

D I O S A / X T P V1.1

A P I 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政府の輸出許可が必要です。

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

# はしがき

本書は、業務システムの構築を支援するD I O S A / X T P 製品群のA P I 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、HP-UX、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。

2012 年 10 月 初版

2017 年 4 月 11 版

本書の関連説明書としては次のものがあります。

- D I O S A / X T P 導入の手引き
- D I O S A / X T P 利用の手引き
- D I O S A / X T P メモリキャッシュ 利用の手引き
- D I O S A / X T P データストア 利用の手引き
- D I O S A / X T P コマンドリファレンス
- D I O S A / X T P 環境定義リファレンス
- D I O S A / X T P メッセージリファレンス

## 備考

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

# 目次

<b>第 I 編</b>	<b>利用者インタフェース</b>	<b>1</b>
第 1 章	利用者インタフェースの説明形式	2
1.1	概要	2
1.2	利用者インタフェースの説明形式	2
第 2 章	利用者インタフェース	3
2.1	アプリケーション実行制御	3
2.1.1	関数一覧	3
2.1.2	C0 制御 C0・利用者出口(受信電文解析出口を除く)	4
2.1.3	C0 制御 受信電文解析出口	5
2.1.4	バッチアプリケーション制御 C0/利用者出口	6
2.2	通信制御	7
2.2.1	関数一覧	7
2.2.2	電文種別決定出口	8
2.2.3	データベース接続出口	9
2.3	メモリキャッシュ	10
2.3.1	関数一覧	10
2.3.2	ハッシュ関数	11
2.3.3	利用者領域初期化利用者出口	12
2.4	データストア基盤	13
2.4.1	関数一覧	13
2.4.2	ログリーダプロセス初期化处理	14
2.4.3	ログリーダトランザクション初期化处理	15
2.4.4	ログリーダログデータ実行メイン処理	16
2.4.5	ログリーダトランザクション終了処理	17
2.4.6	ログリーダプロセス終了処理	18
2.4.7	t_diosa_lrduca(ログデータ処理インタフェース構造体)	19
<b>第 II 編</b>	<b>C 言語インタフェース</b>	<b>21</b>
第 1 章	C 言語インタフェースの構成と記述形式	22
1.1	概要	22
1.2	C 言語インタフェースの構成	23
1.3	C 言語インタフェースの説明形式	25
第 2 章	C 言語インタフェース	26
2.1	アプリケーション実行制御	26
2.1.1	関数一覧	26
2.1.2	diosaaddrconv(メモリアドレス取得)	29
2.1.3	diosaapptref(ファイル直接トレース情報出力)	30
2.1.4	diosaapptrcm(メモリ経由トレース情報出力)	31

2.1.5	diosaapptrget (共通領域取得).....	32
2.1.6	diosaapptrset (共通領域設定).....	33
2.1.7	diosacmdconf (コマンド配信結果取得関数).....	34
2.1.8	diosacmdsend (コマンド配信関数).....	35
2.1.9	diosacommit (コミット).....	36
2.1.10	diosaetgreset (経過時間リセット).....	37
2.1.11	diosaetgstart (経過時間監視再開).....	38
2.1.12	diosaetgstop (経過時間監視停止).....	39
2.1.13	diosaetgusinfo (経過時間監視ユーザ情報登録).....	40
2.1.14	diosagetlnodeid (論理ノード ID 取得関数).....	41
2.1.15	diosagetsrctx (トランザクション開始時電文の取得).....	42
2.1.16	diosagoback (C0 制御への強制リターン) .....	43
2.1.17	diosalock (ロック取得).....	44
2.1.18	diosamaddr (メモリアドレス取得).....	46
2.1.19	diosamalloc (メモリ割り当て).....	47
2.1.20	diosamcopy (保護属性書き込み).....	49
2.1.21	diosamfree (メモリ解放).....	50
2.1.22	diosamsbufalloc (電文バッファ確保).....	51
2.1.23	diosamsbuffree (電文バッファ解放).....	52
2.1.24	diosamsgdisp (メッセージ出力関数).....	53
2.1.25	diosamsgedit (メッセージ編集関数).....	55
2.1.26	diosaopsusiput (稼動統計ユーザ情報登録).....	57
2.1.27	diosaprcforkinit (fork 後プロセス初期化関数) .....	58
2.1.28	diosaprcinit (プロセス初期化関数).....	59
2.1.29	diosaprcpreexecterm (exec 前プロセス終了関数) .....	60
2.1.30	diosaprcrterm (プロセス終了関数).....	61
2.1.31	diosarealloc (メモリ再割り当て).....	62
2.1.32	diosarecvtx (電文受信).....	64
2.1.33	diosarollback (ロールバック).....	66
2.1.34	diosasendtx (電文送信).....	67
2.1.35	diosasgclose (SG オブジェクトファイルクローズ) .....	70
2.1.36	diosasgget (SG オブジェクトレコード取得) .....	71
2.1.37	diosasgopen (SG オブジェクトファイルオープン) .....	72
2.1.38	diosathrinit (スレッド初期化関数).....	73
2.1.39	diosathrterm (スレッド終了関数).....	74
2.1.40	diosatmcactv (タイマ保留解除).....	75
2.1.41	diosatmccmdquery (コマンドタイマ照会).....	76
2.1.42	diosatmccmdset (コマンドタイマ登録).....	77
2.1.43	diosatmccommit (タイマコミット処理).....	78
2.1.44	diosatmccoquery (C0 タイマ照会) .....	79
2.1.45	diosatmccoset (C0 タイマ登録) .....	80
2.1.46	diosatmchold (タイマ保留).....	81

2.1.47	diosatmcreset(タイマ削除).....	82
2.1.48	diosatmcrollback(タイマロールバック処理).....	83
2.1.49	diosatrnnit(トランザクション初期化関数).....	84
2.1.50	diosatrnterm(トランザクション終了関数).....	85
2.1.51	diosaucaget(DIOSAUCA アドレス取得) .....	86
2.1.52	diosaunlock(ロック解放).....	87
2.1.53	diosavcall(AP 動的置換対象関数呼び出し) .....	88
2.1.54	diosavdnameget(VD 名取得) .....	89
2.1.55	t_diosa_addrinfo(送受信アドレス構造体).....	90
2.1.56	t_diosa_analyze(受信電文解析出口構造体).....	91
2.1.57	t_diosa_cmdconfuca(コマンド配信結果情報構造体).....	92
2.1.58	t_diosa_cmdnodeconf(配信先ノード単位結果情報構造体).....	93
2.1.59	t_diosa_cmdqueryuca(コマンドタイマ照会用構造体).....	94
2.1.60	t_diosa_cmdresultinfo(配信結果確認情報構造体).....	95
2.1.61	t_diosa_cmdsendinfo(コマンド配信先情報構造体).....	96
2.1.62	t_diosa_cmdsenduca(コマンド配信情報構造体).....	98
2.1.63	t_diosa_cmdsetuca(コマンドタイマ登録用構造体).....	99
2.1.64	t_diosa_coqueryuca(CO 制御タイマ照会用構造体) .....	100
2.1.65	t_diosa_cosetuca(CO 制御タイマ登録用構造体) .....	101
2.1.66	t_diosa_dcuca(電文送受信構造体).....	102
2.1.67	t_diosa_keyinfo(キー情報構造体).....	104
2.1.68	t_diosa_msgbuf(電文バッファ構造体).....	105
2.1.69	t_diosa_sgarea(SG レコード情報格納バッファ構造体) .....	106
2.1.70	t_diosa_sgctrl(SG オブジェクトファイル制御情報構造体) .....	107
2.1.71	t_diosa_sginfo(SG オブジェクトファイル情報構造体) .....	108
2.1.72	t_diosa_tmctime(タイマ時間設定用構造体).....	109
2.1.73	t_diosa_tmcuca(タイマ制御インタフェース構造体).....	110
2.1.74	t_diosa_tpstatus(TPBASE リターン情報構造体) .....	112
2.1.75	t_diosa_uca(CO、利用者出口呼び出し構造体).....	113
2.2	通信制御 .....	119
2.2.1	関数一覧.....	119
2.2.2	diosadbchangeconnect(DB 接続先切り替え関数) .....	120
2.2.3	diosadbconnect(DB 接続関数[シングルコネクション]) .....	121
2.2.4	diosadbdisconnect(DB 切断関数) .....	122
2.2.5	diosadbfaulthnotification(DB 障害通知関数) .....	123
2.2.6	diosadbmulticonnect(DB 接続関数[マルチコネクション]) .....	124
2.2.7	diosadbreconnect(DB 接続関数[再接続]) .....	125
2.2.8	diosagetdbctx(DB コンテキスト取得関数) .....	126
2.2.9	t_diosa_dbconnectuca(DB 接続関数用構造体) .....	127
2.2.10	t_diosa_dbuca(DB 接続先切り替え関数用構造体) .....	128
2.3	メモリキャッシュ .....	129
2.3.1	関数一覧.....	129

2.3.2	diosagethash(メインキーに対応するハッシュ値取得関数)	131
2.3.3	diosagetmap(MAP 情報取得関数)	132
2.3.4	diosagetmaphash(MAP のハッシュ値範囲一覧取得関数)	133
2.3.5	diosagetmaplist(MAP 一覧取得関数)	134
2.3.6	diosagetmapstat(MAP のレプリケーション状態取得関数)	135
2.3.7	diosagettamnode(TAM のノード配置情報取得関数)	136
2.3.8	diosaimclose(IM サーバクローズ関数)	137
2.3.9	diosaimcommit(コミット関数)	138
2.3.10	diosaimcondsetkey(キー指定検索条件設定関数)	139
2.3.11	diosaimcondsetrange(範囲指定検索条件設定関数)	140
2.3.12	diosaimctxclose(検索コンテキストクローズ関数)	141
2.3.13	diosaimctxopen(検索コンテキストオープン関数)	142
2.3.14	diosaimdelete(レコード削除関数)	143
2.3.15	diosaimdeletex1(キー指定レコード削除関数)	145
2.3.16	diosaimgetmap(アクセス先 MAP 取得関数)	147
2.3.17	diosaimgetmaplist(分散先 MAP 一覧照会関数)	148
2.3.18	diosaimgetptblname(物理表名照会関数)	149
2.3.19	diosaimgetreckeyinfo(レコードキー情報照会関数)	150
2.3.20	diosaimgettblid(論理表 ID 照会関数)	151
2.3.21	diosaimgettbllist(論理表一覧照会関数)	152
2.3.22	diosaimopen(IM サーバオープン関数)	153
2.3.23	diosaimread(複数レコード読込関数)	154
2.3.24	diosaimread1(キー指定レコード読込関数)	159
2.3.25	diosaimrewrite(レコード更新関数)	161
2.3.26	diosaimrollback(ロールバック関数)	163
2.3.27	diosaimsetmap(アクセス先 MAP 宣言関数)	164
2.3.28	diosaimtruncate(全レコード削除関数)	165
2.3.29	diosaimtxstart(トランザクション開始関数)	167
2.3.30	diosaimwrite(レコード追加関数)	168
2.3.31	diosasetmap(MAP 情報更新関数)	170
2.3.32	diosatamswitch(マスタ昇格関数)	171
2.3.33	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)	172
2.3.34	t_diosa_getmapuca(MAP 情報取得関数インタフェース構造体)	173
2.3.35	t_diosa_imcond(検索条件構造体)	174
2.3.36	t_diosa_imreckeyinfo(レコードキー情報構造体)	175
2.3.37	t_diosa_imrefctx(照会コンテキスト構造体)	176
2.3.38	t_diosa_imtbllist(論理表一覧構造体)	177
2.3.39	t_diosa_mapent(MAP 情報エントリ構造体)	178
2.3.40	t_diosa_maphashent(MAP のハッシュ値範囲情報エントリ構造体)	179
2.3.41	t_diosa_recinfo(レコード情報構造体)	180
2.3.42	t_diosa_setmapent(MAP 情報更新関数用 MAP 情報エントリ構造体)	181
2.3.43	t_diosa_setmapuca(MAP 情報更新関数インタフェース構造体)	182

2.3.44	t_diosa_tamnodeent (TAM のノード配置情報エントリ構造体)	183
2.4	データストア基盤	184
2.4.1	関数一覧	184
2.4.2	diosadgetlog (ログデータ読み込み)	185
2.4.3	diosadputlog (ログデータ書き込み)	186
2.4.4	diosalrdcommit (ログリーダ強制コミット)	187
2.4.5	t_diosa_dloggetuca (ログデータ読み込み UCA)	188
2.4.6	t_diosa_dloguca (ログデータ書き込み UCA)	189
<b>第 III 編</b>	<b>Java インタフェース</b>	<b>190</b>
第 1 章	Java インタフェース	191
1.1	クラス一覧	191
1.2	DiosaDlogData	192
1.2.1	DiosaDlogData クラス	192
1.2.2	compressFlg フィールド	193
1.2.3	coName フィールド	193
1.2.4	spstName フィールド	193
1.2.5	streamName フィールド	193
1.2.6	textLen フィールド	193
1.2.7	userData フィールド	193
1.2.8	DiosaDlogData コンストラクタ	194
1.2.9	getCompressFlg メソッド	194
1.2.10	getCoName メソッド	194
1.2.11	getSpstName メソッド	195
1.2.12	getStreamName メソッド	195
1.2.13	getTextLen メソッド	195
1.2.14	getUserData メソッド	195
1.2.15	setCompressFlg メソッド	196
1.2.16	setCoName メソッド	196
1.2.17	setSpstName メソッド	196
1.2.18	setStreamName メソッド	196
1.2.19	setTextLen メソッド	197
1.2.20	setUserData メソッド	197
1.3	DiosaDlogDataConst	197
1.3.1	DiosaDlogDataConst クラス	197
1.3.2	DIOSA_DATACOMP_ON フィールド	197
1.3.3	DIOSA_DATACOMP_OFF フィールド	198
1.4	DiosaDPutLog	198
1.4.1	DiosaDPutLog クラス	198
1.4.2	DiosaDPutLog コンストラクタ	198
1.4.3	putlog メソッド	199



