CICS Transaction Server for z/OS

XPI 関数リファレンス

バージョン *5* リリース *5*

CICS Transaction Server for z/OS

XPI 関数リファレンス

バージョン *5* リリース *5*

注記

本書および本書で紹介する製品をご使用になる前に、 143 ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM CICS Transaction Server for z/OS バージョン 5 リリース 5 (製品番号 5655-Y04) および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原典： CICS Transaction Server for z/OS
XPI Function Reference
Version 5 Release 5

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

© Copyright IBM Corporation 1977, 2018.

目次

この PDF について v

第 1 章 ビジネス・アプリケーション・マネージャー・ドメイン XPI 関数 1
INQUIRE_ACTIVATION 呼び出し 1

第 2 章 ディレクトリー・ドメイン XPI 関数 3
BIND_LDAP 呼び出し 3
END_BROWSE_RESULTS 呼び出し 5
FLUSH_LDAP_CACHE 呼び出し 6
FREE_SEARCH_RESULTS 呼び出し 6
GET_ATTRIBUTE_VALUE 呼び出し 7
GET_NEXT_ATTRIBUTE 呼び出し 8
GET_NEXT_ENTRY 呼び出し 9
SEARCH_LDAP 呼び出し 10
START_BROWSE_RESULTS 呼び出し 12
UNBIND_LDAP 呼び出し 13

第 3 章 ディスパッチャー XPI 関数 . . 15
SUSPEND 処理および RESUME 処理の同期プロトコル 15
通常の同期プロトコル 15
同期プロトコルとタスク・パージ 16
ADD_SUSPEND 呼び出し 17
CHANGE_PRIORITY 呼び出し 19
DELETE_SUSPEND 呼び出し 20
RESUME 呼び出し 21
SUSPEND 呼び出し 22
WAIT_MVS 呼び出し 26

第 4 章 ダンプ管理 XPI 関数 31
SYSTEM_DUMP 呼び出し 31
TRANSACTION_DUMP 呼び出し 32

第 5 章 エンキュー・ドメイン XPI 関数 37
DEQUEUE 機能 37
ENQUEUE 関数 37

第 6 章 カーネル・ドメイン XPI 関数 41
START_PURGE_PROTECTION 関数 41
STOP_PURGE_PROTECTION 機能 41
パージ保護呼び出しのネスト 42

第 7 章 ローダー XPI 関数 43
ACQUIRE_PROGRAM 呼び出し 43
DEFINE_PROGRAM 呼び出し 46
DELETE_PROGRAM 呼び出し 49
IDENTIFY_PROGRAM 呼び出し 50
RELEASE_PROGRAM 呼び出し 52

第 8 章 ログ・マネージャー XPI 関数 55
INQUIRE_PARAMETERS 呼び出し 55
SET_PARAMETERS 呼び出し 55

第 9 章 モニター XPI 関数 57
INQUIRE_APP_CONTEXT 呼び出し 57
INQUIRE_MONITORING_DATA 呼び出し . . . 59
MONITOR 呼び出し 60
SET_TRACKING_DATA 呼び出し 63

第 10 章 オブジェクト・トランザクション XPI 関数 65
IMPORT_TRAN 呼び出し 65
COMMIT_ONE_PHASE 呼び出し 66
PREPARE 呼び出し 67
COMMIT 呼び出し 68
ROLLBACK 呼び出し 68
SET_ROLLBACK_ONLY 呼び出し 69
SET_COORDINATOR 呼び出し 69

第 11 章 パラメーター・ドメイン XPI 関数 71
INQUIRE_FEATUREKEY 呼び出し 71

第 12 章 プログラム管理 XPI 関数 . . . 73
INQUIRE_PROGRAM 呼び出し 73
INQUIRE_CURRENT_PROGRAM 呼び出し . . . 82
SET_PROGRAM 呼び出し 85
START_BROWSE_PROGRAM 呼び出し 88
GET_NEXT_PROGRAM 呼び出し 89
END_BROWSE_PROGRAM 呼び出し 91
INQUIRE_AUTOINSTALL 呼び出し 91
SET_AUTOINSTALL 呼び出し 92
BIND_CHANNEL 呼び出し 94

第 13 章 状態データ・アクセス XPI 関数 95
INQ_APPLICATION_DATA 呼び出し 95
INQUIRE_SYSTEM 呼び出し 97
SET_SYSTEM 呼び出し 102

第 14 章 ストレージ管理 XPI 関数 . . 105
GETMAIN 呼び出し 105
FREEMAIN 呼び出し 108
INQUIRE_ACCESS 呼び出し 109
INQUIRE_ELEMENT_LENGTH 呼び出し 109
INQUIRE_SHORT_ON_STORAGE 呼び出し . . . 110
INQUIRE_TASK_STORAGE 呼び出し 111
SWITCH_SUBSPACE 呼び出し 112

© Copyright IBM Corp. 1977, 2018

第 15 章 トレース制御 XPI 関数 . . . 115
TRACE_PUT 呼び出し 115

第 16 章 トランザクション管理 XPI 関数 117
INQUIRE_CONTEXT 呼び出し 117
INQUIRE_DTRTRAN 呼び出し 119
INQUIRE_MXT 呼び出し 119
INQUIRE_TCLASS 呼び出し 121
INQUIRE_TRANDEF 呼び出し 122
INQUIRE_TRANSACTION 呼び出し 131
SET_TRANSACTION 呼び出し 136

第 17 章 ユーザー・ジャーナリング XPI 関数 139
WRITE_JOURNAL_DATA 呼び出し 139

第 18 章 スレッド・セーフ XPI コマンド 141

特記事項. 143

索引 149

この PDF について

この PDF は、いくつかの CICS サービスにアクセスするためにグローバル・ユーザー出口プログラムで使用できる XPI マクロ関数の解説書です。XPI 関数は、通常は CICS ドメインにより、機能の関係に従ってグループ化されています。プログラムでこれらの関数を使用する方法を確認するには、「*CICS* システム・プログラムの開発」という PDF を参照してください。CICS TS V5.4 より前は、この PDF の情報は「カスタマイズ・ガイド」に収録されていました。

本書で使用される用語および表記の詳細については、IBM Knowledge Center の CICS 資料で使用されている表記規則および用語を参照してください。

この PDF の作成日

この PDF は、2018 年 12 月 14 日に作成されました。

© Copyright IBM Corp. 1977, 2018

第 1 章 ビジネス・アプリケーション・マネージャー・ドメイン XPI 関数

XPI は、ビジネス・アプリケーション・マネージャー・ドメイン関数を提供します。これは DFHBABRX 呼び出し INQUIRE_ACTIVATION です。

INQUIRE_ACTIVATION 呼び出し

INQUIRE_ACTIVATION 機能は、DFHBABRX マクロ呼び出しで提供されます。INQUIRE_ACTIVATION 呼び出しを使用すると、現行トランザクションにおけるビジネス・トランザクション・アクティビティーのアクティビティー名およびプロセス・タイプを取得できます。

INQUIRE_ACTIVATION

```
DFHBABRX [CALL,]
    [CLEAR,]
    [IN,
    FUNCTION(INQUIRE_ACTIVATION),
    [TRANSACTION_TOKEN(name8),]]
    [RETURNED_ACTIVITYID(buffer_descriptor)]
    [RETURNED_PROCESS_NAME(buffer_descriptor)]
    [OUT,
    [ACTIVITY_NAME(name16)]
    [PROCESS_TYPE(name8)]
    RESPONSE (name1 | *),
    REASON (name1 | *)]
```

このコマンドはスレッド・セーフです。

ACTIVITY_NAME(name16)
ユーザーが割り当てる 16 文字の BTS アクティビティー名を戻します。

PROCESS_TYPE(name8)
BTS プロセスにおけるタイプ定義の 8 文字の ID を戻します。

RETURNED_ACTIVITYID(buffer_descriptor)
CICS が割り当てる 52 文字の BTS アクティビティーの ID を戻します。RETURNED_ACTIVITYID は出力パラメーター (BAM によって返される) で、データ型はバッファーです。したがって呼び出し元は、バッファーとして使用される領域を、呼び出しへの入力として提供しなければなりません。

RETURNED_PROCESS_NAME(buffer_descriptor)
BTS プロセスの 36 文字の名前を戻します。RETURNED_PROCESS_NAME は出力パラメーター (BAM によって返される) で、データ型はバッファーです。したがって呼び出し元は、バッファーとして使用される領域を、呼び出しへの入力として提供しなければなりません。

TRANSACTION_TOKEN(name8)
照会対象のタスクのトランザクション・トークンを指定します。

© Copyright IBM Corp. 1977, 2018

INQUIRE_ACTIVATION の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	ACTIVITY_NOT_FOUND
DISASTER	なし
INVALID	INVALID_BUFFER_LENGTH
KERNERROR	なし
PURGED	なし

詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。

第 2 章 ディレクトリー・ドメイン XPI 関数

XPI は、ディレクトリー・ドメイン関数を提供します。これらの関数を使用すると、LDAP セッションのオープンとクローズ、資格情報の結果の参照、結果のスキャンと検出、参照のクローズ、正しい値の返し、および検索のクローズを行うことができます。

ディレクトリー・ドメイン関数は、以下の DFHDDAPX 呼び出しです。

- BIND_LDAP
- END_BROWSE_RESULTS
- FLUSH_LDAP_CACHE
- FREE_SEARCH_RESULTS
- GET_ATTRIBUTE_VALUE
- GET_NEXT_ATTRIBUTE
- GET_NEXT_ENTRY
- SEARCH_LDAP
- START_BROWSE_RESULTS
- UNBIND_LDAP

BIND_LDAP 呼び出し

BIND_LDAP 呼び出しは、LDAP サーバーとのセッションを確立します。

LDAP サーバーは、以下のいずれかで識別されます。

- LDAP URL と、予想されるデータの抽出が許可されているユーザーの識別名およびパスワード。
- LDAP URL と識別名およびパスワードを含む LDAPBIND クラスの RACF プロファイル。アプリケーションで LDAP 資格情報をコーディングする必要がないため、これが優先オプションです。

BIND_LDAP

```
DFHDDAPX [CALL],
    [CLEAR],
    [IN,
    FUNCTION(BIND_LDAP),
    {LDAP_BIND_PROFILE(block-descriptor)|
    LDAP_SERVER_URL((block-descriptor),DISTINGUISHED_NAME((block-descriptor),
    PASSWORD(block-descriptor),}
    [CACHE_SIZE(name4),CACHE_TIME_LIMIT(name4),]]
    [OUT,
    LDAP_SESSION_TOKEN(name4),
    [LDAP_RESPONSE(name4),]
    RESPONSE(name1 | *),
    REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

CACHE_SIZE(name4)
LDAP 検索結果のキャッシングに使用可能なバイト数を指定するフルワード。ゼロの値は、無制限のキャッシュ・サイズを示します。CACHE_SIZE が指定さ

© Copyright IBM Corp. 1977, 2018

れている場合、CACHE_TIME_LIMIT も指定する必要があります。どちらのパラメーターも指定しなかった場合、結果はキャッシュに入れられません。

CACHE_TIME_LIMIT(name4)
LDAP 検索結果をキャッシュに保存する時間の長さ (秒単位) を指定するフルワード。値ゼロは、キャッシュの時間制限が無制限であることを示します。

DISTINGUISHED_NAME(block-descriptor)
選択サーバーへのバインドを許可されているユーザーの LDAP 識別名の場所を指定します。ブロック記述子は、2 つのフルワードのデータで、最初のワードにはデータのアドレスが入り、2 番目のワードにはデータの長さ (バイト単位) が入ります。

ブロック記述子について詳しくは、XPI 構文を参照してください。

LDAP_BIND_PROFILE(block-descriptor)
アクセス対象の LDAP サーバーの URL および 資格情報を含む LDAPBIND クラス内の RACF プロファイルの名前の位置を指定します。ブロック記述子は、2 つのフルワードのデータで、最初のワードにはデータのアドレスが入り、2 番目のワードにはデータの長さ (バイト単位) が入ります。

ブロック記述子について詳しくは、XPI 構文を参照してください。
LDAP_BIND_PROFILE を指定するか、または 3 つのパラメーター LDAP_SERVER_URL、DISTINGUISHED_NAME、PASSWORD のすべてを指定する必要があります。

LDAP_RESPONSE(name4)
URL およびユーザー資格情報の受信に対する応答として、LDAP API によって送信された戻りコードを指定します。

LDAP_SERVER_URL(block-descriptor)
アクセス対象の LDAP サーバーの LDAP URL の場所を指定します (ldap://server:port 形式)。コロンとポート番号を省略すると、ポートはデフォルトの 389 になります。ブロック記述子は、2 つのフルワードのデータで、最初のワードにはデータのアドレスが入り、2 番目のワードにはデータの長さ (バイト単位) が入ります。

ブロック記述子について詳しくは、XPI 構文を参照してください。

LDAP_SESSION_TOKEN(name4)
LDAP 接続を指定するフルワードのトークンの名前。

PASSWORD(block-descriptor)
DISTINGUISHED_NAME 入力で識別されるユーザーのパスワードの場所を指定します。ブロック記述子は、2 つのフルワードのデータで、最初のワードにはデータのアドレスが入り、2 番目のワードにはデータの長さ (バイト単位) が入ります。

ブロック記述子について詳しくは、XPI 構文を参照してください。

BIND_LDAP の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	INVALID_BUFFER_LENGTH
	INVALID_LDAP_PROFILE

RESPONSE	*REASON*
	INVALID_LDAP_URL
	LDAP_INACTIVE
	NOTAUTH
	NOTFOUND
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

注: 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。

END_BROWSE_RESULTS 呼び出し

END_BROWSE_RESULTS 呼び出しを使用すると、START_BROWSE_RESULTS 呼び出しによって開始された参照セッションを終了することができます。

END_BROWSE_RESULTS

```
DFHDDAPX [CALL],
    [CLEAR],
    [IN,
    FUNCTION(END_BROWSE_RESULTS),
    SEARCH_TOKEN(name4),]
    [OUT,
    [LDAP_RESPONSE(name4),]
    RESPONSE(name1 | *),
    REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

LDAP_RESPONSE(name4)
LDAP API によって送信された戻りコードを指定します。

SEARCH_TOKEN(name4)
SEARCH_LDAP 関数によって返されたフルワードのトークンの名前。

END_BROWSE_RESULTS の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	INVALID_TOKEN
	INVALID_CALLING_SEQUENCE
	NOTFOUND
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

注: 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。

FLUSH_LDAP_CACHE 呼び出し

FLUSH_LDAP_CACHE 呼び出しは、指定された LDAP 接続に関するすべてのキャッシュされた検索応答の内容を削除します。

FLUSH_LDAP_CACHE

```
DFHDDAPX [CALL],
    [CLEAR],
    [IN,
    FUNCTION(FLUSH_LDAP_CACHE),
    LDAP_SESSION_TOKEN(name4),]
    [OUT,
    [LDAP_RESPONSE(name4),]
    RESPONSE(name1 | *),
    REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

LDAP_RESPONSE(name4)
LDAP API によって送信された戻りコードを指定します。

LDAP_SESSION_TOKEN(name4)
BIND_LDAP 関数によって返されたフルワードのトークンの名前。

FLUSH_LDAP_CACHE の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	INVALID_TOKEN
	LDAP_INACTIVE
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

注: 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。

FREE_SEARCH_RESULTS 呼び出し

FREE_SEARCH_RESULTS 呼び出しは、SEARCH_LDAP 関数によって保持されているすべてのストレージを解放します。検索結果は強制終了され、検索トークンは無効になります。アプリケーションが FREE_SEARCH_RESULTS 関数を呼び出さない場合は、タスクの終了時に CICS® によって呼び出されます。

FREE_SEARCH_RESULTS

```
DFHDDAPX [CALL],
    [CLEAR],
    [IN,
    FUNCTION(FREE_SEARCH_RESULTS),
    SEARCH_TOKEN(name4),]
    [OUT,
    [LDAP_RESPONSE(name4),]
    RESPONSE(name1 | *),
    REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

LDAP_RESPONSE(name4)
LDAP API によって送信された戻りコードを指定します。

SEARCH_TOKEN(name4)
SEARCH_LDAP 関数によって返されたフルワードのトークンの名前。

FREE_SEARCH_RESULTS の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	INVALID_TOKEN
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

注: 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。

GET_ATTRIBUTE_VALUE 呼び出し

GET_ATTRIBUTE_VALUE 呼び出しを使用して、SEARCH_LDAP 呼び出しによって返される属性に関連付けられている値を取得することができます。エントリーは LDAP レコードで、属性はエントリー内のエレメントです。この属性は、GET_NEXT_ATTRIBUTE 関数を使用するか、属性の名前を指定することによって返すことができます。

GET_ATTRIBUTE_VALUE

```
DFHDDAPX [CALL],
    [CLEAR],
    [IN,
    FUNCTION(GET_ATTRIBUTE_VALUE),
    SEARCH_TOKEN(name4),
    LDAP_ATTRIBUTE_NAME(block-descriptor),
    LDAP_ATTRIBUTE_VALUE(buffer-descriptor),
    [ATTRIBUTE_TYPE(name4),]
    [VALUE_ARRAY_POSITION(name4),]]
    [OUT,
    [LDAP_RESPONSE(name4),]
    RESPONSE(name1 | *),
    REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

ATTRIBUTE_TYPE(name4)
属性の形式を示す、キーワード CHARACTER または BINARY を指定します。このパラメーターを指定しなかった場合、値は CHARACTER であると見なされます。

LDAP_ATTRIBUTE_NAME(block-descriptor)
LDAP 属性名の場所を指定します。ブロック記述子は、2 つのフルワードのデータで、最初のワードには属性名のアドレスが入り、2 番目のワードには属性名の長さ (バイト単位) が入ります。ブロック記述子について詳しくは、XPI 構文を参照してください。

LDAP_ATTRIBUTE_VALUE(buffer-descriptor)
属性値が返されるバッファーを示します。バッファー記述子には、以下の 3 つのフルワードのグループを指定します。

- 結果が返されるアドレス。
- 返されるデータの最大サイズ (バイト単位)。
- 結果の実際の長さ (バイト単位)。これは、* で指定することができます。長さは DDAP_LDAP_ATTRIBUTE_VALUE_N に返されます。

バッファー記述子について詳しくは、XPI 構文を参照してください。

LDAP_RESPONSE(name4)
LDAP API によって送信された戻りコードを指定します。

SEARCH_TOKEN(name4)
SEARCH_LDAP 関数によって返されたフルワードのトークンの名前。

VALUE_ARRAY_POSITION(name4)
現在の属性の値配列内での、要求された値の位置を指定します。このパラメーターは、複数の値が予想される場合のみ必須です。配列の索引付けは 1 から開始します。

GET_ATTRIBUTE_VALUE の RESPONSE 値および REASON 値

RESPONSE	REASON
OK	なし
EXCEPTION	INVALID_TOKEN
	NOTFOUND
	INVALID_BUFFER_LENGTH
	INVALID_CALLING_SEQUENCE
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

注: 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。

GET_NEXT_ATTRIBUTE 呼び出し

GET_NEXT_ATTRIBUTE 呼び出しを使用すると、SEARCH_LDAP 呼び出しによって返された一連のエントリーから次の属性を取得することができます。エントリーは LDAP レコードで、属性はエントリー内のエレメントです。

GET_NEXT_ATTRIBUTE

```
DFHDDAPX [CALL],
    [CLEAR],
    [IN,
    FUNCTION(GET_NEXT_ATTRIBUTE),
    SEARCH_TOKEN(name4),
    LDAP_ATTRIBUTE_NAME(buffer-descriptor),]
    [OUT,
    [LDAP_RESPONSE(name4),]
    [VALUE_COUNT(name4),]
    RESPONSE(name1 | *),
    REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

LDAP_ATTRIBUTE_NAME(buffer-descriptor)
属性名が返されるバッファーを示します。バッファー記述子には、以下の 3 つのフルワードのグループを指定します。

- データが返されるアドレス。
- 返されるデータの最大サイズ (バイト単位)。
- データの実際の長さ (バイト単位)。これは、* で指定することができます。長さは DDAP_LDAP_ATTRIBUTE_NAME_N に返されます。

バッファー記述子について詳しくは、XPI 構文を参照してください。

LDAP_RESPONSE(name4)
LDAP API によって送信された戻りコードを指定します。

SEARCH_TOKEN(name4)
SEARCH_LDAP 関数によって返されたフルワードのトークンの名前。

VALUE_COUNT(name4)
この属性に対して返される値の数が入るフルワード。通常、返される値が 1 つあります。

GET_NEXT_ATTRIBUTE の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	BROWSE_END
	INVALID_BUFFER_LENGTH
	INVALID_CALLING_SEQUENCE
	INVALID_TOKEN
	NOT_FOUND
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

注: 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。

GET_NEXT_ENTRY 呼び出し

GET_NEXT_ENTRY 呼び出しを使用すると、SEARCH_LDAP 呼び出しによって返された一連のエントリーから、次のエントリーを取得することができます。エントリーは LDAP レコードです。エントリーに関連付けられている識別名は、この呼び出しによって返されます。

GET_NEXT_ENTRY

```
DFHDDAPX [CALL],
    [CLEAR],
    [IN,
    FUNCTION(GET_NEXT_ENTRY),
    SEARCH_TOKEN(name4),
    [DISTINGUISHED_NAME(buffer-descriptor),]]
    [OUT,
```

```
    [LDAP_RESPONSE(name4),]
    [ATTRIBUTE_COUNT(name4),]
    RESPONSE(name1 | *),
    REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

ATTRIBUTE_COUNT(name4)
取得されたエントリー内にある属性の数を指定します。

DISTINGUISHED_NAME(buffer-descriptor)
検索の次のエントリーの識別名が返されるバッファーを示します。バッファー記述子には、以下の 3 つのフルワードのグループを指定します。

- データが返されるアドレス。
- 返されるデータの最大サイズ (バイト単位)。
- データの実際の長さ (バイト単位)。これは、* で指定することができます。長さは DDAP_DISTINGUISHED_NAME_N に返されます。

バッファー記述子について詳しくは、XPI 構文を参照してください。

LDAP_RESPONSE(name4)
LDAP API によって送信された戻りコードを指定します。

SEARCH_TOKEN(name4)
SEARCH_LDAP 関数によって返されたフルワードのトークンの名前。

GET_NEXT_ENTRY の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	INVALID_TOKEN
	INVALID_BUFFER_LENGTH
	INVALID_CALLING_SEQUENCE
	BROWSE_END
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

注: 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。

SEARCH_LDAP 呼び出し

SEARCH_LDAP 呼び出しは、指定の LDAP サーバーに検索要求を送信します。検索では、検索のターゲットとなる LDAP 識別名を指定します。

この検索は、参照または選択可能な一連の結果 (属性またはエントリー) を返します。エントリーは LDAP レコードで、属性はエントリー内のエレメントです。

SEARCH_LDAP

```
DFHDDAPX [CALL],
    [CLEAR],
    [IN,
    FUNCTION(SEARCH_LDAP),
```

```
LDAP_SESSION_TOKEN(name4),
DISTINGUISHED_NAME(block-descriptor),
[FILTER(block-descriptor),]
[SEARCH_TIME_LIMIT(name4),]]
[OUT,
SEARCH_TOKEN(name4),
[LDAP_RESPONSE(name4),]
[ENTRY_COUNT(name4),]
RESPONSE(name1 | *),
REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

DISTINGUISHED_NAME(block-descriptor)
LDAP 識別名の場所を指定します。ブロック記述子は、2 つのフルワードのデータで、最初のワードにはデータのアドレスが入り、2 番目のワードにはデータの長さ (バイト単位) が入ります。ブロック記述子について詳しくは、XPI 構文を参照してください。

ENTRY_COUNT(name4)
検索で返された LDAP エントリーの数。

FILTER(block-descriptor)
検索を制限する LDAP フィルター・ストリングの場所を指定します。このパラメーターが指定されていない場合、またはゼロの場合、検索フィルターは (objectClass=*) に設定されます。ブロック記述子は、2 つのフルワードのデータで、最初のワードにはデータのアドレスが入り、2 番目のワードにはデータの長さ (バイト単位) が入ります。ブロック記述子について詳しくは、XPI 構文を参照してください。

LDAP_RESPONSE(name4)
LDAP API によって送信された戻りコードを指定します。

LDAP_SESSION_TOKEN(name4)
BIND_LDAP 関数によって返されたフルワードのトークンの名前。

SEARCH_TIME_LIMIT(name4)
検索の時間制限 (秒単位) を指定します。この時間制限内に検索が成功しない場合、検索は中止されます。このパラメーターが指定されていない場合、またはゼロの場合、検索時間は無制限です。

SEARCH_TOKEN(name4)
検索の現在位置を識別して保持するフルワードのトークンの名前。

SEARCH_LDAP の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	INVALID_BUFFER_LENGTH
	INVALID_TOKEN
	NOTFOUND
	TIMED_OUT
	LDAP_INACTIVE
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

注: 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。

START_BROWSE_RESULTS 呼び出し

START_BROWSE_RESULTS 呼び出しを使用すると、SEARCH_LDAP 呼び出しから返された結果 (属性またはエントリー) を参照することができます。START_BROWSE_RESULTS は、返された最初または唯一のエントリーで参照を開始します (検索で複数のエントリーが返される場合があります)。GET_NEXT_ENTRY 呼び出しを使用すると、他のエントリーを検索することができます。

START_BROWSE_RESULTS は、SEARCH_TOKEN に対して複数回発行できます。GET_NEXT_ENTRY 呼び出しまたは GET_NEXT_ATTRIBUTE 呼び出しの後でこの呼び出しが発行された場合、参照位置は検索結果の先頭にリセットされます。

START_BROWSE_RESULTS

```
DFHDDAPX [CALL],
    [CLEAR],
    [IN,
    FUNCTION(START_BROWSE_RESULTS),
    SEARCH_TOKEN(name4),
    [DISTINGUISHED_NAME(buffer-descriptor),]]
    [OUT,
    [LDAP_RESPONSE(name4),]
    [ATTRIBUTE_COUNT(name4),]
    RESPONSE(name1 | *),
    REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

DISTINGUISHED_NAME(buffer-descriptor)
返された最初または唯一の検出済み結果の識別名を入れるバッファーを指定します。バッファー記述子には、以下の 3 つのフルワードのグループを指定します。

- データが返されるアドレス。
- データが返されるバッファーの長さ (バイト単位)。
- データの最大長 (バイト単位)。これは、* で指定することができます。長さは DDAP_DISTINGUISHED_NAME_N に返されます。

バッファー記述子について詳しくは、XPI 構文を参照してください。

ATTRIBUTE_COUNT(name4)
現行のエントリーで参照できる属性の数を示すフルワード。

LDAP_RESPONSE(name4)
LDAP API によって送信された戻りコードを指定します。

SEARCH_TOKEN(name4)
SEARCH_LDAP 関数によって返されたフルワードのトークンの名前。

START_BROWSE_RESULTS の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	INVALID_TOKEN
	INVALID_BUFFER_LENGTH
	INVALID_CALLING_SEQUENCE
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

注: 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。

UNBIND_LDAP 呼び出し

UNBIND_LDAP 呼び出しは、LDAP サーバーとのセッションを終了します。

UNBIND_LDAP

