

t_diatc_dbaccinfo, t_diatc_recinfo, diatcdbgettblid(), diatcdbread1()

2.5.7　diatcdbgetrplmode(更新ログ登録要否モード取得関数)

名前

diatcdbgetrplmode - 更新ログ登録要否モードを取得する

書式

```
#include <diosa.h>
int diatcdbgetrplmode(int *Rplmode);
```

説明

diatcdbgetrplmode() は、DB アクセス API による更新を行う際の更新ログ登録要否モードを **Rplmode** で指定された領域に格納する。

int *Rplmode(出力)　　更新ログ登録要否モード格納先。

DIATC_REPLICATION_MODE_NORMAL	通常モード(規定値、環境定義に従う)
DIATC_REPLICATION_MODE_NONE	更新ログ抑止モード(レプリケーションなし)
DIATC_REPLICATION_MODE_ASYNC	更新ログ登録モード(非同期型レプリケーション)

戻り値

本 API は、以下の戻り値を返却する。

DIOSA_DONE(0)	正常終了。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータ誤り。

注意

- 本 API で取得した更新ログ登録要否モードは、本 API を実行したトランザクション内でのみ有効となる。
- 本 API が異常終了した場合、**Rplmode** に格納される値は不定となる。
- 環境変数 DIATC_CENTER_ID が未設定の場合、更新ログ登録要否モードに関わらず更新ログを登録しない。

関連

diatcdbsetrplmode()

2.5.8　　diatcdbgettblid(テーブル ID 照会関数)

名前

diatcdbgettblid - テーブル名に対応するテーブル ID の取得

書式

```
#include <diosa.h>
int diatcdbgettblid(t_diatc_dbaccinfo *AccInfo, char *LogicalTableName, int *TableId);
```

説明

diatcdbgettblid() は、テーブル名 **LogicalTableName** に対応するテーブル ID を **TableId** に格納する。

戻り値

DIOSA_DONE(0)	正常終了した。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータ誤り。
DIOSA_ENOENT(-8)	指定されたテーブル名が存在しない。
DIOSA_EINVAL(-9)	パラメータ値不正。

関連

diatcdbread1(), diatcdbread(), diatcdbwrite(), diatcdbrewrite(), diatcdbdelete()

2.5.9　diatcdbread(複数レコード読込関数)

名前

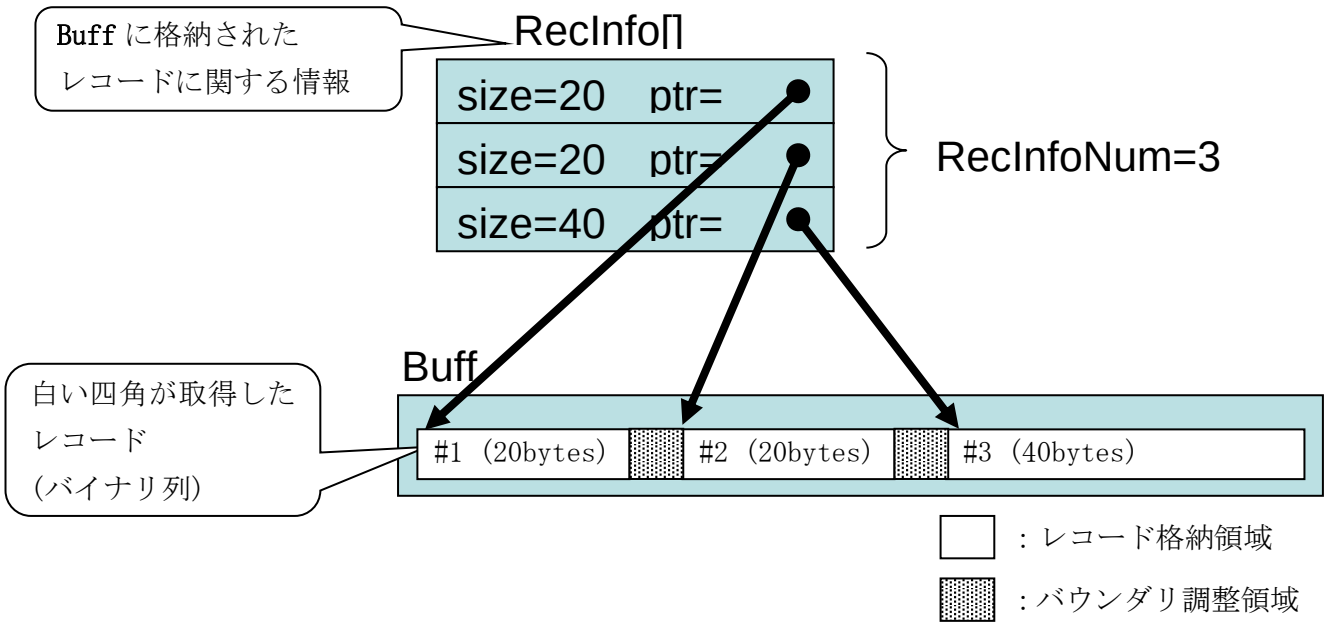
diatcdbread - レコードを取得する（複数件）

書式

```
#include <diosa.h>
int diatcdbread(t_diatc_dbaccinfo *AccInfo, int CtxId, t_diatc_recinfo *RecInfo, int RecInfoNum,
int *FetchedNum, char *Buff, size_t BuffSize);
```

説明

diatcdbread()は、検索コンテキスト **CtxId** に対応した検索条件に合致するレコードを複数件取得する。
取得したいレコード数を **RecInfoNum** に設定する。本関数を実行した結果、実際に取得したレコード数は **FetchedNum** に格納される。
各レコードに関する情報が格納される **RecInfo** は、**RecInfoNum** 分のレコードを格納できる領域を利用者が用意する。
取得したレコードが格納されるバッファ **Buff** も利用者が用意する。ただし、バッファの大きさ **BuffSize** は、レコード１件あたりの最大長と、レコード数 **RecInfoNum** を考慮して算出する必要がある。
検索条件に合致するレコードの数が **RecInfoNum** よりも多い場合は、同じ **CtxId** を指定して本 API を続けて実行することで、全件を取得することができる。
取得したレコードとレコードに関する情報は、指定したバッファ **Buff** とレコード情報構造体 **RecInfo** に返却する。このとき、取得したレコード１件目はバッファ **Buff** の先頭から格納し、２件目以降を格納する際は８バイトのバウンダリ調整を行う。



t_diatc_dbaccinfo *AccInfo(入力)	データベースアクセス情報構造体へのポインタ。
int CtxId(入力)	**diatcdbctxopen**() で生成した検索コンテキストの識別子。
t_diatc_recinfo *RecInfo(出力)	レコード情報構造体へのポインタ。
int RecInfoNum(入力)	取得するレコード数。
int *FetchedNum(出力)	実際に取得したレコード数。
char *Buff(出力)	取得したレコードが格納されるバッファへのポインタ。
size_t BuffSize(入力)	バッファ **Buff** の大きさ。

戻り値

本 API は、以下の戻り値を返却する。

DIOSA_DONE(0)	正常終了。条件に合致するレコードを取得した。

DIOSA_NOENT(20)	該当レコードが読込対象の表に存在しない。または条件に合致するレコードを全て取得し終わった。
DIOSA_SWITCH(21)	計画マスタ切替中である。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータ誤り。
DIOSA_EMEM(-6)	メモリ操作に失敗した。
DIOSA_ENOBUFS(-7)	**Buff** に指定された領域に取得したレコードが 1 件も入らない。
DIOSA_EINVAL(-9)	パラメータ値不正。または、読込対象の表が存在しない。
DIOSA_ELOCK(-14)	ロック制御に失敗した。
DIOSA_ETIMEOUT(-22)	要求がタイムアウトした。
DIOSA_EACCES(-25)	アクセス対象の表がオープンされていない。または、IM の更新ログ出力機能が準備できていないため更新アクセスできない。
DIOSA_EDB(-69)	IM または DB へのアクセス失敗。
DIOSA_ESG(-77)	環境定義に矛盾がある。
DIOSA_ECONFLICT(-110)	レコードサイズが定義情報と不一致。
DIOSA_ESTATE(-114)	指定した **Ctx** が **diatcdbctxopen()** 未実施である。
DIOSA_ESWITCH(-115)	障害時マスタ切替中である。
DIOSA_EREADY(-118)	インメモリサーバの準備ができていない。

注意

- diatcdbread() を複数回呼び出して結果を取得する間に検索対象キーの値を更新すると読込結果が不定になる場合がある。「2.3.23 diosaimread(複数レコード読込関数)」の「注意」を参照のこと。

関連

t_diatc_dbaccinfo, t_diatc_recinfo, diatcdbgettblid(), diatcdbcondsetkey(), diatcdbcondsetrange(), diatcdbctxopen(), diatcdbctxclose()

2.5.10　diatcdbread1(キー指定レコード読込関数)

名前

diatcdbread1 - レコードを取得する（１件ごと、主キーの完全一致)

書式

```
#include <diosa.h>
int diatcdbread1(t_diatc_dbaccinfo *AccInfo, int TableId, t_diatc_dbcond *Cond, t_diatc_recinfo *RecInfo, char *Buff, size_t BuffSize, int Lock);
```

説明

diatcdbread1() は **TableId** で指定されたテーブルから **Cond** で指定されたプライマリキー値に一致するレコードを１件取得する。
取得したレコードが格納されるバッファ **Buff** と、そのレコード情報が格納される **RecInfo** は、１レコード分を格納するための領域を利用者が用意する。
Cond には、**diatcdbcondsetkey()** を実行してプライマリキーを設定する。
Lock には、排他オプションとして以下の値のいずれかを指定する。

DIOSA_NOLOCK	ロックせずに読込を行う。他トランザクションが既に読込対象レコードをロックしていても、待ち合わせすることなく読込を行う。なお、同時に他トランザクションが読込対象レコードを更新している場合は、更新前のレコードを読み込む。
DIOSA_LOCK_WAIT	ロックして読込を行う。他トランザクションで既に読込対象レコードがロックされている場合は、ロックが解放されるまで待ち合わせる。但し、インメモリサーバへのアクセスで応答監視タイマ(環境定義 IMENV 節-USERAP 項-REQTIMEOUT で定義したもの)の時間を経過してもロックが解放されない場合は、DIOSA_ETIMEOUT を返却する。
DIOSA_LOCK_NOWAIT	ロックして読込を行う。他トランザクションで既に読込対象レコードがロックされている場合は、待ち合わせをせずにエラーとなる。

t_diatc_dbaccinfo *AccInfo(入力)	データベースアクセス情報構造体へのポインタ。
int TableId(入力)	**diatcdbgettblid()** で取得するテーブル ID。
t_diatc_dbcond *Cond(入力)	**diatcdbcondsetkey()** で構築する検索条件構造体へのポインタ。
t_diatc_recinfo *RecInfo(出力)	レコード情報構造体へのポインタ。
char *Buff(出力)	取得したレコードが格納されるバッファへのポインタ。
size_t BuffSize(入力)	バッファ **Buff** の大きさ。
int Lock(入力)	排他オプション。