付録 A   A P I 一 覧..... 200

    A. 1   A P I が利用可能な動作環境 ..... 200

# 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、D I O S A / X T P 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

D I O S A / X T P 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

(2) **バッチアプリケーション制御**

CO／利用者出口      バッチアプリケーション制御上で動作するCO／利用者関数の形式

## 2.1.2 C0 制御 C0・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

C0 制御上で動作する C0 および利用者出口である。

**プログラム名**は C0 および各利用者出口ごとに C0 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、C0 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。

## 2.1.4 バッチアプリケーション制御 C0／利用者出口

### 書式

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

### 説明

利用者は、バッチアプリケーション制御上で動作する C0／利用者出口を、書式で示すインタフェースで作成する。

DiosaUca は diosa が用意する。各出口呼び出し時にパラメータとして利用者出口に渡される。

#### DiosaUca（入出力型）

t\_diosa\_uca の項を参照。

## 2.2 通信制御

### 2.2.1 関数一覧

(1) ノード間通信パス管理機能

電文種別決定出口	入力電文を解析し、起動するトランザクション ID を決定する
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



## 2.2.2 電文種別決定出口

### 書式

```
#include <diosa.h>
int GetMsgTypeExit(t_diosa_getmsgtypeuca *GetMsgTypeUca)
```

### 説明

入力された電文を解析し、起動すべきトランザクション ID と電文長を返却する利用者出口。  
同名の関数を作成することで、受信した電文を解析し、トランザクション ID を決定するロジックを変更することができる。

**t\_diosa\_getmsgtypeuca \*GetMsgTypeUca** (入出力型)

○\* GetMsgTypeUca 構造体は以下のメンバを含んでいる。

void *MsgPtr	電文格納領域へのポインタが格納される。(入力)
int LdDtLen	電文格納領域の長さが格納される。(入力)
int MsgLen	総電文長を返却する。(出力)
char TxId[7]	AP ノードで起動するトランザクション ID を利用者が格納する。(出力)

○ユーザ作成出口の使用方法

1. GetMsgTypeExit という関数を含む libdxtpncmgetmsgtype.so というライブラリを作成する。
2. 作成したライブラリを DIOSA のインストールディレクトリよりも読込優先順位の高いパスに配置する。

※規定の電文種別決定出口は何もしない。

DIOSA ヘッダが付加されていない電文を送信すると必ず電文は破棄される。

### 戻り値

利用者出口の処理結果を返却する。

電文種別決定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
DIOSA_BREAK(11)	受信した電文を破棄する (正常終了)
DIOSA_ERROR(-1)	受信した電文を破棄する (異常終了)
その他 (負値)	DIOSA_ERROR と同じ

### 注意

本出口は AP ノードのリスナ EXIT 上で実行されるため、出口名は固定となる。

リスナのパラメータ MSGHD\_SIZE (読込サイズ) には 26 以上を指定すること。

リスナ上で動作するためスレッドセーフな関数として作成すること。

## 2.2.3 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_dbinfouca *DbInfoUca)
```

### 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 が定義されていない場合に呼び出される利用者出口。

### DbInfoUca（入出力型）

t\_diosa\_dbinfouca 構造体により情報の授受を行う。

○t\_diosa\_dbinfouca 構造体は以下のメンバを含む。

char DbName[137]（入力型）

接続先（ネット・サービス名）が格納される。（1～136 バイト+NULL 文字）

sql\_context \* 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。

## 2.3    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
利用者領域初期化利用者出口	利用者領域初期化時の利用者関数の形式

## 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。

## 2.3.3 利用者領域初期化利用者出口

### 書式

```
#include <diosa.h>

int プログラム名( char *CmdParam,
                  unsigned int  DataLen,
                  char *Data )
```

### 説明

インメモリサーバ所在管理テーブルの利用者領域に設定する初期値を指定する利用者出口である。  
-c 指定の diiminit コマンド実行時、および diimchg コマンド実行時に呼び出される。

#### CmdParam (入力型)

diiminit コマンドの -p パラメータで指定された文字列。(NULL 終端)  
diiminit コマンドで -p を省略した場合、および diimchg コマンドの場合は、NULL。

#### DataLen (入力型)

Data の領域長。

#### Data (出力型)

インメモリサーバ所在管理テーブルの利用者領域に設定する初期値。

### 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。コマンドの処理を中断する。

### 注意

DataLen は、環境定義の利用者領域長で指定された値が設定される。

\*Data の領域は、インメモリサーバ所在管理で確保し、確保したサイズを DataLen に設定して利用者領域初期化利用者出口に渡す。

利用者領域初期化利用者出口内で、DataLen の妥当性チェックを行う必要がある。

## 2.4 データストア基盤

### 2.4.1 関数一覧

(1) ログリーダー

プロセス初期化处理	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处理	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处理

### 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	0	I	I	I	I
TrnArea	ユーザ引継ぎエリア		0	I	I	I
TrnEndStatus	トランザクション処理結果			0	I	
PrcType	処理タイプ				I	
TrnType	トランザクションタイプ				I	

I：ログリーダーから利用者出口へ渡す情報

0：利用者出口からログリーダーへ渡す情報

## 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、D I O S A / X T P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 );
```

↑                    ↑                    ↑                    ↑  
変数宣言            パラメータ値      変数宣言      パラメータ値      変数宣言      パラメータ値

### (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

D I O S A / X T P が提供する A P I 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) C O 制御

t_diosa_uca	C0・利用者出口パラメータ (DIOSAUCA)
t_diosa_keyinfo	キー情報
t_diosa_addrinfo	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
diosacommit	トランザクションのコミット
diosagetsrctx	トランザクション開始時電文の取得
diosagoback	C0 制御への強制リターン
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
diosarollback	トランザクションのロールバック
diosasendtx	トランザクション電文の送信
diosaucaget	diosauca の取得
diosavdnameget	TxId に対応する VD 名取得

##### (2) A P 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
diosatmccommit	タイマコミット処理
diosatmccoquery	C0 タイマ照会
diosatmccoset	C0 タイマ登録
diosatmchold	タイマ保留

	diosatmcreset	タイマ削除
	diosatmcrollback	タイマロールバック処理
	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	C0 制御タイマ照会用構造体
	t_diosa_cosetuca	C0 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)	<b>稼動統計</b>	
	diosaopsusiput	稼動統計ユーザ情報登録
(5)	<b>メッセージ出力</b>	
	diosamsdisp	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
	diosamsedit	メッセージを編集し、編集結果を返却する
(6)	<b>D I O S A 共通初期化終了</b>	
	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
	diosatrnnit	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
(7)	<b>経過時間監視</b>	
	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)	<b>メモリ管理</b>	
	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)	<b>A P P トレース</b>	
	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
(10)	<b>ロック制御</b>	
	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)      **環境定義**

t_diosa_sginfo	SG オブジェクトファイル情報構造体
t_diosa_sgctrl	SG オブジェクトファイル制御情報構造体
t_diosa_sgarea	レコード情報格納バッファ構造体
diosasgopen	SG オブジェクトファイルをオープンする
diosasgget	SG オブジェクトファイルからレコードを取得する
diosasgclose	SG オブジェクトファイルをクローズする

(12)      **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

(13)      **起動／停止機能**

diosagetlnodeid	論理ノード ID 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)

メモリアドレス領域ポインタ。

#### 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 diosaapptrcf(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

### 書式

```
#include <diosa.h>

int diosaapptrcf(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。

### 戻り値

DIOSA_DONE(0)	トレース情報の出力に成功した。
DIOSA_IGNORE(9)	<b>level</b> 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)	<b>diapptrcinit</b> 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 <b>DIOSA_APPTRCFFILEOUTPUTMODE</b> の値が ROTATE かつ <b>DIOSA_APPTRCFOVERWRITE</b> の値が OFF の場合のみ)
DIOSA_ERROR(-1)	内部処理に失敗した。

## 2.1.4 diosaapptrcm(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

### 書式

```
#include <diosa.h>

int diosaapptrcm(char *comment, void *data, int size, int errcode, unsigned char level);
```

### 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。

### 戻り値

DIOSA_DONE(0)	トレース情報の格納に成功した。
DIOSA_IGNORE(9)	<b>level</b> 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 が負の値である等)
DIOSA_ENOINIT(-11)	トレース情報保存領域が存在しない。 <b>diapptrcinit</b> 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 <b>DIOSA_APPTRCMOVERWRITE</b> の値が OFF の場合のみ)
DIOSA_ERROR(-1)	内部処理に失敗した。



## 2.1.5 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。

**int \*\*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

## 2.1.6 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。

**int \*\*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**

## 2.1.7 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

## 2.1.8 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_EFUNCAV(-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

## 2.1.9 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, int *UserStatus );
```

### 説明

CO 内でコミットを行う。

本 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。

環境定義にてコミット出口が定義されている場合、本 API によってコミット出口が呼び出される。

コミット出口が異常終了要求を返した場合、本 API は DIOSA\_EABORT をリターンし、UserStatus には、コミット出口が通知した利用者コードが格納される。

### 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_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 A P から要求された
DIOSA_EACCES(-25)	メモリデータ管理アクセスエラー
DIOSA_EABORT(-4)	コミット出口が異常終了要求を通知した
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_EEXIT(-119)	コミット出口が不正な状態コードを通知した
DIOSA_ECALLEXIT(-38)	コミット出口の呼び出しに失敗した
DIOSA_ETPBASE(-120)	TPBASE のエラー
DIOSA_ERROR (-1)	その他 diosa エラー

### 関連

コミット出口

## 2. 1. 10 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過し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。

**diosaetgreset()** を CO 制御またはバッチ AP 制御管理下の AP 以外で実行した場合、または経過時間監視テーブルのエントリを既にオーバーしていた場合、DIOSA\_ENOINIT が返却され、経過時間をリセットすることは出来ない。

## 2.1.11 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

### 注意

diosaetgstart() を CO 制御またはバッチ AP 制御管理下の AP 以外で実行した場合、または経過時間監視テーブルのエントリを既にオーバーしていた場合、DIOSA\_ENOINIT が返却される。

### 関連

diosaetgstop

## 2. 1. 12      **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

### 注意

**diosaetgstop()** を CO 制御またはバッチ AP 制御管理下の AP 以外で実行した場合、または経過時間監視テーブルのエントリを既にオーバーしていた場合、DIOSA\_ENOINIT が返却される。

### 関連

diosaetgstart



## 2. 1. 13 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 CD0 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

### 注意

**diosaetgusinfo()** を CO 制御またはバッチ AP 制御管理下の AP 以外で実行した場合、または経過時間監視テーブルのエントリを既にオーバーしていた場合、DIOSA\_ENOINIT が返却される。

## 2.1.14 diosagetlnodeid(論理ノード ID 取得関数)

### 名前

diosagetlnodeid - 論理ノード ID を取得する。

### 書式

```
#include <diosa.h>
int diosagetlnodeid (int* LNodeId);
```

### 説明

論理ノード ID を取得する。

**LNodeId**                      論理ノード ID を返却する。(出力)

### 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)            未初期化

DIOSA\_EFATAL(-30)            異常終了

### 関連

diosaprcinit

## 2. 1. 15      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_dcua *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 C0 およびアボート出口#1, #2 において、連鎖前の C0 が diosarecvtx により取得した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 C0 が取得した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。

### 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 A P から要求された

### 関連

t\_diosa\_uca, diosarecvtx

## 2. 1. 16      diosagoback (C0 制御への強制リターン)

### 名前

diosagoack - C0 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 C0 制御内のメイン処理へ移動する。

C0 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 A P から要求された

### 関連

t\_diosa\_uca

## 2.1.17 diosaloglock(ロック取得)

### 名前

diosaloglock - 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 diosaloglock(int Id, int Mode);
```

### 説明

**diosaloglock()** 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 diosacommmit/diosarollback 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. OracleDB 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 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。

C0 制御プロセス初期化出口、バッチ AP 制御初期化出口、及びログリーダプロセス初期化出口で取得したロックは、C0 制御 C0、バッチ AP 制御 C0、及びログデータ実行メイン処理出口に引き継ぐことはできない(各初期化出口内でロックの解放を行う必要がある)。

## 関連

diosaunlock

## 2.1.18 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)

<code>DIOSA_APNOPROT</code>	更新可共有メモリ
<code>DIOSA_APPROT</code>	保護属性共有メモリ
<code>DIOSA_PRCNOALL</code>	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)
<code>DIOSA_THRNOALL</code>	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)
<code>DIOSA_SVCALL</code>	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

<code>DIOSA_DONE(0)</code>	処理が正常に終了した。
<code>DIOSA_ERROR(-1)</code>	アドレス取得に失敗した。
<code>DIOSA_EPARAM(-3)</code>	パラメータに誤りがある。
<code>DIOSA_ENOENT(-8)</code>	指定されたメモリ識別子に対応する領域は存在しない。
<code>DIOSA_EFUNCAV(-34)</code>	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`

## 2.1.19 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。



エントリー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ー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. 20      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_EPARAM(-3)	パラメータに誤りがある。
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。
DIOSA_EATTACH(-43)	テーブルアタッチに失敗した。

### 関連

diosamalloc, diosarealloc

## 2.1.21 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. 22      diosmsgbufalloc(電文バッファ確保)

### 名前

diosmsgbufalloc - 電文バッファの確保

### 書式