```
DFHDDAPX [CALL],
    [CLEAR],
    [IN,
    FUNCTION(UNBIND_LDAP),
    LDAP_SESSION_TOKEN(name4),]
    [OUT,
    [LDAP_RESPONSE(name4),]
    RESPONSE(name1 | *),
    REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

LDAP_RESPONSE(name4)
LDAP API によって送信された戻りコードを指定します。

LDAP_SESSION_TOKEN(name4)
BIND_LDAP 関数によって返されたフルワードのトークンの名前。

UNBIND_LDAP の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	INVALID_TOKEN
	LDAP_INACTIVE
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

注: 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。

第 3 章 ディスパッチャー XPI 関数

XPI は、6 つのディスパッチャー関数を提供します。これらの関数は、DFHDSSRX 呼び出し
ADD_SUSPEND、SUSPEND、RESUME、DELETE_SUSPEND、WAIT_MVS、および DFHDSATX 呼び出し CHANGE_PRIORITY です。

これらのディスパッチャー呼び出しの使用には制限があります。関数を使用する前に、Global user exit programsで出口ごとに指定されている詳細を確認してください。

注:

1. SUSPEND 呼び出しまたは RESUME 呼び出しを発行する前に、ADD_SUSPEND 呼び出しを発行して中断トークンを作成する必要があります。
2. 中断されたタスクが取り消された場合、SUSPEND は RESPONSE 値「PURGED」および REASON 値「TASK_CANCELLED」で失敗します。対応する RESUME 呼び出しが、RESPONSE 値「EXCEPTION」および REASON 値「TASK_CANCELLED」を返して戻ります。
3. 中断されたタスクがタイムアウトになった場合、SUSPEND は RESPONSE 値「PURGED」および REASON 値「TIMED_OUT」で失敗します。対応する RESUME 呼び出しが、RESPONSE 値「EXCEPTION」および REASON 値「TIMED_OUT」を返して戻ります。
4. ディスパッチャー・プロトコルでは、SUSPEND が (タスクの取り消しまたはタイムアウトにより) パージされた場合でも、RESUME を発行する必要があります。各 SUSPEND 呼び出しに対して RESUME を 1 回だけ発行する必要があります。

SUSPEND 処理および RESUME 処理の同期プロトコル

XPI の SUSPEND 処理および RESUME 処理を使用する場合は、タスク・パージを効果的に処理できるように、正しいプロトコルに従う必要があります。

通常の同期プロトコル

通常の場合、同期には 2 つのタスクと 3 つの操作が含まれます。

以下の操作例では、タスクは A (保守を要求するタスク) と B (タスク A からの要求を処理するタスク) です。

1. タスク A は、以下により要求を開始します。
   - タスク B が使用するパラメーターの設定
   - タスク B の再開
   - SUSPEND 呼び出しの発行
2. タスク B は、以下によりアクションを実行します。
   - パラメーターの取得

© Copyright IBM Corp. 1977, 2018

- アクションの実行
- 結果の設定
- 終了 (または新規処理の待機)

3. タスク A は、以下により対話を終了します。
   - タスク B で残された結果の取得

このシーケンスは、以下のようになります。

```
Task A:  Set parameters  Resume task B  Suspend ———————————————— Get results
                         ↓                                       ↑
Task B:                  Get parameters  Process request  Set results  Resume  Detach
                                                                      task A
```

再開と中断を無視すると、実行量は以下のようになります。

```
Set parameters; Get parameters; Process request; Set results; Get results
```

これらのアクションは、常に順次です。

同期プロトコルとタスク・パージ

タスクのいずれかをパージする場合、タスク A が中断状態であれば、パージされるタスクはタスク A になります。この場合、SUSPEND が失敗した後のタスク A の実行は、タスク B と並行に行われ、適切な逐次化が行われなくなります。プログラムが未変更のままになると、要求の処理と結果の設定が結果の取得と同時に実行され、結果は予測不能になります。

タスク・パージの代替方法

この問題を回避する方法の 1 つは、タスク A をパージする場合に、タスク A がタスク B の妨げになりそうな処理を実行しないようにすることです。これは、タスク A を切り離すとタスク B がアクセスしなければならないストレージが解放されてしまう場合は、タスク A を切り離してはならないということです。現在関与しているタスクはタスク B のみであるため、タスク B に、両方のタスクをクリーンアップする責任が残ることになります。

このシーケンスを以下の図に示します。

```
Task A:  Set parameters;  Resume task B;  Suspend-fail
                          |
                          ↓
Task B:                   Get parameters;  Process request;  Resume-fail;  Clean up both
```

タスク・パージは、SUSPEND と RESUME の間で実行された場合のみ有効であるため、中断の失敗が再開の失敗の前に発生します。逐次化で通常の同期プロトコルと同じ制約を使用すると、タスク・パージ・プロトコルを論理的に以下のシーケンスに縮小することができます。

```
Set parameters;  Get parameters;  Process request;  Clean up
```

違いは、結果の設定と結果の取得がクリーンアップに置き換わっていることです。これら 2 つのシーケンスのみが発生可能なことが重要です。つまり、両方のプログラムを正確にコーディングする必要があります。CICS は、SUSPEND 処理と RESUME 処理のどちらかが実行されたこと、または失敗したことを両方のタスクに通知します。

以下に、これらの規則に準拠するプログラミング手順を示します。

```
Program for Task A                  Program for Task B
SET PARAMETERS;
RESUME B;                           GET PARAMETERS;
SUSPEND A;                          PROCESS REQUEST;
                                    RESUME A;
if                                  if
  RESPONSE = OK                       RESPONSE ¬= OK
then                                then
  GET RESULTS;                        CLEAN UP;
endif                               endif
```

SUSPEND と RESUME がどちらも「OK」を返した場合、例では、通常の同期の規則に従い、結果の取得で処理が完了します。SUSPEND と RESUME がどちらも「OK」を返さなかった場合、例ではタスク・パージ・プロトコルの規則に従い、クリーンアップで処理が完了します。

前述のシーケンスは、タスク・パージの問題を取り扱う方法の 1 つです。この方法を使用すると、タスク B は、要求を処理しているときに、タスク A がパージされているかどうかを認識しません。つまり、A が所有するリソースを使用する場合、(A がパージされた場合には) B は十分に注意する必要があります。場合によっては、この制限によって問題が発生することがあります。

別の方法は、次のとおりです。タスク A をパージする場合を示します。

1. A は、所有するリソースが既に利用できなくなっていることを B に通知し、A が所有するリソースを使用しないように知らせます。
2. A が、独自のクリーンアップ処理 (ディスパッチャー・プロトコルの要求に従って、パージされた SUSPEND に対して RESUME 呼び出しを発行することを含む) を実行し、異常終了します。
3. B が、独自のクリーンアップ処理を実行します。

ADD_SUSPEND 呼び出し

ADD_SUSPEND は、中断トークンを獲得します。後でこのトークンを使用して SUSPEND/RESUME ペアを識別することができます。

ADD_SUSPEND

```
DFHDSSRX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(ADD_SUSPEND),
      [RESOURCE_NAME(name16 | string | 'string'),]
      [RESOURCE_TYPE(name8 | string | 'string'),]]
```

```
[OUT,
SUSPEND_TOKEN(name4 | (Rn)),
RESPONSE(name1 | *),
REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

RESOURCE_NAME(name16 | string | "string")
処理の中断および再開に関係するリソースの文書化およびトレースに使用できる 16 文字のストリングを指定します。レジスター表記を使用してストリングのアドレスを指定することはできません。

name16
16 バイトの値を格納する場所の名前。

string 間に空白がない文字ストリング。長さが 16 バイトでない場合は、ブランクで拡張されるか、必要に応じて切り捨てられます。

"string"
引用符で囲まれた文字ストリング。囲まれたストリング内ではブランクを使用できます。プログラム内で名前 (ラベル) を文書化する場合は、この形式を使用します。

注**:** ADD_SUSPEND の RESOURCE_NAME は、SUSPEND 呼び出しで RESOURCE_NAME が指定されていない場合に使用されるデフォルト値を提供します。

RESOURCE_TYPE(name8 | string | "string")
処理の中断および再開に関係するリソースの文書化およびトレースに使用できる 8 文字のストリングを指定します。レジスター表記を使用してストリングのアドレスを指定することはできません。

name8
8 バイトの値を格納する場所の名前。

string 間に空白がない文字ストリング。長さが 8 バイトでない場合は、ブランクで拡張されるか、必要に応じて切り捨てられます。

"string"
引用符で囲まれた文字ストリング。囲まれたストリング内ではブランクを使用できます。プログラム内で名前 (ラベル) を文書化する場合は、この形式を使用します。

注**:** ADD_SUSPEND の RESOURCE_TYPE は、SUSPEND 呼び出しで RESOURCE_TYPE が指定されていない場合に使用されるデフォルト値を提供します。

SUSPEND_TOKEN(name4 | (Rn))
タスクで使用される SUSPEND/RESUME ペアの操作を識別するためにシステムで割り当てるトークンを返します。

name4
トークンを格納する 4 バイトのフィールドの名前。

(Rn) トークンの値がロードされるレジスター。

ADD_SUSPEND の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	なし
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

注: 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。

CHANGE_PRIORITY 呼び出し

CHANGE_PRIORITY を使用すると、発行元タスクが自身の優先順位を変更することができます。これを使用して、別のタスクの優先順位を変更することはできません。このコマンドを実行すると、発行元のタスクは制御を放棄して、他のタスクに実行の機会を提供します。

CHANGE_PRIORITY

```
DFHDSATX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(CHANGE_PRIORITY),
      PRIORITY(name1 | (Rn) | decimalint | literalconst),]
      [OUT,
      [OLD_PRIORITY(name1 | (Rn)),]
      RESPONSE(name1 | *),
      REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

OLD_PRIORITY(name1 | (Rn))
発行元タスクの前の優先順位を返します。

name1
タスクの前の優先順位を格納する 1 バイトのフィールドの名前。

(Rn) 下位バイトで前の優先順位の値を受け取り、他のバイトはゼロに設定されるレジスター。

PRIORITY(name1 | (Rn) | decimalint | literalconst)
発行元タスクに割り当てる新規優先順位を指定します。

name1
0 から 255 までの範囲の値を持つ 1 バイトのフィールドの名前。

(Rn) 新規優先順位の値が入る下位バイトを持つレジスター。

decimalint
値が 255 バイトを超えない 10 進数の整数。式も 16 進表記も使用できません。

literalconst
リテラルの形式の数値。例えば、B'00000000'、X'FF'、X'FCF4'、"0"、または類似の値を持つ等価シンボル。

CHANGE_PRIORITY の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
DISASTER	なし
INVALID	なし
KERNERROR	なし

注: 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。

DELETE_SUSPEND 呼び出し

DELETE_SUSPEND は、このタスクに関連付けられている中断トークンを解放します。

DELETE_SUSPEND

```
DFHDSSRX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(DELETE_SUSPEND),
      SUSPEND_TOKEN(name4 | (Rn)),]
      [OUT,
      RESPONSE(name1 | *),
      REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

SUSPEND_TOKEN(name4 | (Rn))
タスクで使用される SUSPEND/RESUME ペアの操作を識別するためにシステムで割り当てるトークンを指定します。

name4
ADD_SUSPEND 呼び出しによって取得されたトークンを格納する 4 バイトのフィールドの名前。

(Rn) 直前に取得されたトークンの値が入るレジスター。

DELETE_SUSPEND の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	なし
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

注: 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。

RESUME 呼び出し

RESUME は、中断したタスクまたはタイムアウトになったタスクの実行を再開します。

各 SUSPEND に対して RESUME 要求は 1 つだけでなければなりません。ただし、これは非同期インターフェースであるため、SUSPEND は、対応する RESUME の前または後に受け取ることができます。出口プログラムから発行される SUSPEND 要求および RESUME 要求のアカウントを確実に保持する必要があります。

RESUME

```
DFHDSSRX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(RESUME),
      SUSPEND_TOKEN(name4 | (Rn)),
      [COMPLETION_CODE(name1 | (Rn)),]]
      [OUT,
      RESPONSE(name1 | *),
      REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

COMPLETION_CODE(name1 | (Rn))
処理の中断および再開時の、ユーザー定義の「RESUME の理由」コードを指定します。

name1
コードを受け取る 1 バイトの領域の名前。

(Rn) 下位バイトに完了コードが入り、他のバイトはゼロに設定されるレジスター。

SUSPEND_TOKEN(name4 | (Rn))
タスクで使用される SUSPEND/RESUME ペアの操作を識別するためにシステムで割り当てるトークンを指定します。

name4
ADD_SUSPEND 呼び出しの出力として直前に取得された 4 バイトのトークンがある場所の名前。

(Rn) トークンの値が入るレジスター。

RESUME の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	TASK_CANCELLED TIMED_OUT
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

注:

1. 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。
2. 「TASK_CANCELLED」は、タスクが中断状態のときにオペレーターの処置によって取り消されたこと、および中断状態のトークンが使用可能であることを示します。

SUSPEND 呼び出し

SUSPEND は、実行中のタスクの実行を中断します。

中断状態のタスクは 2 つの方法の内のいずれかで再開できます。XPI RESUME 呼び出しを発行して再開できます。あるいは、DFHDSSRX マクロで指定した INTERVAL 値の有効期限が切れた場合は、タスクが自動的に再開されます。オペレーター、アプリケーション、またはデッドロック・タイムアウト機能によって、中断状態のタスクをパージすることもできます。

SUSPEND

```
DFHDSSRX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(SUSPEND),
      PURGEABLE(YES|NO),
      SUSPEND_TOKEN(name4 | (Rn)),
      [INTERVAL(name4 | (Rn)),]
      [RESOURCE_NAME(name16 | string | 'string'),]
      [RESOURCE_TYPE(name8 | string | 'string'),]
      [TIME_UNIT(SECOND|MILLI_SECOND),]
      [WLM_WAIT_TYPE,]]
      [OUT,
      [COMPLETION_CODE(name1 | (Rn)),]
      RESPONSE(name1 | *),
      REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

COMPLETION_CODE (name1 | (Rn))
処理の中断および再開時に、ユーザー定義の「アクションの理由」コードを返します。

name1
コードを受け取る 1 バイトの領域の名前。このフィールドの値はユーザー定義であり、CICS では無視されます。

(Rn) 下位バイトに完了コードが入り、他のバイトはゼロに設定されるレジスター。

INTERVAL(name4 | (Rn))
タスクが自動的に再開される時刻 (RESPONSE 値 PURGED と REASON 値 TIMED_OUT を有する時間) を秒単位またはミリ秒単位で指定します。INTERVAL オプションで使用される時間単位は、TIME_UNIT オプションの設定によって決まります。INTERVAL の値は、トランザクションに指定されているタイムアウト (DTIMOUT) の値をオーバーライドします。

name4
バイナリー・フルワードとして解釈される 4 バイトの領域の名前。

(Rn) 間隔の値 (バイナリー・フルワード) が入るレジスター。

PURGEABLE(YES|NO)

パージの結果として異常終了した要求をコードで処理できるかどうかを指定します。表 1 に示されているような 4 タイプのパージがあります。

PURGEABLE(NO) を指定すると、ディスパッチャーに以下のように通知されます。

- タスクを PURGE する試みを拒否する。
- この要求の期間はデッドロック・タイムアウト (DTIMOUT) 機能 (このタスクに該当する場合) を抑止する。

表 *1. SUSPEND* 呼び出し *- RESPONSE(PURGED)*

REASON	CONDITION	PURGEABLE (NO)	PURGEABLE (YES)
TASK_CANCELLED	PURGE	取り消し	正常に進行
	FORCEPURGE	正常に進行	正常に進行
TIMED_OUT	DTIMOUT	取り消し	正常に進行
	INTERVAL	正常に進行	正常に進行

注**:** FORCEPURGE では常に、ユーザーがタスクをパージする必要があるものと見なして、PURGEABLE(NO) オプションをオーバーライドします。ユーザーが INTERVAL を設定した場合、このオプションも PURGEABLE(NO) オプションをオーバーライドします。

RESOURCE_NAME(name16 | string | "string")

処理の中断および再開に関係するリソースの文書化およびトレースに使用できる 16 文字のストリングを指定します。レジスター表記を使用してストリングのアドレスを指定することはできません。

name16
16 バイトの値を格納する場所の名前。

string 間に空白がない文字ストリング。長さが 16 バイトでない場合は、ブランクで拡張されるか、必要に応じて切り捨てられます。

"string"
引用符で囲まれた文字ストリング。囲まれたストリング内ではブランクを使用できます。プログラム内で名前 (ラベル) を文書化する場合は、この形式を使用します。

注**:**

1. CICS は、RESOURCE_NAME 情報を使用しませんが、トレース・エントリーにその情報を組み込み、適切な CEMT 画面に表示して、どのタスクが実行されているかをユーザーが確認できるようにします。CICS 内部では指定値を要求するため、あいまいさを回避するために、ユーザーは別の値を使用する必要があります。CICS 内部要求値については、資料「トラブルシューティング」の『CICS タスクが待機する可能性があるリソース』に説明があります。
2. RESOURCE_NAME が指定されていない場合は、ADD_SUSPEND のデフォルト値 (ある場合) が使用されます。

RESOURCE_TYPE(name8 | string | "string")

処理の中断および再開に関係するリソースの文書化およびトレースに使用できる 8 文字のストリングを指定します。レジスター表記を使用してストリングのアドレスを指定することはできません。

name8
8 バイトの値を格納する場所の名前。

string 間に空白がない文字ストリング。長さが 8 バイトでない場合は、ブランクで拡張されるか、必要に応じて切り捨てられます。

"string"
引用符で囲まれた文字ストリング。囲まれたストリング内ではブランクを使用できます。プログラム内で名前 (ラベル) を文書化する場合は、この形式を使用します。

注:

1. CICS は、RESOURCE_TYPE 情報を使用しませんが、トレース・エントリーにその情報を組み込み、適切な CEMT 画面に表示して、どのタスクが実行されているかをユーザーが確認できるようにします。CICS 内部では指定値を要求するため、あいまいさを回避するために、ユーザーは別の値を使用する必要があります。CICS 内部要求値については、資料「トラブルシューティング」の『CICS タスクが待機する可能性があるリソース』に説明があります。
2. RESOURCE_TYPE が指定されていない場合は、ADD_SUSPEND のデフォルト値 (ある場合) が使用されます。

SUSPEND_TOKEN(name4 | (Rn))

タスクで使用される SUSPEND/RESUME ペアの操作を識別するためにシステムで割り当てるトークンを指定します。

name4
ADD_SUSPEND 呼び出しの出力として直前に取得された 4 バイトのトークンがある場所の名前。

(Rn) トークンの値が入るレジスター。

TIME_UNIT(SECOND | MILLI_SECOND)

INTERVAL オプションで使用する時間単位を指定します。

SECOND
INTERVAL オプションは、タイムアウトになるまでの秒数を指定します。

MILLI_SECOND
INTERVAL オプションは、タイムアウトになるまでのミリ秒数を指定します。

WLM_WAIT_TYPE(name1)

タスクの中断理由を 1 バイトの位置に指定します。この理由は、MVS™ ワークロード・マネージャーに対する待ち状態の性質を示します。

待機タイプの等価の値は、以下のとおりです。

CMDRESP
コマンド応答を待機。

CONV
会話を待機。

DISTRIB
分散要求を待機。

IDLE
CICS タスクは、作業マネージャーとして機能し、モニター環境内での保守が可能な作業要求はありません。例えば、実行するジャーナリング入出力操作がない場合に、ジャーナリング自体を中断するジャーナリング・コードです。

IO 入出力操作または不確定な入出力関連の操作 (ロック、バッファー、ストリングなど) で待機。

LOCK
ロックでの待機。

MISC
未定義リソースを待機。

注: タスクを中断して、かつ WLM_WAIT_TYPE パラメーターを指定しない場合、この値が待機のデフォルトの理由として指定されています。

OTHER_PRODUCT
機能を完了させるために別の製品を待機。例えば、ワークロードが Db2® に渡された場合。

SESS_LOCALMVS
この CICS 領域を実行している MVS イメージ内のセッションの確立待ち。

SESS_NETWORK
ネットワーク内の他の場所の (つまり、この MVS イメージ上にない) セッションの確立待ち。

SESS_SYSPLEX
シスプレックス内の他の場所の (つまり、この MVS イメージ上にない) セッションの確立待ち。

TIMER
タイマー (例えば、タスク自体をスリープ状態にするタスク) のタイムアウトを待機。

CICS を MVS ゴール・モードのワークロード管理環境で実行している場合 (つまり、ゴール指向のパフォーマンス管理を使用している場合) は、WLM_WAIT_TYPE パラメーターでタスクの中断理由を指定してください。

表 2. *SUSPEND* の *RESPONSE* 値および *REASON* 値

RESPONSE	REASON
OK	なし
EXCEPTION	なし
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	TASK_CANCELLED

表 2. *SUSPEND* の *RESPONSE* 値および *REASON* 値 (続き)

RESPONSE	REASON
	TIMED_OUT

注:

1. 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。
2. TASK_CANCELLED は、オペレーターの処置またはアプリケーション・コマンドによってタスクが取り消されたことを示します。
3. PURGED 応答の後、パージされた SUSPEND に対応する RESUME によってリセットされるまで、中断状態のトークンを別の SUSPEND で再使用しないでください。
4. TIMED_OUT は、指定された INTERVAL (またはタスク接続時に指定されたタイムアウト値) の有効期限が切れたために自動的に再開されたことを示します。ただし、トークンは中断状態のままであるため、DELETE_SUSPEND の対象とするには、その前に、RESUME の対象にする必要があります。

WAIT_MVS 呼び出し

WAIT_MVS は、MVS イベント制御ブロック (ECB) または MVS ECB のリストで待機を要求します。たとえば、WAIT_MVS を発行すると、ATTACH を発行してタスク完了 ECB を提供した MVS タスクの完了を待機することができます。

ディスパッチャーは、WAIT_MVS 要求の受信時に ECB をクリアしません。ECB が既にポストされている場合、制御はすぐに出口プログラムに戻り、「OK」の応答を返します。

単一の ECB を、一度に複数の待機の対象にしないでください。WAIT_MVS 要求の受信時に既に 待機中の ECB がある場合、要求が拒否されます。RESPONSE コードは「DSSR_INVALID」で、REASON コードは「DSSR_ALREADY_WAITING」です。

注: WAIT_MVS 要求で使用される ECB は、常に MVS POST マクロを使用してポストする必要があります。

WAIT_MVS

```
DFHDSSRX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(WAIT_MVS),
      {ECB_ADDRESS(name4 | (Ra)) | ECB_LIST_ADDRESS(name4 | (Ra)),}
      PURGEABLE(YES|NO),
      [INTERVAL(name4 | (Rn)),]
      [RESOURCE_NAME(name16 | string | 'string'),]
      [RESOURCE_TYPE(name8 | string | 'string'),]]
      [TIME_UNIT(SECOND|MILLI_SECOND),]
      [WLM_WAIT_TYPE,]
      [OUT,
      RESPONSE(name1 | *),
      REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

ECB_ADDRESS(name4 | (Ra))
待機する対象の ECB のアドレスを指定します。

name4
ECB アドレスが入る場所の名前。

(Ra) ECB のアドレスが入るレジスター。

ECB_LIST_ADDRESS(name4 | (Ra))
待機する対象の ECB アドレスのリストのアドレスを指定します。

name4
ECB アドレスが入る場所の名前。多くの場合、さらに多くの ECB アドレスが続きます。リスト内の最後のアドレス・ワードでは、上位ビットが 1 に設定されます。

(Ra) 上で説明したアドレス・リストを指すレジスター。

INTERVAL(name4 | (Rn))
タスクが自動的に再開される時刻 (RESPONSE 値「PURGED」と REASON 値「TIMED_OUT」を有する時間) を秒単位またはミリ秒単位で指定します。INTERVAL オプションで使用される時間単位は、TIME_UNIT オプションの設定によって決まります。

INTERVAL の値は、トランザクションに指定されているタイムアウト (DTIMOUT) の値をオーバーライドします。

name4
バイナリー・フルワードとして解釈される 4 バイトの領域の名前。

(Rn) 間隔の値 (バイナリー・フルワード) が入るレジスター。

PURGEABLE(YES|NO)
パージの結果として異常終了した要求をコードで処理できるかどうかを指定します。表 3 に示されているような 4 タイプのパージがあります。PURGEABLE(NO) を指定すると、ディスパッチャーに以下のように通知されます。

- タスクを PURGE する試みを拒否する
- この要求の期間はデッドロック・タイムアウト (DTIMOUT) 機能 (このタスクに該当する場合) を抑止する

表 *3. SUSPEND 呼び出し - RESPONSE(PURGED)*

REASON	CONDITION	PURGEABLE (NO)	PURGEABLE (YES)
TASK_CANCELLED	PURGE	取り消し	正常に進行
	FORCEPURGE	正常に進行	正常に進行
TIMED_OUT	DTIMOUT	取り消し	正常に進行
	INTERVAL	正常に進行	正常に進行

注: FORCEPURGE では常に、ユーザーがタスクをパージする必要があるものと見なして、PURGEABLE(NO) オプションをオーバーライドします。ユーザーが INTERVAL を設定した場合、このオプションも PURGEABLE(NO) オプションをオーバーライドします。

RESOURCE_NAME(name16 | string | "string")

処理の中断および再開に関係するリソースの文書化およびトレースに使用できる 16 文字のストリングを指定します。レジスター表記を使用してストリングのアドレスを指定することはできません。

name16
16 バイトの値を格納する場所の名前。

string 間に空白がない文字ストリング。長さが 16 バイトでない場合は、ブランクで拡張されるか、必要に応じて切り捨てられます。

"string"
引用符で囲まれた文字ストリング。囲まれたストリング内ではブランクを使用できます。プログラム内で名前 (ラベル) を文書化する場合は、この形式を使用します。

注: CICS は、RESOURCE_NAME 情報を使用しませんが、トレース・エントリーにその情報を組み込み、適切な CEMT 画面に表示して、どのタスクが実行されているかをユーザーが確認できるようにします。CICS 内部では指定値を要求するため、あいまいさを回避するために、ユーザーは別の値を使用する必要があります。CICS 内部要求値については、資料「トラブルシューティング」の『CICS タスクが待機する可能性があるリソース』に説明があります。

RESOURCE_TYPE(name8 | string | "string")

処理の中断および再開に関係するリソースの文書化およびトレースに使用できる 8 文字のストリングを指定します。レジスター表記を使用してストリングのアドレスを指定することはできません。

name 8 バイトの値を格納する場所の名前。

string 間に空白がない文字ストリング。長さが 8 バイトでない場合は、ブランクで拡張されるか、必要に応じて切り捨てられます。

"string"
引用符で囲まれた文字ストリング。囲まれたストリング内ではブランクを使用できます。プログラム内で名前 (ラベル) を文書化する場合は、この形式を使用します。