戻り値

本 API は、以下の戻り値を返却する。

DIOSA_DONE(0)	正常終了。
DIOSA_NOENT(20)	該当レコードなし。
DIOSA_SWITCH(21)	計画マスタ切替中である。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータ誤り。または、**Cond** に設定された検索キー長がプライマリキー長と異なる。
DIOSA_EMEM(-6)	メモリ操作に失敗した。
DIOSA_ENOBUFS(-7)	**Buff** に指定された領域に取得したレコードが入らない。

DIOSA_EINVAL(-9)	パラメータ値不正。または、**TableId** で指定されたテーブルが存在しない。または、**AccInfo** で指定したメインキーがレコード追加時に指定したメインキーと異なる。
DIOSA_ELOCK(-14)	ロック制御に失敗した。
DIOSA_ETIMEOUT(-22)	要求がタイムアウトした。
DIOSA_EACCES(-25)	アクセス対象の表がオープンされていない。または、IM の更新ログ出力機能が準備できていないため更新アクセスできない。
DIOSA_EBUSY(-31)	既に該当レコードがロックされている。
DIOSA_EFUNCNAV(-34)	更新 DB タイプと運用モードが一致しない。
DIOSA_EDEADLOCK(-36)	デッドロックが発生した。
DIOSA_ECOND(-60)	同一トランザクション内で **diatcdbtruncate()** が実行されている。(**Lock** に DIOSA_LOCK_WAIT、DIOSA_LOCK_NOWAIT のいずれかを指定した場合)
DIOSA_EDB(-69)	IM または DB へのアクセス失敗。
DIOSA_ESG(-77)	環境定義に矛盾がある。
DIOSA_ECONFLICT(-110)	レコードまたはキーサイズが定義情報と不一致。
DIOSA_ESTATE(-114)	**diatctxstart()** が未実行である。検索条件 **Cond** が **diatcdbcondsetkey()** で構築されていない。または、**Lock** に DIOSA_LOCK_WAIT、DIOSA_LOCK_NOWAIT のいずれかを指定した場合は、同一トランザクション内で別 MAP への更新系のアクセス要求 API が実行されている。
DIOSA_ESWITCH(-115)	障害時マスタ切替中である。
DIOSA_EREADY(-118)	インメモリサーバの準備ができていない。

関連

t_diatc_dbaccinfo, t_diatc_dbcond, t_diatc_recinfo, diatcdbgettblid(), diatcdbcondsetkey(), diatctxstart()

2.5.11　diatcdbrewrite(レコード更新関数)

名前

diatcdbrewrite - レコードを更新する（１件ごと）

書式

```
#include <diosa.h>
int diatcdbrewrite(t_diatc_dbaccinfo *AccInfo, int TableId, t_diatc_recinfo *RecInfo);
```

説明

diatcdbrewrite() は、**TableId** に指定されたテーブルに **RecInfo** で指定された１件のレコードを更新する。
事前に **diatcdbread1()** で更新対象のレコードをロックして読み込む必要がある。
RecInfo には、**diatcdbread1()** で取得したレコード情報構造体のポインタを指定する。

t_diatc_dbaccinfo *AccInfo(入力)	データベースアクセス情報構造体へのポインタ。
int TableId(入力)	**diatcdbgettblid()** で取得するテーブル ID。
t_diatc_recinfo *RecInfo(入力)	レコード情報構造体へのポインタ。

戻り値

本 API は、以下の戻り値を返却する。

DIOSA_DONE(0)	正常終了。
DIOSA_SWITCH(21)	計画マスタ切替中である。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータ誤り。
DIOSA_EMEM(-6)	メモリ操作に失敗した。
DIOSA_EINVAL(-9)	パラメータ値不正。または、**TableId** で指定されたテーブルが存在しない。または、**AccInfo** で指定したメインキーがレコード追加時に指定したメインキーと異なる。
DIOSA_ELOCK(-14)	ロック制御に失敗した。
DIOSA_ETIMEOUT(-22)	要求がタイムアウトした。
DIOSA_EACCES(-25)	事前に **diatcdbread1()** が排他指定で呼ばれていない。または、アクセス対象の表がオープンされていない。または、IM の更新ログ出力機能が準備できていないため更新アクセスできない。
DIOSA_ESIZE(-33)	レコードサイズが不正である。
DIOSA_EFUNCNAV(-34)	更新 DB タイプと運用モードが一致しない。
DIOSA_ECOND(-60)	同一トランザクション内で **diatcdbtruncate()**が実行されている。
DIOSA_EDB(-69)	IM または DB へのアクセス失敗。
DIOSA_ESG(-77)	環境定義に矛盾がある。
DIOSA_ECONFLICT(-110)	レコードサイズが定義情報と不一致。
DIOSA_ESTATE(-114)	同一トランザクション内で別 MAP への更新系のアクセス要求 API が実行されている。
DIOSA_ESWITCH(-115)	障害時マスタ切替中である。
DIOSA_EREADY(-118)	インメモリサーバの準備ができていない。

注意

- レコードイメージ中の数値データはホストバイトオーダで作成すること。

関連

t_diatc_dbaccinfo, t_diatc_recinfo, diatcdbgettblid(), diatcdbread1()

2.5.12　diatcdbsetrplmode(更新ログ登録要否モード設定関数)

名前

diatcdbsetrplmode - 更新ログ登録要否モードを設定する

書式

```
#include <diosa.h>
int diatcdbsetrplmode(int Rplmode);
```

説明

diatcdbsetrplmode() は、DB アクセス API による更新を行う際の更新ログ登録要否モードを **Rplmode** で指定された値に変更する。

Rplmode に DIATC_REPLICATION_MODE_NORMAL を指定した場合、以降の更新系 DB アクセス API 実行時は、環境定義 DACENV 節-TABLE 項の REPLICATION パラメータで定義されたテーブル毎のレプリケーション種別に従い、更新ログの登録要否を決定する。

Rplmode に DIATC_REPLICATION_MODE_NORMAL 以外を指定した場合､以降の更新系 DB アクセス API 実行時は、環境定義 DACENV 節-TABLE 項の REPLICATION パラメータで定義されたテーブル毎のレプリケーション種別に依らず、本 API で設定した値により更新ログの登録要否を決定する。

int Rplmode(入力)	更新ログ登録要否モード。
DIATC_REPLICATION_MODE_NORMAL	通常モード(規定値、環境定義に従う)
DIATC_REPLICATION_MODE_NONE	更新ログ抑止モード(レプリケーションなし)
DIATC_REPLICATION_MODE_ASYNC	更新ログ登録モード(非同期型レプリケーション)

戻り値

本 API は、以下の戻り値を返却する。

DIOSA_DONE(0)	正常終了。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EINVAL(-9)	パラメータ値不正。

注意

- 本 API で設定した更新ログ登録要否モードは、本 API を実行したトランザクション内でのみ有効となる。
- 本 API を実行していないトランザクションの更新ログ登録要否モードは「DIATC_REPLICATION_MODE_NORMAL(通常モード)」となる。
- 環境変数 DIATC_CENTER_ID が未設定の場合、更新ログ登録要否モードに関わらず更新ログを登録しない。

関連

diatcdbgetrplmode()

2.5.13　diatcdbtruncate(レコード全件削除関数)

名前

diatcdbtruncate - レコードを全件削除する

書式

```
#include <diosa.h>
int diatcdbtruncate(t_diatc_dbaccinfo *AccInfo, int TableId);
```

説明

diatcdbtruncate() は、**TableId** で指定されたテーブルから、**AccInfo** で指定された MAP に該当するレコードを全件削除する。

t_diatc_dbaccinfo *AccInfo(入力)　　データベースアクセス情報構造体へのポインタ。
int TableId(入力)　　**diatcdbgettblid()** で取得するテーブル ID。

戻り値

本 API は、以下の戻り値を返却する。

DIOSA_DONE(0)	正常終了。
DIOSA_BLOCK(6)	該当 MAP が閉塞している。
DIOSA_SWITCH(21)	計画マスタ切替中である。
DIOSA_ERROR(-1)	異常終了。
DIOSA_ESYS(-2)	システムコールエラーが発生した。
DIOSA_EPARAM(-3)	パラメータ誤り。
DIOSA_EMEM(-6)	メモリ操作に失敗した。
DIOSA_EINVAL(-9)	パラメータ値不正。または、**TableId** で指定されたテーブルが存在しない。
DIOSA_ETIMEOUT(-22)	要求がタイムアウトした。
DIOSA_EACCES(-25)	アクセス対象の表がオープンされていない。または、IM の更新ログ出力機能が準備できていないため更新アクセスできない。
DIOSA_EBUSY(-31)	他トランザクションで削除対象の表のレコードをロックしている。
DIOSA_EFUNCNAV(-34)	機能利用不可。 更新 DB タイプと運用モードが一致しない。
DIOSA_EOVERFLOW(-39)	IMS キューバッファに空きがない。
DIOSA_EMSGQ(-41)	メッセージキューの障害が発生した。
DIOSA_ECOND(-60)	同一トランザクション内で **diatcdbread1()** (**DIOSA_LOCK_WAIT**、**DIOSA_LOCK_NOWAIT** のいずれかを指定)、**diatcdbwrite()**、**diatcdbrewrite()**、**diatcdbdelete()**、**diatcmkread()** (**DIOSA_LOCK_WAIT**、**DIOSA_LOCK_NOWAIT** のいずれかを指定)、**diatcmkadd()**、**diatcmkdelete()**が既に実行されている。
DIOSA_EDB(-69)	DB の処理が異常終了した。
DIOSA_ESOCKET(-70)	ソケット送受信で障害が発生した。
DIOSA_ECONNECT(-71)	通信パスが切断された。
DIOSA_ETAM(-113)	IM の障害によりレコード削除に失敗した。

DIOSA_ESTATE(-114)　**diatctxstart()** が未実行である。または、同一トランザクション内で別 MAP への更新系のアクセス要求 API が実行されている。