```
#include <diosa.h>

int    diosmsgbufalloc( 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 Sizre 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 A P から要求された

### 関連

t\_diosa\_msgbuf, diosmsgbuffree, diossendtx, t\_diosa\_analyze

## 2. 1. 23      diosmsgbuffree(電文バッファ解放)

### 名前

diosmsgbuffree - 電文バッファの解放

### 書式

```
#include <diosa.h>

int    diosmsgbuffree( 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 A P から要求された

### 関連

t\_diosa\_msgbuf, diosmsgbufalloc

## 2. 1. 24      diosamsdisp(メッセージ出力関数)

名前

diosamsdisp - 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 diosamsdisp(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_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_ERECD (-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いうが、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。

置換テキストの表示書式 (**Format**) に誤りがあった場合、誤った書式以降は指定されなかったものとして動作する。

**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. 25      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_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。



置換テキストの表示書式(**Format**)に誤りがあった場合、誤った書式以降は指定されなかったものとして動作する。

**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. 26 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\_ERROR(-1) 異常終了

DIOSA\_EPARAM(-3) パラメータ不正

### 注意

**diosaopsusiput()** は、同一 C0 実行区間内で登録可能なユーザ情報は 1 個のみである。

同一 C0 実行区間内で本関数が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. 27      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(void* Info)
```

### 説明

ユーザアプリケーション(常駐アプリケーションやコマンド等のアプリケーション、CO 制御、バッチ AP 制御上のアプリケーションは除く)のための **fork** 後のプロセス初期処理を行う。

**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)

### 関連

diosaprcpreexecterm

## 2. 1. 28      **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(void* Info)
```

### 説明

ユーザアプリケーション(常駐アプリケーションやコマンド等のアプリケーション、CO 制御、バッチ AP 制御上のアプリケーションは除く)のためのプロセス初期処理を行う。

**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_ALREADY(1)	二重実行
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. 29      **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(常駐アプリケーションやコマンド等のアプリケーション。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. 30      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(常駐アプリケーションやコマンド等のアプリケーション、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(プロセス初期化处理)を呼んだ場合、必ず本関数を実行すること。

### 関連

diosaprcinit

## 2. 1. 31 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. 32 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

C0 およびアポート出口#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。

電文特性ごとに設定される **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_EFUNCNAV (-34)	本マクロを使用できないプロセス上の A P から要求された

### 関連

t\_diosa\_dcuca, diosgetsctx

<diosarecvtx>

t_diosa_dcuca			電文特性				
			論理システム間	システム内派生		TPP間派生	連鎖
				AP→OLTP	OLTP→AP		
TextChar			○	○	○	○	○
SafInfo	LsName		—	○	—	—	—
	LNodeName		—	○	—	—	—
	TpMonitor		—	○	—	—	—
	KeyInfo	Flag	—	—	○	○	—
		MainKey	—	—	○	○	—
		MAP	—	—	○	○	—
DafInfo	LsName		○	○	○	○	—
	LNodeName		—	○	○	○	—
	TpMonitor		—	○	○	○	—
	KeyInfo	Flag	—	○	—	—	—
		MainKey	—	○	—	—	—
		MAP	—	○	—	—	—
TermName			○	—	—	—	—
VdName			—	—	—	○	—
CallId			—	—	—	—	—
TxId			○	○	○	○	—
CoName			○	○	○	○	○
DlayInfo	SStream		—	—	—	—	—
	Stream		—	—	—	—	—
	DivId		—	—	—	—	—
	DivSeq		—	—	—	—	—
RecvId			○	○	○	○	—
MsgSize			○	○	○	○	○
DerivNum			—	○	○	—	—
SimFlag			○	○	○	○	○
SendMode			—	—	—	—	—
TpRet			—	—	—	—	—

## 2. 1. 33 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, int *UserStatus );
```

### 説明

C0 内でロールバックを行う。

本 API 実行前のメモリデータ管理アクセスは無効化され、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。

環境定義にてロールバック出口が定義されている場合、本 API によってロールバック出口が呼び出される。ロールバック出口が異常終了要求を返した場合、本 API は DIOSA\_EABORT をリターンし、UserStatus には、ロールバック出口が通知した利用者コードが格納される。

### 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 A P から要求された
DIOSA_EACCES(-25)	メモリデータ管理アクセスエラー
DIOSA_EABORT(-4)	ロールバック出口が異常終了要求を通知した
DIOSA_EEXIT(-119)	ロールバック出口が不正な状態コードを通知した
DIOSA_ECALLEXIT(-38)	ロールバック出口の呼び出しに失敗した
DIOSA_ETPBASE(-120)	TPBASE のエラー
DIOSA_ERROR (-1)	その他 diosa エラー

### 関連

ロールバック出口

## 2. 1. 34 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、電文特性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_EPARAM(-3)	パラメータエラー
DIOSA_ENOENT(-8)	該当宛先なし
DIOSA_ETPSEND(-120)	TP_send エラー(詳細は CS_STRUCT の status_key, end_key を参照)
DIOSA_EFUNCNAV(-34)	動作環境エラー
DIOSA_EMEM(-6)	メモリアクセスエラー
DIOSA_ECONFLICT(-110)	diosadcuca パラメータエラー
DIOSA_ESG(-77)	diosa 環境定義エラー
DIOSA_ESEND(-15)	TPBASE パス障害
DIOSA_ESIZE(-33)	電文長エラー
DIOSA_EBLOCK(-21)	自ノード閉塞中
DIOSA_EBLOCK(-1)	その他 diosa エラー

### 関連

t\_diosa\_dcuca, diosarecvtx, t\_diosa\_msgbuf, diosamsgbufalloc, diosa\_msgbuffree

<diosasendtx>

t_diosa_dcuca			電文特性					
			論理システム間 注 2	システム 内派生		TPP 間派生	連鎖	保留
				AP→OLTP	OLTP→AP			
TextChar			●	●	●	●	●	●
SafInfo	LsName		—	□	—	—	—	—
	LNodeName		—	□	—	—	—	—
	TpMonitor		—	□	—	—	—	—
	KeyInfo	Flag	—	—	□	□	—	□
		MainKey	—	—	□	□	—	□
		MAP	—	—	□	□	—	□
DafInfo	LsName		●	—	—	—	—	—
	LNodeName		—	★	★	—	—	—
	TpMonitor		—	★	★	—	—	—
	KeyInfo	Flag	—	★	●	—	—	—
		MainKey	—	★	—	—	—	—
		MAP	—	★	—	—	—	—
TermName			● 注 1	—	—	—	—	—
VdName			—	—	—	—	—	—
CallId			—	—	—	—	—	—
TxId			—	●	●	●	—	●
CoName			—	□	□	□	●	□
DlayInfo	SStream		—	—	—	—	—	—
	Stream		—	—	—	—	—	—
	DivId		—	—	—	—	—	—
	DivSeq		—	—	—	—	—	—
RecvId			—	—	—	—	—	—
MsgSize			●	●	●	●	●	●
DerivNum			—	—	—	—	—	—
SimFlag			—	—	—	—	—	—
SendMode			●	●	●	●	—	●
PreBlkFlag			—	★	★	—	—	—
TpRet			○	○	○	○	○	○

●：必須  
□：任意指定可  
（既定値＝無効値）  
★：選択指定（後述）  
○：返却される情報

注 1) 「論理システム間」電文を受信した際に diosarecvtx で渡される「端末名」を指定する必要がある。  
注 2) 本バージョンでは指定できません。

システム内派生において、DafInfo について、キー情報が有効の場合は論理ノード(LNodeName)と TP モニター(TpMonitor)を指定することはできない。

また、キー情報を無効とした場合、論理ノードおよび TP モニタを指定するか、ノードを指定せずに巡回選択にするかを選択することができる。

宛先	KeyInfo			LNodeName	TpMonitor
	KeyFlag	MainKey	MAP		
メインキー	DIOSA_KEY_MAINKEY	指定			
MAP	DIOSA_KEY_MAP		指定		
ノード指定	DIOSA_KEY_UNAVAIL			指定	なし
ノード+TPBASE 指定				指定	指定
ノード巡回選択				なし	なし

## 2. 1. 35      **diosasgclose** (SG オブジェクトファイルクローズ)

### 名前

diosasgclose - SG オブジェクトファイルクローズ

### 書式

```
#include <diosa.h>

Int     diosasgclose( t_diosa_sgctrl* Ctrl );
```

### 説明

**diosasgclose()** は、**diosasgopen()** でオープンした SG オブジェクトファイルをクローズする。

**Ctrl**                      SG オブジェクトファイルの SG オブジェクトファイル制御情報を指定する。（入力）

### 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_ELOCK(-14)	SG オブジェクトファイルのロック処理で異常終了
DIOSA_EFILE(-46)	SG オブジェクトファイルが破壊されている
DIOSA_EFUNCNAV(-34)	SG オブジェクトファイルがオープンされていない

### 注意

本関数は CDO メッセージを出力しない。

**Ctrl** は **diosasgopen()** に渡したものと同一変数でなければならない。

### 関連

diosasgopen, diosasgget

## 2. 1. 36      diosasgget (SG オブジェクトレコード取得)

### 名前

diosasgget - SG オブジェクトレコード取得

### 書式

```
#include <diosa.h>

Int     diosasgget( t_diosa_sgarea* Area, t_diosa_sginfo* Info, t_diosa_sgctrl* Ctrl );
```

### 説明

**diosasgget()** は、**diosasgopen()** でオープンした SG オブジェクトファイルから、1 レコード取得する。

<b>Area</b>	レコード情報格納バッファを指定する。(入力)
<b>Info</b>	SG オブジェクトファイルの SG オブジェクトファイル情報を指定する。(入力/出力)
<b>Ctrl</b>	SG オブジェクトファイルの SG オブジェクトファイル制御情報を指定する。(入力)

### 戻り値

レコードを正常に取得した場合には、0 が返される。全てのレコードを取得した場合には、DIOSA\_DATA LIM 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_DATA LIM(4)	レコードの終端(EOF)を検出
DIOSA_EFILE(-46)	SG オブジェクトファイルが破壊されている。または SG オブジェクトファイル fseek 異常終了
DIOSA_EFUNCAV(-34)	SG オブジェクトファイルがオープンされていない
DIOSA_EINVAL(-9)	パラメータエラー(例：指定した読み込みバッファの指定サイズが不十分)

### 注意

本関数は CDO メッセージを出力しない。

**Ctrl** は **diosasgopen()** に渡したのと同じ変数でなければならない。

### 関連

diosasgopen, diosasclose



## 2. 1. 37 diosasgopen(SG オブジェクトファイルオープン)

### 名前

diosasgopen - SG オブジェクトファイルオープン

### 書式

```
#include <diosa.h>

int diosasgopen( char* Secname, t_diosa_sginfo* Info, t_diosa_sgctrl* Ctrl );
```

### 説明

**diosasgopen()** は、指定したセクションの SG オブジェクトファイルをオープンする。

<b>Secname</b>	オープンする SG オブジェクトファイルのセクション名を指定する。最大 12 バイト +NULL (入力)
<b>Info</b>	オープンした SG オブジェクトファイルの SG オブジェクトファイル情報を返却する。(出力)
<b>Ctrl</b>	オープンした SG オブジェクトファイルの SG オブジェクトファイル制御情報を返却する。(出力)

### 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_ELOCK(-14)	SG オブジェクトファイルのロック処理で異常終了
DIOSA_EMEM(-6)	メモリ確保処理で異常終了
DIOSA_ENOINIT(-11)	SG オブジェクトファイルが見つからない(主な原因 : DIOSA_IRMROOT 未設定、DIOSA_IRMROOT で指定したディレクトリが存在しない)
DIOSA_EFILE(-46)	SG オブジェクトファイルが破壊されている、または SG オブジェクトファイル fseek 異常終了
DIOSA_EINVAL(-9)	指定されたセクションが存在しない

### 注意

本関数は CDO メッセージを出力しない。

本関数を実行するには、環境変数 DIOSA\_IRMROOT の値が設定されている必要である。

パラメータに指定する構造体は、呼び出し元で領域を確保する必要がある。また、各構造体は NULL で初期化する必要がある。

### 関連

diosasgget, diosasgclose

## 2. 1. 38      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(常駐アプリケーションやコマンド等のアプリケーション、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**(プロセス初期化处理)が呼ばれている必要がある。

本関数は **diosaprcinit** (プロセス初期化处理)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(**diosatrnrinit** と **diosatrnrterm** 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, diosaprcrterm, diosathrterm , diosatrnrinit, diosatrnrterm

## 2. 1. 39      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(常駐アプリケーションやコマンド等のアプリケーション、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. 40 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** の項を参照。

### 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

要求が受け付けられるのは、diosatmcactv() を使用しているトランザクションの終了後になる。

各パラメータの指定は必須である。

本 API 実行から 30 秒以上コミットが実行されない場合、タイマデーモンの内部制御によってロールバックされる可能性があるため注意すること。

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. 41 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。

<b>TmcUca</b>	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。 <b>t_diosa_tmcuca</b> 構造体については、 <b>t_diosa_tmcuca</b> の項を参照。
<b>CmdqueryUca</b>	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。 <b>t_diosa_cmdqueryuca</b> 構造体については、 <b>t_diosa_cmdqueryuca</b> の項を参照。
<b>Command</b>	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. 42 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。

<b>TmcUca</b>	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。 <b>t_diosa_tmcuca</b> 構造体については、 <b>t_diosa_tmcuca</b> の項を参照。
<b>CmdsetUca</b>	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。 <b>t_diosa_cmdsetuca</b> 構造体については、 <b>t_diosa_cmdsetuca</b> の項を参照。
<b>Command</b>	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