注: CICS は、RESOURCE_TYPE 情報を使用しませんが、トレース・エントリーにその情報を組み込み、適切な CEMT 画面に表示して、どのタスクが実行されているかをユーザーが確認できるようにします。CICS 内部では指定値を要求するため、あいまいさを回避するために、ユーザーは別の値を使用する必要があります。CICS 内部要求値については、資料「トラブルシューティング」の『CICS タスクが待機する可能性があるリソース』に説明があります。

TIME_UNIT(SECOND | MILLI_SECOND)

INTERVAL オプションで使用する時間単位を指定します。

SECOND
INTERVAL オプションは、タイムアウトになるまでの秒数を指定します。

MILLI_SECOND
INTERVAL オプションは、タイムアウトになるまでのミリ秒数を指定します。

WLM_WAIT_TYPE(name1)

タスクの中断理由を 1 バイトの位置に指定します。これは、MVS ワークロード・マネージャーに対するタスクの待ち状態の性質を示します。

待機タイプの等価の値は、以下のとおりです。

CMDRESP
コマンド応答を待機。

CONV
会話を待機。

DISTRIB
分散要求を待機。

IDLE
CICS タスクは、作業マネージャーとして機能し、モニター環境内での保守が可能な作業要求はありません。例えば、実行するジャーナリング入出力操作がない場合に、ジャーナリング自体を中断するジャーナリング・コードです。

IO 入出力操作または不確定な入出力関連の操作 (ロック、バッファー、ストリングなど) で待機。

LOCK
ロックでの待機。

MISC
未定義リソースを待機。タスクを中断して、かつ WLM_WAIT_TYPE パラメーターを指定しない場合、これが待機のデフォルトの理由として指定されています。

OTHER_PRODUCT
機能を完了させるために別の製品を待機。例えば、ワークロードが Db2 に渡された場合。

SESS_LOCALMVS
この CICS 領域を実行している MVS イメージ内のセッションの確立待ち。

SESS_NETWORK
ネットワーク内の他の場所の (つまり、この MVS イメージ上にない) セッションの確立待ち。

SESS_SYSPLEX
シスプレックス内の他の場所の (つまり、この MVS イメージ上にない) セッションの確立待ち。

TIMER
タイマー (例えば、タスク自体をスリープ状態にするタスク) のタイムアウトを待機。

CICS を MVS ゴール・モードのワークロード管理環境で実行している場合 (つまり、ゴール指向のパフォーマンス管理を使用している場合) は、WLM_WAIT_TYPE パラメーターでタスクの中断理由を指定することをお勧めします。

表 4. *WAIT_MVS* の *RESPONSE* 値および *REASON* 値

RESPONSE	REASON
OK	なし
EXCEPTION	なし
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	TASK_CANCELLED
	TIMED_OUT

注:

1. 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。
2. TIMED_OUT は、INTERVAL の有効期限が切れた場合、またはデッドロック・タイムアウト間隔の有効期限が切れた場合に返されます。
3. TASK_CANCELLED は、オペレーターの処置またはアプリケーション・コマンドによってタスクが取り消されたことを示します。

第 4 章 ダンプ管理 XPI 関数

XPI は、2 つのダンプ管理関数を提供します。これらは、DFHDUDUX マクロ呼び出し SYSTEM_DUMP および TRANSACTION_DUMP です。

制約事項: 以下のグローバル・ユーザー出口点から呼び出された出口プログラムでは、DFHDUDUX 呼び出しを使用することはできません。

- 統計ドメイン
- モニター・ドメイン
- ダンプ・ドメイン
- ディスパッチャー・ドメイン
- 一時データ・プログラム

SYSTEM_DUMP 呼び出し

SYSTEM_DUMP を使用すると、システム・ダンプが取られます。入力で提供したシステム・ダンプ・コードがシステム・ダンプ・コード・テーブル内にある場合は、ダンプを抑止することができます。

ダンプ・テーブルとその動作方法については、Using dumps in problem determinationおよびSET SYSDUMPCODEを参照してください。

SYSTEM_DUMP

```
DFHDUDUX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(SYSTEM_DUMP),
      SYSTEM_DUMPCODE(name8 | string | "string"),
      [CALLER(block-descriptor),]
      [TITLE(block-descriptor),]]
      [OUT,
      DUMPID(name9 | *),
      RESPONSE(name1 | *),
      REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

CALLER(block-descriptor)
システム・ダンプ要求のソースを指定します。ここで指定した情報はダンプ・ヘッダーに表示されるので、システム・ダンプ要求を開始した出口プログラムを識別するために使用することができます。有効なブロック記述子の説明については、XPI 構文を参照してください。

DUMPID(name9 | *)
ダンプ ID を返します。

name9
割り当て済み ID を受け取る 9 バイトのフィールドの名前。

© Copyright IBM Corp. 1977, 2018

SYSTEM_DUMPCODE(name8 | string | "string")
このシステム・ダンプ呼び出しの原因となったエラーに対応するコードを指定します。システム・ダンプ・コードは、ダンプ・テーブルに保持されます。

name8
8 バイトのストリングが入る場所の名前。

string ブランクが介在しない文字ストリング。マクロは、ストリングから長さ 8 バイトのリテラル定数を生成し、ブランクで拡張したり、必要に応じて切り捨てたりします。

"string"
引用符で囲まれたストリングで、ブランクが入ることもあります。この値は、前述の「string」と同じ方法で処理されます。

TITLE(block-descriptor)
システム・ダンプを印刷するときに、ダンプ・ヘッダーに表示するテキストが入る領域を指定します。

SYSTEM_DUMP の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	FESTAE_FAILED
	INSUFFICIENT_STORAGE
	IWMWQWRK_FAILED
	NO_DATASET
	PARTIAL_SYSTEM_DUMP
	SDUMP_BUSY
	SDUMP_FAILED
	SDUMP_NOT_AUTHORIZED
	SUPPRESSED_BY_DUMPOPTION
	SUPPRESSED_BY_DUMPTABLE
	SUPPRESSED_BY_USEREXIT
DISASTER	なし
INVALID	INVALID_DUMPCODE
	INVALID_PROBDESC
	INVALID_SVC_CALL
KERNERROR	なし
PURGED	なし

注: 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。

TRANSACTION_DUMP 呼び出し

TRANSACTION_DUMP を使用すると、トランザクション・ダンプが取られます。入力で提供したトランザクション・ダンプ・コードがトランザクション・ダンプ・コード・テーブル内にある場合は、ダンプを抑止して、必要に応じてシステム・ダンプを取ることができます。

ダンプ・テーブルとその動作方法については、Using dumps in problem determinationおよびSET TRANDUMPCODEを参照してください。

有効な文字は、大文字 (A から Z)、小文字 (a から z)、数字 (0 から 9)、および特殊文字 $ @ # / % & ? ! : ¦ ; , ¢ + * ¬ - および _ です。場合によっては、設定する位置に応じて、文字 < > . = および " も有効です。小文字を入力した場合は、大文字に変換されます。

重要

初期設定段階での XPI の使用には制限があります。PLTPI の第 2 フェーズまでは、XPI 関数の
TRANSACTION_DUMP、WRITE_JOURNAL_DATA、MONITOR、および INQUIRE_MONITOR_DATA を使用する出口プログラムを開始しないでください。PLTPI について詳しくは、Writing initialization and shutdown programsを参照してください。

TRANSACTION_DUMP

```
DFHDUDUX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(TRANSACTION_DUMP),
      TRANSACTION_DUMPCODE(name4 | string | 'string')
      [CSA(NO|YES),]
      [PROGRAM(NO|YES),]
      [SEGMENT(block-descriptor),]
      [SEGMENT_LIST(block-descriptor),]
      [TCA(NO|YES),]
      [TERMINAL(NO|YES),]
      [TRANSACTION(NO|YES),]
      [TRT(NO|YES),]]
      [OUT,
      DUMPID(name9 | *),
      RESPONSE(name1 | *),
      REASON(name1 | *)]
```

注: このコマンドは、スレッド・セーフではありません。

CSA(NO|YES)
共通システム域 (CSA) をトランザクション・ダンプに組み込むかどうかを指定します。デフォルトは NO です。

DUMPID(name9 | *)
ダンプ ID を返します。

name9
割り当て済み ID を受け取る 9 バイトのフィールドの名前。

PROGRAM(NO|YES)
このタスクに関連付けられているすべてのプログラム・ストレージ域をトランザクション・ダンプに組み込むかどうかを指定します。デフォルトは NO です。

SEGMENT(block-descriptor)
ダンプするストレージの単一ブロックのアドレスと長さを指定します。有効なブロック記述子の説明については、XPI 構文を参照してください。SEGMENT と SEGMENT_LIST は同時には使用できません。

SEGMENT_LIST(block-descriptor)
連続する対語のセット のアドレスと長さを指定します。各組みの最初のワードは、ダンプするストレージ・セグメントの長さをバイトで指定します。2 番目の

ワードには、ストレージ・セグメントのアドレスが入ります。リストの末尾は、X'FFFFFFFF' を含むワードでマークを付ける必要があります。SEGMENT と SEGMENT_LIST は同時には使用できません。

TCA(NO|YES)
タスク制御域 (TCA) をトランザクション・ダンプに組み込むかどうかを指定します。デフォルトは NO です。

TERMINAL(NO|YES)
このタスクに関連付けられているすべての端末ストレージ域をトランザクション・ダンプに組み込むかどうかを指定します。デフォルトは NO です。

TRANSACTION(NO|YES)
このタスクに関連付けられているすべてのトランザクション・ストレージ域をトランザクション・ダンプに組み込むかどうかを指定します。デフォルトは NO です。

TRANSACTION_DUMPCODE(name4 | string | "string")
このトランザクション・ダンプ呼び出しの原因となったエラーに対応するコードを指定します。トランザクション・ダンプ・コードは、ダンプ・テーブルに保持されます。

name4
4 バイトのストリングが入る場所の名前。

string
ブランクが介在しない文字ストリング。マクロは、ストリングから長さ 4 バイトのリテラル定数を生成し、ブランクで拡張したり、必要に応じて切り捨てたりします。

"string"
引用符で囲まれたストリングで、ブランクが入ることもあります。この値は、前述の「string」と同じ方法で処理されます。

TRT(NO|YES)
トレース・テーブル (TRT) をトランザクション・ダンプに組み込むかどうかを指定します。デフォルトは NO です。

TRANSACTION_DUMP の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	FESTAE_FAILED
	INSUFFICIENT_STORAGE
	IWMWQWRK_FAILED
	NOT_OPEN
	OPEN_ERROR
	PARTIAL_SYSTEM_DUMP
	PARTIAL_TRANSACTION_DUMP
	SDUMP_BUSY
	SDUMP_FAILED
	SDUMP_NOT_AUTHORIZED
	SUPPRESSED_BY_DUMPOPTION
	SUPPRESSED_BY_DUMPTABLE
	SUPPRESSED_BY_USEREXIT
DISASTER	なし
INVALID	INVALID_DUMPCODE

RESPONSE	*REASON*
	INVALID_PROBDESC
	INVALID_SVC_CALL
KERNERROR	なし
PURGED	なし

注**:**

1. 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。
2. NOT_OPEN は、CICS ダンプ・データ・セットが閉じていることを示します。
3. OPEN_ERROR は、CICS ダンプ・データ・セットを開く途中でエラーが発生したことを示します。
4. PARTIAL は、この要求から生じるトランザクション・ダンプが不完全であることを示します。

第 5 章 エンキュー・ドメイン XPI 関数

XPI は、2 つのエンキュー・ドメイン関数を提供します。これらは、DFHNQEDX 呼び出し DEQUEUE および ENQUEUE です。

DEQUEUE 機能

DEQUEUE 機能は、DFHNQEDX マクロ呼び出しで提供されます。これは、ENQUEUE 機能呼び出しによって以前にエンキューされたリソースを解放します。

DEQUEUE

```
DFHNQEDX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(DEQUEUE),
      {ENQUEUE_TOKEN(name4),|
      ENQUEUE_NAME1(address,length),[ENQUEUE_NAME2(address,length),]}
      MAX_LIFETIME(DISPATCHER_TASK),]
     [ENQUEUE_TYPE (XPI | EXECSTRN | EXECADDR),]
      [OUT,
      RESPONSE (name1 | *),
      REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

ENQUEUE_TOKEN、ENQUEUE_NAME1、ENQUEUE_NAME2、MAX_LIFETIME (DISPATCHER_TASK)、および ENQUEUE_TYPE (XPI | EXECSTRN | EXECADDR) の各パラメーターは、ENQUEUE 関数呼び出しの場合と同じです。

DEQUEUE の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	ENQUEUE_NOT_OWNED ENQUEUE_LOCKED

ENQUEUE 関数

ENQUEUE 関数は、DFHNQEDX マクロ呼び出しで提供されます。これにより、指定のリソース上でのエンキューが可能になります。

デフォルトでは、XPI ENQUEUE コマンドによって作成されたすべてのエンキューは、DISPATCH という特定のエンキュー・プールに割り振られ、CICS 内部処理として扱われます。**EXEC CICS ENQ** コマンドによって作成されたエンキューは、指定されたエンキュー・モデルに応じて異なるエンキュー・プールに追加されるため、それらのエンキューと XPI エンキューが競合することはありません。例えば、ストリング上にアクティブな EXEC CICS ENQ がある場合でも、同じストリング上での XPI ENQUEUE コマンドの取得を妨げることはありません。

© Copyright IBM Corp. 1977, 2018

注:
- XPI エンキューを CICS SPI を使用して参照することはできません。
- XPI エンキューを ENQMODEL を使用して制御することはできません。

オプションの **ENQUEUE_TYPE** パラメーターを使用すると、XPI ENQUEUE コマンドは、EXEC CICS ENQ によってエンキューされているのと同じリソース上にエンキューすることができます。その逆も可能です。アプリケーションは、EXEC CICS コマンドと EXEC XPI コマンドを使用して処理を同期化できます。

ENQUEUE

```
DFHNQEDX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(ENQUEUE),
      ENQUEUE_NAME1(address,length),
      [ENQUEUE_NAME2(address,length),]
      MAX_LIFETIME(DISPATCHER_TASK),
     [ENQUEUE_TYPE (XPI | EXECSTRN | EXECADDR),]
      [WAIT(YES|NO),]
      [PURGEABLE(YES|NO),]]
      [OUT,
      [ENQUEUE_TOKEN(name4),]
      [DUPLICATE_REQUEST,]
      RESPONSE (name1 | *),
      REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

DUPLICATE_REQUEST
要求側ディスパッチャー・タスクが既にエンキューされているリソースを所有していることを示します。

ENQUEUE_NAME1(address,length)
エンキュー対象の名前の上位部分を指定します。

ENQUEUE_NAME2(address,length)
エンキュー対象の名前の下位部分 (ある場合) を指定します。

ENQUEUE_TOKEN(name4)
後続の DEQUEUE 要求が、エンキュー名ではなくトークンでリソースを識別できるようにし、NQ ドメインで直接エンキュー制御ブロックを配置できるようにします。これにより、効率が向上します。

ENQUEUE_TYPE (XPI | EXECSTRN | EXECADDR)
エンキューするリソースのタイプを指定します。XPI オプションは、標準的な DFHNQEDX 動作を指定します。使用されるリソース・プールは XPI 専用で、CICS API がアクセスすることはできません。 ENQUEUE_NAME1 が、EXEC CICS ENQ により使用されているものと同じ名前空間にあるエンキュー・リソースを指定するように指示するには、EXECSTRN または EXECADDR を使用します。EXECSTRN および EXECADDR について詳しくは、「トラブルシューティング」の『CICS タスクが待機する可能性があるリソース』を参照してください。

MAX_LIFETIME(DISPATCHER_TASK)
MAX_LIFETIME(DISPATCHER_TASK) は必須で、これは、すべての XPI エンキューが要求ディスパッチャー・タスクによって所有されるように指定します。

ENQUEUE XPI 呼び出しを使用して、グローバル・ユーザー出口プログラムがスレッド・セーフであることを確認する場合、リソースがエンキューされたグローバル・ユーザー出口プログラムの呼び出しの間は、そのリソースを解放 (デキュー) することをお勧めします。ただし、グローバル・ユーザー出口を停止するためのリカバリー・サービスは提供されていないため、CICS によって、ディスパッチャー・タスクの終了時に未処理の XPI エンキューが自動的にデキューされるようになります。ディスパッチャー・タスクが CICS トランザクションを実行している場合、CICS トランザクションが終了したときには、それが正常終了であろうと異常終了であろうと、ディスパッチャー・タスクは終了します。

通常、エンキューは作業単位（UOW）を含む要求トランザクションによって所有されます。これらの作業単位はエンキュー制御ブロックの固定に使用されます。ただし、XPI はトランザクション環境に必須のものではなく、トランザクションや UOW を持たないディスパッチャー・タスク下でグローバル・ユーザー出口を呼び出すことができます。

PURGEABLE(YES|NO)
要求ディスパッチャー・タスクがエンキューを待機しなければならない場合に、受け入れられるタスクに対する要求をパージ (またはタイムアウト) するかどうかを指定します。

WAIT(YES|NO)
リソースが現在別のディスパッチャー・タスクにエンキューされている場合に、ディスパッチャー・タスクが待機するかどうかを指定します。

ENQUEUE の RESPONSE 値および REASON 値

RESPONSE	REASON
OK	なし
EXCEPTION	
	ENQUEUE_BUSY
	ENQUEUE_LOCKED
	ENQUEUE_DISABLED
	LIMIT_EXCEEDED
	SYSENQ_FAILURE
	INVALID_PHASE
PURGED	
	TASK_CANCELLED
	TIMED_OUT

第 6 章 カーネル・ドメイン XPI 関数

XPI は、2 つのカーネル・ドメイン関数を提供します。これらは、DFHKEDSX 呼び出しの START_PURGE_PROTECTION および STOP_PURGE_PROTECTION です。

START_PURGE_PROTECTION 関数

START_PURGE_PROTECTION 関数は、DFHKEDSX マクロ呼び出しで提供されます。これは、現行タスクに対するパージを禁止します。ただし、強制パージは禁止しません。この関数は、グローバル・ユーザー出口呼び出し時のパージを禁止するために、すべてのグローバル・ユーザー出口プログラムで使用することができます。

一般的に、各 START_PURGE_PROTECTION 呼び出しには、このような保護が必要なプログラム・ロジックの完了時にパージ保護期間を終了するために、対応する STOP_PURGE_PROTECTION 関数呼び出しが必要です。

START_PURGE_PROTECTION

```
DFHKEDSX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION(START_PURGE_PROTECTION),]
       [OUT,
       RESPONSE (name1 | *)]
```

このコマンドはスレッド・セーフです。

この呼び出しには入力パラメーターも出力パラメーターもありません。あるのは RESPONSE のみです。

STOP_PURGE_PROTECTION 機能

STOP_PURGE_PROTECTION 機能は、DFHKEDSX マクロ呼び出しで提供されます。これは、先行する START_PURGE_PROTECTION 機能呼び出しによってパージが中断された後で、現行タスクに対するパージを再び使用可能にします。

STOP_PURGE_PROTECTION

```
DFHKEDSX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION(STOP_PURGE_PROTECTION),]
       [OUT,
       RESPONSE (name1 | *)]
```

このコマンドはスレッド・セーフです。

この呼び出しには入力パラメーターも出力パラメーターもありません。あるのは RESPONSE のみです。

© Copyright IBM Corp. 1977, 2018

パージ保護呼び出しのネスト

START_PURGE_PROTECTION 機能および STOP_PURGE_PROTECTION 機能はネストできます。1 つのタスクに対して複数の START_PURGE_PROTECTION 呼び出しが発行された場合は、パージ保護を取り消すための STOP_PURGE_PROTECTION 呼び出しの発行回数が正しいことを確認する必要があります。

START を 2 回発行して STOP を 1 回しか発行しなかった場合、現行タスクのパージ保護がオンのままになります。

例えば、どの現行タスクの場合でも、複数のグローバル・ユーザー出口プログラムが呼び出されることがあります。パージ保護が正しく取り消されるように出口プログラムを設計する必要があります。ネストの例を以下に示します。

```
XEIIN:
 EXIT_PROG1: Calls START_PURGE_PROTECTION

   XFCREQ:
    EXIT_PROG2: Calls START_PURGE_PROTECTION

   XFCREQC:
    EXIT_PROG3: Calls STOP_PURGE_PROTECTION

XEIOUT:
 EXIT_PROG4: Calls STOP_PURGE_PROTECTION
```

第 7 章 ローダー XPI 関数

XPI は、5 つのローダー関数を提供します。これらの関数は、DFHLDLDX 呼び出し ACQUIRE_PROGRAM、DEFINE_PROGRAM、DELETE_PROGRAM、IDENTIFY_PROGRAM、および RELEASE_PROGRAM です。

XPI を含む CICS ローダー・サービスは、リンクされた AMODE(64) である非 Language Environment (LE) アセンブラー・プログラムを認識します。モジュールのアドレッシング・モードは、戻されるエントリー・ポイント・パラメーターで示されます。ビット 0 が 0、ビット 31 が 1 である場合、AMODE(64) が示されます (z/OS® オペレーティング・システムで使用されるのと同じアドレッシング・モード規則)。

制約事項: 以下のドメインまたはプログラムのグローバル・ユーザー出口点から呼び出された出口プログラムでは、DFHLDLDX 呼び出しを使用することはできません。

- 統計ドメイン
- モニター・ドメイン
- ダンプ・ドメイン
- ディスパッチャー・ドメイン
- 一時データ・プログラム

ACQUIRE_PROGRAM 呼び出し

ACQUIRE_PROGRAM は、指定されたプログラムの使用可能コピー (その名前またはプログラム・トークンのいずれかによって識別できる) のエントリー・ポイントおよびロード・ポイントのアドレス、長さ、および新規プログラム・トークンを返します。

ACQUIRE_PROGRAM

```
DFHLDLDX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(ACQUIRE_PROGRAM),
      {PROGRAM_NAME(name8 | string | 'string')|
      PROGRAM_TOKEN(name8)},
      [SUSPEND(NO|YES),]]
      [OUT,
      ENTRY_POINT(name4 | (Ra)),
      [LOAD_POINT(name4 | (Ra)),]
      [NEW_PROGRAM_TOKEN(name8),]
      [PROGRAM_ATTRIBUTE(name1 | (Rn)),]
      [PROGRAM_LENGTH(name4 | (Rn)),]
      RESPONSE(name1 | *),
      REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

ENTRY_POINT(name4 | (Ra))
プログラムのエントリー・ポイント・アドレスを返します。

© Copyright IBM Corp. 1977, 2018

name4
31 ビットのエントリー・アドレスを受け取る 4 バイトの位置の名前。

(Ra) エントリー・アドレスを受け取るレジスター。

LOAD_POINT(name4 | (Ra))
プログラムのロード・ポイント・アドレスを返します。

name4
ロードされたアドレスを受け取る 4 バイトの位置の名前。

(Ra) ロード・アドレスが入るレジスター。

NEW_PROGRAM_TOKEN(name8)
指定されたプログラムの使用可能なコピーの新規プログラム・トークンを返します。

name8
このプログラムとインスタンスを識別する 8 バイトのトークンを受け取る場所の名前。

PROGRAM_ATTRIBUTE(name1 | (Rn))
プログラム属性を返します。

name1
プログラム属性を受け取る 1 バイトの位置の名前。

(Rn) 下位バイトでプログラム属性を受け取り、他のバイトはゼロに設定されるレジスター。これには、値 RELOAD、RESIDENT、REUSABLE、または TRANSIENT を指定できます。

RELOAD
このプログラムは再使用可能ではないため、複数のプログラムのコピーがロードされることがあります。
RELEASE_PROGRAM 呼び出しが (コピーに対して) 発行されると、そのコピーはストレージから削除されます。

RESIDENT
削除しない限りストレージから除去されない、プログラムの単一コピーがあります。RESIDENT プログラムは、少なくとも準再入可能である必要があります。デフォルトでは、PROGRAM_TYPE SHARED のプログラムはすべて RESIDENT 属性を持っています。DELETE_PROGRAM 呼び出しは、このタイプの RESIDENT プログラムには影響しません。

REUSABLE
ストレージの最適化を目的として使用されていない REUSABLE プログラムを CICS でストレージから削除できることを除いて、RESIDENT と同様です。

TRANSIENT
使用回数がゼロになるとすぐに TRANSIENT プログラムがストレージから削除されることを除いて、RESIDENT と同様です。

PROGRAM_LENGTH(name4 | (Rn))
指定されたプログラムの長さを返します。

name4
長さをバイト単位で受け取る 4 バイトの位置の名前。バイナリーで表されます。

(Rn) バイト単位の長さが入るレジスター。バイナリーで表されます。

PROGRAM_NAME(name8 | string | "string")
獲得するプログラムの名前を指定します。

name8
8 バイトのプログラム名が入る場所の名前。

string プログラムの名前を指定する文字ストリング。

"string"
引用符で囲まれたストリング。ストリングの長さは、ブランク埋め込みまたは切り捨てによって、8 に設定されます。

PROGRAM_TOKEN(name8),
詳細を獲得するプログラムを識別するトークンを指定します。

name8
前述の DEFINE_PROGRAM 呼び出しまたは ACQUIRE_PROGRAM 呼び出しから取得された 8 バイトのトークンが入る場所の名前。

SUSPEND(NO|YES)
要求が認可されるまで実行を中断するかどうかを指定します。

ACQUIRE_PROGRAM の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	NO_STORAGE
	PROGRAM_NOT_DEFINED
	PROGRAM_NOT_FOUND
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

注**:**

1. 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。
2. RESPONSE「EXCEPTION」を伴う REASON「NO_STORAGE」は、この要求を満たすにはストレージが不十分であり、SUSPEND(NO) が指定されたことを示します。
3. REASON「PROGRAM_NOT_FOUND」は、プログラムがライブラリー連結に組み込まれていない場合、またはリンク・エディットが失敗した場合に返されます。このような場合、プログラムには「実行不可」のマークが付きます。このプログラムを正常に獲得するには、その前に再リンクする必要があります。

DEFINE_PROGRAM 呼び出し

DEFINE_PROGRAM 呼び出しを使用して、ローダー・ドメインに対して新規プログラムを定義するか、または既に定義済みのプログラムの詳細を変更することができます。指定する詳細はローカル・カタログに記録され、即時に使用可能になります。それらは、指定されたプログラムのそれ以降の ACQUIRE 要求すべてに使用されます。

DEFINE_PROGRAM を使用して作成されたプログラム定義は、XRF テークオーバーでは保持されません。また、CSD は更新されず、ローダー・ドメイン定義のみが更新されます。