DIOSA_ESWITCH(-115)　障害時マスタ切替中である。

DIOSA_EINACTIVE(-117)　　レプリケーショングループが停止している。

DIOSA_EREADY(-118)　インメモリサーバの準備ができていない。

注意

- 本 API は DB アクセスモードでは使用できない。
- 正常終了した場合は、**diosarollback()**によるキャンセルは出来ない。
- 異常終了(負値)が返却された場合は、必ず **diosarollback()**を実行すること。但し、**DIOSA_ETIMEOUT** が返却された場合、レコード全件削除要求が確定している可能性があり、その場合はキャンセル出来ない。
- 当 API を使用する場合､同一トランザクション内で **diatcdbread1()** (**DIOSA_LOCK_WAIT**､**DIOSA_LOCK_NOWAIT** のいずれかを指定)、**diatcdbwrite()**、**diatcdbrewrite()**、**diatcdbdelete()**、**diatcmkread()** (**DIOSA_LOCK_WAIT**､**DIOSA_LOCK_NOWAIT** のいずれかを指定)､**diatcmkadd()**､**diatcmkdelete()**を使用することはできない。

関連

t_diatc_dbaccinfo,　diatcdbgettblid()

2.5.14　diatcdbwrite(レコード追加関数)

名前

diatcdbwrite - レコードを追加する（１件ごと）

書式

```
#include <diosa.h>
int diatcdbwrite(t_diatc_dbaccinfo *AccInfo, int TableId, t_diatc_recinfo *RecInfo);
```

説明

diatcdbwrite() は、**TableId** に指定されたテーブルに **RecInfo** で指定された１件のレコードを追加する。
RecInfo で指定されたレコードが既にテーブル上に存在した場合はエラーとなる。
本 API の実行時、処理対象のレコードをトランザクションがコミットまたはロールバックされる時までロックする。
本 API の実行時、処理対象のレコードが既にロックされている場合は、ロックが解除されるまで待機し続ける。但し、インメモリサーバへのアクセスで応答監視タイマ(環境定義 IMENV 節-USERAP 項-REQTIMEOUT で定義したもの)の時間を経過してもロックが解放されない場合は、DIOSA_ETIMEOUT を返却する。

t_diatc_dbaccinfo *AccInfo(入力)	データベースアクセス情報構造体へのポインタ。
int TableId(入力)	**diatcdbgettblid()** で取得するテーブル ID。
t_diatc_recinfo *RecInfo(入力)	レコード情報構造体へのポインタ。

戻り値

本 API は、以下の戻り値を返却する。

DIOSA_DONE(0)	正常終了。
DIOSA_SWITCH(21)	計画マスタ切替中である。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータ誤り。
DIOSA_EMEM(-6)	メモリ操作に失敗した。
DIOSA_EINVAL(-9)	パラメータ値不正。または、**TableId** で指定されたテーブルが存在しない。
DIOSA_ELOCK(-14)	ロック制御に失敗した。
DIOSA_ETIMEOUT(-22)	要求がタイムアウトした。
DIOSA_EACCES(-25)	アクセス対象の表がオープンされていない。または、IM の更新ログ出力機能が準備できていないため更新アクセスできない。
DIOSA_EEXIST(-27)	指定されたレコードがすでに存在する。
DIOSA_ESIZE(-33)	レコードサイズが不正である。
DIOSA_EFUNCNAV(-34)	更新 DB タイプと運用モードが一致しない。
DIOSA_EDEADLOCK(-36)	デッドロックが発生した。
DIOSA_ECOND(-60)	同一トランザクション内で **diatcdbtruncate()**が実行されている。
DIOSA_EDB(-69)	IM または DB へのアクセス失敗。
DIOSA_ESG(-77)	環境定義に矛盾がある。
DIOSA_ECONFLICT(-110)	レコードサイズが定義情報と不一致。
DIOSA_ESTATE(-114)	**diatctxstart()** が未実行である。または、同一トランザクション内で別 MAP への更新系のアクセス要求 API が実行されている。

DIOSA_ESWITCH(-115)　　障害時マスタ切替中である。

DIOSA_EREADY(-118)　　インメモリサーバの準備ができていない。

注意

- 一つの論理表内でプラリマリキーは一意である必要がある。指定されたレコードと同じプライマリキーのレコードがアクセス先の MAP 以外に存在しても本関数は DIOSA_EEXIST を返却しないので注意されたい。
- 環境定義 DACENV 節において属性が CHAR のプライマリキー項目には空文字列を格納できない。
- レコードイメージ中の数値データはホストバイトオーダで作成すること。

関連

t_diatc_dbaccinfo,　t_diatc_recinfo,　diatcdbgettblid(),　diatctxstart()

2.5.15　diatcmkadd(メインキー登録関数)

名前

diatcmkadd - ユーザデータ状態管理表にメインキーを登録する

書式

```
#include <diosa.h>
int diatcmkadd(t_diatc_dbaccinfo *AccInfo, int Status);
```

説明

diatcmkadd() はユーザデータ状態管理表に **AccInfo** で指定されたメインキーのレコードを追加する。
AccInfo で指定されたメインキーのレコードが既にユーザデータ状態管理表に存在した場合はエラーとなる。
本 API の実行時、処理対象のレコードをトランザクションがコミットまたはロールバックされる時までロックする。
本 API の実行時、処理対象のレコードが既にロックされている場合は、ロックが解除されるまで待機し続ける。但し、インメモリサーバへのアクセスで応答監視タイマ(環境定義 IMENV 節-USERAP 項-REQTIMEOUT で定義したもの)の時間を経過してもロックが解放されない場合は、DIOSA_ETIMEOUT を返却する。

t_diatc_dbaccinfo *AccInfo(入力)	データベースアクセス情報構造体へのポインタ。
int Status(入力)	データ登録状態。DIATC_STATUS_TAM_ORACLE を指定する。
DIATC_STATUS_TAM_ORACLE	IM と DB

戻り値

本 API は、以下の戻り値を返却する。

DIOSA_DONE(0)	正常終了。
DIOSA_SWITCH(21)	計画マスタ切替中である。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータ誤り。
DIOSA_EMEM(-6)	メモリ操作に失敗した。
DIOSA_EINVAL(-9)	パラメータ値不正。
DIOSA_ELOCK(-14)	ロック制御に失敗した。
DIOSA_ETIMEOUT(-22)	要求がタイムアウトした。
DIOSA_EACCES(-25)	アクセス対象の表がオープンされていない。または、IM の更新ログ出力機能が準備できていないため更新アクセスできない。
DIOSA_EEXIST(-27)	指定されたレコードがすでに存在する。
DIOSA_EFUNCNAV(-34)	更新 DB タイプと運用モードが一致しない。
DIOSA_EDEADLOCK(-36)	デッドロックが発生した。
DIOSA_ECOND(-60)	同一トランザクション内で **diatcdbtruncate()**が実行されている。
DIOSA_EDB(-69)	IM または DB へのアクセス失敗。
DIOSA_ESTATE(-114)	**diatctxstart()** が未実行である。または、同一トランザクション内で別 MAP への更新系のアクセス要求 API が実行されている。
DIOSA_ESWITCH(-115)	障害時マスタ切替中である。
DIOSA_EREADY(-118)	インメモリサーバの準備ができていない。

関連

t_diatc_dbaccinfo

2.5.16　diatcmkdelete(メインキー削除関数)

名前

diatcmkdelete - ユーザデータ状態管理表からメインキーを削除する

書式

```
#include <diosa.h>
int diatcmkdelete(t_diatc_dbaccinfo *AccInfo);
```

説明

diatcmkdelete() はユーザデータ状態管理表から **AccInfo** で指定されたメインキーのレコードを削除する。
AccInfo で指定されたメインキーのレコードがテーブル上に存在しなかった場合はエラーとなる。
本 API の実行時、処理対象のレコードをトランザクションがコミットまたはロールバックされる時までロックする。
本 API の実行時、処理対象のレコードが既にロックされている場合は、ロックが解除されるまで待機し続ける。但し、インメモリサーバへのアクセスで応答監視タイマ(環境定義 IMENV 節-USERAP 項-REQTIMEOUT で定義したもの)の時間を経過してもロックが解放されない場合は、DIOSA_ETIMEOUT を返却する。

t_diatc_dbaccinfo *AccInfo(入力)　　データベースアクセス情報構造体へのポインタ。

戻り値

本 API は、以下の戻り値を返却する。

DIOSA_DONE(0)	正常終了。
DIOSA_SWITCH(21)	計画マスタ切替中である。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータ誤り。
DIOSA_EMEM(-6)	メモリ操作に失敗した。
DIOSA_ENOENT(-8)	該当レコードなし。
DIOSA_EINVAL(-9)	パラメータ値不正。
DIOSA_ELOCK(-14)	ロック制御に失敗した。
DIOSA_ETIMEOUT(-22)	要求がタイムアウトした。
DIOSA_EACCES(-25)	アクセス対象の表がオープンされていない。または、IM の更新ログ出力機能が準備できていないため更新アクセスできない。
DIOSA_EFUNCNAV(-34)	更新 DB タイプと運用モードが一致しない。
DIOSA_EDEADLOCK(-36)	デッドロックが発生した。
DIOSA_ECOND(-60)	同一トランザクション内で **diatcdbtruncate()**が実行されている。
DIOSA_EDB(-69)	IM または DB へのアクセス失敗。
DIOSA_ESTATE(-114)	**diatctxstart()** が未実行である。または、同一トランザクション内で別 MAP への更新系のアクセス要求 API が実行されている。
DIOSA_ESWITCH(-115)	障害時マスタ切替中である。
DIOSA_EREADY(-118)	インメモリサーバの準備ができていない。

関連

t_diatc_dbaccinfo

2.5.17　diatcmkgettblid(ユーザデータ状態管理表テーブル ID 照会関数)

名前

diatcmkgettblid - ユーザデータ状態管理表のテーブル ID を取得する

書式

```
#include <diosa.h>
int diatcmkgettblid(int *TableId);
```

説明

diatcmkgettblid() は、ユーザデータ状態管理表のテーブル ID を **TableId** に格納する。この **TableId** を使って **diatcdbread()** でユーザデータ状態管理表からレコードを取得する。

戻り値

本 API は、以下の戻り値を返却する。

DIOSA_DONE(0)	正常終了した。
DIOSA_ERROR(-1)	異常終了した。
DIOSA_EPARAM(-3)	パラメータ誤り。
DIOSA_ENOENT(-8)	ユーザデータ状態管理表が存在しない。

関連

diatcdbread()

2.5.18　diatcmkread(メインキー照会関数)

名前

diatcmkread - ユーザデータ状態管理表からデータ登録状態を読み込む（メインキー指定）

書式

```
#include <diosa.h>
int diatcmkread(t_diatc_dbaccinfo *AccInfo, int Lock, int *Status);
```

説明

diatcmkread() は、ユーザデータ状態管理表から **AccInfo** に指定されたメインキーの情報を取得する。
Lock には、排他オプションとして以下の値のいずれかを指定する。