Mode (時刻形式) に時刻指定 (DIOSA\_CLOCK) を指定した場合、通知回数を指定することはできない。  
Mode (時刻形式) に時刻指定を指定し、タイマ値に過去の時刻を設定又は、存在しない時刻を設定した場合、DIOSA\_ETIMEFORM となる。  
タイマ制御の仕様上、実行対象のコマンドが “126, 127” で exit すると、コマンド実行エラーとの判別ができないため注意が必要である。  
本 API 実行から 30 秒以上コミットが実行されない場合、タイマデーモンの内部制御によってロールバックされる可能性があるため注意すること。  
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, diosatmccreset, diosatmccmdquery

## 2.1.43 diosatmccommit(タイマコミット処理)

### 名前

diosatmccommit - タイマコミット処理

### 書式

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

### 説明

**diosatmccommit()** は、タイマ情報の実登録を行う。

### 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_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

C0 制御およびバッチ AP 制御上で動作する C0 では、**diosatmccommit()** は不要である。

C0 制御、バッチ AP 制御以外のプロセスで、タイマの登録／削除／保留／保留解除が正常終了し該当の処理を確定させる場合は、トランザクションの終了時に **diosatmccommit()** を呼び出すこととする。

本 API はシングルスレッド上での動作を保証する。

### 関連

diosatmccoset, diosatmccmdset, diosatmcreset,  
diosatmchold, diosatmcactv, diosatmcrollback

## 2. 1. 44    diosatmccoquery (C0 タイマ照会)

### 名前

diosatmccoquery - C0 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、C0 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。

<b>TmcUca</b>	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。 <b>t_diosa_tmcuca</b> 構造体については、 <b>t_diosa_tmcuca</b> の項を参照。
<b>CoqueryUca</b>	C0 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。 <b>t_diosa_coqueryuca</b> 構造体については、 <b>t_diosa_coqueryuca</b> の項を参照。
<b>Data</b>	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

C0 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. 45      diosatmccoset (C0 タイマ登録)

### 名前

diosatmccoset - C0 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、C0 タイマの登録を行う。

<b>TmcUca</b>	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。 <b>t_diosa_tmcuca</b> 構造体については、 <b>t_diosa_tmcuca</b> の項を参照。
<b>CosetUca</b>	C0 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。 <b>t_diosa_cosetuca</b> 構造体については、 <b>t_diosa_cosetuca</b> の項を参照。
<b>Data</b>	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)	エントリをオーバした。
DIOSA_ETIMEFORM(-92)	指定された時刻(時間)が不正である。

### 注意

Mode(時刻形式)に時刻指定(DIOSA\_CLOCK)を指定した場合、通知回数を指定することはできない。

Mode(時刻形式)に時刻指定を指定し、タイマ値に過去の時刻を設定又は、存在しない時刻を設定した場合、DIOSA\_ETIMEFORM となる。

本 API 実行から 30 秒以上コミットが実行されない場合、タイマデーモンの内部制御によってロールバックされる可能性があるため注意すること。

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, diosatmccreset, diosatmccoquery

## 2. 1. 46 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** の項を参照。

### 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

要求が受け付けられるのは、diosatmchold() を使用しているトランザクションの終了後になる。

各パラメータの指定は必須である。

本 API 実行から 30 秒以上コミットが実行されない場合、タイマデーモンの内部制御によってロールバックされる可能性があるため注意すること。

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

## 2.1.47 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** の項を参照。

### 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

各パラメータの指定は必須である。

本 API 実行から 30 秒以上コミットが実行されない場合、タイマデーモンの内部制御によってロールバックされる可能性があるため注意すること。

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. 48      diosatmcrollback(タイマロールバック処理)

### 名前

diosatmcrollback - タイマロールバック処理

### 書式

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

### 説明

**diosatmcrollback()** は、タイマ情報の破棄を行う。

### 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_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

C0 制御およびバッチ AP 制御上で動作する C0 では、diosatmcrollback() は不要である。

C0 制御、バッチ AP 制御以外のプロセスで、タイマの登録／削除／保留／保留解除が異常終了等で該当の処理を破棄したい場合は、トランザクションの終了時に diosatmcrollback() を呼び出すこととする。

本 API はシングルスレッド上での動作を保証する。

### 関連

diosatmccoset, diosatmccmdset, diosatmcreset,  
diosatmchold, diosatmcactv, diosatmccommit

## 2. 1. 49      diosatrnninit(トランザクション初期化関数)

### 名前

diosatrnninit - 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 diosatrnninit(void* Info)
```

### 説明

ユーザアプリケーション(常駐アプリケーションやコマンド等のアプリケーション、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前後で、それぞれ本関数 **diosatrnninit**(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。

**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

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. 50      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(常駐アプリケーションやコマンド等のアプリケーション、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。

### 戻り値

DIOSA_DONE(0)	正常終了
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. 51      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・C0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・C0 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 A P から要求された

### 関連

t\_diosa\_uca, diosagoback

## 2. 1. 52      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。

C0 制御プロセス初期化出口、バッチ AP 制御初期化出口、及びログリーダプロセス初期化出口で取得したロックは、C0 制御 C0、バッチ AP 制御 C0、及びログデータ実行メイン処理出口に引き継ぐことはできない(各初期化出口内でロックの解放を行う必要がある)。

### 関連

diosalock



## 2. 1. 53      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(), diosatrnninit(), diosatrnrterm()

## 2. 1. 54      diosavdnameget (VD 名取得)

### 名前

diosavdnameget - CO 制御管理下の TxId に対応する VD 名の取得

### 書式

```
#include <diosa.h>

int     diosavdnameget( const char *TxId,   char *VdName );
```

### 説明

任意のプロセスにおいて、CO 制御が管理する TxID に対応するトランザクション型 VD の VD 名を取得する。  
TxId によって指定される TxID はヌル文字により終端された文字列でなければならない。  
VD 名が格納される VdName が指す領域は 25 バイト以上の領域でなければならない。

### 戻り値

DIOSA_DONE (0)	正常に実行が終了した
DIOSA_EPARAM (-3)	TxId, VdName 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

## 2. 1. 55     t\_diosa\_addrinfo (送受信アドレス構造体)

### 名前

t\_diosa\_addrinfo – アドレス情報

### 書式

```
#include <diosa.h>

t_diosa_addrinfo    アドレス情報;
```

### 説明

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)
char    TpMonitor[8+1];	TPBASE モニタ名 (最大 8 文字)
t_diosa_keyinfo    KeyInfo;	キー情報

キー情報については、t\_diosa\_keyinfo の章を参照

電文の送信先／送信元の情報をアドレス形式で示す

### 関連

t\_diosa\_dcuca、diosarecvtx、diosagetsrctx、diosasendtx

2. 1. 56      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

以下のメンバ変数を持つ。

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   InMsgFlag;	diosa 論理システム内から送信された電文識別
char    CoName[30+1];	C0 名（最大 30 文字）

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。
InMsgFlag	○	—	diosa 論理システム内から送信された電文識別 DIOSA_ON :diosa 論理システム内電文 DIOSA_OFF:diosa 外電文
CoName	○	○	受信電文が、実行する C0 が指定された電文であった場合、その C0 名が設定される。そうでない場合は空文字列が設定される。 利用者は、任意の C0 名に書き換えて出口を終了することにより、実行する C0 を変更することができる。

関連

t\_diosa\_msgbuf, diosamsgbufalloc, diosamsgbuffree

### 2.1.57 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. 58      **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_EFUNCAV (-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\_reuslt\_要求元ノード`\_要求プロセス ID\_要求スロット` ID\_日時\_配信先ノード`名.std

stderr :

diosa\_cdd\_reuslt\_要求元ノード`\_要求プロセス ID\_要求スロット` ID\_日時\_配信先ノード`名.err

(※日時は YYYYMMDDHHMMSSsssss)

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. 59     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
char Typeout[9]	タイマエントリ時間形式が返却される。 CLOCK           時刻指定 INTERVAL        インターバル指定または即時形式
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

## 2.1.60 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



## 2.1.61 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								
	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;	<p>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)</p> <p>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。</p>								
int ExecTimeOut;	<p>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)</p> <p>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。</p>								
int RtryCnt;	<p>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)</p> <p>0 を指定した場合、リトライしない。</p> <p>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。</p>								
int RtryIntvl;	<p>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)</p> <p>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。</p>								
char InFileOption[3];	<p>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)</p> <p>入力ファイルのオプションが不要な場合、NULL を設定する。</p>								
char InFileDelMode;	<p>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。</p> <table> <tr> <td>DIOSA_CMDSND_DELMODE_NO</td><td>配信元、配信先の入力ファイルを削除しない</td></tr> <tr> <td>DIOSA_CMDSND_DELMODE_ORG</td><td>配信元の入力ファイルを削除する</td></tr> <tr> <td>DIOSA_CMDSND_DELMODE_DST</td><td>配信先の入力ファイルを削除する</td></tr> <tr> <td>DIOSA_CMDSND_DELMODE_BOTH</td><td>配信元、配信先の入力ファイルを削除する</td></tr> </table>	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
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];	<p>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)</p> <p>入力ファイルの転送が不要な場合、NULL を設定する。</p>								

## 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. 62 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

## 2.1.63 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. 64 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
char Typeout[9]	タイマエントリ時間形式が返却される。 CLOCK           時刻指定 INTERVAL       インターバル指定または即時形式
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. 65     t\_diosa\_cosetuca (C0 制御タイマ登録用構造体)

### 名前

t\_diosa\_cosetuca - C0 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、C0 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 C0 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. 66 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    diosadcuca 領域名;
```

説明

電文送受信時に CO 制御と AP 間の情報受け渡しをおこなうの領域(t\_diosa\_dcuca 領域)である。

```
short    TextChar;        電文特性

                                DIOSA_MSG_LS           : 論理システム間
                                DIOSA_MSG_DERIV        : システム内派生
                                DIOSA_MSG_TPDERIV      : TPP 間派生
                                DIOSA_MSG_CHAIN        : 連鎖
                                DIOSA_MSG_REST         : 保留

t_diosa_addrinfo  SafInfo;   送信元情報
t_diosa_addrinfo  DafInfo;   送信先情報

char    TermName[31+1];    端末名 (最大 31 文字)
char    VdName[24+1];     VD 名 (最大 24 文字)
char    CallId[8+1];      TP_CALL 識別子 (最大 8 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    RecvId; 受信識別子

                                DIOSA_RECVFROM_TERM    (TERMNAME)
                                DIOSA_RECVFROM_TPP      (VDNAME)
                                DIOSA_RECVFROM_TPCALL   (CALLID)

short    DerivNum;        派生回数
char     SimFlag;         シミュレータフラグ : DIOSA_ON / DIOSA_OFF

(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    PreBlkFlag;      論理ノード予閉塞フラグ

                                DIOSA_YES  : 予閉塞中
                                DIOSA_NO   : 閉塞なし

t_diosa_tpstatus  TpStatus;  TP_send リターン情報
```

アドレス情報

```
typedef struct t_diosa_addrinfo {    送信元情報
    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）
char    TpMonitor[8+1];    TPBASE モニタ名（最大 8 文字）
t_diosa_keyinfo  KeyInfo;    キー情報
} t_diosa_addrinfo;
```

キー情報については、t\_diosa\_uca の項を参照。

#### TPBASE システムからの状態通知

```
typedef 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
} t_diosa_tpstatus;
```

StatusKey[2]およびEndKey は、TPBASE の CS\_STRUCT の status\_key[2]および end\_key に相当する。



## 2.1.67 t\_diosa\_keyinfo(キー情報構造体)

### 名前

t\_diosa\_keyinfo - キー情報

### 書式

```
#include <diosa.h>

t_diosa_keyinfo キー名;
```

### 説明

char	Flag;	キー情報フラグ
		DIOSA_KEY_UNAVAIL(0) : キー情報が無効
		DIOSA_KEY_MAP(1) : MAP 指定
		DIOSA_KEY_MAINKEY(2) : メインキー指定
char	rfu[3];	
int	MAP;	MAP
char	MainKey[32];	メインキー(32 バイト)

Flag が DIOSA\_KEY\_MAINKEY の場合は、MainKey が有効であり、DIOSA\_KEY\_MAP の場合は、MAP が有効となる。  
Flag が DIOSA\_KEY\_UNAVAIL の場合は、MainKey と MAP は両方とも無効となる。

## 2.1.68 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

diosamsgbuffalloc 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

diosamsgbuffalloc, diosamsgbufffree, diosasendtx, t\_diosa\_analyze

## 2. 1. 69     t\_diosa\_sgarea (SG レコード情報格納バッファ構造体)

### 名前

t\_diosa\_sgarea - SG レコード情報格納バッファ構造体

### 書式

```
#include <diosa.h>
t_diosa_sgarea Area;
```

### 説明

t\_diosa\_sgarea 構造体は、SG オブジェクトファイルから取得するレコード情報を格納するバッファを指定する。**diosasgget()**によって正常にレコードを取得した場合、取得したレコード情報は t\_diosa\_sgarea で指定したバッファに格納される。

char *cArea;	レコード情報を格納するバッファの先頭アドレス
int nSize;	レコード情報を格納するバッファのサイズ

### 関連

diosasgget

## 2.1.70 t\_diosa\_sgctrl (SG オブジェクトファイル制御情報構造体)

### 名前

t\_diosa\_sgctrl - SG オブジェクトファイル制御情報構造体

### 書式

```
#include <diosa.h>
t_diosa_sgctrl Ctrl;
```

### 説明

t\_diosa\_sgctrl 構造体は、SG オブジェクトファイルの制御情報が格納される。利用者は **diosasgopen()** で返却された制御情報を **diosasgget()**、**diosasgclose()** に指定する必要がある。

```
char nSgCtrl[496];          SG オブジェクトファイルの制御情報
```