DEFINE_PROGRAM

```
DFHLDLDX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(DEFINE_PROGRAM),
      PROGRAM_NAME(name8 | string | 'string' ),
      [EXECUTION_KEY(CICS|USER),]
      [PROGRAM_ATTRIBUTE(RELOAD|RESIDENT|REUSABLE|TRANSIENT),]
      [PROGRAM_TYPE(PRIVATE|SHARED|TYPE_ANY),]
      [REQUIRED_AMODE(24|31|AMODE_ANY|64),]
      [REQUIRED_RMODE(24|RMODE_ANY),]]
      [OUT,
      [NEW_PROGRAM_TOKEN(name8),]
      RESPONSE(name1 | *),
      REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

EXECUTION_KEY(CICS|USER)

他のプログラム属性と連携して、ローダーがプログラムをロードする動的ストレージ域 (DSA) のタイプを指定します。

CICS 再入不可プログラムの場合、プログラムは、16 MB 境界より上または下の CICS DSA (つまり CDSA または ECDSA) にロードされます。CICS DSA の選択は、リンケージ・エディターに定義されているとおり、プログラムの常駐モード (RMODE) 属性によって決まります。

再入可能 RMODE(24) プログラムの場合、プログラムは CDSA にロードされます。

USER 再入不可プログラムの場合、プログラムは、16 MB 境界より上または下のユーザー DSA (つまり UDSA または EUDSA) にロードされます。ユーザー DSA の選択は、リンケージ・エディターに定義されているとおり、プログラムの常駐モード (RMODE) 属性によって決まります。

再入可能 RMODE(24) プログラムの場合、プログラムは UDSA にロードされます。

16 MB 境界より上に常駐するのに適格な再入可能プログラム: プログラムが AMODE(31),RMODE(ANY) を指定して再入可能としてリンク・エディットされている場合、EXECUTION_KEY オプションは無視され、読み取り専用 DSA (RDSA または ERDSA) にロードされます。ERDSA に割り振られたストレージのタイプについて詳しくは、RENTPGM システム初期設定パラメーターを参照してください。

他の要因と関係する EXECUTION_KEY オプションの影響については、表 5に要約してあるので、参照してください。

表 5. *DSA* の適格性を定義する属性の要約

EXECUTION_KEY オプション	再入可能	**16 MB** 境界より上または下	動的ストレージ域 **(DSA)**
CICS	いいえ	Below	CDSA
CICS	はい	Below	RDSA
CICS	いいえ	上	ECDSA
CICS	はい	上	ERDSA
USER	いいえ	Below	UDSA
USER	はい	Below	RDSA
USER	いいえ	上	EUDSA
USER	はい	上	ERDSA

NEW_PROGRAM_TOKEN(name8)

新しく定義したプログラムに対して提供されるトークンを返します。

name8

取得された 8 バイトのトークンが入る場所の名前。

PROGRAM_ATTRIBUTE(RELOAD|RESIDENT|REUSABLE|TRANSIENT)

プログラムの常駐状況を指定します。

RELOAD

ストレージに新規コピーをロードすることで、このプログラムのすべての ACQUIRE_PROGRAM 要求が満たされます。コピーに対して RELEASE 要求が発行されると、そのコピーはストレージから削除されます。

注: 出口プログラムを定義するときは、この属性を使用しないでください。

RESIDENT

削除しない限りストレージから除去されない、プログラムの単一コピーがあります。RESIDENT プログラムは、少なくとも準再入可能である必要があります。

REUSABLE

プログラムは少なくとも準再入可能であり、ストレージ内の単一コピーをシステム内の複数のタスクによって使用することができます。REUSABLE プログラムの使用回数がゼロになると、このプログラムは、通常の動的プログラム・ストレージ圧縮 (DPSC) スキームの一部として ストレージからの削除に適格になります。

TRANSIENT

使用回数がゼロになるとすぐにプログラムがストレージから削除されることを除いて、REUSABLE と同様です。プログラムの使用頻度が少ない場合、またはシステム内のプログラムが重大なストレージ不足の状況にある場合のみ、このオプションを指定してください。

PROGRAM_NAME(name8 | string | "string")
定義するプログラムの名前を指定します。

name8
8 バイトのプログラム名が入る場所の名前。

string プログラムの名前を示す、間にブランクのない文字ストリング。

"string"
引用符で囲まれた文字ストリング。ストリングの長さは、ブランク埋め込みまたは切り捨てによって、8 に設定されます。

PROGRAM_TYPE(PRIVATE|SHARED|TYPE_ANY)
プログラムのロード元を指定します。

PRIVATE
プログラムは、DFHRPL または動的 LIBRARY 連結内にあります。PRIVATE プログラムは再入可能である必要はなく、無許可の上書きに対する制限付き保護のみが指定されます。保護の度合いは、プログラムがロードされる動的ストレージ域 (DSA) のタイプによって異なります (EXECUTION_KEY オプションを参照してください)。

DSA 無許可の上書きに対する保護

CDSA
USER タスクでは上書き不可。

ECDSA
USER タスクでは上書き不可。

ERDSA
完了: USER タスクまたは CICS タスクでは上書き不可。

EUDSA
なし

RDSA
完了: USER タスクまたは CICS タスクでは上書き不可。

UDSA
なし

SHARED
プログラムはリンク・パック域 (LPA) 内にあり、再入可能で、保護されています。

TYPE_ANY
DFHRPL または動的 LIBRARY 連結を使用するか、プログラムの LPA コピーを使用することができます。ただし、LPA コピーが優先されます。

REQUIRED_AMODE(24|31|AMODE_ANY|64)
プログラムのアドレッシング・モードを指定します。後続の ACQUIRE_PROGRAM 処理の際に、定義済みアドレッシング要件を満たすプログラムのコピーが見つからない場合、ACQUIRE_PROGRAM 呼び出しは EXCEPTION 応答と REASON 値 PROGRAM_NOT_FOUND を受け取ります。

注:

1. この関数の AMODE_ANY と AMODE 31 は同じ意味です。
2. このオプションを使用して、プログラムのリンク・エディット済みアドレッシング・モードをオーバーライドすることはできません。

REQUIRED_RMODE(24|RMODE_ANY)
プログラムの常駐モードを指定します。後続の ACQUIRE_PROGRAM 処理の際に、定義済みアドレッシング要件を満たすプログラムのコピーが見つからない場合、ACQUIRE_PROGRAM 呼び出しは EXCEPTION 応答と REASON 値 PROGRAM_NOT_FOUND を受け取ります。

注: このオプションを使用して、プログラムのリンク・エディットされている常駐モードをオーバーライドすることはできません。

DEFINE_PROGRAM の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	CATALOG_ERROR
	CATALOG_NOT_OPERATIONAL
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

注: 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。

DELETE_PROGRAM 呼び出し

DELETE_PROGRAM は、カタログや現行プログラムのリストから、指定されたプログラムの定義を削除します。この要求が正常に実行されると、後続の ACQUIRE_PROGRAM 要求は、REASON 値「PROGRAM_NOT_DEFINED」で失敗します。

DELETE_PROGRAM

```
DFHLDLDX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(DELETE_PROGRAM),
      PROGRAM_NAME(name8 | string | 'string' ),]
      [OUT,
      RESPONSE(name1 | *),
      REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

PROGRAM_NAME(name8 | string | "string")
削除するプログラムの名前を指定します。

name8
8 バイトのプログラム名が入る場所の名前。

string プログラムの名前を指定する文字ストリング。

"string"
引用符で囲まれたストリング。ストリングの長さは、ブランク埋め込みまたは切り捨てによって、8 に設定されます。

DELETE_PROGRAM の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	PROGRAM_NOT_DEFINED
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

注: 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。

IDENTIFY_PROGRAM 呼び出し

IDENTIFY_PROGRAM はアドレスに関連付けられているプログラムを検出します。アドレスが CICS 定義のプログラム内にある場合、呼び出しはそのプログラムに関する情報を返します。

アドレスがローダー・ドメイン CICS 定義のプログラムに関連付けられていない場合、要求は REASON 値 INSTANCE_NOT_FOUND で失敗します。

IDENTIFY_PROGRAM

```
IDENTIFY_PROGRAM
DFHLDLDX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(IDENTIFY_PROGRAM),
      ADDRESS(name4 | (Rn)| *),]
      [OUT,
      [PROGRAM_NAME(name8 | *),]
      [PROGRAM_ATTRIBUTE(name1 | (Rn) | *),]
      [PROGRAM_LENGTH(name4 | (Rn) | *),]
      [LOAD_POINT(name4 | (Ra) | *),]
      [ENTRY_POINT(name4 | (Ra) | *),]
      RESPONSE(name1 | *),
      REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

ADDRESS(name4 | (Rn) | *)
プログラムの識別に使用されるストレージ・アドレス。

name4
ストレージ・アドレスが格納される 4 バイトのフルワードの名前。

(Rn) ストレージ・アドレスに設定されるレジスター。

PROGRAM_NAME(name8 | *)
ストレージ・アドレスが入るプログラムの名前を返します。PROGRAM_NAME は CSECT ではなく、CICS 定義のプログラム名に対応します。

name8
8 バイトのプログラム名が入る場所の名前。

PROGRAM_ATTRIBUTE(name1 | (Rn) | *)
プログラム属性を返します。

name1
プログラム属性を受け取る 1 バイトの位置の名前。

(Rn) 下位バイトでプログラム属性を受け取り、他のバイトはゼロに設定されるレジスター。このレジスターには、値
RELOAD、RESIDENT、REUSABLE、または TRANSIENT を指定できます。

RELOAD
このプログラムは再使用可能ではないため、複数のプログラムのコピーがロードされることがあります。
RELEASE_PROGRAM 呼び出しが (コピーに対して) 発行されると、そのコピーはストレージから削除されます。

RESIDENT
削除しない限りストレージから除去されない、プログラムの単一コピーがあります。RESIDENT プログラムは、少なくとも準再入可能である必要があります。デフォルトでは、PROGRAM_TYPE SHARED のプログラムはすべて RESIDENT 属性を持っています。DELETE_PROGRAM 呼び出しは、このタイプの RESIDENT プログラムには影響しません。

REUSABLE
このプログラムは、プログラムが使用されていない場合に、ストレージの使用を最適化するために、CICS でそのプログラムをストレージから削除できることを除いて、RESIDENT と同様です。

TRANSIENT
このプログラムは、使用回数がゼロになるとすぐにストレージから削除されることを除いて、RESIDENT と同様です。

PROGRAM_LENGTH(name4 | (Rn) | *)
指定されたプログラムの長さを返します。

name4
長さをバイト単位で受け取る 4 バイトの位置の名前。バイナリーで表されます。

(Rn) バイト単位の長さが入るレジスター。バイナリーで表されます。

LOAD_POINT(name4 | (Ra) | *)
プログラムのロード・ポイント・アドレスを返します。

name4
ロードされたアドレスを受け取る 4 バイトの位置の名前。

(Ra) ロード・アドレスが入るレジスター。

ENTRY_POINT(name4 | (Ra) | *)
プログラムのエントリー・ポイント・アドレスを返します。

name4
31 ビットのエントリー・アドレスを受け取る 4 バイトの位置の名前。

(Ra) エントリー・アドレスを受け取るレジスター。

IDENTIFY_PROGRAM の RESPONSE 値および REASON 値

RESPONSE	REASON
OK	なし
EXCEPTION	INSTANCE_NOT_FOUND
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

注: 詳しくは、XPI 呼び出しの実行にある RESPONSE および REASON の説明を参照してください。

RELEASE_PROGRAM 呼び出し

RELEASE_PROGRAM は、現在ロードされているプログラムの使用回数を 1 ずつ減分します。

プログラムが RELOAD 属性を指定して定義されている場合、このプログラムのコピーによって占有されていたストレージは解放されます。

同じ出口プログラムの実行時に、単一プログラムに対して ACQUIRE_PROGRAM 要求および RELEASE_PROGRAM 要求を発行する必要があります。これを実行したくない場合は、CICS 初期設定時にプログラムを一度獲得し、CICS が終了するまでそのプログラムを常駐させたままにする必要があります。

RELEASE_PROGRAM

```
DFHLDLDX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(RELEASE_PROGRAM),
      ENTRY_POINT(pointer),
      {PROGRAM_NAME(name8 | string | 'string')|
      PROGRAM_TOKEN(name8)},]
      [OUT,
      RESPONSE(name1 | *),
      REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

ENTRY_POINT(pointer)
指定されたプログラムのコピーのエントリー・ポイントのアドレスを指定します。

PROGRAM_NAME(name8 | string | "string")
解放するプログラムの名前を指定します。

name8
8 バイトのプログラム名が入る場所の名前。

string プログラムの名前を指定する文字ストリング。

"string"
引用符で囲まれたストリング。ストリングの長さは、ブランク埋め込みまたは切り捨てによって、8 に設定されます。

PROGRAM_TOKEN(name8),
解放するプログラムを識別するトークンを指定します。

name8
前述の DEFINE_PROGRAM 呼び出しまたは ACQUIRE_PROGRAM 呼び出しから取得された 8 バイトのトークンが入る場所の名前。

RELEASE_PROGRAM の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	PROGRAM_NOT_DEFINED
	PROGRAM_NOT_IN_USE
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

注:

1. 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。
2. 「PROGRAM_NOT_DEFINED」は、ユーザー名がシステムに認識されていないプログラムの場合に返されます。
3. 「PROGRAM_NOT_IN_USE」は、指定されたプログラムの使用回数が既にゼロになっている場合に返されます。

第 8 章 ログ・マネージャー XPI 関数

XPI は、2 つのログ・マネージャー関数を提供します。これらは、DFHLGPAX 呼び出しの INQUIRE_PARAMETERS および SET_PARAMETERS です。これらの呼び出しを使用し、ログ・マネージャー・パラメーター KEYPOINT_FREQUENCY を照会して設定することができます。このパラメーターにより、CICS 領域のアクティビティー・キーポイント処理の頻度を指定します。

INQUIRE_PARAMETERS 呼び出し

INQUIRE_PARAMETERS は、システムのアクティビティー・キーポイントの頻度に関する情報を返します。

INQUIRE_PARAMETERS

```
DFHLGPAX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(INQUIRE_PARAMETERS),
      [OUT,
      [KEYPOINT_FREQUENCY(name4 | *),]
      RESPONSE(name1 | *),
      REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

KEYPOINT_FREQUENCY(name4 | *)
CICS 領域のアクティビティー・キーポイント処理の頻度を返します。

name4
頻度の値を受け取る 4 バイトの位置の名前。

INQUIRE_PARAMETERS の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
DISASTER	なし
INVALID	なし
KERNERROR	なし

SET_PARAMETERS 呼び出し

SET_PARAMETERS では、CICS 領域のアクティビティー・キーポイントの頻度を設定することができます。

SET_PARAMETERS

```
DFHLGPAX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(SET_PARAMETERS),
```

© Copyright IBM Corp. 1977, 2018

```
[KEYPOINT_FREQUENCY(name4 | (Rn) ),]]
[OUT,
RESPONSE(name1 | *),
REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

KEYPOINT_FREQUENCY(name4 | *)

CICS 領域のアクティビティー・キーポイント処理の頻度を指定します。

有効な値は 0、または 200 から 65535 までの任意の整数です。

name4
新規の頻度の値が入る 4 バイトの位置の名前。

(Rn) 新規の頻度の値が入るレジスター。

SET_PARAMETERS の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	OUT_OF_RANGE
DISASTER	なし
INVALID	なし
KERNERROR	なし

注: 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。

第 9 章 モニター XPI 関数

XPI は、4 つのモニター関数 (DFHMNMNX 呼び出し INQUIRE_MONITORING_DATA および MONITOR、DFHMNTDX 呼び出し SET_TRACKING_DATA、および DFHMNIAX 呼び出し INQUIRE_APP_CONTEXT) を提供します。

制約事項:

以下のドメインのグローバル・ユーザー出口点から呼び出された出口プログラムでは、DFHMNMNX、DFHMNTDX、および DFHMNIAX の各呼び出しを使用することはできません。

- ディスパッチャー・ドメイン
- ダンプ・ドメイン
- モニター・ドメイン
- 統計ドメイン
- 一時データ・プログラム

また、以下のドメインのグローバル・ユーザー出口点から呼び出された出口プログラムでは、SET_TRACKING_DATA 呼び出しおよび INQUIRE_APP_CONTEXT 呼び出しを使用することはできません。

- トランザクション・マネージャー・ドメイン

DFHTCP または DFHZCP のグローバル・ユーザー出口点から呼び出された出口プログラムでは、INQUIRE_APP_CONTEXT、INQUIRE_MONITORING_DATA、および SET_TRACKING_DATA の各呼び出しを使用することもできません。

INQUIRE_APP_CONTEXT 呼び出し

INQUIRE_APP_CONTEXT 呼び出しは、発行するタスクを実行している現行のアプリケーションに関連付けられたアプリケーション・コンテキスト・データを出口プログラムに返します。

制約事項

PLTPI の第 2 フェーズまでは、INQUIRE_APP_CONTEXT 関数を使用する出口プログラムを開始しないでください。PLTPI について詳しくは、Writing initialization and shutdown programsを参照してください。

INQUIRE_APP_CONTEXT

```
DFHMNIAX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(INQUIRE_APP_CONTEXT),
      CONTEXT(INITIAL | CURRENT)]
      [OUT,
      [APPLNAME(name64),]
      [PLATNAME(name64),]
```

© Copyright IBM Corp. 1977, 2018

```
[OPERNAME(name64),]
[MAJORVER(name4 | (Rn)),]
[MINORVER(name4 | (Rn)),]
[MICROVER(name4 | (Rn)),]
RESPONSE(name1 | *),
REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

APPLNAME(name64)
タスクに関連付けられているアプリケーションの 64 バイトの名前を返します。

CONTEXT(INITIAL | CURRENT)
CONTEXT(INITIAL | CURRENT) パラメーターは、返されるアプリケーション・コンテキスト値がタスクの初期コンテキストのものか現行コンテキストのものかを判別します。**CONTEXT** パラメーターを指定しない場合、デフォルトでは、タスクの現行アプリケーション・コンテキストを返します。

MAJORVER(name4 | (Rn))
タスクに関連付けられているアプリケーションのメジャー・バージョン番号を返します。

name4
メジャー・バージョン番号がバイナリー値として入る 4 バイトのフィールドの名前。

(Rn) メジャー・バージョン番号を受け取るレジスター。

MICROVER(name4 | (Rn))
タスクに関連付けられているアプリケーションのマイクロ・バージョン番号を返します。

name4
マイクロ・バージョン番号がバイナリー値として入る 4 バイトのフィールドの名前。

(Rn) マイクロ・バージョン番号を受け取るレジスター。

MINORVER(name4 | (Rn))
タスクに関連付けられているアプリケーションのマイナー・バージョン番号を返します。

name4
マイナー・バージョン番号がバイナリー値として入る 4 バイトのフィールドの名前。

(Rn) マイナー・バージョン番号を受け取るレジスター。

OPERNAME(name64)
タスクに関連付けられている操作の 64 バイトの名前を返します。

PLATNAME(name64)
タスクに関連付けられているプラットフォームの 64 バイトの名前を返します。

INQUIRE_APP_CONTEXT の RESPONSE 値および REASON 値

RESPONSE	REASON
OK	なし

RESPONSE	REASON
EXCEPTION	APP_CONTEXT_UNAVAILABLE
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。

INQUIRE_MONITORING_DATA 呼び出し

INQUIRE_MONITORING_DATA 関数は出口プログラムに、発行するタスクに累積されたパフォーマンス・クラス・モニター・データを返します。

データをマップする DFHMNTDS DSECT は、固定フォーマットです。次の点に注意してください。

- パフォーマンス・レコード内のすべての CICS システム定義フィールド (DFHMCT TYPE=RECORD マクロの EXCLUDE オプションを使用して除外用に指定したフィールドを含む) がリストされます。
- ユーザー定義データ・フィールドはリストされません。

INQUIRE_MONITORING_DATA

```
DFHMNMNX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(INQUIRE_MONITORING_DATA),
      DATA_BUFFER(buffer-descriptor),]
      [OUT,
      RESPONSE(name1 | *),
      REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

重要

初期設定段階での XPI の使用には制限があります。PLTPI の第 2 フェーズまでは、XPI 関数の
TRANSACTION_DUMP、WRITE_JOURNAL_DATA、MONITOR、および INQUIRE_MONITOR_DATA を使用する出口プログラムを開始しないでください。PLTPI について詳しくは、Writing initialization and shutdown programsを参照してください。

DATA_BUFFER(buffer-descriptor)
返されたモニター・データが入るバッファーのアドレスと長さを指定します。バッファー記述子の完全な定義については、XPI 構文を参照してください。DSECT DFHMNTDS はモニター・データをマップします。

INQUIRE_MONITORING_DATA の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	LENGTH_ERROR
	MONITOR_DATA_UNAVAILABLE
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

注:

1. 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。
2. 「LENGTH_ERROR」は、バッファー記述子で指定されている長さが、XPI 呼び出しから返されたモニター・データに対して短すぎることを意味します。

MONITOR 呼び出し

MONITOR XPI 呼び出しは、**EXEC CICS MONITOR** コマンドと類似しています。これにより、出口プログラムでユーザー・イベント・モニター・ポイント (EMP) を呼び出すことができます。

ユーザー・イベント・モニター・ポイントは、DFHMCT TYPE=EMP マクロを使用してモニター管理テーブル (MCT) で定義するか、DFHMCT TYPE=INITIAL マクロの **APPLNAME** パラメーターで生成する必要があります。CICS モニターについて詳しくは、CICS モニター機能: パフォーマンスおよび調整を参照してください。

ユーザー EMP では、ユーザーは、パフォーマンス・クラス・モニター・データ・レコードで無条件に予約されたフィールドに、ユーザーの独自データを追加できます (最大 256 カウンター、最大 256 クロック、および最大 256 バイトの単一文字ストリング)。DFHAPPL EMP で最大 12 バイトのユーザー情報を追加することもできます。

MONITOR

```
DFHMNMNX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(MONITOR),
      POINT(expression | name2 | (Rn)),
      [DATA1(expression | name4 | (Ra) | *),]
      [DATA2(expression | name4 | (Ra) | *),]
      [ENTRYNAME(name8 | string | 'string'),]]
      [OUT,
      RESPONSE(name1 | *),
      REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

重要

初期設定段階での XPI の使用には制限があります。PLTPI の第 2 フェーズまでは、XPI 関数の
TRANSACTION_DUMP、WRITE_JOURNAL_DATA、MONITOR、および INQUIRE_MONITOR_DATA を使用する出口プログラムを開始しないでください。PLTPI について詳しくは、Writing initialization and shutdown programsを参照してください。

DATA1(expression | name4 | (Ra) | *)

使用されているユーザー EMP のタイプによって内容が決まるフルワード・バイナリーの変数を指定します。

- MCT ユーザー EMP 定義に、ADDCNT、SUBCNT、NACNT、EXCNT、または ORCNT のいずれかのオプションが含まれる場合、DATA1 変数は、ユーザー EMP 定義による定義に従って使用される領域になります。
- MCT ユーザー EMP 定義に MLTCNT オプションが含まれる場合、DATA1 変数は、一連の隣接フルワードのアドレスを伴う領域になります。そのフルワードには、ユーザー EMP 定義によって定義されるユーザー・カウント・フィールドに追加される値が入ります。
- MCT ユーザー EMP 定義に MOVE オプションが指定されている場合には、 DATA1 の変数は、移動する文字ストリングのアドレスをもつ領域です。この規則は、DFHAPPL EMP にも適用されます。

ユーザー EMP オプションについて詳しくは、Monitoring control table (MCT) を参照してください。

expression
この EMP のフルワード・バイナリー変数を指定する有効なアセンブラー言語の式。

name4
この EMP のフルワード・バイナリー変数が入る 4 バイトのフィールドの名前。

(Ra) この EMP のフルワード・バイナリー変数が入るレジスター。

***** このオプションの値は既にパラメーター・リストに存在しているか、または、この EMP のオプションが指定されていません。

DATA2(expression | name4 | (Rn) | *)

使用されているユーザー EMP のタイプによって内容が決まるフルワード・バイナリーの変数を指定します。

- MCT ユーザー EMP 定義に、ADDCNT、SUBCNT、NACNT、EXCNT、または ORCNT のいずれかのオプションが含まれる場合、DATA2 変数は、ユーザー EMP 定義による定義に従って使用される領域になります。
- MCT ユーザー EMP 定義に MLTCNT オプションが指定されていれば、DATA2 の変数は更新されるユーザー・カウント・フィールドの数を持つ領域です。

  DATA2 で指定される数値は、操作用に MCT で定義されているデフォルト値をオーバーライドします。0 の値を指定した場合は、モニターにデフォルトが使用されます。DATA2 の値を指定しなくても、MLTCNT 操作の成功

が妨げられることはありませんが、DATA2_NOT_SPECIFIED 例外応答が返されます。注 5 を参照してください。

- MCT ユーザー EMP 定義に MOVE オプションが含まれる場合、DATA2 変数は、移動される文字ストリングの長さを伴う領域になります。

  DATA2 で指定される長さは、操作用に MCT で定義されているデフォルト値をオーバーライドします。0 の値を指定した場合は、モニターにデフォルトが使用されます。DATA2 の値を指定しなくても、MOVE 操作の成功が妨げられることはありませんが、DATA2_NOT_SPECIFIED 例外応答が返されます。注 5 を参照してください。

ユーザー EMP オプションについて詳しくは、Monitoring control table (MCT) を参照してください。

expression
この EMP のフルワード・バイナリー変数を指定する有効なアセンブラー言語の式。

name4
この EMP のフルワード・バイナリー変数が入る 4 バイトのフィールドの名前。

(Rn) この EMP のフルワード・バイナリー変数が入るレジスター。

***** このオプションの値は既にパラメーター・リストに存在しているか、または、この EMP のオプションが指定されていません。

ENTRYNAME(name8 | string | "string")
モニター・ポイントのエントリー名を指定します。これは、POINT 値を修飾するもので、モニター管理テーブル (MCT) に定義されています。

name8
8 バイトのストリングが入る場所の名前。

string ブランクが介在しない文字ストリング。マクロは、ストリングから長さ 8 バイトのリテラル定数を生成し、ブランクで拡張したり、必要に応じて切り捨てたりします。

"string"
引用符で囲まれたストリング。ブランクが入ることもあります。この値は、前述の「string」と同じ方法で処理されます。

注: MCT 内に EMP を定義する際に、エントリー名を指定しないと、エントリー名はデフォルトの「USER」になります。ENTRYNAME も同様に、指定しない場合はデフォルトの「USER」になります。

POINT(expression | name2 | (Rn))
MCT に定義されているモニター・ポイント ID を 0 から 255 の範囲で指定します。200 から 255 の間の点 ID は、IBM® プログラム・プロダクトで使用するため予約されています。

expression
2 バイトで表される有効なアセンブラー言語の式。

name2
ポイント・データの 2 バイトのソースの名前。

(Rn) 下位 2 バイトにポイント・データが入るレジスター。

MONITOR の RESPONSE 値および REASON 値

RESPONSE	REASON
OK	なし
EXCEPTION	DATA1_NOT_SPECIFIED DATA2_NOT_SPECIFIED POINT_NOT_DEFINED INVALID_DATA1_VALUE INVALID_DATA2_VALUE
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

注:

1. 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。
2. POINT_NOT_DEFINED は、指定した EMP が MCT に定義されていないことを示します。
3. INVALID_DATA1_VALUE およびINVALID_DATA2_VALUE は、不正なアドレスが提供されたことによって発生したと考えられます。これは、プログラム・チェックを引き起こします。
4. DATA1_NOT_SPECIFIED および DATA2_NOT_SPECIFIED は、DATA1 または DATA2 が、それぞれ操作に必要なときに指定されていなかったことを示します。DATA2 の説明を参照してください。
5. エラー応答により EMP の処理が強制終了します。障害発生時点よりも前に実行するように定義されている操作は実行されます。後の操作は取り消されます。

SET_TRACKING_DATA 呼び出し

SET_TRACKING_DATA 呼び出し関数は、発行するタスクのためにトランザクション・トラッキングの起点データ・タグを設定します。

SET_TRACKING_DATA