DIOSA_NOLOCK	ロックせずに読込を行う。他トランザクションが既に読込対象レコードをロックしていても、待ち合わせすることなく読込を行う。なお、同時に他トランザクションが読込対象レコードを更新している場合は、更新前のレコードを読み込む。
DIOSA_LOCK_WAIT	ロックして読込を行う。他トランザクションで既に読込対象レコードがロックされている場合は、ロックが解放されるまで待ち合わせる。但し、インメモリサーバへのアクセスで応答監視タイマ(環境定義 IMENV 節-USERAP 項-REQTIMEOUT で定義したもの)の時間を経過してもロックが解放されない場合は、DIOSA_ETIMEOUT を返却する。
DIOSA_LOCK_NOWAIT	ロックして読込を行う。他トランザクションで既に読込対象レコードがロックされている場合は、待ち合わせをせずにエラーとなる。

t_diatc_dbaccinfo *AccInfo(入力)	データベースアクセス情報構造体へのポインタ。
int Lock(入力)	排他オプション。
int *Status	取得したデータ登録状態を格納する変数へのポインタ。
DIATC_STATUS_TAM_ORACLE	IM と DB

戻り値

本 API は、以下の戻り値を返却する。

DIOSA_DONE(0)	正常終了。
DIOSA_SWITCH(21)	計画マスタ切替中である。
DIOSA_ERROR(-1)	異常終了。
DIOSA_EPARAM(-3)	パラメータ誤り。
DIOSA_EMEM(-6)	メモリ操作に失敗した。
DIOSA_ENOENT(-8)	該当レコードなし。
DIOSA_EINVAL(-9)	パラメータ値不正。
DIOSA_ELOCK(-14)	ロック制御に失敗した。
DIOSA_ETIMEOUT(-22)	要求がタイムアウトした。
DIOSA_EACCES(-25)	アクセス対象の表がオープンされていない。または、IM の更新ログ出力機能が準備できていないため更新アクセスできない。
DIOSA_EBUSY(-31)	既に該当レコードがロックされている。
DIOSA_EFUNCNAV(-34)	更新 DB タイプと運用モードが一致しない。
DIOSA_EDEADLOCK(-36)	デッドロックが発生した。
DIOSA_ECOND(-60)	同一トランザクション内で **diatcdbtruncate()** が実行されている。(**Lock** に DIOSA_LOCK_WAIT、DIOSA_LOCK_NOWAIT のいずれかを指定した場合)

DIOSA_EDB(-69)	IM または DB へのアクセス失敗。
DIOSA_ESTATE(-114)	**diatctxstart()** が未実行である。または、同一トランザクション内で別 MAP への更新系のアクセス要求 API が実行されている。(**Lock** に DIOSA_LOCK_WAIT、DIOSA_LOCK_NOWAIT のいずれかを指定した場合)
DIOSA_ESWITCH(-115)	障害時マスタ切替中である。
DIOSA_EREADY(-118)	インメモリサーバの準備ができていない。

関連

t_diatc_dbaccinfo

2.5.19　t_diatc_dbcond(検索条件構造体)

名前

t_diatc_dbcond - 検索条件構造体

書式

```
#include <diosa.h>
t_diatc_dbcond cond;
```

説明

t_diatc_dbcond 構造体は、ヘッダー <diosa.h> で定義されていて、検索条件を設定する構造体である。
diatcdbcondsetkey()、diatcdbcondsetrange() により検索条件が設定される。
利用者はメンバーを直接参照する必要はない。

関連

diatcdbcondsetkey(), diatcdbcondsetrange(), diatcdbctxopen(), diatcdbread1()

2.5.20　t_diatc_recinfo(レコード情報構造体)

名前

t_diatc_recinfo - レコード情報構造体

書式

```
#include <diosa.h>
t_diatc_recinfo recinfo;
```

説明

t_diatc_recinfo 構造体は、ヘッダー <diosa.h> で定義されていて、その中にレコードに関する情報を保持する。

t_diatc_recinfo 構造体は、以下のメンバーを含んでいる。

char *RecordPtr;	レコードイメージ格納アドレス
size_t RecordSize;	レコードサイズ レコードサイズには、ユーザデータサイズと DB アクセス制御機能の制御情報 40 バイトの合計値を指定する。

関連

diatcdbdelete(), diatcdbread(), diatcdbread1(), diatcdbrewrite(), diatcdbwrite()

2.5.21　t_diatc_dbaccinfo(データベースアクセス情報構造体)

名前

t_diatc_dbaccinfo - データベースアクセス情報構造体

書式

```
#include <diosa.h>
t_diatc_dbaccinfo info;
```

説明

t_diatc_dbaccinfo 構造体は、ヘッダー <diosa.h> で定義されていて、その中にデータベースアクセスに関連する情報が格納される。
t_diatc_dbaccinfo 構造体は、以下のメンバーを含んでいる。

int Method;　分散キーの指定方法。以下のいずれかを設定する。

DIATC_BY_MAINKEY	メインキーによる指定を行う。 この場合、メンバーMainkey の値が有効となり、 メンバーMapId の値は無視される。
DIATC_BY_MAPID	MAPID による指定を行う。 この場合、メンバーMapId の値が有効となり、 メンバーMainkey の値は無視される。

char Mainkey[32];　メインキー

int MapId;　MAPID

int Target;　アクセス対象。以下のいずれかを指定する。

DIATC_TARGET_TAM_ORACLE	IM と DB の両方
DIATC_TARGET_TAM	IM のみ

int TrnBorderFlag　OracleDB トランザクション境界フラグ。以下のいずれかを指定する。

DIOSA_NO	何もしない
DIOSA_YES	DB のデータ更新時にコミットの実行を要求する

int SqlTemplateExecFlag

SQL 雛形実行フラグ。以下のいずれかを指定する。

DIOSA_NO	何もしない
DIOSA_YES	DB のデータ更新時に SQL 雛形ファイルに定義した SQL を実行する。

int Version;　環境定義のリビジョン番号を指定する。環境定義 DACENV 節の TABLESET 項の VERSION パラメータの値が本パラメータと等しい定義を利用する。最新のリビジョンを利用する場合は -1 を指定する。

呼び出し時に本構造体を指定する関数について、各関数で指定が必要なメンバーを下表に示す。

（■：必須項目、□：選択必須項目、－：不要）

関数名	Method	Mainkey	MapId	Target	TrnBorder Flag	SqlTemplate ExecFlag	Version
diatcdbctxopen	■	□	□	－	－	－	■
diatcdbdelete	■	■	－	■	■	■	■
diatcdbgettblid	－	－	－	－	－	－	■
diatcdbread	■	□	□	－	－	－	■
diatcdbread1	■	□(*1)	□(*1)	－	－	－	■
diatcdbrewrite	■	■	－	－	■	■	■
diatcdbtruncate	■	□	□	－	－	－	■
diatcdbwrite	■	■	－	－	■	■	■
diatcmkadd	■	■	－	－	■	－	－
diatcmkdelete	■	■	－	－	■	－	－
diatcmkread	■	■	－	－	－	－	－

(*1)排他あり読込の場合は Mainkey の指定が必須、MapId の指定は不要である。

関連

diatcdbctxopen(), diatcdbdelete(), diatcdbgettblid(), diatcdbread(), diatcdbread1(), diatcdbrewrite(), diatcdbtruncate(), diatcdbwrite(), diatcmkadd(), diatcmkdelete(), diatcmkread()

第III編 Java インタフェース

第1章　Javaインタフェース

1.1　クラス一覧

(1)　データストア基盤

DiosaDlogData	ディレード転送機能のプールファイルに書き込むログデータを作成する。
DiosaDlogDataConst	DiosaDlogDataクラスのフィールドで指定する定数を定義したクラス。
DiosaDPutLog	DiosaDlogDataクラスで作成したログデータをプールファイルに書き込むためのログデータ登録実行クラス。

(2)　データ変換・通信オプション

DiatcDbUpdateLog	更新ログを生成するクラス。
DiatcDataReplication	更新ログをストリームごとに結合し、トランザクションごとにログデータ登録をおこなうクラス。
DiatcDataReplicationConst	センタ間レプリケーションのメソッドに指定する定数を定義したクラス。
DiatcDataReplicationFactory	DiatcDataReplicationクラスのインスタンスを生成、管理するクラス。

1.2 DiosaDlogData

1.2.1 DiosaDlogData クラス

クラス概要

ディレード転送機能のプールファイルに書き込むログデータを作成する。DiosaDlogData クラスはユーザデータとユーザデータ属性情報(スーパーストリーム名など)から構成される。

パッケージ情報

com.nec.jp.diosa.delayed.DiosaDlogData

関連

DiosaDPutLog

フィールド概要

short compressFlg
String coName
String spstName
String streamName
int textLen
byte[] userData

コンストラクタ概要

DiosaDlogData(byte[] userData, int textLen, String spstName, String streamtName, short compressFlg)	必須パラメータを設定してログデータを新規作成する。

メソッド概要

short getCompressFlg()	データ圧縮指定を取得する。
String getCoName()	CO 名を取得する。
String getSpstName()	スーパーストリーム名を取得する。
String getStreamName()	ストリーム名を取得する。
int getTextLen()	ユーザデータ長を取得する。
byte[] getUserData()	ユーザデータを取得する。
void setCompressFlg(short compressFlg)	データ圧縮指定を設定する。
void setCoName(String coName)	CO 名を設定する。
void setSpstName(String spstName)	スーパーストリーム名を設定する。
void setStreamName(String streamName)	ストリーム名を設定する。
void setTextLen(int textLen)	ユーザデータ長を設定する。
void setUserData(byte[] userData)	ユーザデータを設定する。

1.2.2 compressFlg フィールド

書式

```
private short compressFlg
```

説明

ログデータをプールファイルに登録する際にユーザデータを圧縮するかしないかを指定する。ユーザデータを圧縮する場合は DiosaComConst.DIOSA_DATACOMP_ON を指定する。ユーザデータを圧縮しない場合は DiosaComConst.DIOSA_DATACOMP_OFF を指定する。

1.2.3 coName フィールド

書式

```
private String coName
```

説明

CO 名を指定する。最大文字数は 30 文字である。省略してもログデータ登録は可能である。

1.2.4 spstName フィールド

書式

```
private String spstName
```

説明

ログデータを処理するスーパーストリーム名を指定する。最大文字数は 15 文字である。

1.2.5 streamName フィールド

書式

```
private String streamName
```

説明

ログデータを処理するストリーム名を指定する。最大文字数は 15 文字である。

1.2.6 textLen フィールド

書式

```
private int textLen
```

説明

ユーザデータのデータ長を指定する。最大値は 2,147,483,640 バイトである。

1.2.7 userData フィールド

書式

```
private byte[] userData
```

説明

ユーザデータを指定する。

1.2.8　DiosaDlogData コンストラクタ

書式

public DiosaDlogData(byte[] userData, int textLen, String spstName, String streamName, short compressFlg)