### 関連

diosasgopen, diosasgget, diosasgclose

## 2. 1. 71 t\_diosa\_sginfo (SG オブジェクトファイル情報構造体)

### 名前

t\_diosa\_sginfo - SG オブジェクトファイル情報構造体

### 書式

```
#include <diosa.h>
t_diosa_sginfo Info;
```

### 説明

t\_diosa\_sginfo 構造体は、SG オブジェクトファイル情報が格納される。

int nMaxSize;	<b>diosasgopen()</b> でオープンした SG オブジェクトファイルの 最大レコード長(バイト)
int nVolume;	<b>diosasgopen()</b> でオープンした SG オブジェクトファイルの 総データサイズ(バイト)
int nRecnum;	<b>diosasgopen()</b> でオープンした SG オブジェクトファイルの総レコード数
int nRecsize;	<b>diosasgget()</b> で取り出したレコードサイズ(バイト)
int nBasesys;	環境定義機能の内部情報(利用者使用不可)
char cSrcrev[4];	環境定義機能の内部情報(利用者使用不可)
char cObjrev[4];	<b>diosasgopen()</b> でオープンした SG オブジェクトファイルのリビジョン リビジョン情報は、環境定義生成コマンドによる SG オブジェクトファイルの 更新回数を表す。
char cRectype[3];	<b>diosasgget()</b> で取り出したレコードの種別

### 関連

diosasgopen, diosasgget

## 2.1.72 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。[1-31]の範囲で設定する。
short Month	月を示す。[1-12]の範囲で設定する。
short Year	年を示す。[1990-2089]の範囲で設定する。

### 注意

タイマ制御インタフェース構造体において、Mode 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。

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。  
各項目の設定を省略することはできない。

### 関連

t\_diosa\_tmcuca, diosatmccoset, diosatmccmdset, diosatmccoquery, diosatmccmdquery

## 2.1.73 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_CLOCK	時刻指定
	DIOSA_IMMEDIATE	即時指定
int Count	通知回数を示す。 <b>Mode</b> が DIOSA_CLOCK (時刻指定) の場合は指定できない。	
	DIOSA_NOLIM	無限回の通知
t_diosa_tmctime Time	タイマ値を示す。	
	<b>Mode</b> が DIOSA_INTER、DIOSA_IMMEDIATE の場合は、タイマ実行間隔を設定する。	
	<b>Mode</b> が DIOSA_CLOCK の場合は、タイマ通知日時を設定する。	
	<b>t_diosa_tmctime</b> 構造体については、 <b>t_diosa_tmctime</b> の項を参照。	
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	—	—	—	—	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)を指定した場合、通知回数を指定することはできない。  
Mode(時刻形式)に時刻指定(DIOSA\_CLOCK)を指定し、タイマ値に過去の時刻を設定又は、存在しない時刻を設定した場合、DIOSA\_ETIMEFORM となる。  
各パラメータの指定は必須である。

## 関連

diosatmccoset, diosatmccmdset, diosatmcreset, diosatmccoquery,  
diosatmccmdquery, diosatmchold, diosatmcactv, t\_diosa\_tmctime



## 2.1.74 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. 75 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
                                DIOSA_EXIT_COMMIT         : コミット出口
                                DIOSA_EXIT_ROLLBACK       : ロールバック出口

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;     論理ノードタイプ

                                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)
t_diosa_keyinfo  KeyInfo; トランザクション開始時のキー情報
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     DerivNum;      トランザクションの派生回数
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 (C0・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)

プロセス初期化出口内で diosaapptrset により通知された AP 共通領域が、他の利用者出口・C0 において 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)

各利用者出口・C0 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・C0 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。

表中のA～Dは、後述の状態コードの表の同記号と対応する。

ー：対象外 ○：対象 A～D：利用者コード (UserExitKey)参照可  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	コミット	ロール バック	ア ボ ー ト # 1	ア ボ ー ト # 2
正常 DIOSA_EK_NORMAL	○	○	○	○	○	○	○	○	—	—
AP からの異常終了要求 DIOSA_EK_APREQ	—	—	—	—	—	—	A	A	A	A
リトライオーバー DIOSA_EK_RETRYOV	—	—	—	—	—	—	B D	B D	B D	B D
エラーCO 呼び出しエラー DIOSA_EK_ERRCO	—	—	—	—	—	—	○	○	○	○
継続不可能な障害発生 DIOSA_EK_COCREQ	—	—	—	—	—	—	○	○	○	○
例外発生 (シグナル発生) DIOSA_EK_EXCP	—	—	—	—	—	—	—	○	○	—
CPU 時間超過 DIOSA_EK_CPUOV	—	—	—	—	—	—	—	○	○	—
CO 閉塞 DIOSA_EK_COBLOCK	—	—	—	—	—	○	—	—	—	—
CO 呼び出しエラー DIOSA_EK_CALLERR	—	—	—	—	—	○	—	—	—	—
CO が決定できない DIOSA_EK_ECODEC	—	—	—	—	—	○	—	—	—	—
ロールバックリトライ DIOSA_EK_ROLLBACK	—	—	—	B	—	—	—	B C	—	—
デッドロックリトライ DIOSA_EK_DEADLOCK	—	—	—	D	—	—	—	D	—	—

状態コード<C0 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。

表中のA～Dは、前述のトランザクション処理結果の表の同記号と対応する。

○：指定可 ×：指定不可 －：指定は無効 A～D： 利用者コード (UserStatus)指定可  状態コード	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	〇	コミット(※2)	ロールバック(※2)	アボート#1	アボート#2
正常終了 DIOSA_ST_DONE	○	×	○	○	○	○	○	○	×	○
異常終了 DIOSA_ST_ABORT	○	×	○	○	A	A	A	A	×	○
オプション付き異常終了 DIOSA_ST_ABORTCONT：継続 DIOSA_ST_ABORTSTOP：終了	○	×	○	○	A	A	A	A	×	－
ロールバックリトライ DIOSA_ST_ROLLBACK	×	×	×	×	×	B	×	×	×	×
デッドロックリトライ DIOSA_ST_DEADLOCK	×	×	×	×	D	D	×	×	×	×
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 diosaccommit, diosarollback から呼び出されるコミット、ロールバック出口から diosagoback を実行した場合、その時に設定されている状態コードは C0 から返却された状態コードとして扱われる。diosagoback を実行しない場合は上記表指定可否に従う。

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。

また、正常以外の状態コードは最新の状態コード、利用者コードで上書きされるため、上表は状態コード、利用者コードを照会できる目安である。C0 が DIOSA\_ST\_ABORTSONT (UserStatus=1) を要求して異常終了処理のアボート#2 出口が DIOSA\_ST\_ABORT (UserStatus=2) で再び異常終了要求した場合は、その後呼び出されるロールバック出口 (出口定義要) には (UserStatus=2) が渡される。

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 C O を再度呼び出す) C O からのリターン時のみ有効

利用者出口との対応

C0 および各利用者出口ごと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	コミット出口	ロールバック出口	プロセス終了出口	初期化出口	CO	正常終了	アボート出口	コミット出口	ロールバック出口	
UcaLen		I									I							
UcaRev		I									I							
ExitId		I									I							
LsName		I									I							
LNodeName		I									I							
LNodeType		I									I							
DiosaMode		I									I							
TpMonitor		I									—							
ClassName		I									—							
TpsUcaPtr		I									—							
TxId		—		I						—		—						
CoName		—		I						—		—	I					
KeyInfo	Flag	—		I						—		0	I					
	MainKey	—		I						—		0	I					
	MAP	—		I						—		0	I					
Exitkey		—		I						—		—		I				
UserExitKey		—				I					—		—					
RetryLim		—		I						—		—	I					
RetryCount		—		I						—		—	I					
ChainNum		—		I						—		—						
CommitNum		—		I						—		—						
DerivNum		—	I							—		—	I					
MsgSizeMax		I									—							
ApComarea		—	I							—								
ApAreaPtr		0	I/O							I		—						
BatchApInfo		—									I/O							
Status		0				—		0		—		0	I/O		I	I/O		
UserStatus		—		0		—		0		—		0	I/O		I	I/O		

## 2.2 通信制御

### 2.2.1 関数一覧

#### (1) ノード間通信パス管理機能

diosadbmulticonnect	活性と判断されたデータベース・インスタンスに接続する。
diosadbconnect	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。
diosabddisconnect	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。
diosadbchangeconnect	指定データベース・インスタンスにコンテキストを切り替える。
diosagetdbctx	カレントのデータベースコンテキストを返却する。
diosadbfaultnotification	DB 障害検出時に障害通知を行う。
t_diosa_dbconnectuca	データベース接続関数で使用する構造体。
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。



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

### 名前

`diosadbchangeconnect` - 指定データベース・インスタンスにコンテキストを切り替える。

### 書式

```
#include <diosa.h>

int diosadbchangeconnect(t_diosa_dbuca *DbUca, long *Dstatus);
```

### 説明

リソースグループ ID、あるいはリソースグループセット名を指定し、対応するデータベース・インスタンスにコンテキストを切り替える。

**t\_diosa\_dbuca \*DbConnectUca** (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_ENOENT (-8)	カレントデータベースが決定していない
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)	DB ヘルスチェック機能非作動のため判定不能
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。

指定されたデータベース・インスタンスに未接続である場合、本関数内でデータベース・インスタンスに接続する。この際、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。本関数を実行する前に、コミットもしくはロールバックを実行して、処理を完了させる必要がある。

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_dbconnectuca *p_DbConnectUca, long *Dstatus);
```

### 説明

リソースグループ ID、もしくはリソースグループセット名を指定し、対応するデータベース・インスタンスにのみ接続する。

**t\_diosa\_dbconnectuca \*p\_DbConnectUca**(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
<b>Dstatus</b> に詳細ステータスが返却される (DB 接続出口の呼び出しエラー詳細)	
-1	ノード種別エラー、または処理継続不可のエラー
-3	パラメータエラー
-8	接続出口関数が見つからない
-10	範囲不正
-34	初期化処理が行われていない
-78	環境変数が見つからない
-82	指定されたリビジョンの関数は存在しない
その他	Oracle の sqlcode が返却される
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)	全ノード接続失敗
DIOSA_EFUNCNAV(-34)	機能利用不可
DIOSA_EALREADY(-35)	二重実行エラー
DIOSA_EATTACH(-43)	初期化処理エラー

### 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。  
本関数を呼び出した際は、次の接続を行う前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必要がある。

### 関連

t\_diosa\_dbconnectuca, diosadbdconnect

## 2. 2. 4      diosabdbdisconnect (DB 切断関数)

### 名前

diosabdbdisconnect - 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 diosabdbdisconnect(long *Dstatus);
```

### 説明

現在接続されているデータベース・インスタンスの内、デフォルト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。

**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, t\_diosa\_dbconnectuca

## 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(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・インスタスとの接続を切断する。

#### char \*DbStatus (出力)

データベース・インスタスの状態を返却する。

DIOSA\_DBSTATUS\_DBACTIVE (0x00)           : データベース・インスタス使用可能  
DIOSA\_DBSTATUS\_DBDOWN (0x01)           : 全データベース・インスタス障害  
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_ENOENT(-8)	カレントデータベースが決定していない
DIOSA_ENOINIT(-11)	未初期化エラー
DIOSA_EFUNCNAV(-34)	機能利用不可

## 2. 2. 6 diosadbmultipconnect (DB 接続関数[マルチコネクション])

### 名前

`diosadbmultipconnect` - 活性と判断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 diosadbmultipconnect(long *Dstatus)
```

### 説明

デフォルト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に対して、データベースヘルスチェックの結果を確認し、活性と判断される全データベース・インスタンスに接続する。

**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
<b>Dstatus</b> に詳細ステータスが返却される (DB 接続出口の呼び出しエラー詳細)	
-1	ノード種別エラー、または処理継続不可のエラー
-3	パラメータエラー
-8	接続出口関数が見つからない
-10	範囲不正
-34	初期化処理が行われていない
-78	環境変数が見つからない
-82	指定されたリビジョンの関数は存在しない
その他	Oracle の <code>sqlcode</code> が返却される
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)	全ノード接続失敗
DIOSA_EFUNCNAV(-34)	機能利用不可
DIOSA_EALREADY(-35)	二重実行エラー
DIOSA_EATTACH(-43)	初期化処理エラー

### 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。  
本関数を呼び出した際は、次の接続を行う前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必要がある。

### 関連

`diosadbreconnect`, `diosadbchangeconnect`, `diosadbdisconnect`, `t_diosa_dbconnectuca`

## 2. 2. 7 diosadbconnect (DB 接続関数[再接続])

### 名前

diosadbconnect - 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 diosadbconnect(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)	DB ヘルスチェック機能非作動のため判定不能

### 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)

データベース接続先切り替え関数(diosadbchangeconnect)の戻り値により、必要に応じて本関数を呼び出すこと。

### 関連

diosadbchangeconnect

## 2.2.8 diosagetdbctx (DB コンテキスト取得関数)

### 名前

diosagetdbctx - カレントのデータベースコンテキストを返却する。

### 書式

```
#include <diosa.h>

int diosagetdbctx(sql_context *SqlCtx);
```

### 説明

カレントのデータベースコンテキストを返却する。

**sql\_context \*SqlCtx**(出力)

カレントのデータベースコンテキスト。

### 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。

### 関連

diosadbchangeconnect

## 2.2.9 t\_diosa\_dbconnectuca (DB 接続関数用構造体)

### 名前

t\_diosa\_dbconnectuca - データベース接続関数で使用する構造体。

### 書式