```
DFHMNTDX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(SET_TRACKING_DATA),
      {TRACKING_TAG (MOBILE)|TRACKING_TAG_VALUE(name1 | *)},]
      [OUT,
      RESPONSE(name1 | *),
      REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

重要

初期設定段階での XPI の使用には制限があります。PLTPI の第 2 フェーズまでは、SET_TRACKING_DATA 関数を使用する出口プログラムを開始しないでくださ

い。PLTPI について詳しくは、Writing initialization and shutdown programsを参照してください。

TRACKING_TAG (MOBILE)

トランザクション・トラッキングの起点データで設定されるトランザクションの追跡の起点データ・タグ情報を指定します。

MOBILE
トランザクション・トラッキングの起点データ・タグはモバイルです。

SET_TRACKING_DATA の RESPONSE 値および REASON 値

RESPONSE	REASON
OK	なし
EXCEPTION	NO_ASSOCIATION_DATA TRACKING_TAG_ALREADY_SET INVALID_TRACKING_TAG INVALID_TRACKING_TAG_VALUE
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

注:

1. 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。
2. NO_ASSOCIATION_DATA は、この XPI 呼び出しが呼び出されたトランザクションにトランザクション・トラッキングの関連データがないことを示します。関連データとトランザクション・トラッキングについて詳しくは、CICS 相互通信の紹介を参照してください。
3. TRACKING_TAG_ALREADY_SET は、発行するタスクに対してトランザクション・トラッキングの起点データ・タグが既に設定されていることを示します。
4. INVALID_TRACKING_TAG は、トランザクション・トラッキングの起点データ・タグに無効な値があることを示します。
5. INVALID_TRACKING_TAG_VALUE は、トランザクション・トラッキングの起点データ・タグの値が 129 から 255 までの範囲内にないことを示します。

第 10 章 オブジェクト・トランザクション XPI 関数

オブジェクト・トランザクション XPI 呼び出しを使用して、CICS 作業単位とリモート・トランザクション・コーディネーターとの間の呼び出しに応答する TRUE プログラムを実装できます。これらは、DFHOTTRX 呼び出しの COMMIT、COMMIT_ONE_PHASE、IMPORT_TRAN 、PREPARE、ROLLBACK、SET_ROLLBACK_ONLY、および DFHOTCOX 呼び出しの SET_COORDINATOR です。これらの関数は、CICS 作業単位の同期点処理に対する強力な制御を提供します。これらの呼び出しの使用法を誤ると、CICS リカバリー・マネージャーが CICS を即時に強制終了するため、再同期処理が必要になります。

IMPORT_TRAN 呼び出し

外部トランザクションにタスクの現在の作業単位をリンクします。外部トランザクションに関する情報の一部は、現在の作業単位に記録されます。

IMPORT_TRAN

```
DFHOTTRX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION(IMPORT_TRAN),
       [FORMAT_ID,(name4|Rn),]
       [BQUAL_LEN,(name4|Rn),]
       [TID_BLOCK_IN,(block-descriptor),]
       [TIMEOUT,(name4|Rn),]
       [LOGICAL_SERVER,(name4|string|'string'),]
       [PUBLIC_ID,(name64|string|'string'),]]
       [OUT,
       UOW_ID,(name8 | *),
       RESPONSE (name1 | *),
       REASON (name1 | *)]
```

このコマンドはスレッド・セーフです。

BQUAL_LEN (name4 | Rn)
OTS トランザクション ID (TID) のブランチ修飾子の長さを指定します。

name4
ブランチ修飾子の長さが入る 4 バイトの位置の名前。

Rn
ブランチ修飾子の長さが入るレジスター。

FORMAT_ID (name4 | Rn)
OTS トランザクションのフォーマット ID を指定します。

name4
フォーマット ID が入る 4 バイトの位置の名前。

Rn
フォーマット ID が入るレジスター。

LOGICAL_SERVER (name4 | string | 'string')
トランザクションが実行される論理サーバーの名前を指定します。この XPI のユーザーは、CICS システムにインストールされている CorbaServer 定義とは異なる値を選択する必要があります。

PUBLIC_ID (name64 | string | 'string')
トランザクションに関連した公開 ID を指定します。

© Copyright IBM Corp. 1977, 2018

TID_BLOCK_IN (block-descriptor)
タスクの作業単位に関連付けられている外部トランザクションの、固有の OTS トランザクション ID (TID) を指定します。ブロック記述子は、2 つのフルワードのデータで、最初のワードにはデータのアドレスが入り、2 番目のワードにはデータの長さ (バイト単位) が入ります。

TIMEOUT (name4 | Rn)
OTS トランザクション・タイムアウト値 (秒) を指定します。

name4
タイムアウト値が入る 4 バイトの位置の名前。

Rn
タイムアウト値が入るレジスター。

UOW_ID (name8 | *)
インポートされた OTS トランザクション内の CICS 作業単位の ID を指定します。

IMPORT_TRAN の RESPONSE 値および REASON 値

RESPONSE	REASON
OK	なし
EXCEPTION	TID_TOO_LONG
	OTS_TRAN_ALREADY
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

COMMIT_ONE_PHASE 呼び出し

外部コーディネーターを参照せずに、現行タスクの作業単位上の同期点を実行します。現行タスクの作業単位にコーディネーターに関する情報を追加するために SET_COORDINATOR 呼び出しを使用した場合は、COMMIT_ONE_PHASE 呼び出しを使用することはできません。

COMMIT_ONE_PHASE

```
DFHOTTRX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION(COMMIT_ONE_PHASE),]]
       [OUT,
       STATUS (name1 | *),
       RESPONSE (name1 | *),
       REASON (name1 | *)]
```

このコマンドはスレッド・セーフです。

STATUS (name1 | *)
CICS 作業単位の結果。

このパラメーターには、以下の値を指定できます。

COMMITTED

ROLLEDBACK

COMMIT_ONE_PHASE の RESPONSE 値および REASON 値

RESPONSE	REASON
OK	なし
EXCEPTION	なし
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

PREPARE 呼び出し

OTS トランザクションに代わって、CICS 作業単位上の同期点の最初のフェーズを実行します。この機能によって返される票は、OTS トランザクションのコーディネーターがトランザクション全体の結果を判別するために使用することを意図しています。PREPARE が呼び出された場合、CICS 作業単位は未確定 状態に入り、後続の COMMIT 関数または ROLLBACK 関数が呼び出されるまでは未確定のままになります。CICS が失敗し、再始動の必要がある場合は、CICS 作業単位はシステム・ログからリカバリーされます。そのため、未確定の作業単位を解決するには再同期する必要があります。

PREPARE

```
DFHOTTRX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION(PREPARE),]
       [OUT,
       VOTE (name1 | *),
       RESPONSE (name1 | *),
       REASON (name1 | *)]
```

このコマンドはスレッド・セーフです。

VOTE (name1 | *)

OTS トランザクションの最初のフェーズの結果。

このパラメーターには、以下の値を指定できます。

YES

NO

READ_ONLY

HEURISTIC_MIXED

PREPARE の RESPONSE 値および REASON 値

RESPONSE	REASON
OK	なし
EXCEPTION	なし
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

COMMIT 呼び出し

OTS トランザクションの同期点の第 2 フェーズを実行して、トランザクションが確実にコミットされるようにします。

COMMIT

```
DFHOTTRX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION(COMMIT),]
       [OUT,
       RESPONSE (name1 | *),
       REASON (name1 | *)]
```

このコマンドはスレッド・セーフです。

COMMIT の RESPONSE 値および REASON 値

RESPONSE	REASON
OK	なし
EXCEPTION	UOW_ROLLEDBACK
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

ROLLBACK 呼び出し

OTS トランザクションのロールバック

ROLLBACK

```
DFHOTTRX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION(ROLLBACK),]
       [OUT,
       RESPONSE (name1 | *),
       REASON (name1 | *)]
```

このコマンドはスレッド・セーフです。

ROLLBACK の RESPONSE 値および REASON 値

RESPONSE	REASON
OK	なし
EXCEPTION	UOW_COMMITTED
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

SET_ROLLBACK_ONLY 呼び出し

CICS 作業単位にマークを付けて、同期点プロトコルが実行されたときに OTS コーディネーターに「いいえ」の票を与え、グローバル・トランザクションのロールバックを強制します。CICS 作業単位は未完了 状態を継続しますが、後続のリカバリー可能リソース更新は、残りのグローバル・トランザクションと共にロールバックされます。

SET_ROLLBACK_ONLY

```
DFHOTTRX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION(SET_ROLLBACK_ONLY),]
       [OUT,
       RESPONSE (name1 | *),
       REASON (name1 | *)]
```

このコマンドはスレッド・セーフです。

SET_ROLLBACK_ONLY の RESPONSE 値および REASON 値

RESPONSE	REASON
OK	なし
EXCEPTION	なし
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

SET_COORDINATOR 呼び出し

現行タスクの作業単位にリンクを関連付けて、リモート・コーディネーターを表示します。

SET_COORDINATOR

リモート・コーディネーターのアクションへの応答で、同期点処理を介して現在の作業単位を呼び出すために、フェーズ 1 (OTTR_PREPARE) およびフェーズ 2 (OTTR_COMMIT または OTTR_ROLLBACK) のオブジェクト・トランザクション XPI 関数が使用されるようにしてください。SET_COORDINATOR 関数を使用してコーディネーターに関する情報を現行タスクの作業単位に追加した場合は、OTTR_COMMIT_ONE_PHASE 関数を使用することはできません。また、正しいオブジェクト・トランザクション XPI を使用して現行作業単位の同期点を取らない限り、タスクを終了させることはできません。

```
DFHOTCOX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION(SET_COORDINATOR),
       [IOR_BLOCK,(block-descriptor)]
       [HOST_BLOCK,(block-descriptor)]]
       [OUT,
       COORDINATOR_TOKEN,(name1 | *),
       RESPONSE (name1 | *),
       REASON (name1 | *)]
```

このコマンドはスレッド・セーフです。

HOST_BLOCK (block-descriptor)
システム (コーディネーター・インスタンスが入っている) の ID を表す文字ストリングへのポインターおよびその文字ストリングの長さ。サポートされるこのパラメーターの最大長は 4096 バイトです。

IOR_BLOCK (block-descriptor)
ホスト・システム内のコーディネーター・インスタンスを表す文字ストリングへのポインターとその文字ストリングの長さ。サポートされるこのパラメーターの最大長は 4096 バイトです。

COORDINATOR_TOKEN (name1 | *)
コーディネーターを表すトークン。

SET_COORDINATOR の RESPONSE 値および REASON 値

RESPONSE	REASON
OK	なし
EXCEPTION	IOR_TOO LONG
	HOST_TOO_LONG
	LINK_UNKNOWN
	COORDINATOR_NOT_FOUND
	COORDINATOR_ALREADY
	INVALID_SYNCPOINT_STATE
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

第 11 章 パラメーター・ドメイン XPI 関数

XPI には、パラメーター・ドメイン関数が 1 つ用意されています。機能トグルのための DFHPAIQX 呼び出し INQUIRE_FEATUREKEY です。

INQUIRE_FEATUREKEY 呼び出し

INQUIRE_FEATUREKEY は、機能トグルの値を取得します。

切り替え可能な機能の CICS リリース別のリストについては、トグル対応機能、リリース別のサポートを参照してください。機能リスト表のリンクをたどると、切り替え可能な特定の機能を有効にしたり構成オプションを設定したりするための機能トグルについて説明している情報が見つかります。

INQUIRE_FEATUREKEY

```
DFHPAIQX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(INQUIRE_FEATUREKEY),
      OPTION(name255|'string'),
      [STRING(buffer-descriptor),]
      [UPPERCASE (YES|NO),]
      [OUT,
      [NUMBER(name4),]
      [BOOLEAN(name1)]
      RESPONSE(name1 | *),
      REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

OPTION(name255|'string')
機能トグルの名前を指定します。

STRING(buffer-descriptor)
トグルについて返された文字ストリング値を入れるバッファーのアドレスと長さを指定します。バッファー記述子の定義の詳細については、XPI 構文を参照してください。

UPPERCASE (YES|NO)
STRING に返された値を大文字に変更するかどうかを指定します。

NUMBER(name4)
数値のトグル値をバイナリーで返します。

BOOLEAN(name1)
boolean のトグル値を boolean で返します。

TRUE 値は true です。

FALSE
値は false です。

© Copyright IBM Corp. 1977, 2018

INQUIRE の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	NOT_FOUND
	BUFFER_TOO_SMALL
	BAD_OPTION
	NOT_BOOLEAN
	NOT_NUMBER
	NO_LIST
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

第 12 章 プログラム管理 XPI 関数

XPI は、DFHPGISX、DFHPGAQX、および DFHPGCHX 呼び出しを含む 8 つのプログラム管理関数を提供します。これらの関数は、ローダー関数と共に、プログラムを操作するための一連の包括的なツールを提供します。

プログラム管理関数には、以下の DFHPGISX 呼び出しが含まれます。

- END_BROWSE_PROGRAM
- GET_NEXT_PROGRAM
- INQUIRE_CURRENT_PROGRAM
- INQUIRE_PROGRAM
- SET_PROGRAM
- START_BROWSE_PROGRAM

プログラム管理関数には、以下の DFHPGAQX 呼び出しが含まれます。

- INQUIRE_AUTOINSTALL
- SET_AUTOINSTALL

プログラム管理関数には、以下の DFHPGCHX 呼び出しが含まれます。

- BIND_CHANNEL

これらの関数をローダー関数 (DFHLDLDX 呼び出し) と共に使用して、プログラムを操作することができます。ただし、DFHPGISX 呼び出しの NEW_PROGRAM_TOKEN フィールドに返されるトークンは、DFHLDLDX 呼び出しによって返されるトークンとは異なります。DFHPGISX 呼び出しによって取得したトークンを DFHLDLDX 呼び出しで使用したり、逆に DFHLDLDX 呼び出しによって取得したトークンを DFHPGISX 呼び出しで使用したりしないでください。

INQUIRE_PROGRAM 呼び出し

INQUIRE_PROGRAM は、指定されたプログラムの属性に関する情報を返します。

INQUIRE_PROGRAM

```
DFHPGISX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(INQUIRE_PROGRAM),
      [AC_APPLICATION_NAME(block-descriptor),]
      [AC_MAJOR_VERSION(name4),]
      [AC_MICRO_VERSION(name4),]
      [AC_MINOR_VERSION(name4),]
      [AC_PLATFORM_NAME(block-descriptor),]
      [{PROGRAM_NAME(name8 | string | 'string')|
      PROGRAM_TOKEN(name4)},]
      [SHOW_PROGRAMS(PRIVATE|PRIVATE_AND_PUBLIC),]
      [OUT,
      [ACCESS(CICS|NONE|READ_ONLY|USER),]
      [APIST(CICSAPI|OPENAPI),]
```

© Copyright IBM Corp. 1977, 2018

```
[AVAIL_STATUS(DISABLED|ENABLED),]
[CEDF_STATUS(CEDF|NOCEDF|NOT_APPLIC),]
[CONCURRENCY(QUASIRENT|THREADSAFE),]
[DATA_LOCATION(ANY|BELOW|NOT_APPLIC),]
[DYNAMIC_STATUS(DYNAMIC|NOT_DYNAMIC),]
[ENTRY_POINT(name4),]
[EXECUTION_KEY(CICS|NOT_APPLIC|USER),]
[EXECUTION_SET(DPLSUBSET|FULLAPI|NOT_APPLIC),]
[HOLD_STATUS(CICS_LIFE|NOT_APPLIC|TASK_LIFE),]
[INSTALL_TYPE(AUTO|CATALOG|GROUPLIST|MANUAL|RDO|SYSAUTO),]
[LANGUAGE_DEDUCED(ASSEMBLER|C370|COBOL|
                  COBOL2|LE370|NOT_APPLIC|NOT_DEDUCED|PLI),]
[LANGUAGE_DEFINED(ASSEMBLER|C370|COBOL|
                  LE370|NOT_APPLIC|NOT_DEFINED|PLI),]
[LIBRARY(name8),]
[LIBRARYDSN(name44),]
[LOAD_POINT(name4),]
[LOAD_STATUS(LOADABLE|NOT_APPLIC|NOT_LOADABLE|NOT_LOADED),]
[LOCATION(CDSA|ECDSA|ELPA|ERDSA|ESDSA|LPA|NONE|RDSA|SDSA),]
[MODULE_TYPE(MAPSET|PARTITIONSET|PROGRAM),]
[NEW_PROGRAM_TOKEN(name4),]
[PROGRAM_ATTRIBUTE(RELOAD|RESIDENT|REUSABLE|TRANSIENT),]
[PROGRAM_LENGTH(name4),]
[PROGRAM_TYPE(NOT_APPLIC|PRIVATE|SHARED|TYPE_ANY),]
[PROGRAM_USAGE(APPLICATION|NUCLEUS),]
[PROGRAM_USE_COUNT(name4),]
[PROGRAM_USER_COUNT(name4),]
[REMOTE_DEFINITION(LOCAL|REMOTE),]
[REMOTE_PROGID(name8),]
[REMOTE_SYSID(name4),]
[REMOTE_TRANID(name4),]
[SPECIFIED_AMODE(24|31|AMODE_ANY|AMODE_NOT_SPECIFIED|64),]
[SPECIFIED_RMODE(24|RMODE_ANY|RMODE_NOT_SPECIFIED),]
RESPONSE(name1 | *),
REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

AC_APPLICATION_NAME(block-descriptor)
このプログラムに関連付けられているアプリケーションのアドレスと名前の長さを指定します。プラットフォームにデプロイされたアプリケーションの専用プログラムについて照会するには、完全なアプリケーション・コンテキストを提供するために、 AC_APPLICATION_NAME、 AC_MAJOR_VERSION、 AC_MINOR_VERSION、 AC_MICRO_VERSION、および AC_PLATFORM_NAME フィールドを指定する必要があります。ブロック記述子について詳しくは、XPI 構文を参照してください。

AC_MAJOR_VERSION(name4)
アプリケーション・メジャー・バージョンをバイナリーで指定します。

AC_MICRO_VERSION(name4)
アプリケーション・マイクロ・バージョンをバイナリーで指定します。

AC_MINOR_VERSION(name4)
アプリケーション・マイナー・バージョンをバイナリーで指定します。

AC_PLATFORM_NAME(block-descriptor)
このプログラムに関連付けられているプラットフォームのアドレスと名前の長さを指定します。ブロック記述子について詳しくは、XPI 構文を参照してください。

ACCESS(CICS|NONE|READ_ONLY|USER)
プログラムのロード先のストレージのタイプを示す値を返します。

CICS CICS キー。

なし プログラムはロードされませんでした。

READ_ONLY
読み取り専用。

USER ユーザー・キー。

APIST(CICSAPI|OPENAPI)
インストール済みプログラム定義の API 属性を示す値を返します。

CICSAPI
プログラムは、CICS が許可したアプリケーション・プログラミング・インターフェースだけを使用するように制限されます。

OPENAPI
プログラムは、CICS が許可したアプリケーション・プログラミング・インターフェースのみを使用するようには制限されません。プログラムはスレッド・セーフ標準に適合するようにコード化され、CONCURRENCY(THREADSAFE) を指定して定義される必要があります。

AVAIL_STATUS(DISABLED|ENABLED)
プログラムが使用可能 (ENABLED) か、使用不可 (DISABLED) かを示す値を返します。

CEDF_STATUS(CEDF|NOCEDF|NOT_APPLIC)
プログラムの実行診断機能 (EDF) 状況を返します。

CEDF プログラムが CICS EDF の制御下で実行されている場合、EDF 診断画面が表示されます。

NOCEDF
EDF 診断画面は表示されません。

NOT_APPLIC
適用外。このモジュールは、マップ・セット、区分セット、またはリモート・プログラムです。

CONCURRENCY(QUASIRENT|THREADSAFE)
インストール済みプログラム定義の並行性属性を示す値を返します。

QUASIRENT
プログラムは、準再入可能として定義され、CICS QR TCB 下でのみ実行できます。

THREADSAFE
プログラムはスレッド・セーフとして定義されており、プログラムに制御が付与されるときにユーザー・タスクによって使用されるどの TCB の下でも実行できます。オープン TCB または CICS QR TCB が該当します。

注: しかし、Language Environment に準拠するプログラムの場合、もともと定義されている並行性は、後でプログラムをロードする際にオーバーライドできます。

DATA_LOCATION(ANY|BELOW|NOT_APPLIC)
16 MB 境界上に置かれているデータにプログラムでアクセスできるかどうかを示す値を返します。

ANY プログラムは、31 ビット・アドレスを処理することができ、16 MB 境界より上または下に位置するデータを受け渡すことができます。

BELOW
プログラムは 24 ビット・アドレスのみを処理します。したがって、16 MB 境界より下に位置するデータのみを渡す必要があります。

NOT_APPLIC
適用外。このモジュールは、マップ・セット、区分セット、またはリモート・プログラムです。

DYNAMIC_STATUS(DYNAMIC|NOT_DYNAMIC)
プログラムがプログラム・リンク要求の対象である場合に、要求を動的にルーティングできるかどうかを示す値を返します。

DYNAMIC
プログラムがプログラム・リンク要求の対象である場合、CICS 動的ルーティング・プログラムが呼び出されます。リモート・サーバー領域が EXEC CICS LINK コマンドの SYSID オプションで明示的に名前指定されていない場合は、ルーティング・プログラムはそのプログラムが実行する領域に要求をルーティングできます。

NOT_DYNAMIC
プログラムがプログラム・リンク要求の対象である場合、動的ルーティング・プログラムが呼び出されません。

分散プログラム・リンク (DPL) 要求の場合、PROGRAM 定義の REMOTESYSTEM オプションか EXEC CICS LINK コマンドの SYSID オプションで、プログラムが実行するサーバー領域を明示的に指定しなければなりません。指定しないと、デフォルトのローカル領域になります。

DPL 要求の動的ルーティングに関する情報は、DPL 要求の動的ルーティングを参照してください。

ENTRY_POINT(name4)
ローダー・ドメイン ACQUIRE_PROGRAM 呼び出しから返されたプログラム・エントリー・ポイント・アドレスのエントリー・ポイント・アドレスを返します。

EXECUTION_KEY(CICS|NOT_APPLIC|USER)
CICS がプログラムに制御を与えて、プログラムが CICS キー・ストレージを変更できるかどうかを決めるためのキーを返します。

CICS CICS は、CICS キーでプログラムに制御を渡します。プログラムは、リンケージ・エディターに定義されている常駐モード (RMODE) 属性に応じて、 16 MB 境界の上または下 (つまり CDSA または ECDSA) にある CICS 動的ストレージ域 (DSA) にロードされます。

NOT_APPLIC
適用外。このモジュールは、マップ・セット、区分セット、またはリモート・プログラムです。

USER CICS は、ユーザー・キーでプログラムに制御を渡します。プログラムは、リンケージ・エディターに定義されている常駐モード (RMODE) 属性に応じて、 16 MB 境界の上または下 (つまり UDSA または EUDSA) にある ユーザー DSA にロードされます。

EXECUTION_SET(DPLSUBSET|FULLAPI|NOT_APPLIC)
CICS をプログラムにリンクし、リモート CICS 領域で実行しているかのようにプログラムを実行するかどうかを示す値を返します。

DPLSUBSET
CICS は、リモート DPL プログラムの API 制限を使用してプログラムにリンクし、そのプログラムを実行します。このプログラムでは、CICS API のサブセットのみを使用できます。

FULLAPI
CICS は、リモート DPL プログラムの API 制限を使用せずにプログラムにリンクし、そのプログラムを実行します。プログラムは、全 CICS API を使用できます。

NOT_APPLIC
適用外。このモジュールは、マップ・セット、区分セット、またはリモート・プログラムです。(DEFINE PROGRAM の EXECUTIONSET オプションは、ローカル・プログラム定義にのみ適用されます。その目的は、DPL プログラムとして実行しているかのように、プログラムをローカル CICS 環境でテストすることです。)

HOLD_STATUS(CICS_LIFE|NOT_APPLIC|TASK_LIFE)
プログラムがロードされたままの状態でいられる時間の長さを示す値を返します。

CICS_LIFE
プログラムは、CICS がシャットダウンされるまでロードされたままになります。

NOT_APPLIC
適用外。プログラムはロードされていないか、リモートです。

TASK_LIFE
プログラムがタスクの存続期間中はロードされたままになります。

INSTALL_TYPE(AUTO|CATALOG|GROUPLIST|MANUAL|RDO|SYSAUTO)
PROGRAM リソース定義のインストールに使用された方法を返します。

AUTO
自動インストール。

CATALOG
CICS グローバル・カタログ (再始動後)。

GROUPLIST
CICS 開始グループ・リスト。

MANUAL
プログラムは、別の CICS コンポーネントによって明示的に定義された CICS 内部モジュールです。

RDO RDO コマンド。

SYSAUTO
システム自動インストール (つまり、自動インストール・ユーザー・プログラムを呼び出さずに CICS によって自動インストールされます)。プログラムは、CICS 内部モジュールや、第 1 段階の PLTPI プログラムなどである場合があります。

LANGUAGE_DEDUCED(ASSEMBLER|C370|COBOL|COBOL2|LE370|NOT_APPLIC|NOT_DEDUCED|PLI)
CICS を構成しているプログラム言語を返します。COBOL は OS/VS COBOL で、この CICS バージョンでは実行できません。COBOL2 は Enterprise COBOL または VS COBOL II のいずれかです。

LANGUAGE_DEFINED(ASSEMBLER|C370|COBOL|LE370| NOT_APPLIC|NOT_DEFINED|PLI)
リソース定義で指定されたプログラミング言語を返します。

LIBRARY(name)
このプログラムがロードされた LIBRARY リソースの 8 文字の名前を返します。このフィールドは、プログラムがロードされていない場合か、LPASTATUS が LPA (プログラムが LPA からロードされていることを示す) の場合はブランクです。

LIBRARYDSN(name44)
プログラムのロード元のデータ・セットの 44 文字の名前を返します。このフィールドは、プログラムがロードされていない場合か、 LPASTATUS が LPA (プログラムが LPA からロードされていることを示す) の場合はブランクです。

- インストール済みの LIBRARY からプログラムがロードされた場合は、LIBRARY と LIBRARYDSN の名前が返されます。
- 使用不可になっている LIBRARY からプログラムがロードされた場合は、LIBRARY 名は返されますが、LIBRARYDSN はブランクになります。
- 廃棄されている LIBRARY からプログラムがロードされた場合は、LIBRARY と LIBRARYDSN は両方ともブランクになります。