説明

ログデータの必須パラメータ(ユーザデータ、ユーザデータ長、スーパーストリーム名、ストリーム名、データ圧縮指定)を設定して、ログデータを新規作成する。

パラメータ

userData

　ユーザデータ

textLen

　ユーザデータ長

spstName

　スーパーストリーム名

streamName

　ストリーム名

compressFlg

　データ圧縮指定

1.2.9　getCompressFlg メソッド

書式

public short getCompressFlg()

説明

設定されているデータ圧縮指定(compressFlg フィールド)を取得する。

戻り値

設定されている compressFlg フィールドの値が返却される。

1.2.10　getCoName メソッド

書式

public String getCoName()

説明

設定されている CO 名(coName フィールド)を取得する。

戻り値

設定されている coName フィールドの値が返却される。

1.2.11　getSpstName メソッド

書式

public String getSpstName()

説明

設定されているスーパーストリーム名(spstName フィールド)を取得する。

戻り値

設定されている spstName フィールドの値が返却される。

1.2.12　getStreamName メソッド

書式

public String getStreamName()

説明

設定されているストリーム名(streamName フィールド)を取得する。

戻り値

設定されている streamName フィールドの値が返却される。

1.2.13　getTextLen メソッド

書式

public int getTextLen()

説明

設定されているユーザデータ長(textLen フィールド)を取得する。

戻り値

設定されている textLen フィールドの値が返却される。

1.2.14　getUserData メソッド

書式

public byte[] getUserdata()

説明

設定されているユーザデータ(userData フィールド)を取得する。

戻り値

設定されている userData フィールドの値が返却される。

1.2.15　setCompressFlg メソッド

書式

public void setCompressFlg(short compressFlg)

説明

データ圧縮指定を設定する。

パラメータ

compressFlg

設定するデータ圧縮指定。

1.2.16　setCoName メソッド

書式

public void setCoName(String coName)

説明

CO 名を設定する。

パラメータ

coName

設定する CO 名。

1.2.17　setSpstName メソッド

書式

public void setSpstName(String spstName)

説明

スーパーストリーム名を設定する。

パラメータ

spstName

設定するスーパーストリーム名。

1.2.18　setStreamName メソッド

書式

public void setStreamName(String streamName)

説明

ストリーム名を設定する。

パラメータ

streamName

設定するストリーム名。

1.2.19　setTextLen メソッド

書式

public void setTextLen(int textLen)

説明

ユーザデータ長を設定する。

パラメータ

textLen

設定するユーザデータ長。

1.2.20　setUserData メソッド

書式

public void setUserData(byte[] userData)

説明

ユーザデータを設定する。

パラメータ

userData

設定するユーザデータ。

1.3　DiosaDlogDataConst

1.3.1　DiosaDlogDataConst クラス

クラス概要

DiosaDlogData クラスのフィールドで指定する定数を定義したクラス。

パッケージ情報

com.nec.jp.diosa.com.DiosaComConst

関連

DiosaDlogData

フィールド概要

DIOSA_DATACOMP_ON
DIOSA_DATACOMP_OFF

1.3.2　DIOSA_DATACOMP_ON フィールド

書式

public static final short DIOSA_DATACOMP_ON

説明

ユーザデータを圧縮する場合に DiosaDlogData.CompressFlg に指定する定数。

1.3.3　DIOSA_DATACOMP_OFF フィールド

書式

public static final short DIOSA_DATACOMP_OFF

説明

ユーザデータを圧縮する場合に DiosaDlogData.CompressFlg に指定する定数。

1.4　DiosaDPutLog

1.4.1　DiosaDPutLog クラス

クラス概要

DiosaDlogData クラスで作成したログデータをプールファイルに書き込むためのログデータ登録実行クラス。

パッケージ情報

com.nec.jp.diosa.delayed.DiosaDPutLog

関連

DiosaDlogData

コンストラクタ概要

DiosaDPutLog()	ログデータ登録実行オブジェクトを作成する。

メソッド概要

int[] putlog(Connection con, DiosaDlogData logdata)	ログデータをプールファイルに書き込む。

1.4.2　DiosaDPutLog コンストラクタ

書式

public DiosaDPutLog()

説明

ログデータをプールファイルへ書き込むための処理を行うログデータ登録実行オブジェクトを作成する。

1.4.3　putlog メソッド

書式

public int[] putlog(Connection con, DiosaDlogData logdata)

説明

logdata で指定したログデータ(DiosaDlogData クラスのオブジェクト)をプールファイルへ書き込む処理を実行する。

パラメータ

con

データベースとのコネクション

logdata

プールファイルへ書き込むログデータ

戻り値

戻り値は配列で返却する。
1 番目は以下の値を返却する。2 番目はエラー発生時の詳細コードを返却する。

0	正常終了
-1	その他エラー
-3	パラメータエラー
-34	メンテナンス中/無効化中ため書込み不可
-39	プールファイルオーバーフロー
-93	プールファイルアクセスエラー

1.5　DiatcDbUpdateLog

1.5.1　　DiatcDbUpdateLog クラス

クラス概要

public class DiatcDbUpdateLog

更新ログを生成するためのクラスです。

パッケージ情報

com.nec.jp.diatc.dac.DiatcDbUpdateLog

メソッド概要

static <T> byte[] makeUpdateLog(String sql, List<T> params)	バインド変数を含む SQL とバインドする値の一覧を指定して更新ログを生成します。
static byte[] makeUpdateLog(String sql)	SQL（バインド変数を含まない）を指定して更新ログを生成します。

1.5.2　　makeUpdateLog メソッド

書式

public static <T> byte[] makeUpdateLog(String sql, List<T> params) throws IOException, SQLException

説明

バインド変数を含む SQL とバインドする値の一覧を指定して更新ログを生成します。

パラメータ

sql

SQL 文を指定します。sql には、１つ以上のバインド変数を含めることができます。バインド変数の出現箇所には プレースホルダを表す'?' を指定します。'?'の前後には英数字を連結しないでください。
シングルクォート(')で囲まれた範囲に記述した'?'は、プレースホルダではなく文字として扱われます。

params

バインドする値の一覧を指定します。最大 1032 個のバインド変数を指定することができます。sql における '?' に値をバインドするのに必要な情報（データ型と値）を持つオブジェクトを params の各要素に登録します。sql における '?' の出現順序と params の要素の登録順序は一致する必要があります。
OracleDB のデータ型とそれに対応する params の各要素の Java のデータ型は下表のとおりです。また、params に要素 null を登録すると NULL をバインドします。

OracleDB のデータ型	Java のデータ型
VARCHAR2	java.lang.String
CHAR	java.lang.String
NUMBER	java.lang.Byte　(※1)
	java.lang.Short　(※1)
	java.lang.Integer　(※1)
	java.lang.Long　(※1)
	java.math.BigDecimal
RAW	byte[]
BLOB	java.sql.Blob　(※2)
CLOB	java.sql.Clob　(※2)
DATE	java.util.Date　(※3)
TIMESTAMP	java.sql.Timestamp (※4)

※1 プリミティブ型も指定可(Java のオートボクシング機能による変換のため)
※2 BLOB/CLOB に対し、それぞれ byte[]/java.lang.String の指定も可能(Oracle による自動変換のため)
※3 OracleDB の NLS_DATE_FORMAT に則した文字列であれば java.lang.String も指定可
※4 OracleDB の NLS_TIMESTAMP_FORMAT に則した文字列であれば java.lang.String も指定可

戻り値

更新ログのバイト配列を返却します。

例外

IllegalArgumentException

パラメータ sql や params が null の場合にスローされます。

UnsupportedOperationException

params の要素に利用可能な Java のデータ型（上表に示す Java のデータ型、および null）以外のデータ型が含まれていた場合にスローされます。

IOException

何らかの入出力例外が発生した場合にスローされます。

SQLException

BLOB や CLOB データのアクセスで例外が発生した場合にスローされます。

注意

BLOB データや CLOB データをバインドする場合(*)、下記のような SQL はパラメータ sql に指定できません。
(*)パラメータ params に java.sql.Blob 型または java.sql.Clob 型を含む場合を指す。

・DML RETURNING 句を含む SQL
・複数レコードの更新を行う SQL
・PL/SQL ブロック
・ストアドプロシージャ呼び出し

なお、BLOB データや CLOB データをバインドしなければ、BLOB 型や CLOB 型の項目があるテーブルを更新する場合でも上記のような SQL は指定可能です。

1.5.3　　makeUpdateLog メソッド

書式

public static byte[] makeUpdateLog(String sql) throws IOException

説明

指定した SQL から更新ログを生成します。

パラメータ

sql

SQL 文を指定します。

戻り値

更新ログのバイト配列を返却します。

例外

IllegalArgumentException

パラメータ sql が null の場合にスローされます。

IOException

何らかの入出力例外が発生した場合にスローされます。

1.6　DiatcDataReplication

1.6.1　　DiatcDataReplication クラス

クラス概要

public class DiatcDataReplication

更新ログをストリームごとに結合し、トランザクションごとにログデータ登録をおこなうクラス。

パッケージ情報

com.nec.jp.diatc.dsc.DiatcDataReplication

フィールド概要

String szStreamName
List<DiosaDLogData> listLogData

コンストラクタ概要

DiatcDataReplication()	DiatcDataReplication オブジェクトを構築し、DiosaDLogData オブジェクトの初期化をおこなう

メソッド概要

boolean isAvailable()	センタ間レプリケーションの動作環境かを返却する。
int[] tranInit()	トランザクション初期化処理をおこなう。
int[] tranInit(Connection apConnection)	
int[] setStream(String streamName)	ストリームを設定する。
int[] saveULogData(String nSubComponentID, int nUpdateLogSize, byte[] cUpdateLog)	更新ログを同一ストリーム名で纏めて登録する
int[] tranTerm(boolean isRollbackOnly)	トランザクション終了処理をおこなう。 更新ログを、ディレード転送するログデータとしてDB に出力する。
int[] tranTerm(Connection apConnection, boolean isRollbackOnly)	

1.6.2　DiatcDataReplication コンストラクタ

書式

private DiatcDataReplication()

説明

更新ログ登録用のデータオブジェクトを初期化する。

1.6.3　isAvailable メソッド

書式

public static boolean isAvailable()

説明

センタ間レプリケーション動作可能環境かを返却する。

戻り値

true	動作可能
false	動作不可

1.6.4　tranInit メソッド

書式

public int[] tranInit()

説明

トランザクション初期化処理をおこなう。

Spring Framework 上でトランザクションを実行する場合、トランザクションの開始時に必ず呼び出す必要がある。

パラメータ

なし

戻り値

戻り値は配列で返却する。

1 番目は以下の値を返却する。2 番目はエラー発生時の詳細コードを返却する。

0	正常終了
-34	センタ間レプリケーション動作不可

1.6.5　tranInit メソッド

書式

public int[] tranInit(Connection apConnection)