```
#include <diosa.h>

t_diosa_dbconnectuca DbConnectUca
```

### 説明

データベース接続関数で使用する構造体。

#### char InputFlag(入力)

入力情報の種別を指定する。

DIOSA_DBCONNECT_RGID (0x00)	リソースグループ ID 指定
DIOSA_DBCONNECT_RGSET (0x01)	リソースグループセット指定
DIOSA_DBCONNECT_DEFAULTRGSET (0x02)	デフォルト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)

### 関連

diosadbconnect, diosadbmultipconnect, diosadbdisconnect



## 2. 2. 10 t\_diosa\_dbuca (DB 接続先切り替え関数用構造体)

### 名前

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。

### 書式

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

### 説明

データベース接続先切り替え関数で使用する構造体。

#### char InputFlag(入力)

入力情報の種別を指定する。

DIOSA_DBCONNECT_RGID (0x00)	リソースグループ ID 指定
DIOSA_DBCONNECT_RGSET (0x01)	リソースグループセット指定
DIOSA_DBCONNECT_DEFAULTRGSET (0x02)	デフォルト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

#### char ReconnectFlag(出力)

データベース・インスタンスと再接続の必要があるかどうかを返却する。

再接続の必要がある場合は、トランザクション終了後にデータベース接続関数(diosadbreconnect)を実行する。

DIOSA_DBNORECONNECT (0x00)	再接続の必要なし
DIOSA_DBRECONNECT (0x01)	再接続の必要あり

#### long ReconfigCnt(出力)

要求世代管理番号を返却する。

### 関連

diosadbchangeconnect, diosadbreconnect

## 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、`diosatrnrinit()` 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 `diosaimreadl()` である。

#### 接続処理

<code>diosaimopen</code>	インメモリサーバとの接続を確立する。
<code>diosaimclose</code>	インメモリサーバとの接続を切断する。

#### アクセス要求

<code>diosaimreadl</code>	1 レコードを取得する(主キーの完全一致)。
<code>diosaimread</code>	複数レコードを取得する。
<code>diosaimwrite</code>	レコードを挿入する。
<code>diosaimrewrite</code>	レコードを更新する。
<code>diosaimdelete</code>	レコードを削除する(レコード情報で指定)。
<code>diosaimdeletexl</code>	レコードを削除する(主キーの完全一致)。
<code>diosaimtruncate</code>	全レコードを削除する。

#### トランザクション制御

<code>diosaimtxstart</code>	トランザクションの開始を宣言する。
<code>diosaimcommit</code>	更新要求を確定する。
<code>diosaimrollback</code>	更新要求をロールバックする。

#### アクセス先 MAP

<code>diosaimsetmap</code>	アクセス先の MAP を設定する。
<code>diosaimgetmap</code>	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。
diosaimgetreckeyinfo	レコードのキー情報を照会する。
diosaimgetptblname	物理表名を照会する。
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。

### (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
diosasetmap	MAP 情報更新関数
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_setmapent	MAP 情報更新関数インタフェース (MAP エントリ) 構造体
t_diosa_setmapuca	MAP 情報更新関数インタフェース構造体
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

C0 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 の状態と利用者領域情報を **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)	<b>GetMapUca</b> 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, diosasetmap, 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

C0 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。同ー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, diosaprcrterm, 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。

<b>RepGrpId</b>	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。
<b>MapNum</b>	MAP 数が返却される。
<b>MapEnt</b>	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

C0 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。同ー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

C0 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, diosaprcrterm, 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

C0 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。同ー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**, **diosaprcrterm**, **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()
```

### 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_MSGQ(-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_ETIME(-114)	<b>diosaprcinit()</b> 、または <b>diosathrinit()</b> 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()
```

### 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()** 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_MSGQ(-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)	TAM の障害により更新確定が失敗した。
DIOSA_ESTATE(-114)	<b>diosaimtxstart()</b> 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)	<b>diosaprcinit()</b> 、 <b>diosathrinit()</b> (マルチスレッドの場合)、 <b>diosaimopen()</b> 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)	<b>CtxId</b> で指定された検索コンテキストが <b>diosaimctxopen()</b> 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 <b>Cond</b> が <b>diosaimcondsetkey()</b> 、 <b>diosaimcondsetrange()</b> のいずれかで構築されていない。もしくは、 <b>diosaimsetmap()</b> 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_MSGQ(-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 <b>diosaimtruncate()</b> 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)	TAM の障害によりレコード削除に失敗した。
DIOSA_ESTATE(-114)	<b>diosaimtxstart()</b> 、または <b>diosaimsetmap()</b> 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

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。

#### 関連

diosaimsetmap(),   diosaimgettblid(),   diosaimreadl(),   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

<b>DIOSA_DONE(0)</b>	正常終了した。
<b>DIOSA_BLOCK(6)</b>	該当 MAP が閉塞している。
<b>DIOSA_NOENT(20)</b>	該当レコードが削除対象の表に存在しなかった。
<b>DIOSA_SWITCH(21)</b>	計画マスタ切替中である。
<b>DIOSA_ERROR(-1)</b>	その他エラーが発生した。
<b>DIOSA_ESYS(-2)</b>	システムコールエラーが発生した。
<b>DIOSA_EPARAM(-3)</b>	パラメータが不正である。
<b>DIOSA_EMEM(-6)</b>	メモリ操作に失敗した。
<b>DIOSA_EINVAL(-9)</b>	削除対象の表が存在しない。
<b>DIOSA_ELOCK(-14)</b>	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。
<b>DIOSA_ETIMEOUT(-22)</b>	要求がタイムアウトした。
<b>DIOSA_EACCES(-25)</b>	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。
<b>DIOSA_EBUSY(-31)</b>	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。 <b>Lock</b> に <b>DIOSA_LOCK_NOWAIT</b> が指定されていた場合にのみ発生する。
<b>DIOSA_EDEADLOCK(-36)</b>	デッドロックが発生した。 <b>Lock</b> に <b>DIOSA_LOCK_WAIT</b> を指定した場合のみ発生する。
<b>DIOSA_EOVERFLOW(-39)</b>	IMS キューバッファに空きがない。
<b>DIOSA_MSGQ(-41)</b>	メッセージキューの障害が発生した。
<b>DIOSA_ECOND(-60)</b>	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

クシヨン内で **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) TAM 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

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。

## 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)	<b>diosaimopen()</b> 、または <b>diosatrnninit()</b> 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\_DATA LIM** 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_DATA LIM(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)	<b>diosaprcinit()</b> 、または <b>diosathrinit()</b> が未実施である。

### 関連

t\_diosa\_imrefctx

## 2.3.18 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)	<b>diosaprcinit()</b> 、または <b>diosathrinit()</b> が未実施である。

## 2.3.19 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、必ずプライマリー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\_DATA LIM** が返却される。

### 戻り値

DIOSA_DONE(0)	キー情報を取得できた。
DIOSA_DATA LIM(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)	<b>diosaprcinit()</b> 、または <b>diosathrinit()</b> が未実施である。

### 関連

t\_diosa\_imreckeyinfo, t\_diosa\_imrefctx

## 2.3.20 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)	<b>diosaprcinit()</b> 、または <b>diosathrinit()</b> が未実施である。

### 関連

diosaimread1(), diosaimread(), diosaimwrite(), diosaimrewrite(), diosaimdelete(),  
diosaimdeletex1(), diosaimtruncate(), diosaimctxopen()



## 2.3.21 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\_DATA LIM** が返却される。

### 戻り値

**DIOSA\_DONE**(0)                論理表を取得できた。

**DIOSA\_DATA LIM**(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

## 2.3.22 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_EMMSGQ(-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)	<b>diosaprcinit()</b> 、または <b>diosathrinit()</b> 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. 23      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

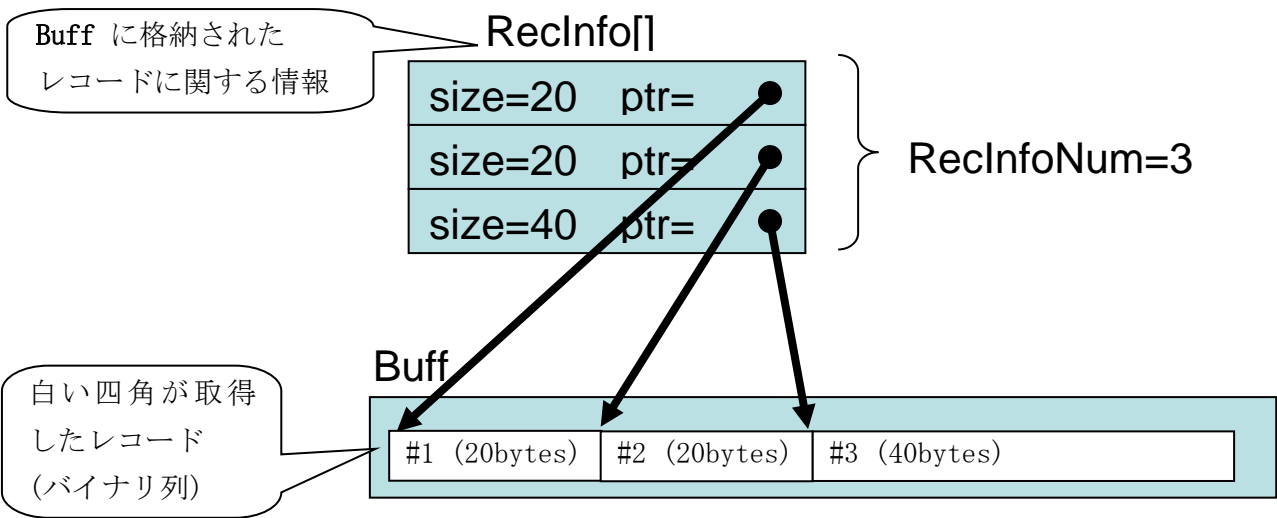
```
#include <diosa.h>

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。



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)	<b>Buff</b> 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_MSGQ(-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)	TAM 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 <b>CtxId</b> は <b>diosaimctxopen()</b> 未実施である。もしくは、 <b>diosaimsetmap()</b> 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。