LOAD_POINT(name4)
ローダー・ドメイン ACQUIRE_PROGRAM 呼び出しから返されたプログラムのロード・ポイント・アドレスを返します。

LOAD_STATUS(LOADABLE|NOT_APPLIC|NOT_LOADABLE|NOT_LOADED)
プログラムがロード可能かどうかを示す値を返します。

LOADABLE
プログラムはロード可能です。

NOT_APPLIC
適用外。プログラムはリモートです。

NOT_LOADABLE
CICS がプログラムをロードしようとして失敗しました。プログラムがライブラリーにありません。

NOT_LOADED
CICS は、まだプログラムのロードを試行していません。

LOCATION(CDSA|ECDSA|ELPA|ERDSA|ESDSA|LPA|NONE|RDSA|SDSA)
ロードされている最新のプログラムのコピーがある場所を示す値を返します。

CDSA
CICS 動的ストレージ域

ECDSA
拡張 CICS 動的ストレージ域

ELPA 拡張リンク・パック域

ERDSA
拡張読取専用動的ストレージ域

ESDSA
拡張共用動的ストレージ域

LPA リンク・パック域

なし プログラムはロードされませんでした。

RDSA
読み取り専用動的ストレージ域

SDSA
共用動的ストレージ域

MODULE_TYPE(MAPSET|PARTITIONSET|PROGRAM)
プログラム・リソースのタイプを返します。

NEW_PROGRAM_TOKEN(name4)
指定されたプログラムの識別に使用可能なトークンを返します。

name4
このプログラムを識別する 4 バイトのトークンを受け取る場所の名前。

要求で PROGRAM_NAME が指定された場合、NEW_PROGRAM_TOKEN はプログラム・トークンに設定されます。このプログラム・トークンは、同じプログラムの後続の要求で使用できます。要求で PROGRAM_TOKEN が指定された場合、NEW_PROGRAM_TOKEN は同じ値に設定されます。

PROGRAM_ATTRIBUTE(RELOAD|RESIDENT|REUSABLE|TRANSIENT)
プログラムの常駐ステータスを返します。つまり、プログラムのストレージが解放されるときを示します。

RELOAD
このプログラムは再使用可能ではないため、複数のコピーがロードされることがあります。RELEASE_PROGRAM 呼び出しが (コピーに対して) 発行されると、そのコピーはストレージから削除されます。

RESIDENT
削除しない限りストレージから除去されない、プログラムの単一コピー

があります。RESIDENT プログラムは、少なくとも準再入可能である必要があります。デフォルトでは、PROGRAM_TYPE SHARED のプログラムはすべて RESIDENT です。

REUSABLE
ストレージの最適化を目的としてストレージで使用されていない REUSABLE プログラムを CICS で除去できることを除いて、RESIDENT と同様です。

TRANSIENT
ユーザー・カウントがゼロになるとすぐに TRANSIENT プログラムがストレージから削除されることを除いて、RESIDENT と同様です。

PROGRAM_LENGTH(name4)
プログラムの長さをバイト単位で返します。バイナリーで表されます。

PROGRAM_NAME(name8 | string | 'string')
照会するプログラムの名前を指定します。

name8
8 バイトのプログラム名が入る場所の名前。

string プログラムの名前を指定する文字ストリング。

'string'
引用符で囲まれた文字ストリング。ストリングの長さは、ブランク埋め込みまたは切り捨てによって、8 に設定されます。

PROGRAM_TOKEN(name4)
照会するプログラムを識別するトークンを指定します。

name4
直前の INQUIRE_PROGRAM 呼び出しから取得された 4 バイトのトークンが入る場所の名前。

PROGRAM_TYPE(NOT_APPLIC|PRIVATE|SHARED|TYPE_ANY)
次に新しいプログラムのコピーがロードされる場所を示す値を返します。

NOT_APPLIC
適用外。プログラムはリモートです。

PRIVATE
プログラムは、DFHRPL または動的 LIBRARY 連結からロードされます。. PRIVATE プログラムは再入可能である必要はなく、無許可の上書きに対する制限付き保護のみが与えられます。保護の度合いは、プログラムのロード先の動的ストレージ域のタイプによって異なります (DEFINE_PROGRAM 呼び出しの PROGRAM_TYPE オプションの説明を参照してください)。

SHARED
プログラムは、リンク・パック域 (LPA) からロードされます。SHARED プログラムは再入可能でなければならず、保護されています。

次に NEWCOPY または PHASEIN が受信されるときに、使用可能な場合はプログラムの LPA コピーが使用されます。使用可能な LPA バージョンがない場合、プログラムは、DFHRPL または動的 LIBRARY 連結からロードされます。

TYPE_ANY
DFHRPL または動的 LIBRARY 連結を使用するか、プログラムの LPA コピーを使用することができます。ただし、LPA コピーが優先されます。

PROGRAM_USAGE(APPLICATION|NUCLEUS)
プログラムを CICS 中核プログラムとして使用するか、ユーザー・アプリケーション・プログラムとして使用するかを示す値を返します。

PROGRAM_USE_COUNT(name4)
プログラムを別々のユーザーが呼び出した回数を返します。

PROGRAM_USER_COUNT(name4)
プログラムの現在のユーザー数を返します。

REMOTE_DEFINITION(LOCAL|REMOTE)
このプログラムがローカル・リソースであるのかリモート・リソースであるのかを示す値を返します。リモート・リソースである場合、CICS は、プログラムへのリンク要求を分散プログラム・リンク (DPL) 要求として処理し、リモート領域に伝送します。

REMOTE_PROGID(name8)
プログラムがリモート・リソースである場合、プログラムがリモート CICS 領域内で認識されるための名前を返します。PROGRAM 定義で REMOTESYSTEM が指定されていて、REMOTENAME が省略された場合、リモート名はローカル名と同じになります (つまり、REMOTE_PROGID のデフォルト値は PROGRAM_NAME になります)。

REMOTE_SYSID(name4)
プログラムがリモート・リソースである場合、プログラムを所有するリモート CICS 領域の名前を返します。

REMOTE_TRANID(name4)
プログラムがリモート・リソースである場合、リモート CICS の接続先であり、その下でプログラムを実行するトランザクションの名前を返します。

SHOW_PROGRAMS(PRIVATE|PRIVATE_AND_PUBLIC)
INQUIRE_PROGRAM にアプリケーション・コンテキスト・フィールドが指定されている場合、 SHOW_PROGRAMS は検索の範囲を定義します。

PRIVATE
専用プログラムについてのみ検索します。

PRIVATE_AND_PUBLIC
最初に専用プログラムを検索し、次に公開プログラムを検索します。

SPECIFIED_AMODE(24|31|AMODE_ANY|AMODE_NOT_SPECIFIED|64)
DEFINE_PROGRAM 呼び出しで指定されたアドレッシング・モードを返します。

SPECIFIED_RMODE(24|RMODE_ANY|RMODE_NOT_SPECIFIED)
DEFINE_PROGRAM 呼び出しで指定された常駐モード (つまり、プログラムを 16 MB 境界より上にロードするか下にロードするか) を返します。

INQUIRE_PROGRAM の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	PROGRAM_NOT_DEFINED_TO_LD
	PROGRAM_NOT_DEFINED_TO_PG
	APP_CONTEXT_NOT_FOUND
DISASTER	ABEND
	LOCK_ERROR
INVALID	INVALID_PROGRAM_TOKEN
KERNERROR	なし
PURGED	なし

INQUIRE_CURRENT_PROGRAM 呼び出し

INQUIRE_CURRENT_PROGRAM は、現在実行されているプログラムの属性に関する情報を返します。この呼び出しがグローバルまたはタスク関連ユーザー出口ルーチン内から発行される場合、グローバルまたはタスク関連ユーザー出口ルーチン・プログラム自体の属性が返されます。

INQUIRE_CURRENT_PROGRAM

```
DFHPGISX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(INQUIRE_CURRENT_PROGRAM),]
      [IGNORE_EXITS(YES|NO),]
      [OUT,
      [AVAIL_STATUS(DISABLED|ENABLED),]
      [CEDF_STATUS(CEDF|NOCEDF|NOT_APPLIC),]
      [CURRENT_AMODE(24|31|64),]
      [CURRENT_CEDF_STATUS(CEDF|NOCEDF),]
      [CURRENT_ENTRY_POINT(name4),]
      [CURRENT_ENVIRONMENT(EXEC|GLUE|PLT|SYSTEM|TRUE|URM),]
      [CURRENT_EXECUTION_SET(DPLSUBSET|FULLAPI),]
      [CURRENT_LOAD_POINT(name4),]
      [CURRENT_PROGRAM_LENGTH(name4),]
      [CURRENT_PROGRAM_NAME(name8),]
      [DATA_LOCATION(ANY|BELOW|NOT_APPLIC),]
      [DYNAMIC_STATUS(DYNAMIC|NOT_DYNAMIC),]
      [EXECUTION_KEY(CICS|NOT_APPLIC|USER),]
      [EXECUTION_SET(DPLSUBSET|FULLAPI|NOT_APPLIC),]
      [HOLD_STATUS(CICS_LIFE|NOT_APPLIC|TASK_LIFE),]
      [IGNORE_EXITS(YES|NO),]
      [INSTALL_TYPE(AUTO|CATALOG|GROUPLIST|MANUAL|RDO|SYSAUTO),]
      [INVOKING_ENVIRONMENT (EXEC|GLUE|PLT|SYSTEM|TRUE|URM),]
      [INVOKING_PROGRAM_NAME(name8),]
      [LANGUAGE_DEDUCED(ASSEMBLER|C370|COBOL|
                        COBOL2|LE370|NOT_APPLIC|NOT_DEDUCED|PLI),]
      [LANGUAGE_DEFINED(ASSEMBLER|C370|COBOL|
                        LE370|NOT_APPLIC|NOT_DEFINED|PLI),]
      [LIBRARY(name8),]
      [LIBRARYDSN(name44),]
      [LOAD_STATUS(LOADABLE|NOT_APPLIC|NOT_LOADABLE|NOT_LOADED),]
      [MODULE_TYPE(MAPSET|PARTITIONSET|PROGRAM),]
      [NEW_PROGRAM_TOKEN(name4),]
```

```
[REMOTE_DEFINITION(LOCAL|REMOTE),]
[REMOTE_PROGID(name8),]
[REMOTE_SYSID(name4),]
[REMOTE_TRANID(name4),]
[RETURN_PROGRAM_NAME(name8),]
RESPONSE(name1 | *),
REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

注: 以下のリストに記述されていないオプションは、INQUIRE_PROGRAM 呼び出しの等価のオプションと同じです。 73 ページの『INQUIRE_PROGRAM 呼び出し』を参照してください。

CURRENT_AMODE(24|31|64)
実行中のプログラムが現在使用しているアドレッシング・モードを返します。

CURRENT_CEDF_STATUS(CEDF|NOCEDF)
プログラムの現行インスタンスの EDF 状況を返します。返される値は、CEDF_STATUS (プログラム定義で指定される EDF状況) の場合と同じです。INQUIRE_PROGRAM の CEDF_STATUS オプションを参照してください。

CURRENT_ENTRY_POINT(name4)
現行プログラムのエントリー・ポイント・アドレスを返します。

CURRENT_ENVIRONMENT(EXEC|GLUE|PLT|SYSTEM|TRUE|URM)
現行プログラムを実行中環境、つまり、プログラムのタイプを返します。

EXEC ユーザー・アプリケーション・プログラム。

GLUE
グローバル・ユーザー出口プログラム。

PLT プログラム・リスト・テーブル・プログラム。

SYSTEM
CICS システム・コード。

TRUE タスク関連ユーザー出口プログラム。

URM ユーザー置換可能プログラム。

CURRENT_EXECUTION_SET(DPLSUBSET|FULLAPI)
プログラムの現行インスタンスで使用される API 実行セットを返します。返される値は、トランザクションの最初のプログラムでない限り、EXECUTION_SET (プログラム定義で指定された API 実行セット) と同じです。トランザクションの最初のプログラムの場合は、値が異なる可能性があります。これは、DPLSUBSET 属性がリンク先のプログラムのみに適用されるためです。トランザクションの最初のプログラムは、DPL 呼び出しのターゲットにはできないため、このオプションは無視されます。したがって、トランザクションの最初のプログラムでは、EXECUTION_SET が DPLSUBSET を返した場合でも、CURRENT_EXECUTION_SET は FULLAPI を返します。INQUIRE_PROGRAM の EXECUTION_SET オプションを参照してください。

CURRENT_LOAD_POINT(name4)
現行プログラムのロード・ポイント・アドレスを返します。

CURRENT_PROGRAM_LENGTH(name4)
現行プログラムの長さをバイト単位で返します。バイナリーで表されます。

CURRENT_PROGRAM_NAME(name8)
現在実行されているプログラムの名前を返します。

IGNORE_EXITS(YES|NO)
現行プログラムを呼び出したプログラムおよび制御が返されるプログラムに関する情報を返すときに、グローバル・ユーザー出口プログラムとタスク関連ユーザー出口プログラムを無視するかどうかを指定します。このオプションの設定は、INVOKING_ENVIRONMENT、INVOKING_PROGRAM_NAME、および RETURN_PROGRAM_NAME の各オプションで返される値に影響します。YES を指定した場合 (デフォルト)、これらのオプションではグローバル・ユーザー出口プログラムとタスク関連ユーザー出口プログラムが無視されます。NO を指定すると、グローバル・ユーザー出口プログラムとタスク関連ユーザー出口プログラムが関わる場合は、このオプションは出口プログラムに関する情報を返します。

INVOKING_ENVIRONMENT (EXEC|GLUE|PLT|SYSTEM|TRUE|URM)
現行プログラムの呼び出し元の環境を返します。つまり、INVOKING_PROGRAM_NAME という名前のプログラムに対応する環境を返します。値は CURRENT_ENVIRONMENT の説明のとおりです。

INVOKING_PROGRAM_NAME(name8)
現行プログラムを呼び出すための最新プログラムの名前を返します。IGNORE_EXITS(NO) を指定した場合、これは、関連する場合はグローバル・ユーザー出口プログラムまたはタスク関連ユーザー出口プログラムになります。IGNORE_EXITS(YES) を指定した場合 (デフォルト)、これは、グローバル・ユーザー出口プログラムまたはタスク関連ユーザー出口プログラムではない 最新プログラムになります。

LIBRARY(name)
このプログラムがロードされた LIBRARY リソースの 8 文字の名前を返します。このフィールドは、プログラムがロードされていない場合か、LPASTATUS が LPA (プログラムが LPA からロードされていることを示す) の場合はブランクです。インストール済み LIBRARY からプログラムがロードされた場合は、LIBRARY と LIBRARYDSN 名が返される。

LIBRARYDSN(*data-area*)
プログラムのロード元のデータ・セットの 44 文字の名前を返します。このフィールドは、プログラムがロードされていない場合か、LPASTATUS が LPA (プログラムが LPA からロードされていることを示す) の場合はブランクです。インストール済み LIBRARY からプログラムがロードされた場合は、LIBRARY と LIBRARYDSN 名が返される。

RETURN_PROGRAM_NAME(name8)
制御が返されるプログラムの名前を返します。IGNORE_EXITS(NO) を指定した場合、これは、グローバル・ユーザー出口プログラムまたはタスク関連ユーザー出口プログラムになります。IGNORE_EXITS(YES) を指定した場合 (デフォルト)、これは、中間のグローバル・ユーザー出口プログラムまたはタスク関連ユーザー出口プログラムが完了した後で制御が返されるプログラムです。

INQUIRE_CURRENT_PROGRAM の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	NO_CURRENT_PROGRAM
DISASTER	LOCK_ERROR
	ABEND
INVALID	なし
KERNERROR	なし
PURGED	なし

SET_PROGRAM 呼び出し

SET_PROGRAM を使用して、指定されたプログラムの定義の選択属性を設定します。

SET_PROGRAM

```
DFHPGISX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(SET_PROGRAM),
      {PROGRAM_NAME(name8 | string | 'string')|
      PROGRAM_TOKEN(name4)},]
      [AVAIL_STATUS(DISABLED|ENABLED),]
      [CEDF_STATUS(CEDF|NOCEDF),]
      [EXECUTION_KEY(CICS|USER),]
      [EXECUTION_SET(DPLSUBSET|FULLAPI),]
      [PROGRAM_ATTRIBUTE(RELOAD|RESIDENT|REUSABLE|TRANSIENT),]
      [PROGRAM_TYPE(PRIVATE|SHARED|TYPE_ANY),]
      [PROGRAM_USAGE(APPLICATION|NUCLEUS),]
      [REQUIRED_AMODE(24|31|AMODE_ANY|64),]
      [REQUIRED_RMODE(24|RMODE_ANY),]]
      [OUT,
      RESPONSE(name1 | *),
      REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

AVAIL_STATUS(DISABLED|ENABLED)
プログラムが使用可能 (ENABLED) か、使用不可 (DISABLED) かを示します。

CEDF_STATUS(CEDF|NOCEDF)
プログラムが CICS 実行診断機能 (EDF) の制御下で実行されている場合、診断画面を表示するかしないかを指定します。

EXECUTION_KEY(CICS|USER)
CICS がプログラムに制御を渡すために使用するキーを指定します。このキーは、プログラムで CICS キー・ストレージを変更可能かどうかを決定します。

CICS CICS は、CICS キーでプログラムに制御を渡します。プログラムは、リンケージ・エディターに定義されている常駐モード (RMODE) 属性に応じて、 16 MB 境界の上または下 (つまり CDSA または ECDSA) にある CICS 動的ストレージ域 (DSA) にロードされます。

USER CICS は、ユーザー・キーでプログラムに制御を渡します。プログラム

は、リンケージ・エディターに定義されている常駐モード (RMODE) 属性に応じて、 16 MB 境界の上または下 (つまり UDSA または EUDSA) にある ユーザー DSA にロードされます。

注: AMODE(31)、RMODE(ANY) を指定してプログラムを再入可能としてリンク・エディットした場合、EXECUTION_KEY オプションは無視され、プログラムは拡張読み取り専用 DSA (ERDSA) にロードされます。ERDSA に割り振られるストレージのタイプについて詳しくは、RENTPGM システム初期設定パラメーターを参照してください。

EXECUTION_SET(DPLSUBSET|FULLAPI)

CICS をプログラムにリンクし、リモート CICS 領域で実行しているかのようにプログラムを実行するかどうかを指定します。

注: EXECUTION_SET は、ローカル・プログラム定義にのみ適用されます。その目的は、DPL プログラムとして実行しているかのように、プログラムをローカル CICS 環境でテストすることです。

DPLSUBSET

CICS は、リモート DPL プログラムの API 制限を使用してプログラムにリンクし、そのプログラムを実行します。このプログラムでは、CICS API のサブセットのみを使用できます。

FULLAPI

CICS は、リモート DPL プログラムの API 制限を使用せずにプログラムにリンクし、そのプログラムを実行します。プログラムは、全 CICS API を使用できます。

PROGRAM_ATTRIBUTE(RELOAD|RESIDENT|REUSABLE|TRANSIENT)

プログラムの常駐状況、つまり、プログラムのストレージをいつ解放するかを指定します。

RELOAD

このプログラムは再使用可能ではないため、複数のコピーがロードされることがあります。RELEASE_PROGRAM 呼び出しが (コピーに対して) 発行されると、そのコピーはストレージから削除されます。

RESIDENT

任意の一時点でストレージ内にあるプログラムのコピーは 1 つ以下で、このコピーは削除しない限り除去されません。RESIDENT プログラムは、少なくとも準再入可能である必要があります。デフォルトでは、PROGRAM_TYPE SHARED のプログラムはすべて RESIDENT です。

REUSABLE

ストレージの最適化を目的としてストレージで使用されていない REUSABLE プログラムを CICS で除去できることを除いて、RESIDENT と同様です。

TRANSIENT

ユーザー・カウントがゼロになるとすぐに TRANSIENT プログラムがストレージから削除されることを除いて、RESIDENT と同様です。

PROGRAM_NAME(name8 | string | 'string')
属性を変更するプログラムの名前を指定します。

name8
8 バイトのプログラム名が入る場所の名前。

string プログラムの名前を指定する文字ストリング。

'string'
引用符で囲まれた文字ストリング。ストリングの長さは、ブランク埋め込みまたは切り捨てによって、8 に設定されます。

PROGRAM_TOKEN(name4)
プログラムを識別するトークンを指定します。

name4
直前の INQUIRE_PROGRAM、INQUIRE_CURRENT_PROGRAM、START_BROWSE_PROGRAM、または GET_NEXT_PROGRAM のいずれかの呼び出しから取得された 4 バイトのトークンが入る場所の名前。

PROGRAM_TYPE(PRIVATE|SHARED|TYPE_ANY)
プログラムをどの場所からロードするのかを指定します。

PRIVATE
プログラムは、DFHRPL または動的 LIBRARY 連結内にあります。PRIVATE プログラムは再入可能である必要はなく、無許可の上書きに対する制限付き保護のみが与えられます。保護の度合いは、プログラムのロード先の動的ストレージ域のタイプによって異なります (DEFINE_PROGRAM 呼び出しの PROGRAM_TYPE オプションの説明を参照してください)。

SHARED
プログラムはリンク・パック域 (LPA) 内にあり、再入可能で、保護されています。

TYPE_ANY
DFHRPL または動的 LIBRARY 連結を使用するか、プログラムの LPA コピーを使用することができます。ただし、LPA コピーが優先されます。

PROGRAM_USAGE(APPLICATION|NUCLEUS)
プログラムを CICS 中核プログラムとして使用するか、ユーザー・アプリケーション・プログラムとして使用するかを指定します。

REQUIRED_AMODE(24|31|AMODE_ANY|64)
プログラムのアドレッシング・モードを指定します。後続の処理中に、定義されたアドレッシング要件を満たすプログラムのコピーが見つからない場合、例外が発生します。

注:

1. この関数の AMODE_ANY と 31 は同じ意味です。
2. このオプションを使用して、プログラムのリンク・エディット済みアドレッシング・モードをオーバーライドすることはできません。

REQUIRED_RMODE(24|AMODE_ANY)

プログラムの常駐モード (つまり、16MB 境界より上にロードするか下にロードするか) を指定します。後続の処理中に、定義された常駐要件を満たすプログラムのコピーが見つからない場合、例外が発生します。

注: このオプションを使用して、プログラムのリンク・エディット済み常駐モードをオーバーライドすることはできません。

SET_PROGRAM の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	CEDF_STATUS_NOT_FOR_MAPSET
	CEDF_STATUS_NOT_FOR_PTNSET
	CEDF_STATUS_NOT_FOR_REMOTE
	EXEC_KEY_NOT_FOR_MAPSET
	EXEC_KEY_NOT_FOR_PTNSET
	EXEC_KEY_NOT_FOR_REMOTE
	EXEC_SET_NOT_FOR_MAPSET
	EXEC_SET_NOT_FOR_PTNSET
	EXEC_SET_NOT_FOR_REMOTE
	INCOMPATIBLE_BUNDLE_SET
	PROGRAM_NOT_DEFINED_TO_LD
	PROGRAM_NOT_DEFINED_TO_PG
DISASTER	ABEND
	CATALOG_ERROR
	CATALOG_NOT_OPERATIONAL
	LOCK_ERROR
INVALID	INVALID_MODE_COMBINATION
	INVALID_PROGRAM_NAME
	INVALID_PROGRAM_TOKEN
	INVALID_TYPE_ATTRIB_COMBIN
KERNERROR	なし
PURGED	なし

START_BROWSE_PROGRAM 呼び出し

START_BROWSE_PROGRAM は、プログラム定義を介した参照を開始し、必要に応じて指定のプログラムの定義で開始できるようにするトークンを返します。

START_BROWSE_PROGRAM

```
DFHPGISX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(START_BROWSE_PROGRAM),
      [PROGRAM_NAME(name8 | string | 'string'),]
      [OUT,
      BROWSE_TOKEN(name4)
      RESPONSE(name1 | *),
      REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

BROWSE_TOKEN(name4)
プログラム定義の順次参照を開始するために GET_NEXT_PROGRAM 呼び出しで使用するトークンを返します。

name4
4 バイトのトークンを受け取る場所の名前。

PROGRAM_NAME(name8 | string | 'string')
最初に参照する定義を持つプログラムの名前を指定します。参照シーケンスは、アルファベット順です。指定された名前のプログラムがない場合、CICS は、アルファベット順で次にくる定義のトークンを返します。プログラムを指定しない場合、CICS は、最初の定義のトークンを返します。

name8
8 バイトのプログラム名が入る場所の名前。

string プログラムの名前を指定する文字ストリング。

'string'
引用符で囲まれた文字ストリング。ストリングの長さは、ブランク埋め込みまたは切り捨てによって、8 に設定されます。

START_BROWSE_PROGRAM の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	
DISASTER	ABEND
	INVALID_DIRECTORY
	LOCK_ERROR
INVALID	なし
KERNERROR	なし
PURGED	なし

GET_NEXT_PROGRAM 呼び出し

GET_NEXT_PROGRAM を使用して、START_BROWSE_PROGRAM によって開始される参照シーケンスの際に次のプログラム定義を照会します。参照シーケンスは、アルファベット順です。定義のアルファベット順リストの終了は、END_LIST 例外応答によって示されます。

GET_NEXT_PROGRAM

```
DFHPGISX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(GET_NEXT_PROGRAM),
      BROWSE_TOKEN(name4),]
      [OUT,
      PROGRAM_NAME(name8),
      [ACCESS(CICS|NONE|READ_ONLY|USER),]
      [AVAIL_STATUS(DISABLED|ENABLED),]
      [CEDF_STATUS(CEDF|NOCEDF|NOT_APPLIC),]
      [DATA_LOCATION(ANY|BELOW|NOT_APPLIC),]
      [ENTRY_POINT(name4),]
      [EXECUTION_KEY(CICS|NOT_APPLIC|USER),]
```

```
[EXECUTION_SET(DPLSUBSET|FULLAPI|NOT_APPLIC),]
[HOLD_STATUS(CICS_LIFE|NOT_APPLIC|TASK_LIFE),]
[INSTALL_TYPE(AUTO|CATALOG|GROUPLIST|MANUAL|RDO|SYSAUTO),]
[LANGUAGE_DEDUCED(ASSEMBLER|C370|COBOL|
                  COBOL2|LE370|NOT_APPLIC|NOT_DEDUCED|PLI),]
[LANGUAGE_DEFINED(ASSEMBLER|C370|COBOL|
                  LE370|NOT_APPLIC|NOT_DEFINED|PLI),]
[LOAD_POINT(name4),]
[LOAD_STATUS(LOADABLE|NOT_APPLIC|NOT_LOADABLE|NOT_LOADED),]
[LOCATION(CDSA|ECDSA|ELPA|ERDSA|ESDSA|LPA|NONE|RDSA|SDSA),]
[MODULE_TYPE(MAPSET|PARTITIONSET|PROGRAM),]
[NEW_PROGRAM_TOKEN(name4),]
[PROGRAM_ATTRIBUTE(RELOAD|RESIDENT|REUSABLE|TRANSIENT),]
[PROGRAM_LENGTH(name4),]
[PROGRAM_TYPE(NOT_APPLIC|PRIVATE|SHARED|TYPE_ANY),]
[PROGRAM_USAGE(APPLICATION|NUCLEUS),]
[PROGRAM_USE_COUNT(name4),]
[PROGRAM_USER_COUNT(name4),]
[REMOTE_DEFINITION(LOCAL|REMOTE),]
[REMOTE_PROGID(name8),]
[REMOTE_SYSID(name4),]
[REMOTE_TRANID(name4),]
[SPECIFIED_AMODE(24|31|AMODE_ANY|AMODE_NOT_SPECIFIED|64),]
[SPECIFIED_RMODE(24|RMODE_ANY|RMODE_NOT_SPECIFIED),]
RESPONSE(name1 | *),
REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