説明

トランザクション初期化処理をおこなう。
本メソッドは、トランザクションの開始時に必ず呼び出す必要がある。
Spring Framework 以外でトランザクションを実行する場合、トランザクションの開始時に必ず呼び出す必要がある。

パラメータ

Connection apConnection

アプリケーションで使用する DB への接続

戻り値

戻り値は配列で返却する。
1 番目は以下の値を返却する。2 番目はエラー発生時の詳細コードを返却する。

0	正常終了
-34	センタ間レプリケーション動作不可

1.6.6　setStream メソッド

書式

public int[] setStream(String szStreamName)

説明

通常運用時に、指定されたストリーム名 **szStreamName** を DiosaDLogData に設定する。
SaveULogData メソッドを呼び出す前に必ず呼び出す必要がある。
以降、SaveULogData メソッドは本メソッドで指定されたストリームに対して更新ログを蓄積する。また、ストリーム名を変更して再度本メソッドを呼び出すことにより SaveULogData メソッドは変更されたストリームに対して処理をおこなう。
szStreamName で指定した文字列から、以下の命名規則で生成した文字列を DIOSA/XTP のデータストア基盤のスーパーストリーム名として定義すること。
ORA_{szStreamName 指定名}_{拠点識別情報(環境変数 DIATC_CENTER_ID の指定)}

パラメータ

String szStreamName

ストリーム名 ： 9 文字以内

戻り値

戻り値は配列で返却する。
1 番目は以下の値を返却する。2 番目はエラー発生時の詳細コードを返却する。

0	正常終了
-3	パラメータエラー
-11	トランザクション区間外

1.6.7　　saveULogData メソッド

書式

```
public int[] saveULogData(  String szSubComponentID,
                            int nUpdateLogSize,
                            byte[] cUpdateLog)
```

説明

setStream メソッドで決定されたストリームの DiosaDLogData オブジェクトにパラメータで渡された更新ログを追加する。

パラメータ

String szSubComponentID

呼出元機能 ID

int nUpdateLogSize

更新ログデータサイズ

byte[] cUpdateLog

更新ログデータ

戻り値

戻り値は配列で返却する。
1 番目は以下の値を返却する。2 番目はエラー発生時の詳細コードを返却する。

0	正常終了
-3	パラメータエラー
-8	ストリーム未設定
-11	トランザクション区間外

1.6.8　tranTerm メソッド

書式

```
public int[] tranTerm(boolean isRollbackOnly)
```

説明

ストリーム単位に格納されているログデータの登録要求をおこなう。
Spring Framework 上でトランザクションを実行する場合、トランザクションの終了時に必ず呼び出す必要がある。

パラメータ

boolean isRollbackOnly

トランザクションがコミットされるかロールバックされるかを指定する。
コミットの場合 false、ロールバックの場合 true を指定する。

戻り値

戻り値は配列で返却する。
1 番目は以下の値を返却する。2 番目はエラー発生時の詳細コードを返却する。
プールファイルオーバーフローについては、業務処理を異常終了させないこと。

0	正常終了
6	プールファイルオーバーフローによる書き込み抑止中
-1	その他エラー
-3	パラメータエラー
-8	ストリーム未定義
-11	トランザクション区間外
-34	ストリームメンテナンス/閉塞中により書き込み不可
-69	DB アクセスエラー（トランザクションタイムアウトを含む）
-93	プールファイルアクセスエラー

1.6.9　tranTerm メソッド

書式

```
public int[] tranTerm(Connection apConnection, boolean isRollbackOnly)
```

説明

ストリーム単位に格納されているログデータの登録要求をおこなう。
Spring Framework 以外でトランザクションを実行する場合、トランザクションの終了時に必ず呼び出す必要がある。

パラメータ

Connection apConnection

アプリケーションで使用する DB への接続

boolean isRollbackOnly

トランザクションがコミットされるかロールバックされるかを指定する。
コミットの場合 false、ロールバックの場合 true を指定する。

戻り値

戻り値は配列で返却する。
1 番目は以下の値を返却する。2 番目はエラー発生時の詳細コードを返却する。
プールファイルオーバーフローについては、業務処理を異常終了させないこと。

0	正常終了
-1	その他エラー
6	プールファイルオーバーフローによる書き込み抑止中
-3	パラメータエラー
-8	ストリーム未定義
-11	トランザクション区間外
-34	ストリームメンテナンス/閉塞中により書き込み不可
-69	DB アクセスエラー
-93	プールファイルアクセスエラー

1.7　DiatcDataReplicationConst

1.7.1　　DiatcDataReplicationConst クラス

クラス概要

public class DiatcDataReplicationConst

センタ間レプリケーションのメソッドに指定する定数を定義したクラス。

パッケージ情報

com.nec.jp.diatc.dsc.DiatcDataReplicationConst

フィールド概要

FUNCID_DAC
FUNCID_GNT

1.7.2　　FUNCID_DAC フィールド

書式

public static final String FUNCID_DAC

説明

DB アクセス制御機能を表す定数。

1.7.3　　FUNCID_GNT フィールド

書式

public static final String FUNCID_GNT

説明

電文保証機能を表す定数。

1.8　DiatcDataReplicationFactory

1.8.1　　DiatcDataReplicationFactory クラス

クラス概要

public class DiatcDataReplicationFactory

DiatcDataReplication インスタンスを生成、管理するクラス。

パッケージ情報

com.nec.jp.diatc.dsc.DiatcDataReplicationFactory

メソッド概要

DiatcDataReplication createInstance()	カレントスレッドの DiatcDataReplication インスタンスを生成して取得する。
DiatcDataReplication getInstance()	カレントスレッドの カレント DiatcDataReplication インスタンスを取得する。
void removeInstance()	カレントスレッドの カレント DiatcDataReplication インスタンスを削除する。

1.8.2　createInstance メソッド

書式

public DiatcDataReplication createInstance()

説明

カレントスレッドの DiatcDataReplication インスタンスを生成して取得する。

戻り値

DiatcDataReplication インスタンス

1.8.3　getInstance メソッド

書式

public static DiatcDataReplication getInstance()

説明

カレントスレッドの カレント DiatcDataReplication インスタンスを取得する。

戻り値

DiatcDataReplication インスタンス
※createInstance メソッドを実行していない場合は、null インスタンスが返却される。

1.8.4　removeInstance メソッド

書式

public static void removeInstance()

説明

カレントスレッドの カレント DiatcDataReplication インスタンスを削除する。

戻り値

なし

付録A　API一覧

A.1　APIが利用可能な動作環境

A.1.1　アプリケーション実行制御

アプリケーション実行制御	関数名	CO制御								バッチAP制御				ログリーダ					通信制御											メモリキャッシュ	ユーザAP			備考
		プロセス終了出口	アボート出口#2	アボート出口#1	トランザクション終了出口	CO	トランザクション初期化出口	プロセス初期化出口	受信電文解析出口	アボート出口	正常終了出口	CO	初期化出口	プロセス終了出口	トランザクション終了出口	ログデータ実行メイン処理出口	トランザクション初期化出口	プロセス初期化出口	電文保証TPPプロセス終了出口	電文保証TPPプロセス初期化出口	保証電文送信有無判定出口	送受信エラー終了出口	送受信エラー初期化出口	送受信エラー出口	データベース接続出口	リスナ終了出口	リスナ初期化出口	リスナ電文送受信時出口	ハッシュ関数	diosaprcinit/term 区間内	diosatrninit/term 区間内	diosatxstart/commit 区間内	
	diosaaddrconv	－	－	－	○	－	－	－	－	－	○	－	－	○	○	○	○	○	－	－	－	－	－	－	－	－	－	－	－	○	○		
	diosaapegetitem	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	－	－	－	－	－	－	－	－	－	－	－	○	○	○	
	diosaapegettbl	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	－	－	－	－	－	－	－	－	－	－	－	○	○	○	
	diosaapptrcclose	－	－	－	○	－	－	－	－	－	○	－	－	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	○	
	diosaapptrcf	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	－	○	○	○	○	○	○	○	○	○	
	diosaapptrcf_position	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	－	○	○	○	○	○	○	○	○	○	
	diosaapptrcm	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	－	○	○	○	○	○	○	○	○	○	
	diosaapptrcm_position	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	－	○	○	○	○	○	○	○	○	○	
	diosaapptrcopen	－	－	－	○	－	－	－	－	－	○	－	－	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	○	
	diosaapptrcread	－	－	－	○	－	－	－	－	－	○	－	－	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	○	
	diosaapptrget	－	－	－	○	－	－	－	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	
	diosaapptrset	－	－	－	○	－	－	－	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	
	diosacmdconf	－	－	－	○	－	－	－	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	
	diosacmdsend	－	－	－	○	－	－	－	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	
	diosacommit	－	－	－	○	－	－	－	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	－	
	diosaetgreset	－	－	－	○	－	－	－	－	－	○	－	－	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	
	diosaetgstart	－	－	－	○	－	－	－	－	－	○	－	－	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	

アプリケーション実行制御	関数名	CO制御								バッチAP制御				ログリーダ					通信制御										メモリキャッシュ	ユーザAP			備考
		受信電文解析出口	プロセス初期化出口	トランザクション初期化出口	CO	トランザクション終了出口	アボート出口#1	アボート出口#2	プロセス終了出口	初期化出口	CO	正常終了出口	アボート出口	プロセス初期化出口	トランザクション初期化出口	ログデータ実行メイン処理出口	トランザクション終了出口	プロセス終了出口	リスナ電文送受信時出口	リスナ初期化出口	リスナ終了出口	データベース接続出口	送受信エラー出口	送受信エラー初期化出口	送受信エラー終了出口	保証電文送信有無判定出口	電文保証TPPプロセス初期化出口	電文保証TPPプロセス終了出口	ハッシュ関数	diosaprcinit/term区間内	diosatrninit/term区間内	diosatxstart/commit区間内	
	diosaetgstop	−	−	−	○	−	−	−	−	−	○	−	−	−	−	○	−	−	−	−	−	−	−	−	−	−	−	−	−	○	○	
	diosaetgusinfo	−	−	−	○	−	−	−	−	−	○	−	−	−	−	○	−	−	−	−	−	−	−	−	−	−	−	−	−	○	○	
	diosagetlnodeinfo	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	−	○	○	○	○	○	○	−	○	○	
	diosagetsrctx	−	−	○	○	○	○	○	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	
	diosagoback	○	○	○	○	○	○	○	○	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	
	diosalock	−	−	−	○	−	−	−	−	−	○	−	−	−	−	○	−	−	−	−	−	−	−	−	−	○	−	−	−	−	○	○	△※1
	diosamaddr	−	−	−	○	−	−	−	−	−	○	−	−	○	○	○	○	○	−	−	−	−	−	−	−	−	−	−	−	−	○	○	
	diosamalloc	−	−	−	○	−	−	−	−	−	○	−	−	○	○	○	○	○	−	−	−	−	−	−	−	−	−	−	−	−	○	○	
	diosamcopy	−	−	−	○	−	−	−	−	−	○	−	−	○	○	○	○	○	−	−	−	−	−	−	−	−	−	−	−	−	○	○	
	diosamfree	−	−	−	○	−	−	−	−	−	○	−	−	○	○	○	○	○	−	−	−	−	−	−	−	−	−	−	−	−	○	○	
	diosamsgbufalloc	○	○	○	○	○	○	○	○	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	○	−	
	diosamsgbuffree	○	○	○	○	○	○	○	○	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	○	−	
	diosamsgdisp	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	−	○	○	
	diosamsgedit	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	−	○	○	○	○	○	−	○	○	
	diosaopsusiput	−	−	−	○	−	−	−	−	−	○	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	○	○	
	diosaopsperfend	−	−	○	○	○	○	○	−	−	○	○	○	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	○	○	
	diosaperfstart	−	−	○	○	○	○	○	−	−	○	○	○	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	○	○	
	diosaprcforkinit	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	○	○	
	diosaprcinit	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	
	diosaprcpreexecterm	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	○	○	
	diosaprcterm	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	
	diosarealloc	−	−	−	○	−	−	−	−	−	○	−	−	○	○	○	○	○	−	−	−	−	−	−	−	−	−	−	−	−	○	○	