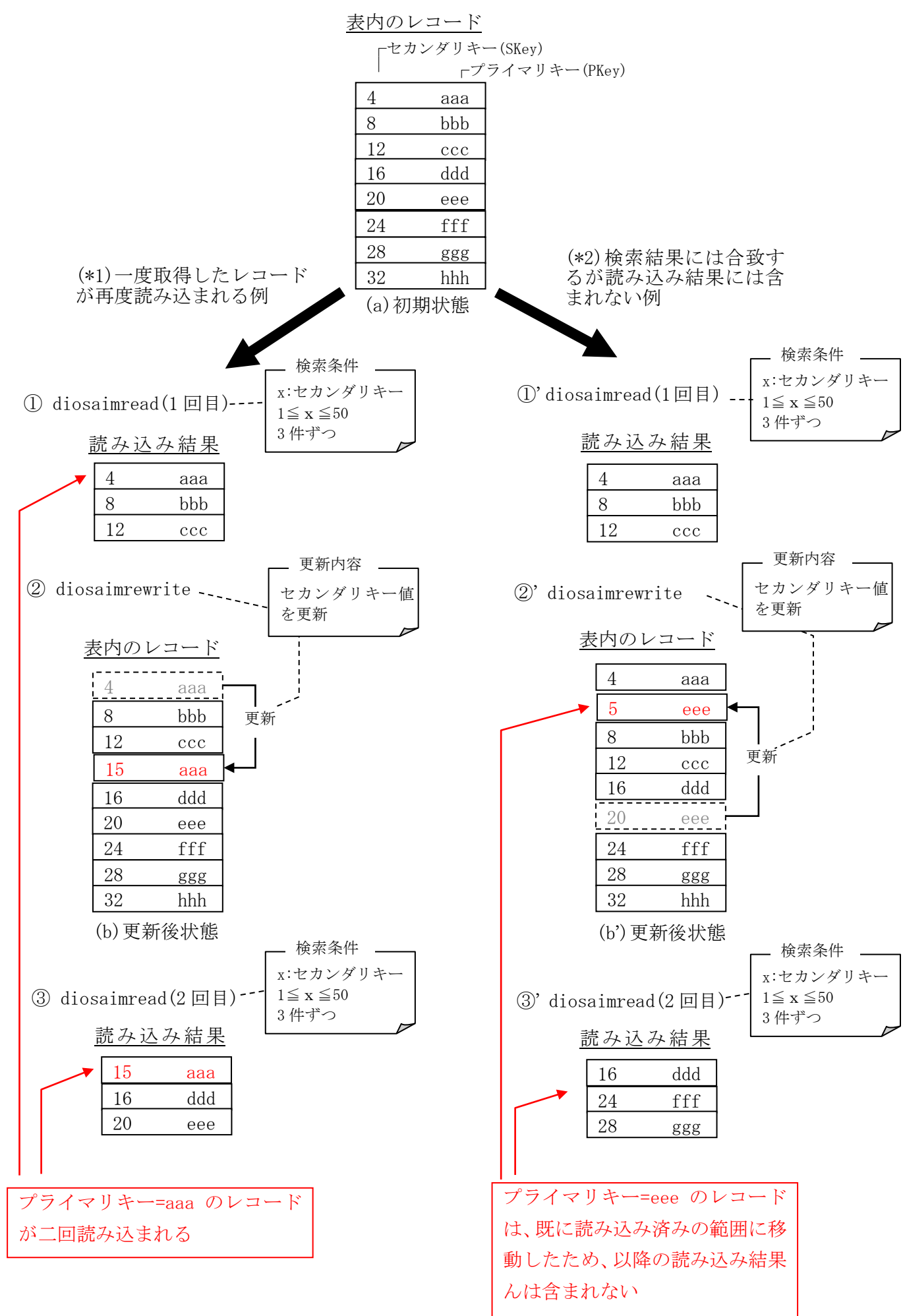
## 注意

**diosaimtxstart()** を実行済みの状態で、異常終了(負値)が返却された時は、必ず **diosaimrollback()** を実行すること。

**DIOSA\_SWITCH** など一部の戻り値が返却された場合は、**diosatrnterm()** 実行後、**diosatrnnit()** から実行する必要がある。本ケースに該当する戻り値については、メモリキャッシュ利用の手引きの「3.1.7 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 件ごとの読み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 バイトのキーに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);
```

## 2.3.24 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ー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

<b>DIOSA_DONE(0)</b>	正常終了した。
<b>DIOSA_BLOCK(6)</b>	該当 MAP が閉塞している。
<b>DIOSA_NOENT(20)</b>	該当レコードが読込対象の表に存在しなかった。
<b>DIOSA_SWITCH(21)</b>	計画マスタ切替中である。
<b>DIOSA_ERROR(-1)</b>	その他エラーが発生した。
<b>DIOSA_ESYS(-2)</b>	システムコールエラーが発生した。
<b>DIOSA_EPARAM(-3)</b>	パラメータが不正である。
<b>DIOSA_ENOBUFS(-7)</b>	<b>Buff</b> に指定された領域に取得したレコードが入らない。
<b>DIOSA_EINVAL(-9)</b>	読込対象の表が存在しない。
<b>DIOSA_ELOCK(-14)</b>	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。
<b>DIOSA_ETIMEOUT(-22)</b>	要求がタイムアウトした。
<b>DIOSA_EACCES(-25)</b>	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。
<b>DIOSA_EBUSY(-31)</b>	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。 <b>Lock</b> 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\_MSGQ**(-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) TAM 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。

**diosaimtxstart()** を実行済みの状態で、異常終了(負値)が返却された時は、必ず **diosaimrollback()** を実行すること。

**DIOSA\_SWITCH** など一部の戻り値が返却された場合は、**diosatrnterm()** 実行後、**diosatrninit()** から実行する必要がある。本ケースに該当する戻り値については、メモリキャッシュ利用の手引きの「3.1.7 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**

## 2.3.25 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_MSGQ(-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 <b>diosaimtruncate()</b> 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)	TAM の障害によりレコード更新に失敗した。
DIOSA_ESTATE(-114)	<b>diosaimtxstart()</b> 、または <b>diosaimsetmap()</b> 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

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。

#### 関連

diosaimsetmap(),   diosaimtxstart(),   diosaimgettblid(),   diosaimread1(),   t\_diosa\_recinfo

## 2.3.26 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()
```

### 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_MSGQ(-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)	TAM の障害によりロールバックが失敗した。
DIOSA_ETIME(-114)	<b>diosatrninit()</b> 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()** から実行する必要がある。

### 関連

**diosaimtxstart()**, **diosaimcommit()**

## 2.3.27 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、**diosatrnnit()** 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_MSGQ(-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)	<b>MapId</b> に指定されたサーバに接続できない。
DIOSA_ETIME(-114)	<b>diosaimopen()</b> 、または <b>diosatrnnit()</b> 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、**diosatrnnit()** から実行する必要がある。

### 関連

**diosatrnnit()**, **diosaimgetmap()**, **diosaimopen()**

## 2.3.28 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_MSGQ(-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 <b>diosaimread1(DIOSA_LOCK_WAIT, DIOSA_LOCK_NOWAIT)</b> のいずれかが指定)、 <b>diosaimwrite()</b> 、 <b>diosaimrewrite()</b> 、 <b>diosaimdelete()</b> 、 <b>diosaimdeletex1()</b> 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)	TAM の障害によりレコード削除に失敗した。
DIOSA_ETIMEOUT(-114)	<b>diosaimtxstart()</b> 、または <b>diosaimsetmap()</b> 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_EREDY(-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は出来ない。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()`

## 2.3.29 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()
```

### 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)	<b>diosaimopen()</b> 、または <b>diosatrnnit()</b> が未実施である。

### 関連

**diosaimcommit()**, **diosaimrollback()**



## 2.3.30 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。

<b>DIOSA_LOCK_WAIT</b>	他トランザクションで同じプライマリー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、 <b>DIOSA_ETIMEOUT</b> が返却される。
<b>DIOSA_LOCK_NOWAIT</b>	他トランザクションで同じプライマリー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)	同じプライマリーキーのレコードが、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。 <b>Lock</b> に <b>DIOSA_LOCK_NOWAIT</b> 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。 <b>Lock</b> に <b>DIOSA_LOCK_WAIT</b> 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_MSGQ(-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 <b>diosaimtruncate()</b> 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)	TAM の障害によりレコード追加に失敗した。
DIOSA_ESTATE(-114)	<b>diosaimtxstart()</b> 、または <b>diosaimsetmap()</b> 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

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。

#### 関連

**diosaimsetmap()**, **diosaimtxstart()**, **diosaimgettblid()**, **t\_diosa\_recinfo**

## 2. 3. 31 diosasetmap (MAP 情報更新関数)

### 名前

diosasetmap - MAP 情報更新関数

### 書式

```
#include <diosa.h>

int diosasetmap( t_diosa_setmapuca *SetMapUca, t_diosa_setmapent *SetMapEnt );
```

### 説明

**diosasetmap()** は、**SetMapUca** で指定されたレプリケーショングループを検索し、そのレプリケーショングループ内の **SetMapEnt** で指定された MAP の利用者領域情報を **SetMapEnt** で指定された内容に更新する。**SetMapEnt** は複数 MAP を指定可能である。また、**SetMapUca** の ChgHashFlag に DIOSA\_YES を指定された場合、**SetMapEnt** で指定された MAP のハッシュ値切替も行う。

**SetMapUca**                    MAP 情報更新関数インタフェース構造体のポインタを指定する。

**SetMapEnt**                    MAP 情報更新関数用 MAP 情報エントリ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\_ENOENT(-8)             指定された MAP 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。

DIOSA\_ETIMEOUT(-22)         一定時間内に応答がない。

DIOSA\_EBUSY(-31)             他要求によるテーブル更新処理中で対応できない。

DIOSA\_ESIZE(-33)             **SetMapUca** の UserDataLen に指定されたサイズが利用者領域データ長と異なる。

DIOSA\_EFUNCNAV(-34)         マスタが存在しないノードで実行された。

### 注意

C0 制御、バッチ AP 制御以外の環境で diosasetmap() 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。

diosasetmap() は、指定レプリケーショングループのマスタが存在するノードのみ実行可能である。

本関数は、事業部 PP 専用の API であり、業務 AP では使用不可である。

### 関連

diosagetmap, diosaprcinit, diosaprcrterm, diosathrinit, diosathrterm, t\_diosa\_setmapent, t\_diosa\_setmapuca

## 2. 3. 32 diosatamswitch(マスタ昇格関数)

### 名前

diosatamswitch - TAM 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 TAM をマスタに昇格する。本関数は、昇格させるスレーブの TAM 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 TAM をマスタに切り替える場合に、対象の TAM 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 TAM 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\_ERECD(-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

C0 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

## 2.3.33 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 TAM 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

## 2.3.34 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;	UserArea で指定した利用者領域のサイズを指定する。
unsigned int UserDataLen;	利用者領域データ長が返却される。
void *UserArea;	利用者領域データを格納する領域のポインタを指定する。 利用者領域情報が返却される。

### 注意

UserArea の領域は、利用者側で確保し、確保した領域長を UserAreaLen に指定する。  
利用者領域情報が不要な場合は、UserAreaLen=0、\*UserArea=NULL を指定する。

### 関連

diosagetmap

## 2. 3. 35 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()

## 2.3.36 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

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()



## 2.3.37 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

## 2.3.38 t\_diosa\_imtblist(論理表一覧構造体)

### 名前

t\_diosa\_imtblist - 論理表一覧の構造体

### 書式

```
#include <diosa.h>

t_diosa_imtblist tblist;
```

### 説明

**t\_diosa\_imtblist** 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

diosaimgettblist()

## 2. 3. 39 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 TAM 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

## 2.3.40 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

## 2.3.41 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()

## 2.3.42 t\_diosa\_setmapent (MAP 情報更新関数用 MAP 情報エントリ構造体)

### 名前

t\_diosa\_setmapent - MAP 情報更新関数用 MAP 情報エントリ構造体

### 書式

```
#include <diosa.h>

t_diosa_setmapent SetMapEnt;
```

### 説明

t\_diosa\_setmapent 構造体は、ヘッダ<diosa.h>で定義されていて、その中に MAP 状態更新関数で更新する MAP に関連する情報が格納される。

int MapId;	MAPID を指定する。
void *UserArea;	利用者領域データを格納する領域を指定する。 更新後の利用者領域情報を指定する。 利用者領域の更新が不要な場合 NULL を指定する。

### 注意

UserArea の領域は、利用者領域データ長と同じ大きさで利用者側が確保し、確保した領域長を SetMapUca の UserDataLen に指定する。  
ハッシュ値切替を行うが、利用者領域の更新が必要な MAP は、エントリとして指定を行い、\*UserArea=NULL を指定する。

### 関連

diosasetmap, t\_diosa\_setmapuca

## 2. 3. 43 t\_diosa\_setmapuca (MAP 情報更新関数インタフェース構造体)

### 名前

t\_diosa\_setmapuca - MAP 情報更新関数インタフェース構造体

### 書式