注: 以下のリストに記述されていないオプションは、INQUIRE_PROGRAM 呼び出しの等価のオプションと同じです。 73 ページの『INQUIRE_PROGRAM 呼び出し』を参照してください。

BROWSE_TOKEN(name4)
参照する定義を識別するトークンを指定します。これは、最後の GET_NEXT_PROGRAM 呼び出しの NEW_PROGRAM_TOKEN フィールドで返されたトークン、または START_BROWSE_PROGRAM 呼び出しの BROWSE_TOKEN フィールドで返されたトークン (このトークンは GET_PROGRAM 呼び出しの後で毎回更新されます) のいずれかになります。

name4
4 バイトのトークンが入る場所の名前。

NEW_PROGRAM_TOKEN(name4)
参照シーケンス内の次の定義を識別するトークンを返します。これは、次の GET_NEXT_PROGRAM 呼び出し (または、シーケンスを終了する場合は END_BROWSE_PROGRAM 呼び出し) の BROWSE_TOKEN フィールドで使用できます。INQUIRE_PROGRAM 呼び出しおよび SET_PROGRAM 呼び出しの PROGRAM_TOKEN フィールドで使用することもできます。

name4
次のプログラム定義を識別する 4 バイトのトークンを受け取る場所の名前。

GET_NEXT_PROGRAM の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	END_LIST

RESPONSE	*REASON*
	INVALID_BROWSE_TOKEN
	PROGRAM_NOT_DEFINED_TO_LD
DISASTER	ABEND
	LOCK_ERROR
INVALID	なし
KERNERROR	なし
PURGED	なし

END_BROWSE_PROGRAM 呼び出し

END_BROWSE_PROGRAM を使用すると、START_BROWSE_PROGRAM によって開始されるプログラム定義の参照を終了することができます。

END_BROWSE_PROGRAM

```
DFHPGISX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(END_BROWSE_PROGRAM),
      BROWSE_TOKEN(name4),]
      [OUT,
      RESPONSE(name1 | *),
      REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

BROWSE_TOKEN(name4)
最後の GET_NEXT_PROGRAM 呼び出しの NEW_PROGRAM_TOKEN フィールドで返されたトークン、または START_BROWSE_PROGRAM 呼び出しの BROWSE_TOKEN フィールドで返されたトークン (このトークンは GET_NEXT_PROGRAM 呼び出しの後で毎回更新されます) のいずれかを指定します。

END_BROWSE_PROGRAM の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	INVALID_BROWSE_TOKEN
DISASTER	ABEND
	LOCK_ERROR
INVALID	なし
KERNERROR	なし
PURGED	なし

INQUIRE_AUTOINSTALL 呼び出し

INQUIRE_AUTOINSTALL は、プログラム、マップ・セット、および区分セットの自動インストール機能の現在の設定に関する情報を返します。

INQUIRE_AUTOINSTALL

```
DFHPGAQX [CALL,]
      [CLEAR,]
      [IN,
```

```
FUNCTION(INQUIRE_AUTOINSTALL),]
[OUT,
[AUTOINSTALL_CATALOG (ALL|MODIFY|NONE),]
[AUTOINSTALL_EXIT_NAME(name8),]
[AUTOINSTALL_STATE (ACTIVE|INACTIVE),]
RESPONSE(name1 | *),
REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

AUTOINSTALL_CATALOG(ALL|MODIFY|NONE)
自動インストールされたプログラム定義のカタログ状況を返します。

ALL 自動インストールされたすべてのプログラム、マップ、および区分セットの定義がカタログされます。

MODIFY
自動インストールされたプログラム、マップ、および区分セットの定義は、自動インストール後に SET PROGRAM コマンドによって変更された場合のみ、CICS グローバル・カタログに記録されます。

なし 自動インストールされたプログラム、マップ、および区分セットの定義はカタログされません。

AUTOINSTALL_EXIT_NAME(name8)
プログラム、マップ・セット、および区分セットのユーザーが置き換え可能な自動インストール制御プログラムの名前を返します。

AUTOINSTALL_STATE(ACTIVE|INACTIVE)
プログラムの自動インストール機能の状況を返します。

ACTIVE
プログラム、マップ・セット、および区分セットの自動インストールが使用可能です。

INACTIVE
プログラム、マップ・セット、および区分セットの自動インストールが使用不可です。

INQUIRE_AUTOINSTALL の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	なし
DISASTER	なし
INVALID	INVALID_FUNCTION
KERNERROR	なし
PURGED	なし

SET_AUTOINSTALL 呼び出し

SET_AUTOINSTALL を使用すると、プログラム、マップ・セット、および区分セットの自動インストール機能の設定を変更することができます。

SET_AUTOINSTALL

```
DFHPGAQX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(SET_AUTOINSTALL),
      [AUTOINSTALL_CATALOG (ALL|MODIFY|NONE),]
      [AUTOINSTALL_EXIT_NAME(name8),]
      [AUTOINSTALL_STATE (ACTIVE|INACTIVE),]
      [LANGUAGES_AVAILABLE(NO|YES),]]
      [OUT,
      RESPONSE(name1 | *),
      REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

AUTOINSTALL_CATALOG(ALL|MODIFY|NONE)
自動インストールされたプログラム定義のカタログ状況を指定します。

ALL 自動インストールされたすべてのプログラム、マップ、および区分セットの定義がカタログされます。

MODIFY
自動インストールされたプログラム、マップ、および区分セットの定義は、自動インストール後に SET PROGRAM コマンドによって変更された場合のみ、CICS グローバル・カタログに記録されます。

なし 自動インストールされたプログラム、マップ、および区分セットの定義はカタログされません。

AUTOINSTALL_EXIT_NAME(name8)
プログラム、マップ・セット、および区分セットについて、ユーザーが置き換え可能な自動インストール制御プログラムの名前を指定します。

AUTOINSTALL_STATE(ACTIVE|INACTIVE)
プログラムの自動インストール機能の状況を指定します。

ACTIVE
プログラム、マップ・セット、および区分セットの自動インストールを有効にします。

INACTIVE
プログラム、マップ・セット、および区分セットの自動インストールを無効にします。

LANGUAGES_AVAILABLE(NO|YES)
自動インストール制御プログラムが呼び出し可能かどうかを指定します。これは、言語が確立した後のみ呼び出すことができます。

NO 制御プログラムを呼び出すことはできません。

YES 制御プログラムを呼び出すことができます。

SET_AUTOINSTALL の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	なし
DISASTER	なし
INVALID	INVALID_FUNCTION

RESPONSE	*REASON*
KERNERROR	なし
PURGED	なし

BIND_CHANNEL 呼び出し

BIND_CHANNEL は、チャネルをタスクにバインドします。この呼び出しは、タスク内の最初のプログラムの前に発行する必要があります。

BIND_CHANNEL

```
DFHPGCHX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(BIND_CHANNEL),
      CHANNEL_TOKEN(name4)]
      [OUT,
      RESPONSE(name1 | *),
      REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

CHANNEL_TOKEN(name4)
タスクにバインドされるチャネルを表す 4 バイトのトークンが入る場所の名前を指定します。

BIND_CHANNEL の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	INVALID_TOKEN CHANNEL_ALREADY_SET CHANNEL_ON_RESTART
DISASTER	なし
INVALID	INVALID_LINK_LEVEL
KERNERROR	なし
PURGED	なし

第 13 章 状態データ・アクセス XPI 関数

XPI は、状態データ・アクセス関数を提供します。この関数を使用して、AP ドメイン内の特定のシステム・データの紹介および設定を行うことができます。これらは、DFHAPIQX 呼び出し INQ_APPLICATION_DATA、INQUIRE_SYSTEM、および SET_SYSTEM です。

INQ_APPLICATION_DATA 呼び出し

INQ_APPLICATION_DATA 呼び出しを使用すると、AP ドメインのアプリケーション・システム・データを照会することができます。

INQ_APPLICATION_DATA

```
DFHAPIQX [CALL,]
   [CLEAR,]
   [IN,
   FUNCTION(INQ_APPLICATION_DATA),]
   [OUT,
   [ACEE(name4 | (Rn) | * ),]     [DSA(name4 | (Rn) | * ),]
   [EIB(name4 | (Rn) | * ),]
   [RSA(name4 | (Rn) | * ),]
   [SYSEIB(name4 | (Rn) | * ),]
   [TCTUA(name4 | (Rn) | * ),]
   [TCTUASIZE(name4 | * ),]
   [TWA(name4 | (Rn) | * ),]
   [TWASIZE(name4 | (Rn) | * ),]
   RESPONSE (name1 | * ),
   REASON (name1 | * )]
```

このコマンドはスレッド・セーフです。

ACEE(name4 | (Rn | *)

アクセス制御環境エレメント (ACEE) のアドレスを返します。

name4
ACEE のアドレスを受け取るフルワード領域の名前。

(Rn) ACEE アドレスを受け取るレジスター。

***** APIQ_ACEE という名前のパラメーター・リスト自体が、アドレスの保持に使用されます。

DSA(name4 | (Rn | *)

アプリケーション・プログラムを再入可能にするために、そのアプリケーション・プログラムによって使用される動的ストレージのチェーンのヘッドを返します (例えば、アセンブラー・プログラムの場合は DFHEISTG ストレージ)。

name4
動的ストレージ・チェーンのヘッドのアドレスを受け取る 4 バイトの領域の名前。

(Rn) DSA アドレスを受け取るレジスター。

***** APIQ_DSA という名前のパラメーター・リスト自体が、アドレスの保持に使用されます。

© Copyright IBM Corp. 1977, 2018

EIB(name4 | (Rn) | *)
EXEC インターフェース・ブロック (EIB) のアドレスを返します。

name4
EIB のアドレスを受け取るフルワード領域の名前。

(Rn) EIB のアドレスを受け取るレジスター。

* APIQ_EIB という名前のパラメーター・リスト自体が、アドレスの保持に使用されます。

RSA(name4 | (Rn | *)
現行タスクのレジスター保存域のアドレスを返します。

name4
レジスター保存域のアドレスを受け取るフルワード領域の名前。

(Rn) レジスター保存域のアドレスを受け取るレジスター。

* APIQ_RSA という名前のパラメーター・リスト自体が、アドレスの保持に使用されます。

SYSEIB(name4 | (Rn) | *)
現行タスクのシステム EXEC インターフェース・ブロック (EIB) のアドレスを返します。

name4
システム EXEC インターフェース・ブロックのアドレスを受け取るフルワード領域の名前。

(Rn) システム EXEC インターフェース・ブロックのアドレスを受け取るレジスター。

* APIQ_SYSEIB という名前のパラメーター・リスト自体が、アドレスの保持に使用されます。

TCTUA(name4 | (Rn) | *)
現行タスクの端末管理テーブル・ユーザー域 (TCTUA) のアドレスを返します。

name4
TCTUA のアドレスを受け取るフルワード領域の名前。

(Rn) TCTUA のアドレスを受け取るレジスター。

* APIQ_TCTUA という名前のパラメーター・リスト自体が、アドレスの保持に使用されます。

TCTUASIZE(name4 | (Rn) | *)
現行タスクの TCTUA の長さをバイト単位で返します。

name4
TCTUA の長さをバイト単位で受け取る 4 バイトの領域の名前。

(Rn) TCTUA の長さを受け取るレジスター。

* APIQ_TCTUASIZE という名前のパラメーター・リスト自体が、TCTUA の長さの保持に使用されます。

TWA(name4 | (Rn) | *)
トランザクション作業域のアドレスを返します。

name4
TWA のアドレスを受け取るフルワード領域の名前。

(Rn) TWA のアドレスを受け取るレジスター。

* APIQ_TWA という名前のパラメーター・リスト自体が、TWA の長さの保持に使用されます。

TWASIZE(name4 | (Rn) | *)
トランザクション作業域 (TWA) の長さをバイト単位で返します。

name4
TWA の長さをバイト単位で受け取る 4 バイトの領域の名前。

(Rn) TWA の長さを受け取るレジスター。

* APIQ_TWASIZE という名前のパラメーター・リスト自体が、TWA の長さの保持に使用されます。

INQ_APPLICATION_DATA の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	DPL_PROGRAM
	NO_TRANSACTION_ENVIRONMENT
	TRANSACTION_DOMAIN_ERROR
DISASTER	ABEND
	LOOP
	INQ_FAILED
INVALID	INVALID_FUNCTION
KERNERROR	なし
PURGED	なし

INQUIRE_SYSTEM 呼び出し

INQUIRE_SYSTEM 呼び出しを使用すると、AP ドメインの CICS システム・データにアクセスすることができます。

INQUIRE_SYSTEM

```
DFHSAIQX [CALL,]
   [CLEAR,]
   [IN,
   FUNCTION(INQUIRE_SYSTEM),
   [GMMTEXT(name4),]]
   [OUT,
   [CICSREL(name4 | *),]
   [CICSSTATUS(ACTIVE | FINALQUIESCE |
                  FIRSTQUIESCE| INITIALIZING),]
   [CICSSYS(name1 | *),]
   [CICTSLEVEL(name6 | *),]
   [CWA(name4 | (Rn) | *),]
   [CWALENGTH(name2 | *),]
   [DATE(name4|*),]
   [DTRPRGRM(name8 | *),]
   [GMMLENGTH(name2 | *),]
   [GMMTRANID(name4 | *),]
   [INITSTATUS(FIRSTINIT | INITCOMPLETE | SECONDINIT |
                  THIRDINIT),]
   [JOBNAME(name8 | *),]
```

```
[OPREL(name2 | *),]
[OPSYS(name1 | *),]
[OSLEVEL(name4 | *),]
[PLTPI(name2 | *),]
[SDTRAN(name4 | *),]
[SECURITYMGR(EXTSECURITY | NOSECURITY),]
[SHUTSTATUS(CONTROLSHUT | NOTSHUTDOWN | SHUTDOWN),]
[STARTUP(COLDSTART | EMERGENCY | WARMSTART),]
[STARTUPDATE(name4| *),]
[TERMURM(name8 | *),]
[TIMEOFDAY(name4| *),]
[XRFSTATUS(NOXRF | PRIMARY | TAKEOVER),]
RESPONSE (name1 | * ),
REASON (name1 | * )]
```

このコマンドはスレッド・セーフです。

CICSREL(name4 | *)
CICS 領域が実行している CICS コードのレベル番号を返します。

name4
レベル番号の文字を 16 進値で受け取る 4 バイトの位置の名前。

CICSSTATUS(ACTIVE|FINALQUIESE|FIRSTQUIESCE|INITIALIZING)
CICS 領域の状況を返します。

ACTIVE
CICS 領域はアクティブで、処理を受け取る準備ができています。

FINALQUIESCE
CICS 領域はシャットダウン中で、静止の最終ステージにあります。

FIRSTQUIESCE
CICS 領域はシャットダウン中で、静止の第 1 ステージにあります。

INITIALIZING
CICS 領域を初期化しています。

CICSSYS(name1 | *)
実行中の CICS が作成される対象となったオペレーティング・システムを返します。

name1
オペレーティング・システムの 16 進文字を受け取る 1 バイトの領域の名前。値「X」は MVS を表します。

CICSTSLEVEL(name6 | *)
CICS が実行している CICS Transaction Server のリリースを返します。

name6
リリース文字を 16 進値で受け取る 6 バイトの領域の名前。

CWA(name4 | (Rn) | *)
共通作業域のアドレスを返します。

name4
CWA のアドレスを受け取る 4 バイトのフィールドの名前。

(Rn) CWA のアドレスを受け取るレジスター。

CWALENGTH(name2 | *)
CWA の長さをバイト単位で返します。

name2
CWA の長さを受け取る 2 バイトのフィールドの名前。

DATE(name4 | *)
今日の日付をパック 10 進形式で 4 バイトの **0Cyyddds** で返します。この値は以下のようになります。

- **C** は世紀の指標。(0=1900、1=2000、2=2100 のようになります。)
- **yy**= 年。
- **ddd**= 日。
- **s** は符号。

name4
日付を受け取る 4 バイトの位置の名前。

DTRPRGRM(name8 | *)
動的ルーティング・プログラムの名前を返します。

name8
動的ルーティング・プログラムの名前を受け取る 8 バイトの領域の名前。

GMMLENGTH(name2 | *)
「good morning」メッセージの長さをバイト単位で返します。

name2
good morning メッセージの長さを受け取る 2 バイトの領域の名前。

GMMTEXT(name4)
CICS が good morning メッセージを返すストレージ域 (少なくとも長さ 244 バイトで呼び出し元が所有するもの) のアドレスを指定します。

name4
good morning メッセージを受け取るストレージ域のアドレス。

注: GMMTEXT パラメーターは、入力パラメーターとして IN ステートメントに続いていなければなりません。

GMMTRANID(name4 | *)
CICS good morning トランザクションのトランザクション ID を返します。

name4
CICS good morning トランザクション ID を受け取る 4 バイトの領域の名前。

INITSTATUS(FIRSTINIT|INITCOMPLETE|SECONDINIT|THIRDINIT)
CICS 初期設定の際に到達したステージを示す値を返します。

FIRSTINIT
CICS 初期設定の第 1 ステージ。

INITCOMPLETE
CICS の初期設定が完了しました。

SECONDINIT
CICS 初期設定の第 2 ステージ。この段階は第 1 フェーズの PLTPI

プログラムの実行時に一致します。これらの PLT 内のプログラムは DFHDELIM ステートメントの前に定義されます。

THIRDINIT
CICS 初期設定の第 3 ステージ。この段階は第 2 フェーズの PLTPI プログラムの実行時に一致します。これらの PLT 内のプログラムは DFHDELIM ステートメントの後に定義されます。

JOBNAME(name8 | *)
CICS 領域が実行している 8 文字の MVS ジョブ名を返します。

name8
MVS ジョブ名を受け取る 8 バイトの領域の名前。

OPREL(name2 | *)
CICS 領域が実行している z/OS の MVS エレメントのレベル番号の最後の 2 桁を返します。

name2
z/OS の MVS エレメントのレベル番号をハーフワード・バイナリー値で受け取る 2 バイトの領域の名前。例えば、z/OS リリース 3 MVS は 03 で表されます。

注: このフィールドは互換性の目的でのみサポートされます。情報は、MVS CVTPRODN フィールドに保持された最後の 2 つの数字から派生します。例えば、CVTPRODN は、MVS/ESA SP バージョン 5 リリース 2.2 の場合は SP5.2.2 を保持し (この場合 OPREL は 22 を返す)、z/OS リリース 3 の場合は SP6.0.3 を保持します。z/OS 製品の完全なバージョンおよびリリース番号については、OSLEVEL フィールドを使用することをお勧めします。

OPSYS(name1 | *)
CICS 領域を実行中のオペレーティング・システムのタイプを返します。

name1
CICS を実行中のオペレーティング・システムの 16 進文字を受け取る 1 バイトの領域の名前。値「X」は MVS を表します。

OSLEVEL(name4 | *)
CICS を実行中の z/OS 製品のバージョン、リリース、およびモディフィケーション・レベルです。

name1
CICS を実行中の z/OS のバージョンおよびリリース番号を受け取る 4 バイトの領域の名前。値「0240」は z/OS リリース 4 を表します。

PLTPI(name2 | *)
CICS 初期化の最中に実行されるプログラムのリスト (プログラム・リスト・テーブル初期化後 (PLTPI) リスト) を含むプログラム・リスト・テーブル (PLT) を識別する接尾部を返します。

name2
接尾部を受け取る 2 バイトの領域の名前。

SDTRAN(name4 | *)
通常または即時シャットダウンの開始時に実行される「シャットダウン補助」ト

ランザクションの名前を返します。シャットダウン補助トランザクションについては、シャットダウン補助ユーティリティー・プログラム、DFHCESDで説明します。

name4
名前を受け取る 4 バイトの領域の名前。

SECURITYMGR(EXTSECURITY|NOSECURITY)
セキュリティーがアクティブかどうかを示す値を返します。

EXTSECURITY
CICS が外部セキュリティー・マネージャー (RACF® など) を使用しています。

NOSECURITY
CICS 領域でセキュリティーが使用されていません。システム初期設定パラメーターとして SEC=NO が指定されています。

SHUTSTATUS(CONTROLSHUT|NOTSHUTDOWN|SHUTDOWN)
CICS 領域のシャットダウン状況を返します。

CONTROLSHUT
CICS は制御されたシャットダウン、つまり、ウォーム・キーポイントによる通常シャットダウンを実行しています。

NOTSHUTDOWN
CICS はシャットダウン・モードではありません。

SHUTDOWN
CICS は即時シャットダウンを実行しています。

STARTUP(COLDSTART|EMERGENCY|WARMSTART)
CICS 領域が実行した始動のタイプを返します。

COLDSTART
CICS がコールド・スタートを実行しました。これは、システム初期設定パラメーターで明示的に指定されたためか、またはグローバル・カタログの状態が理由で CICS がコールド・スタートを強制したためです。

EMERGENCY
前回の実行がウォーム・キーポイントを使用して正常にシャットダウンしなかったため、CICS は緊急時再始動を実行しました。

WARMSTART
CICS は前回の実行の通常シャットダウンの後にウォーム・リスタートを実行しました。

STARTUPDATE(name4 | *)
この CICS 領域の開始日をパック 10 進数形式 (4 バイトの **00yydddc**、ここで、**yy**= 年、**ddd**= 日、**c** は符号) で返します。

name4
この CICS システムの開始日を受け取る 4 バイトの位置の名前。

TERMURM(name8 | *)
端末の自動インストール・ユーザー・プログラムの名前を返します。

name8
端末の自動インストール・ユーザー・プログラムの名前を受け取る 8 バイトの領域の名前。

TIMEOFDAY(name4 | *)
現在時刻を パック 10 進数形式 (4 バイトの **hhmmsstc**、ここで、**hh**= 時、**mm**= 分、**ss**= 秒、**t**= 10 分の 1 秒、**c** は符号) で返します。

name4
時間を受け取る 4 バイトの位置の名前。

XRFSTATUS(NOXRF|PRIMARY|TAKEOVER)
CICS 領域の XRF 状況を返します。

NOXRF
CICS は、システム初期設定パラメーター XRF=NO が指定された状態で開始されました。XRF はアクティブではありません。

PRIMARY
CICS 領域は、XRF 環境でアクティブな CICS として開始されました。

TAKEOVER
CICS 領域は、START=STANDBY システム初期設定パラメーターを使用して、代替 CICS として開始されました。

INQUIRE_SYSTEM の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
INVALID	INVALID_FUNCTION
EXCEPTION	LENGTH_ERROR
	UNKNOWN_DATA
DISASTER	INQ_FAILED
PURGED	なし

SET_SYSTEM 呼び出し

SET_SYSTEM 呼び出しを使用すると、AP ドメインで CICS システム・データ値を設定することができます。

SET_SYSTEM

```
DFHSAIQX [CALL,]
   [CLEAR,]
   [IN,
   FUNCTION(SET_SYSTEM),
   [DTRPRGRM(name8 | string | 'string'),]
   [GMMLENGTH(name2 | (Rn) | expression),]
   [GMMTEXT(name8 | (Rn)),]]
   [OUT,
   RESPONSE (name1 | * ),
   REASON (name1 | * )]
```

このコマンドはスレッド・セーフです。

DTRPRGRM(name8 | string | 'string')
動的ルーティング・プログラムの名前を指定します。

name8
動的ルーティング・プログラムの名前が含まれた 8 バイトの領域の名前。

string ブランクが介在しない文字ストリング。設定されている動的ルーティング・プログラムの名前を定義します。

'string'
ブランクが介在しない文字ストリング。プログラム内で名前 (ラベル) を文書化する場合は、この形式を使用します。

GMMLENGTH(name2 | (Rn))
GMMTEXT パラメーターで指定されている新規の「Good morning」メッセージの長さを指定します。

name2
新規の Good morning メッセージの長さがハーフワードのバイナリー値として含まれた 2 バイトの領域の名前。

(Rn) 新規の Good morning メッセージの長さが含まれたレジスター。

GMMTEXT(name4 | (Rn))
新規の Good morning メッセージを指定します。

name4
Good morning メッセージが入るストレージ域のアドレス (最大 246 バイト長) が含まれた 4 バイトの位置の名前。

(Rn) Good morning メッセージが入るストレージ域のアドレス (最大 246 バイト長) が含まれたレジスター。

SET_SYSTEM の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
INVALID	INVALID_FUNCTION
EXCEPTION	AKP_SIZE_ERROR
	NO_KEYPOINT
DISASTER	SET_FAILED
PURGED	なし

第 14 章 ストレージ管理 XPI 関数

XPI は、7 つのストレージ管理関数を提供します。これらは、GETMAIN、FREEMAIN、INQUIRE_ELEMENT_LENGTH、INQUIRE_TASK_STORAGE を呼び出す DFHSMMCX マクロと、INQUIRE_ACCESS、INQUIRE_SHORT_ON_STORAGE、SWITCH_SUBSPACE を呼び出す DFHSMSRX です。

以下のドメインまたはプログラムのグローバル・ユーザー出口点から呼び出された出口プログラムで、DFHSMMCX 呼び出しを使用することはできません。

- ディスパッチャー・ドメイン
- ダンプ・ドメイン
- モニター・ドメイン
- 統計ドメイン
- 一時データ・プログラム

GETMAIN 呼び出し

GETMAIN は、出口プログラムで使用するためのストレージのエレメントを獲得します。ストレージの特定の CLASS を要求することができます。また、その CLASS を 1 バイトの値に初期化するように要求することもできます。

GETMAIN 呼び出しを使用して獲得されたストレージと、以下のクラス内にあるストレージは、獲得終了時に TCA が使用されているときに、CICS によって解放されます。

- CICS
- CICS24
- USER
- USER24

これに対して、以下のクラス内にあるストレージは、タスクの終了時に自動的には解放されません。FREEMAIN 呼び出しを使用して解放する必要があります。

- SHARED_CICS
- SHARED_CICS24
- SHARED_USER
- SHARED_USER24
- TERMINAL

また、システム・タスクから呼び出せるユーザー出口もあります。このような環境では、次に CICS がシャットダウンするまで、ストレージは解放されません。したがって、GETMAIN 呼び出しで獲得されたすべてのストレージ域を、そのストレージ域の使用完了後すぐに解放するには、FREEMAIN 呼び出しを使用してください。