アプリケーション実行制御	関数名	CO制御								バッチAP制御				ログリーダ					通信制御										メモリキャッシュ	ユーザAP			備考
		受信電文解析出口	プロセス初期化出口	トランザクション初期化出口	CO	トランザクション終了出口	アボート出口#1	アボート出口#2	プロセス終了出口	初期化出口	CO	正常終了出口	アボート出口	プロセス初期化出口	トランザクション初期化出口	ログデータ実行メイン処理出口	トランザクション終了出口	プロセス終了出口	リスナ電文送受信時出口	リスナ初期化出口	リスナ終了出口	データベース接続出口	送受信エラー出口	送受信エラー初期化出口	送受信エラー終了出口	保証電文送信有無判定出口	電文保証TPPプロセス初期化出口	電文保証TPPプロセス終了出口	ハッシュ関数	diosaprcinit/term区間内	diosatrninit/term区間内	diosatxstart/commit区間内	
	diosarecvtx	－	－	－	○	○	○	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	
	diosarollback	－	－	－	○	－	－	－	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	－	
	diosasendtx	－	－	－	○	○	○	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	－	
	diosasetblockage	－	－	－	○	－	－	－	－	－	○	－	－	○	○	○	○	○	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	
	diosathrinit	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	－	－	
	diosathrterm	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	－	－	
	diosatmcactv	－	－	－	○	－	－	－	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	
	diosatmccmdquery	－	－	－	○	－	－	－	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	
	diosatmccmdset	－	－	－	○	－	－	－	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	
	diosatmccoquery	－	－	－	○	－	－	－	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	
	diosatmccoset	－	－	－	○	－	－	－	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	
	diosatmchold	－	－	－	○	－	－	－	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	
	diosatmcreset	－	－	－	○	－	－	－	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	
	diosatrninit	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	－	－	
	diosatrnterm	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	－	－	
	diosatxstart	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	－	
	diosaucaget	○	○	○	○	○	○	○	○	○	○	○	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	
	diosaunlock	－	－	－	○	－	－	－	－	－	○	－	－	－	－	○	－	－	－	－	－	－	－	－	－	○	－	－	－	－	○	○	
	diosavcall	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	－	－	－	－	－	－	－	－	－	－	－	－	○	○	

○ … 使用可能　　－ … 使用不可　　△ … 条件付使用可能

※1　ファイル型のみ

A.1.2　　通信制御

通信制御	関数名	CO制御								バッチAP制御				ログリーダ					通信制御											メモリキャッシュ	ユーザAP			備考
		受信電文解析出口	プロセス初期化出口	トランザクション初期化出口	CO	トランザクション終了出口	アボート出口#1	アボート出口#2	プロセス終了出口	初期化出口	CO	正常終了出口	アボート出口	プロセス初期化出口	トランザクション初期化出口	ログデータ実行メイン処理出口	トランザクション終了出口	プロセス終了出口	リスナ電文送受信時出口	リスナ初期化出口	リスナ終了出口	データベース接続出口	送受信エラー出口	送受信エラー初期化出口	送受信エラー終了出口	保証電文送信有無判定出口	電文保証TPPプロセス初期化出口	電文保証TPPプロセス終了出口	ハッシュ関数	diosaprcinit/term区間内	diosatrninit/term区間内	diosatxstart/commit区間内	
	diosadbchangeconnect	―	―	○	○	―	―	○	―	○	○	―	○	―	○	○	―	―	―	―	―	―	―	―	―	○	―	―	―	―	○	○	
	diosadbconnect	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	○	―	
	diosadbdisconnect	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	○	―	
	diosadbfaultnotification	―	―	○	○	―	―	○	―	―	○	―	○	―	○	○	―	―	―	―	―	―	―	―	―	○	―	―	―	―	○	○	
	diosadbmulticonnect	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	○	―	
	diosadbreconnect	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	○	―	
	diosagetacspinfo	○	○	○	○	○	○	○	○	―	―	―	―	―	―	―	―	―	―	―	―	―	―	○	○	―	―	―	―	―	△	△	※1
	diosagetdbctx	―	―	○	○	―	―	○	―	―	○	―	○	―	○	○	―	―	―	―	―	―	―	―	―	○	―	―	―	―	○	○	
	diosagetnodepathstatus	―	―	―	○	○	○	○	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	
	diosagntcheck(_mk)	―	―	―	○	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	
	diosagntdel(_mk)	―	―	―	○	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	
	diosagntfin	―	―	―	○	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	
	diosagntshiftnextsend(_mk)	―	―	―	○	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	
	diosalspathctrl	―	―	―	○	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	―	

○ … 使用可能　　― … 使用不可　　△ … 条件付使用可能

※1 ユーザ AP ではプロトコル情報の取得のみ可能

A.1.3　　メモリキャッシュ

メモリキャッシュ	関数名	CO制御								バッチAP制御				ログリーダ					通信制御										メモリキャッシュ	ユーザAP			備考
		受信電文解析出口	プロセス初期化出口	トランザクション初期化出口	CO	トランザクション終了出口	アボート出口#1	アボート出口#2	プロセス終了出口	初期化出口	CO	正常終了出口	アボート出口	プロセス初期化出口	トランザクション初期化出口	ログデータ実行メイン処理出口	トランザクション終了出口	プロセス終了出口	リスナ電文送受信時出口	リスナ初期化出口	リスナ終了出口	データベース接続出口	送受信エラー出口	送受信エラー初期化出口	送受信エラー終了出口	保証電文送信有無判定出口	電文保証TPPプロセス初期化出口	電文保証TPPプロセス終了出口	ハッシュ関数	diosaprcinit/term 区間内	diosatrninit/term 区間内	diosatxstart/commit 区間内	
	diosagethash	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	－	－	－	－	－	－	－	－	－	－	－	－	○	○	
	diosagetmap	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	－	－	－	－	－	－	－	－	－	－	－	－	○	○	
	diosagetmaphash	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	－	－	－	－	－	－	－	－	－	－	－	－	○	○	
	diosagetmaplist	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	－	－	－	－	－	－	－	－	－	－	－	－	○	○	
	diosagetmapstat	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	－	－	－	－	－	－	－	－	－	－	－	－	○	○	
	diosagettamnode	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	－	－	－	－	－	－	－	－	－	－	－	－	○	○	
	diosaimclose	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	－	－	
	diosaimcommit	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	－	
	diosaimcondsetkey	○	－	○	○	○	－	○	－	－	○	○	○	－	－	△	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	※1
	diosaimcondsetrange	○	－	○	○	○	－	○	－	－	○	○	○	－	－	△	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	※1
	diosaimctxclose	○	－	○	○	○	－	○	－	－	○	○	○	－	－	△	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	※1
	diosaimctxopen	○	－	○	○	○	－	○	－	－	○	○	○	－	－	△	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	※1
	diosaimdelete	－	－	－	○	○	－	○	－	－	○	○	○	－	－	△	－	－	－	－	－	－	－	－	－	－	－	－	－	－	△	○	※1 ※3
	diosaimdeletex1	－	－	－	○	○	－	○	－	－	○	○	○	－	－	△	－	－	－	－	－	－	－	－	－	－	－	－	－	－	△	○	※1 ※3
	diosaimgetmap	○	－	○	○	○	－	○	－	－	○	○	○	－	－	△	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	※1
	diosaimgetmaplist	○	－	○	○	○	－	○	－	－	○	○	○	－	－	△	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	※1
	diosaimgetperf	－	－	－	－	－	－	－	－	－	－	－	－	－	－	△	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	※1
	diosaimgetptblname	○	－	○	○	○	－	○	－	－	○	○	○	－	－	△	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	※1
	diosaimgetreckeyinfo	○	－	○	○	○	－	○	－	－	○	○	○	－	－	△	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	※1
	diosaimgetsvcstatus	○	－	○	○	○	－	○	－	－	○	○	○	－	－	△	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	※1

メモリキャッシュ	関数名	CO制御								バッチAP制御				ログリーダ					通信制御										メモリキャッシュ	ユーザAP			備考
		受信電文解析出口	プロセス初期化出口	トランザクション初期化出口	CO	トランザクション終了出口	アボート出口#1	アボート出口#2	プロセス終了出口	初期化出口	CO	正常終了出口	アボート出口	プロセス初期化出口	トランザクション初期化出口	ログデータ実行メイン処理出口	トランザクション終了出口	プロセス終了出口	リスナ電文送受信時出口	リスナ初期化出口	リスナ終了出口	データベース接続出口	送受信エラー出口	送受信エラー初期化出口	送受信エラー終了出口	保証電文送信有無判定出口	電文保証TPPプロセス初期化出口	電文保証TPPプロセス終了出口	ハッシュ関数	diosaprcinit/term 区間内	diosatrninit/term 区間内	diosatxstart/commit 区間内	
	diosaimgettblid	○	－	○	○	○	－	○	－	－	○	○	○	－	－	△	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	※1
	diosaimgettbllist	○	－	○	○	○	－	○	－	－	○	○	○	－	－	△	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	※1
	diosaimopen	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	－	－	
	diosaimread	○	－	○	○	○	－	○	－	－	○	○	○	－	－	△	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	※1
	diosaimread1	△	－	△	○	○	－	○	－	－	○	○	○	－	－	△	－	－	－	－	－	－	－	－	－	－	－	－	－	－	△	○	※1 ※2 ※3
	diosaimrewrite	－	－	－	○	○	－	○	－	－	○	○	○	－	－	△	－	－	－	－	－	－	－	－	－	－	－	－	－	－	△	○	※1 ※3
	diosaimrollback	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	－	
	diosaimsetmap	○	－	○	○	○	－	○	－	－	○	○	○	－	－	△	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	※1
	diosaimtruncate	－	－	－	○	○	－	○	－	－	○	○	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	△	○	※3
	diosaimtxstart	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	－	
	diosaimwrite	－	－	－	○	○	－	○	－	－	○	○	○	－	－	△	－	－	－	－	－	－	－	－	－	－	－	－	－	－	△	○	※1 ※3
	diosatamswitch	－	－	－	○	－	－	－	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	