```
#include <diosa.h>

t_diosa_setmapuca SetMapUca;
```

### 説明

t\_diosa\_setmapuca 構造体は、ヘッダ<diosa.h>で定義されていて、その中に MAP 状態更新関数のインタフェース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 を指定する。
unsigned int UserDataLen;	利用者領域のサイズを指定する。
char RspFlag;	全ノードに更新情報が反映されたことを待ち合わせるかを指定する。 DIOSA_YES        待ち合わせをする DIOSA_NO        待ち合わせをしない
char ChgHashFlag;	SetMapEnt で指定された MAP のハッシュ値切替を行うかを指定する。 DIOSA_YES        ハッシュ値切替を行う DIOSA_NO        ハッシュ値切替を行わない
int Timeout;	応答待ち時間を秒単位で指定する。(0～3600) 0 を指定した場合、無限待ちとする。
int SetMapEntNum;	更新対象 MAP のエントリ数を指定する。

### 関連

diosasetmap, t\_diosa\_setmapent

## 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) ログライター

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) ログリーダー

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)	Oracle 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)	TAM 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。

### 関連

diosaprcinit, diosathrinit, diosatrnnit, t\_diosa\_Dloggetuca

## 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。

<b>DIOSA_DONE</b> (0)	ログデータの登録に成功した。
<b>DIOSA_SWITCH</b> (21)	計画マスタ切替中である。
<b>DIOSA_ERROR</b> (-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。
<b>DIOSA_EPARAM</b> (-3)	指定したスーパーストリーム名が存在しない。
<b>DIOSA_ENOINIT</b> (-11)	ディレード転送機能が起動されていない。
<b>DIOSA_ESIZE</b> (-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。
<b>DIOSA_EFUNCNAV</b> (-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。 <b>POOLFILE=RECEIVER</b> 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。
<b>DIOSA_EDEADLOCK</b> (-36)	デッドロックを検出した(パッケージ破棄含む)。CO 制御サーバ上ではリトライ処理が自動的に行われるため、本戻り値は返却されない。
<b>DIOSA_EOVERFLOW</b> (-39)	プールファイルがオーバーフローした。
<b>DIOSA_EDB</b> (-69)	Oracle 制御ファイルのアクセスに失敗した。
<b>DIOSA_ETAM</b> (-113)	TAM 制御ファイルのアクセスに失敗した。
<b>DIOSA_ESWITCH</b> (-115)	障害時マスタ切替中である。
<b>DIOSA_EREADY</b> (-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。

### 関連

diosaprcinit, diosathrinit, diosatrinit, t\_diosa\_Dloguca

## 2.4.4 diosalrdcommit(ログリーダー強制コミット)

### 名前

diosalrdcommit - ログデータ処理の途中であっても Oracle または TAM 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 Oracle または TAM 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)	Oracle 制御ファイルのアクセスに失敗した。
DIOSA_ETAM(-113)	TAM 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	0	読み込んだログデータのユーザ通番を返却する。
size_t	DataSize	0	読み込んだログデータのサイズを返却する。
size_t	ShortageSize	0	領域サイズ不足時の不足サイズを返却する。
int	Dstatus	0	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	0	登録したログデータのディビジョン ID を返却する。
long	UserDataNo	0	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	0	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

## 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。



## 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 streamName, 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()	C0 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)	C0 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

C0 名を設定する。

### パラメータ

**coName**

設定する C0 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



# 付録A A P I 一覧

## A.1 A P I が利用可能な動作環境

アプリケーション実行制御	関数名	C O 制御										バッチ A P 制御						ログリーダー					通信制御		メモリキヤッシュ		ユーザ A P	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	C O	トランザクション終了出口名	アボート出口#1	アボート出口#2	コミット出口	ロールバック出口	プロセス終了出口名	初期化出口	C O	正常終了出口	アボート出口	コミット出口	ロールバック出口	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	電文種別決定出口	データベース接続出口	ハッシュ関数	利用者領域初期化利用者出口		
	diosaaddrconv	—	—	—	○	—	—	—	—	—	—	—	○	—	—	—	—	○	○	○	○	○	—	—	—	—	○	
	diosaapptrcf	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	diosaapptrcm	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	diosaapptrget	—	—	—	○	—	—	—	—	—	—	—	○	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
	diosaapptrset	—	—	—	○	—	—	—	—	—	—	—	○	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
	diosacmdconf	—	—	—	○	—	—	—	—	—	—	—	○	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
	diosacmdsend	—	—	—	○	—	—	—	—	—	—	—	○	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
	diosacommit	—	—	—	○	—	—	—	—	—	—	—	○	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
	diosaetgreset	—	—	—	○	—	—	—	—	—	—	—	○	—	—	—	—	—	—	○	—	—	—	—	—	—	—	
	diosaetgstart	—	—	—	○	—	—	—	—	—	—	—	○	—	—	—	—	—	—	○	—	—	—	—	—	—	—	
	diosaetgstop	—	—	—	○	—	—	—	—	—	—	—	○	—	—	—	—	—	—	○	—	—	—	—	—	—	—	
	diosaetgusinfo	—	—	—	○	—	—	—	—	—	—	—	○	—	—	—	—	—	—	○	—	—	—	—	—	—	—	
	diosagetlnodeid	—	—	○	○	○	○	○	○	○	—	—	○	○	○	○	○	—	○	○	○	—	—	—	—	—	○	
	diosagetsrctx	—	—	○	○	○	○	○	○	○	—	—	—	—	—	—	—	—	—	—	—	—					—	
	diosagoback	○	○	○	○	○	○	○	○	○	○	—	—	—	—	—	—	—	—	—	—	—					—	
	diosalock	—	△	—	○	—	—	—	—	—	—	△	○	—	—	—	—	△	—	○	—	—	—	—	—	—	○	※1

アプリケーション実行制御	関数名	C O制御									バッチA P制御						ログリーダー					通信制御		メモリキヤッシュ		ユーザA P	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	C O	トランザクション終了出口名	アポート出口#1	アポート出口#2	コミット出口	ロールバック出口	プロセス終了出口名	初期化出口	C O	正常終了出口	アポート出口	コミット出口	ロールバック出口	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	電文種別決定出口	データベース接続出口	ハッシュ関数		
	diosamaddr	－	－	－	○	－	－	－	－	－	－	○	－	－	－	－	○	○	○	○	○	－	－	－	－	○	
	diosamalloc	－	－	－	○	－	－	－	－	－	－	○	－	－	－	－	○	○	○	○	○	－	－	－	－	○	
	diosamcopy	－	－	－	○	－	－	－	－	－	－	○	－	－	－	－	○	○	○	○	○	－	－	－	－	○	
	diosamfree	－	－	－	○	－	－	－	－	－	－	○	－	－	－	－	○	○	○	○	○	－	－	－	－	○	
	diosamsgbufalloc	○	○	○	○	○	○	○	○	○	－	－	－	－	－	－	－	－	－	－	－					－	
	diosamsgbuffree	○	○	○	○	○	○	○	○	○	－	－	－	－	－	－	－	－	－	－	－					－	
	diosamsgdisp	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	diosamsgedit	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	diosaopsusiput	－	－	－	○	－	－	－	－	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	
	diosaprcforkinit	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	
	diosaprcinit	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	
	diosaprcpreexecterm	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	
	diosaprcrterm	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	－	○	
	diosarealloc	－	－	－	○	－	－	－	－	－	－	○	－	－	－	－	○	○	○	○	○	－	－	－	－	○	
	diosarecvtx	－	－	－	○	○	○	○	○	－	－	－	－	－	－	－	－	－	－	－	－					－	
	diosarollback	－	－	－	○	－	－	－	－	－	－	○	－	－	－	－	－	－	－	－	－	－	－	－	－	－	
	diosasendtx	－	－	－	○	○	○	○	－	－	－	－	－	－	－	－	－	－	－	－	－					－	
	diosasgclose	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	diosasgget	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	diosasgopen	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	

アプリケーション実行制御	関数名	C O制御									バッチA P制御						ログリーダー					通信制御		メモリキヤッシュ		ユーザA P	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	C O	トランザクション終了出口名	アポート出口#1	アポート出口#2	コミット出口	ロールバック出口	プロセス終了出口名	初期化出口	C O	正常終了出口	アポート出口	コミット出口	ロールバック出口	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	電文種別決定出口	データベース接続出口	ハッシュ関数	利用者領域初期化利用者出口	
アプリケーション実行制御	diosathrinit	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	○	
	diosathrterm	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	○	
	diosatmcactv	—	—	—	○	—	—	—	—	—	—	—	○	—	—	—	—	—	—	—	—	—	—	—	—	○	
	diosatmccmdquery	—	—	—	○	—	—	—	—	—	—	—	○	—	—	—	—	—	—	—	—	—	—	—	—	○	
	diosatmccmdset	—	—	—	○	—	—	—	—	—	—	—	○	—	—	—	—	—	—	—	—	—	—	—	—	○	
	diosatmccommit	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	○	
	diosatmccoquery	—	—	—	○	—	—	—	—	—	—	—	○	—	—	—	—	—	—	—	—	—	—	—	—	○	
	diosatmccoset	—	—	—	○	—	—	—	—	—	—	—	○	—	—	—	—	—	—	—	—	—	—	—	—	○	
	diosatmchold	—	—	—	○	—	—	—	—	—	—	—	○	—	—	—	—	—	—	—	—	—	—	—	—	○	
	diosatmcreset	—	—	—	○	—	—	—	—	—	—	—	○	—	—	—	—	—	—	—	—	—	—	—	—	○	
	diosatmcrollback	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	○	
	diosatrnnit	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	○	
	diosatrnterm	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	○	
	diosaucaget	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	—	—	—	—	—				—	
	diosaunlock	—	△	—	○	—	—	—	—	—	—	△	○	—	—	—	—	△	—	○	—	—	—	—	—	○	※1
	diosavcall	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	—	—	—	—	○	
	diosavdnameget	○	○	○	○	○	○	○	○	○	○	—	—	—	—	—	—	—	—	—	—					—	

○ … 使用可能    — … 使用不可    △ … 条件付使用可能

※1 … △は、該当する出口内でロック取得（diosalock）し、ロック解放（diosaunlock）する場合のみ使用可能

通信制御	関数名	C O制御									バッチA P制御						ログリーダー					通信制御		メモリキヤッシュ		ユーザA P	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	C O	トランザクション終了出口名	アボート出口#1	アボート出口#2	コミット出口	ロールバック出口	プロセス終了出口名	初期化出口	C O	正常終了出口	アボート出口	コミット出口	ロールバック出口	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	電文種別決定出口	データベース接続出口	ハッシュ関数			利用者領域初期化利用者出口
	diosadbchangeconnect	—	—	○	—	—	—	—	—	—	○	—	—	—	—	—	—	—	—	—	—	—	—	—	—	○		
	diosadbconnect	—	○	—	—	—	—	—	—	—	○	—	—	—	—	—	—	—	—	—	—	—	—	—	—	○		
	diosadbdisconnect	—	—	—	—	—	—	—	—	○	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	○		
	diosadbfaulthnotification	—	—	—	—	—	—	—	○	○	—	—	—	—	○	○	—	—	—	—	—	—	—	—	—	—	○	
	diosadbmulticonnect	—	○		—	—	—	—	—	—	○	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	○	
	diosadbreconnect	—	—	—	—	—	—	—	○	○	—	—	—	—	○	○	—	—	—	—	—	—	—	—	—	—	○	
	diosagetdbctx	—	—	—	○	—	—	—	○	○	—	—	○	—	—	○	○	—	—	○	—	—	—	—	—	—	○	

○ … 使用可能    — … 使用不可    △ … 条件付使用可能

メモリキャッシュ	関数名	CO制御										バッチAP制御						ログリーダー					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名	アポート出口#1	アポート出口#2	コミット出口	ロールバック出口	プロセス終了出口名	初期化出口	CO	正常終了出口	アポート出口	コミット出口	ロールバック出口	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	電文種別決定出口	データベース接続出口	ハッシュ関数	利用者領域初期化利用者出口		
	diosagethash	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	diosagetmap	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	diosagetmaphash	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	diosagetmaplist	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	diosagetmapstat	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	diosagettamnode	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	diosaimclose	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	○	
	diosaimcommit	—	—	—	—	—	—	—	○	—	—	—	—	—	—	○	—	—	—	—	—	—	—	—	—	—	○	
	diosaimcondsetkey	○	—	○	○	○	—	○	○	—	—	—	○	○	○	○	—	—	—	○	—	—	—	—	—	—	○	※1
	diosaimcondsetrange	○	—	○	○	○	—	○	○	—	—	—	○	○	○	○	—	—	—	○	—	—	—	—	—	—	○	※1
	diosaimctxclose	○	—	○	○	○	—	○	○	—	—	—	○	○	○	○	—	—	—	○	—	—	—	—	—	—	○	※1
	diosaimctxopen	○	—	○	○	○	—	○	○	—	—	—	○	○	○	○	—	—	—	○	—	—	—	—	—	—	○	※1
	diosaimdelete	—	—	—	○	○	—	○	○	—	—	—	○	○	○	○	—	—	—	○	—	—	—	—	—	—	○	※1
	diosaimdeletex1	—	—	—	○	○	—	○	○	—	—	—	○	○	○	○	—	—	—	○	—	—	—	—	—	—	○	※1
	diosaimgetmap	○	—	○	○	○	—	○	○	—	—	—	○	○	○	○	—	—	—	○	—	—	—	—	—	—	○	※1
	diosaimgetmaplist	○	—	○	○	○	—	○	○	—	—	—	○	○	○	○	—	—	—	○	—	—	—	—	—	—	○	※1
	diosaimgetptblname	○	—	○	○	○	—	○	○	—	—	—	○	○	○	○	—	—	—	○	—	—	—	—	—	—	○	※1
	diosaimgetreckeyinfo	○	—	○	○	○	—	○	○	—	—	—	○	○	○	○	—	—	—	○	—	—	—	—	—	—	○	※1
	diosaimgettblid	○	—	○	○	○	—	○	○	—	—	—	○	○	○	○	—	—	—	○	—	—	—	—	—	—	○	※1
	diosaimgettbllist	○	—	○	○	○	—	○	○	—	—	—	○	○	○	○	—	—	—	○	—	—	—	—	—	—	○	※1

メモリキャッシュ	関数名	C O制御										バッチA P制御						ログリーダー					通信制御		メモリキャッシュ		ユーザA P	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	C O	トランザクション終了出口名	アボート出口#1	アボート出口#2	コミット出口	ロールバック出口	プロセス終了出口名	初期化出口	C O	正常終了出口	アボート出口	コミット出口	ロールバック出口	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	電文種別決定出口	データベース接続出口	ハッシュ関数	利用者領域初期化利用者出口		
メモリキャッシュ	diosaimopen	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	○	
	diosaimread	○	—	○	○	○	—	○	○	—	—	—	○	○	○	○	—	—	—	○	—	—	—	—	—	—	○	※1
	diosaimread1	△	—	△	○	○	—	○	○	—	—	—	○	○	○	○	—	—	—	○	—	—	—	—	—	—	○	※1※3
	diosaimrewrite	—	—	—	○	○	—	○	○	—	—	—	○	○	○	○	—	—	—	○	—	—	—	—	—	—	○	※1
	diosaimrollback	—	—	—	—	—	—	—	—	○	—	—	—	—	—	—	○	—	—	—	—	—	—	—	—	—	○	
	diosaimsetmap	○	—	○	○	○	—	○	○	—	—	—	○	○	○	○	—	—	—	○	—	—	—	—	—	—	○	※1
	diosaimtruncate	—	—	—	○	○	—	○	—	—	—	—	○	○	○	—	—	—	—	—	—	—	—	—	—	—	○	
	diosaimtxstart	—	—	—	—	—	—	—	○	○	—	—	—	—	—	○	○	—	—	—	—	—	—	—	—	—	○	
	diosaimwrite	—	—	—	○	○	—	○	○	—	—	—	○	○	○	○	—	—	—	○	—	—	—	—	—	—	○	※1
	diosasetmap	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	—	—	—	—	—	—	—	—	—	○	※2
	diosatamswitch	—	—	—	○	—	—	—	—	—	—	—	○	—	—	—	—	—	—	—	—	—	—	—	—	—	○	

○ … 使用可能    — … 使用不可    △ … 条件付使用可能

※1 … ログリーダー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 … 事業部 PP 専用の API であり、業務 AP では使用不可

※3 … △は、DIOSA\_NOLOCK 指定での diosaimread1 のみ使用可能

データストア	関数名	CO制御										バッチAP制御						ログリーダー					通信制御		メモリキヤッシュ		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名	アポート出口#1	アポート出口#2	コミット出口	ロールバック出口	プロセス終了出口名	初期化出口	CO	正常終了出口	アポート出口	コミット出口	ロールバック出口	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	電文種別決定出口	データベース接続出口	ハッシュ関数	利用者領域初期化利用者出口		
	diosadgetlog	—	—	—	○	—	—	—	—	—	—	—	○	—	—	—	—	—	—	—	—	—	—	—	—	—	○	
	diosadputlog	—	—	—	○	—	—	—	—	—	—	—	○	—	—	—	—	—	—	—	—	—	—	—	—	—	○	
	diosalrdcommit	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	○	—	—	—	—	—	—	—	—	

○ … 使用可能    — … 使用不可    △ … 条件付使用可能

D I O S A / X T P V 1.1  
A P I リファレンス

2017 年 4 月 11 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)

©NEC Corporation 2011, 2017

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