○ … 使用可能　　－ … 使用不可　　△ … 条件付使用可能

※1 … ログリーダユニットのユーザデータ更新先が IM の場合のみ呼び出し可

※2 … △は、DIOSA_NOLOCK 指定での diosaimread1 のみ使用可能

※3 … diosatrninit/term 区間内での更新系 API(排他あり読み込みを含む)は、diosaimtxstart/commit 区間内であれば呼び出し可

A.1.4　　データストア

データストア	関数名	ＣＯ制御								バッチＡＰ制御				ログリーダ					通信制御										メモリキャッシュ	ユーザＡＰ			備考
		受信電文解析出口	プロセス初期化出口	トランザクション初期化出口	ＣＯ	トランザクション終了出口	アボート出口＃１	アボート出口＃２	プロセス終了出口	初期化出口	ＣＯ	正常終了出口	アボート出口	プロセス初期化出口	トランザクション初期化出口	ログデータ実行メイン処理出口	トランザクション終了出口	プロセス終了出口	リスナ電文送受信時出口	リスナ初期化出口	リスナ終了出口	データベース接続出口	送受信エラー出口	送受信エラー初期化出口	送受信エラー終了出口	保証電文送信有無判定出口	電文保証 TPP プロセス初期化出口	電文保証 TPP プロセス終了出口	ハッシュ関数	diosaprcinit/term 区間内	diosatrninit/term 区間内	diosatxstart/commit 区間内	
	diosadgetlog	－	－	－	○	－	－	－	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	
	diosadputlog	－	－	－	○	－	－	－	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	
	diosalrdcommit	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	

○ … 使用可能　　－ … 使用不可　　△ … 条件付使用可能

A.1.5　　データ変換・通信オプション

データ変換・通信オプション	関数名	CO制御								バッチAP制御				ログリーダ					通信制御										メモリキャッシュ	ユーザAP			備考
		受信電文解析出口	プロセス初期化出口	トランザクション初期化出口	CO	トランザクション終了出口	アボート出口#1	アボート出口#2	プロセス終了出口	初期化出口	CO	正常終了出口	アボート出口	プロセス初期化出口	トランザクション初期化出口	ログデータ実行メイン処理出口	トランザクション終了出口	プロセス終了出口	リスナ電文送受信時出口	リスナ初期化出口	リスナ終了出口	データベース接続出口	送受信エラー出口	送受信エラー初期化出口	送受信エラー終了出口	保証電文送信有無判定出口	電文保証TPPプロセス初期化出口	電文保証TPPプロセス終了出口	ハッシュ関数	diosaprcinit/term区間内	diosatrninit/term区間内	diosatxstart/commit区間内	
	diatcdbcondsetkey	－	－	－	○	－	－	○	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	
	diatcdbcondsetrange	－	－	－	○	－	－	○	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	
	diatcdbctxclose	－	－	－	○	－	－	○	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	
	diatcdbctxopen	－	－	－	○	－	－	○	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	
	diatcdbdelete	－	－	－	○	－	－	○	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	
	diatcdbgetrplmode	－	－	－	○	－	－	○	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	
	diatcdbgettblid	－	－	－	○	－	－	○	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	
	diatcdbread	－	－	－	○	－	－	○	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	
	diatcdbread1	－	－	－	○	－	－	○	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	△	○	※1
	diatcdbrewrite	－	－	－	○	－	－	○	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	
	diatcdbsetrplmode	－	－	－	○	－	－	○	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	
	diatcdbtruncate	－	－	－	○	－	－	○	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	
	diatcdbwrite	－	－	－	○	－	－	○	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	
	diatcmkadd	－	－	－	○	－	－	○	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	
	diatcmkdelete	－	－	－	○	－	－	○	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	
	diatcmkgettblid	－	－	－	○	－	－	○	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	○	
	diatcmkread	－	－	－	○	－	－	○	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	△	○	※1

○ … 使用可能　　－ … 使用不可

※1 … diosatrninit/term 区間内では排他なし読み込みのみ呼び出し可

A.2　API 呼び出し可能ノード

A.2.1　　アプリケーション実行制御

関数名	論理ノード種別			備考
	AP	OLTP	DB	
diosaaddrconv	○	○	○	
diosaapegetitem	○	○	○	
diosaapegettbl	○	○	○	
diosaappclose	○	○	○	
diosaapptrcf	○	○	○	
diosaapptrcf_position	○	○	○	
diosaapptrcm	○	○	○	
diosaapptrcm_position	○	○	○	
diosaappopen	○	○	○	
diosaappread	○	○	○	
diosaapptrget	○	○	○	
diosaapptrset	○	○	○	
diosacmdconf	○	○	○	
diosacmdsend	○	○	○	
diosacommit	○	○	－	
diosaetgreset	○	○	－	
diosaetgstart	○	○	－	
diosaetgstop	○	○	－	
diosaetgusinfo	○	○	－	
diosagetlnodeinfo	○	○	○	
diosagetsrctx	○	○	－	
diosagoback	○	○	－	
diosalock	○	○	○	
diosamaddr	○	○	○	
diosamalloc	○	○	○	
diosamcopy	○	○	○	
diosamfree	○	○	○	
diosamsgbufalloc	○	○	－	
diosamsgbuffree	○	○	－	
diosamsgdisp	○	○	○	
diosamsgedit	○	○	○	
diosaopsusiput	○	○	－	
diosaperfend	○	○	－	
diosaperfstart	○	○	－	
diosaprcforkinit	○	○	○	
diosaprcinit	○	○	○	
diosaprcpreexecterm	○	○	○	
diosaprcterm	○	○	○	
diosarealloc	○	○	○	
diosarecvtx	○	○	－	
diosarollback	○	○	－	
diosasendtx	○	○	－	
diosasetblockage	○	○	○	

関数名	論理ノード種別			備考
	AP	OLTP	DB	
diosathrinit	○	○	○	
diosathrterm	○	○	○	
diosatmcactv	○	○	－	
diosatmccmdquery	○	○	－	
diosatmccmdset	○	○	－	
diosatmccoquery	○	○	－	
diosatmccoset	○	○	－	
diosatmchold	○	○	－	
diosatmcreset	○	○	－	
diosatrninit	○	○	○	
diosatrnterm	○	○	○	
diosatxstart	○	○	○	
diosaucaget	○	○	－	
diosaunlock	○	○	○	
diosavcall	○	○	○	

○ … 使用可能　－ … 使用不可　△ … 条件付使用可能

A.2.2　　通信制御

関数名	論理ノード種別			備考
	AP	OLTP	DB	
diosadbchangeconnect	○	○	△	※1
diosadbconnect	○	○	△	※1
diosadbdisconnect	○	○	△	※1
diosadbfaultnotification	○	○	△	※1
diosadbmulticonnect	○	○	△	※1
diosadbreconnect	○	○	△	※1
diosagetacspinfo	○	○	－	
diosagetdbctx	○	○	△	※1
diosagetnodepathstatus	○	○	－	
diosagntcheck(_mk)	－	○	－	
diosagntdel(_mk)	－	○	－	
diosagntfin	－	○	－	
diosagntshiftnextsend(_mk)	－	○	－	
diosalspathctrl	○	○	－	

○ … 使用可能　　－ … 使用不可　　△ … 条件付使用可能

※1 … DB ノードの場合、自インスタンスへのアクセスのみ可能とする

A.2.3　　メモリキャッシュ

関数名	論理ノード種別			備考
	AP	OLTP	DB	
diosagethash	○	○	－	
diosagetmap	○	○	－	
diosagetmaphash	○	○	－	
diosagetmaplist	○	○	－	
diosagetmapstat	○	○	－	
diosagettamnode	○	○	－	
diosaimclose	－	○	－	
diosaimcommit	－	○	－	
diosaimcondsetkey	－	○	－	
diosaimcondsetrange	－	○	－	
diosaimctxclose	－	○	－	
diosaimctxopen	－	○	－	
diosaimdelete	－	○	－	
diosaimdeletex1	－	○	－	
diosaimgetmap	－	○	－	
diosaimgetmaplist	－	○	－	
diosaimgetperf	－	○	－	
diosaimgetptblname	－	○	－	
diosaimgetreckeyinfo	－	○	－	
diosaimgetsvcstatus	－	○	－	
diosaimgettblid	－	○	－	
diosaimgettbllist	－	○	－	
diosaimopen	－	○	－	
diosaimread	－	○	－	
diosaimread1	－	○	－	
diosaimrewrite	－	○	－	
diosaimrollback	－	○	－	
diosaimsetmap	－	○	－	
diosaimtruncate	－	○	－	
diosaimtxstart	－	○	－	
diosaimwrite	－	○	－	
diosatamswitch	－	○	－	

○ … 使用可能　　－ … 使用不可　　△ … 条件付使用可能

A.2.4　　データストア

関数名	論理ノード種別			備考
	AP	OLTP	DB	
diosadgetlog	○	○	－	
diosadputlog	○	○	－	
diosalrdcommit	○	○	－	

○ … 使用可能　　－ … 使用不可　　△ … 条件付使用可能

A.2.5　　データ変換・通信オプション

関数名	論理ノード種別			備考
	AP	OLTP	DB	
diatcdbcondsetkey	－	○	－	
diatcdbcondsetrange	－	○	－	
diatcdbctxclose	－	○	－	
diatcdbctxopen	－	○	－	
diatcdbdelete	－	○	－	
diatcdbgetrplmode	－	○	－	
diatcdbgettblid	－	○	－	
diatcdbread	－	○	－	
diatcdbread1	－	○	－	
diatcdbrewrite	－	○	－	
diatcdbsetrplmode	－	○	－	
diatcdbtruncate	－	○	－	
diatcdbwrite	－	○	－	
diatcmkadd	－	○	－	
diatcmkdelete	－	○	－	
diatcmkgettblid	－	○	－	
diatcmkread	－	○	－	

○ … 使用可能　　－ … 使用不可

DIOSA/XTP V3.1
API リファレンス

2022 年　9 月　3 版

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