© Copyright IBM Corp. 1977, 2018

GETMAIN

```
DFHSMMCX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(GETMAIN),
      GET_LENGTH(name4 | (Rn) | expression),
      STORAGE_CLASS(CICS|CICS24|SHARED_CICS|SHARED_CICS24|
                 SHARED_USER|SHARED_USER24|USER|USER24|TERMINAL),
      SUSPEND(NO|YES),
      [INITIAL_IMAGE(name1 | literalconst),]
      [TCTTE_ADDRESS(name4 | (Ra)),]]
      [OUT,
      ADDRESS(name4 | (Rn) | *),
      RESPONSE(name1 | *),
      REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

ADDRESS(name4 | (Rn) | *)
呼び出しで取得されたストレージのアドレスを返します。

name4
取得されたストレージ・アドレスを保管するフルワードの名前。

(Rn) 取得されたストレージを指すように設定されているレジスター。

***** SMMC_ADDRESS という名前のパラメーター・リスト自体が、アドレスの保持に使用されます。

GET_LENGTH(name4 | (Rn) | expression)
必要なストレージのバイト数を指定します。以下のいずれかの方法で表示されます。

name4
バイト数をバイナリーで指定したフルワードの名前。

(Rn) バイト数をバイナリーで含むレジスター。

expression
有効なアセンブラー言語の式。例えば、数式、シンボリック式、またはその 2 つの組み合わせ。

TERMINAL ストレージを要求する場合、ユーザーが指定する長さには、ストレージ・アカウンティング域 (SAA) の長さは含まれません。指定できる最大長は 65,515 バイトです。CICS ストレージ管理によって 8 バイトの SAA が追加されます。XPI 呼び出しで返されるアドレスは SAA の開始のアドレスです。

CICS24、CICS、USER24、USER、SHARED_CICS24、SHARED_CICS、SHARED_USER24、または SHARED_USER のいずれかのストレージを要求する場合は、プログラムで必要な長さのみを指定する必要があります。返されるアドレスは、データ・ストレージの開始のアドレスです。これらのストレージ・クラスのストレージの最大サイズは、割り当てられている DSA のサイズと同じです。

INITIAL_IMAGE(name1 | literalconst)
初期化パターンを指定します。例えば、獲得したストレージを 2 進ゼロに設定するとします。

name1
1 バイトの初期化パターンが格納される場所の名前。

literalconst
リテラルの形式の数値。例えば、B'00000000'、X'FF'、X'FC'、"0"、または類似の値を持つ等価シンボル。

STORAGE_CLASS(CICS|CICS24|SHARED_CICS|SHARED_CICS24|SHARED_USER|SHARED_USER24|USER|USER24|TERMINAL)
呼び出しの対象となるストレージのクラスを指定します。このオプションに割り当て可能な値と、それぞれが表すストレージのタイプを表 6 にリストします。

表 *6. CICS* ストレージ・クラス

STORAGE_CLASS	ストレージのタイプ
CICS	16 MB より上で 2 GB より下のタスク存続期間 CICS キー・ストレージ
CICS24	16 MB より下のタスク存続期間 CICS キー・ストレージ
SHARED_CICS	16 MB より上で 2 GB より下の共用 CICS キー・ストレージ
SHARED_CICS24	16 MB より下の共用 CICS キー・ストレージ
SHARED_USER	16 MB より上で 2 GB より下の共用ユーザー・キー・ストレージ
SHARED_USER24	16 MB より下の共用ユーザー・キー・ストレージ
TERMINAL	このクラスのストレージには、8 バイトの SAA があります。
USER	16 MB より上で 2 GB より下のタスク存続期間ユーザー・キー・ストレージ
USER24	16 MB より下のタスク存続期間ユーザー・キー・ストレージ

GETMAIN 要求でストレージ・クラスを指定する必要があります。FREEMAIN 要求では、これはオプション・パラメーターであり、ユーザー指定の値は CICS によってチェックされません。

SUSPEND(YES|NO)
使用可能なストレージが GET_LENGTH オプションで要求した量より少ない場合に、要求を中断するかしないかを指定します。

TCTTE_ADDRESS(name4 | (Ra))
端末制御テーブル・エントリー (TCTTE) のアドレスを指定します。GETMAIN 要求では、STORAGE_CLASS オプションで TERMINAL のクラスを指定する場合に、このオプションをコーディングする必要があります。FREEMAIN 要求では、TERMINAL クラス・ストレージを解放する場合に、このオプションをコーディングする必要があります。

注**:** TERMINAL クラス・ストレージを取得する前に、TCAFCI ビット 7 を調べて、TCA が端末で実行されていることを確認してください。

name4
アドレスが入るフルワードの名前。

(Ra) TCTTE を指すレジスター。

GETMAIN の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	INSUFFICIENT_STORAGE
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

注:

1. 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。
2. GETMAIN 要求が SUSPEND(NO) と共に指定され、要求を満たすために十分な使用可能なストレージがなかった場合、INSUFFICIENT_STORAGE が返されます。
3. GETMAIN 要求が SUSPEND(YES) と共に指定され、要求を満たすために十分なストレージがなかったためにタスクがパージされた場合、PURGED が返されます。

FREEMAIN 呼び出し

FREEMAIN は、現在ご使用の出口プログラムに割り振られたストレージ域を解放します。

FREEMAIN

```
DFHSMMCX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(FREEMAIN),
      ADDRESS(name4 | (Rn) | *),
      [STORAGE_CLASS(CICS|CICS24|SHARED_CICS|SHARED_CICS24|
                 SHARED_USER|SHARED_USER24|USER|USER24|TERMINAL),]
      [TCTTE_ADDRESS(pointer),]]
      [OUT,
      RESPONSE(name1 | *),
      REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

このオプションの説明については、 105 ページの『GETMAIN 呼び出し』を参照してください。

FREEMAIN の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	なし
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

注: 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。

INQUIRE_ACCESS 呼び出し

INQUIRE_ACCESS は、開始アドレスおよび長さによって指定された、ストレージのエレメントのアクセス・キーを返します。エレメントの全体がいずれかの CICS 動的ストレージ域 (DSA) に含まれていない場合は、CICS は例外応答を返します。

INQUIRE_ACCESS

```
DFHSMSRX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(INQUIRE_ACCESS),
      ELEMENT_ADDRESS(name4 | (Rn) | *),
      ELEMENT_LENGTH(name4 | (Rn) | *),]
      [OUT,
      ACCESS(CICS | READ_ONLY | USER),
      RESPONSE(name1 | *),
      REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

ACCESS(CICS|READ_ONLY|USER)
ストレージ・エレメントのアクセス・キーを返します。

CICS CICS キー

READ_ONLY
読み取り専用ストレージ

USER ユーザー・キー。

ELEMENT_ADDRESS(name4 | (Rn) | *)
ストレージ・エレメントのアドレスを指定します。

ELEMENT_LENGTH(name4 | (Rn) | *)
ストレージ・エレメントの長さをバイト単位で指定します。長さゼロは、長さ 1 として処理されます。

INQUIRE_ACCESS の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	INVALID_ELEMENT
DISASTER	なし
INVALID	なし
KERNERROR	なし

INQUIRE_ELEMENT_LENGTH 呼び出し

INQUIRE_ELEMENT_LENGTH を使用すると、タスク存続期間ストレージのエレメントの任意部分のアドレスを渡すことができます。また、渡したアドレスが含まれるストレージ・エレメントの開始アドレスと長さを CICSから取得することができます。

INQUIRE_ELEMENT_LENGTH

```
DFHSMMCX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION (INQUIRE_ELEMENT_LENGTH),
       ADDRESS (name4 | (Rn) | *),]
       [OUT,
       ELEMENT_ADDRESS(name4 | (Rn) | *),
       ELEMENT_LENGTH(name4 | (Rn) | *),
       RESPONSE (name1 | *),
       REASON (name1 | *)]
```

このコマンドはスレッド・セーフです。

ADDRESS(name4 | (Rn) | *)
現行タスクのタスク存続期間ストレージのエレメント内にあるアドレスを指定します。

CICS は、先頭のチェック・ゾーンと末尾のチェック・ゾーンを参照するアドレスを、照会するストレージのエレメントの有効なアドレスとして受け入れます。

ELEMENT_ADDRESS(name4 | (Rn) | *)
ADDRESS パラメーターで参照されるタスク存続期間ストレージのエレメントの開始アドレスを返します。返された開始アドレスには、先行チェック・ゾーンは含まれません。

ELEMENT_LENGTH(name4 | (Rn) | *)
ADDRESS パラメーターで参照されるタスク存続期間ストレージのエレメントの長さを返します。返された長さには、先行または後続チェック・ゾーンは含まれません。

INQUIRE_ELEMENT_LENGTH の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	INVALID_ADDRESS
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

INQUIRE_SHORT_ON_STORAGE 呼び出し

INQUIRE_SHORT_ON_STORAGE を使用すると、64 ビット (2 GB 境界より上) のストレージ、16 MB より上で 2 GB より下 (16 MB 境界より上) のストレージ、16 MB より下 (16 MB 境界より下) のストレージで、CICS のストレージ不足が発生しているかどうかを確認できます。

INQUIRE_SHORT_ON_STORAGE

```
DFHSMSRX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION(INQUIRE_SHORT_ON_STORAGE),]
       [OUT,
       SOS_ABOVE_THE_BAR(NO|YES),
```

```
      SOS_ABOVE_THE_LINE(NO|YES),
      SOS_BELOW_THE_LINE(NO|YES),
      RESPONSE (name1 | *),
      REASON (name1 | *)]
```

このコマンドはスレッド・セーフです。

SOS_ABOVE_THE_BAR(NO|YES),
64 ビット (2 GB 境界より上) のストレージで CICS のストレージ不足が現在発生している場合は YES、そうでない場合は NO が返されます。

SOS_ABOVE_THE_LINE(NO|YES),
16 MB より上で 2 GB より下のストレージで CICS のストレージ不足が現在発生している場合は YES、そうでない場合は NO が返されます。

SOS_BELOW_THE_LINE(NO|YES),
CICS が現在 16 MB 未満のストレージ不足である場合は YES を返し、そうでない場合は NO を返します。

INQUIRE_SHORT_ON_STORAGE の RESPONSE と REASON の値

RESPONSE	*REASON*
OK	なし
DISASTER	なし
KERNERROR	なし

INQUIRE_TASK_STORAGE 呼び出し

INQUIRE_TASK_STORAGE を使用すると、タスクに属しているタスク存続期間ストレージの全エレメントの詳細を要求することができます。呼び出しで明示的にタスクのトランザクション番号を指定することも、デフォルトの現行タスクを使用することもできます。

INQUIRE_TASK_STORAGE

```
DFHSMMCX  [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION (INQUIRE_TASK_STORAGE),
      [TRANSACTION_NUMBER(name4 | (Rn) | *),]
      ELEMENT_BUFFER(buffer-descriptor),
      LENGTH_BUFFER(buffer-descriptor),]
      [OUT,
      NUMBER_OF_ELEMENTS(name4 | (Rn) | *),
      RESPONSE (name1 | *),
      REASON (name1 | *)]
```

このコマンドはスレッド・セーフです。

ELEMENT_BUFFER(buffer-descriptor)
指定のタスクまたはデフォルトの現行タスクに属しているタスク存続期間ストレージについて、CICS が返す全エレメントの開始アドレスのリストを入れるバッファーのアドレスおよび長さを定義します。

返された開始アドレスには、先行チェック・ゾーンは含まれません。バッファー記述子の説明については、XPI 構文を参照してください。

LENGTH_BUFFER(buffer-descriptor)
指定のタスクまたはデフォルトの現行タスクに属しているタスク存続期間ストレージについて、CICS が返すエレメントの長さのリストを入れるバッファーのアドレスおよび長さを定義します。返された長さには、先行または後続チェック・ゾーンは含まれません。

バッファー記述子の説明については、XPI 構文を参照してください。

NUMBER_OF_ELEMENTS(name4 | (Rn) | *)
2 つのバッファー ELEMENT_BUFFER および LENGTH_BUFFER のそれぞれのエントリーの数をフルワード・バイナリー値として返します。

TRANSACTION_NUMBER(name4 | (Rn) | *)
ストレージが属するタスクのトランザクション番号を 4 バイトのパック 10 進値として指定します。

トランザクション (タスク) 番号を省略すると、CICS は現行タスクが指定されたものと見なします。

INQUIRE_TASK_STORAGE の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	INSUFFICIENT_STORAGE NO_TRANSACTION_ENVIRONMENT
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

SWITCH_SUBSPACE 呼び出し

SWITCH_SUBSPACE を指定すると、タスクがまだ基本スペースで実行されていない場合に、CICS がサブスペースから基本スペースに切り替わります。タスクが既に基本スペース内にある場合、ストレージ・マネージャーはこの呼び出しを無視します。

この関数は、サブスペースで制御を受け取るグローバル・ユーザー出口プログラムで使用することも、何らかの理由で基本スペースに切り替える必要がある場合に使用することもできます。

SWITCH_SUBSPACE

```
DFHSMSRX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION(SWITCH_SUBSPACE),
       SPACE(BASESPACE),]
       [OUT,
       RESPONSE (name1 | *),
       REASON (name1 | *)]
```

このコマンドはスレッド・セーフです。

SPACE(BASESPACE)
CICS が現在サブスペース内で実行されている場合には、基本スペースの呼び出

しを実行してタスクを切り替えることを指定します。これにより、タスクは、別のタスクのユーザー・キー・タスク存続期間ストレージの読み取りおよび書き込みを行うことができます。

SWITCH_SUBSPACE の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
DISASTER	なし
KERNERROR	なし

第 15 章 トレース制御 XPI 関数

XPI は、1 つのトレース制御関数を提供します。これは DFHTRPTX 呼び出し TRACE_PUT です。

制約事項: 以下のグローバル・ユーザー出口点から呼び出された出口プログラムでは、DFHTRPTX 呼び出しを使用することはできません。

- ディスパッチャー・ドメイン
- ダンプ・ドメイン
- モニター・ドメイン
- 統計ドメイン
- 一時データ・プログラム

TRACE_PUT 呼び出し

TRACE_PUT は、アクティブ・トレース宛先にトレース・エントリーを書き込みます。

UEPTRON が出口プログラムを含む機能に対してトレースがアクティブになっていることを示している場合のみ、TRACE_PUT 呼び出しを行います (DFHUEPAR の UEPTRON を参照)。重大なエラーが発生した場合は、UEPTRON をテストせずに、例外トレース・エントリーを作成することをお勧めします。

TRACE_PUT を使用して例外トレース・エントリーを書き込む場合は、例外トレース・エントリーを識別して、トレース・フォーマット・ユーティリティー・プログラムによって例外トレース・エントリーが強調表示されるようにします。例外トレース・エントリーを識別するには、DFHTRPTX 呼び出しでリテラル・ストリング「USEREXC」を DATA1 ブロック記述子フィールドに入力します。

TRACE_PUT

```
DFHTRPTX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(TRACE_PUT),
      POINT_ID(literalconst | name2 | (Rn)),
      [DATA1(block-descriptor),]
      [DATA2(block-descriptor),]
      [DATA3(block-descriptor),]
      [DATA4(block-descriptor),]
      [DATA5(block-descriptor),]
      [DATA6(block-descriptor),]
      [DATA7(block-descriptor),]
      [RETURN_ADDR(expression | name4 | (Ra)),]]
      [OUT,
      RESPONSE(name1 | *)]
```

このコマンドはスレッド・セーフです。

DATA*n*(block-descriptor)
トレース・エントリーのデータ・セクションに組み込む領域を最大 7 つ指定し

© Copyright IBM Corp. 1977, 2018

ます。有効なブロック記述子の説明については、XPI 構文を参照してください。特定の DATA*n* を指定する場合は、DATA1 から DATA(*n*-1) を DATA*n* の前にコーディングする必要があります。指定の DATA 項目は、指定の順序 (DATA1 から DATA*n* の順序) でトレース出力に表示されます。データ・フィールド自体の前に、2 バイトの長さフィールドが表示されます。1 回の呼び出しでトレース可能なデータの最大合計長は、4000 バイトです。すべてのデータ・フィールドと、それらすべての 2 バイトの長さフィールドの合計長を、この制限内に収める必要があります。

POINT_ID(literalconst|name2|(Rn))
この要求の結果として作成されたトレース・エントリーを指定します。呼び出しドメイン内のすべての TRACE_PUT 呼び出しが、固有の POINT_ID を指定している必要があります。これにより、定様式トレースを検査する場合にトレース呼び出しの発信元を検出することができます。POINT_ID は、256 から 511 (X'100' から X'1FF') の範囲の 10 進数でなければなりません。この範囲は、CICS モジュールでは使用されませんが、ユーザー出口用に予約されています。

literalconst
ID が入るリテラル形式の番号

name2
ID が入る 2 バイトのフィールドの名前

(Rn) レジスター (ID が入る下位の 2 バイト)

RETURN_ADDR(expression|name4|(Ra))
トレース・エントリーのリターン・アドレス・フィールドに表示される値を指定します。

expression
アドレスを生成する有効なアセンブラー言語の式

name4
アドレスが入るフルワードの名前

(Ra) アドレスが入るレジスター

第 16 章 トランザクション管理 XPI 関数

XPI はトランザクション管理関数を提供します。ユーザーはこの関数を使用して、トランザクションを照会し、特定のトランザクションのパラメーターを設定することができます。

INQUIRE_CONTEXT 呼び出し

INQUIRE_CONTEXT は、トランザクションを実行中の環境に関する情報を返します。つまりブリッジ環境で実行するトランザクションについての情報を提供します。

INQUIRE_CONTEXT

```
DFHBRIQX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION(INQUIRE_CONTEXT),]
       [OUT,
       [CONTEXT(byte1),]
       [BRIDGE_TRANSACTION_ID(name4),]
       [BRIDGE_EXIT_PROGRAM(name8),]
       [BFB_TOKEN(name4),]
       [BRXA_TOKEN(name4),]
       [FACILITYTOKEN(name8),]
       [START_TYPE(byte1),]
       RESPONSE (name1 | *),
       REASON (name1 | *)]
```

このコマンドはスレッド・セーフです。

BFB_TOKEN(name4)
このタスクで使用されるブリッジ機能のアドレスが含まれたポインターを返します。ブリッジ機能は実際の端末ではありませんが、これは TCTTE と同じ形式のデータ構造で表され、DSECT DFHTCTTE を使用してマップできます。CONTEXT が NORMAL を返す場合、このフィールドの内容は無意味です。

注: 以前のリリースの CICS では、このフィールドは、BRIDGE_FACILITY_TOKEN と呼ばれていました。

name4
トークンを受け取る 4 バイトの位置の名前。

BRIDGE_EXIT_PROGRAM(name8)
このタスクで使用されるブリッジ出口プログラムの名前を返します。CONTEXT が NORMAL を返す場合、このフィールドの内容は無意味です。

name8
ブリッジ出口プログラムの名前を受け取る 8 バイトの位置の名前。

BRIDGE_TRANSACTION_ID(name4)
このトランザクションを開始する START BREXIT TRANSID コマンドを発行したブリッジ・モニター・トランザクションの名前を返します。CONTEXT が NORMAL を返す場合、このフィールドの内容は無意味です。

© Copyright IBM Corp. 1977, 2018

name4
ブリッジ・モニター・トランザクションの名前を受け取る 4 バイトの位置の名前。

BRXA_TOKEN(name4)
このタスクで使用されるブリッジ出口領域のアドレスが含まれたトークンを返します。BRXA は、Link3270 ブリッジ (START_TYPE=BRIQ_LINK) には適用されません。BRXA の形式は、DFHBRARx コピーブックによって定義されます。CONTEXT が NORMAL を返す場合、このフィールドの内容は無意味です。

name4
トークンを受け取る 4 バイトの位置の名前。

CONTEXT(byte1)
トランザクションを実行している環境のタイプを 1 バイトの位置 (*byte1*) に返します。

BRIDGE
ブリッジを使用して開始されたユーザー・トランザクション。

BREXIT
ブリッジ出口プログラム。

NORMAL
ブリッジ環境で実行していないトランザクション。

FACILITYTOKEN(name8)
facilitytoken (ブリッジ機能に関連付けられている ID) を返します。CONTEXT が NORMAL を返す場合、このフィールドの内容は無意味です。

name8
facilitytoken を受け取る 8 バイトの位置の名前。

START_TYPE(byte1)
3270 ブリッジがどのように開始されたかを示す標識を 1 バイトの位置 (*byte1*) に返します。CONTEXT が NORMAL を返す場合、このフィールドの内容は無意味です。

BRIQ_START
ブリッジは START BREXIT を使用して開始されました。

BRIQ_LINK
ブリッジは Link3270 メカニズムを使用して開始されました。

INQUIRE_CONTEXT の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
DISASTER	ABEND
	LOOP
INVALID	なし
EXCEPTION	NO_TRANSACTION_ENVIRONMENT
KERNERROR	なし

INQUIRE_DTRTRAN 呼び出し

INQUIRE_DTRTRAN は、ダイナミックトランザクション・ルーティング (DTR) トランザクション定義の名前を返します。

DTR トランザクション定義は、動的にルーティングされ、特定のトランザクション定義を持たないトランザクションの共通属性を提供します。これは、DTRTRAN システム初期設定パラメーターで指定されます。CICS 提供のデフォルト定義は CRTX です。

INQUIRE_DTRTRAN

```
DFHXMSRX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION(INQUIRE_DTRTRAN),]
       [OUT,
       DTRTRAN(name4),
       RESPONSE (name1 | *),
       REASON (name1 | *)]
```

このコマンドはスレッド・セーフです。

DTRTRAN(name4)
明示的なトランザクション・リソース定義で定義されていないトランザクションのルーティングに使用される DTR トランザクション定義の名前を返します。

name4
DTR トランザクション定義の名前を受け取る 4 バイトの位置の名前。DTRTRAN システム初期設定パラメーターで「NO」が指定された場合、「NO」はこのフィールドに入ります。

INQUIRE_DTRTRAN の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
DISASTER	ABEND
	LOGIC_ERROR
	LOOP
INVALID	INVALID_FUNCTION
KERNERROR	なし
PURGED	なし

INQUIRE_MXT 呼び出し

INQUIRE_MXT 関数は、DFHXMSRX マクロ呼び出しで提供されます。目的は、MXT パラメーターの現行値を提供することです。

INQUIRE_MXT

```
DFHXMSRX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION(INQUIRE_MXT),]
       [OUT,
       CURRENT_ACTIVE(name4 | (Rn) ),
       MXT_LIMIT(name4 | (Rn)),
```

```
MXT_QUEUED(name4 | (Rn) ),
TCLASS_QUEUED(name4 | (Rn) ),
RESPONSE (name1 | *),
REASON (name1 | *)]
```

このコマンドはスレッド・セーフです。

CURRENT_ACTIVE(name4 | (Rn))
現在のすべてのアクティブ・ユーザー・タスクの数を返します。

name4
アクティブ・ユーザー・タスクの現在の数を受け取る 4 バイトの位置の名前。バイナリー値で表されます。

(Rn) アクティブ・ユーザー・タスクの現在の数を受け取るレジスター。バイナリー値で表されます。

MXT_LIMIT(name4 | (Rn))
MXT パラメーターの現在の数を返します。

name4
現在許可されるすべてのユーザー・タスクの最大数を受け取る 4 バイトの位置の名前。バイナリー値で表されます。

(Rn) 現在許可されるすべてのタスクの最大数を受け取るレジスター。バイナリー値で表されます。

MXT_QUEUED(name4 | (Rn))
最大タスク (MXT) に達した結果、キューに入れられたユーザー・トランザクションの現在の数を返します。

name4
キューに入れられたユーザー・タスクの現在の数を受け取る 4 バイトの位置の名前。バイナリー値で表されます。

(Rn) キューに入れられたユーザー・タスクの現在の数を受け取るレジスター。バイナリー値で表されます。

TCLASS_QUEUED(name4 | (Rn))
トランザクション・クラスのメンバーシップのためにキューに入れられたすべてのトランザクションの現在の数を返します。

name4
キューに入れられたトランザクション・クラス・メンバーの現在の数を受け取る 4 バイトの位置の名前。バイナリー値で表されます。

(Rn) キューに入れられたトランザクション・クラス・メンバーの現在の数を受け取るレジスター。バイナリー値で表されます。

INQUIRE_MXT の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
DISASTER	LOGIC_ERROR
	ABEND
	LOOP
INVALID	INVALID_FUNCTION
KERNERROR	なし

RESPONSE	*REASON*
PURGED	なし

INQUIRE_TCLASS 呼び出し

INQUIRE_TCLASS 関数は、DFHXMCLX マクロ呼び出しで提供されます。目的は、指定のトランザクション・クラス (TCLASS) に関する現行情報を提供することです。

INQUIRE_TCLASS

```
DFHXMCLX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION(INQUIRE_TCLASS),
       INQ_TCLASS_NAME(name8 | string | 'string'),]
       [OUT,
       [CURRENT_ACTIVE(name4 | (Rn)),]
       [CURRENT_QUEUED(name4 | (Rn)),]
       [MAX_ACTIVE(name4 | (Rn)),]
       [PURGE_THRESHOLD(name4 | (Rn)),]
       RESPONSE (name1 | *),
       REASON (name1 | *)]
```

このコマンドはスレッド・セーフです。

CURRENT_ACTIVE(name4 | (Rn))
このトランザクション・クラス内のアクティブ・ユーザー・タスクの現在の数を返します。

name4
このトランザクション・クラスに関するアクティブ・ユーザー・タスクの現在の数を受け取る 4 バイトの位置の名前。バイナリー値で表されます。

(Rn) このトランザクション・クラスに関するアクティブ・ユーザー・タスクの現在の数を受け取るレジスター。バイナリー値で表されます。

CURRENT_QUEUED(name4 | (Rn))
キューに入れられたユーザー・タスクの現在の数を返します。

name4
このトランザクション・クラス内のキューに入れられたユーザー・タスクの現在の数を受け取る 4バイトの位置の名前。バイナリー値で表されます。

(Rn) キューに入れられたユーザー・タスクの現在の数を受け取るレジスター。バイナリー値で表されます。

INQ_TCLASS_NAME(name8 | string | 'string')
この照会のトランザクション・クラスの名前を指定します。

name8
照会中のトランザクション・クラスの名前が入る 8 バイトの位置の名前。

string トランザクション・クラスの名前を示す、間にブランクのない文字ストリング。

'string'
トランザクション・クラスの名前を示す、引用符で囲まれた文字ストリング。ストリングの長さは、引用符内にブランクを埋め込むことにより 8 に設定されます。

MAX_ACTIVE(name4 | (Rn))
トランザクション・クラスに許可されているアクティブ・タスクの現行最大数を返します。

name4
このトランザクション・クラスに現在許可されているアクティブ・タスクの現行最大数を受け取る 4 バイトの位置の名前。バイナリー値で表されます。

(Rn) このトランザクション・クラスに現在許可されているアクティブ・タスクの現行最大数を受け取るレジスター。バイナリー値で表されます。

PURGE_THRESHOLD(name4 | (Rn))
このトランザクション・クラスのパージしきい値限度を返します。

name4
このトランザクション・クラスに関する現在のパージしきい値限度を受け取る 4 バイトの位置の名前。バイナリー値で表されます。

(Rn) このトランザクション・クラスに関する現在のパージしきい値限度を受け取るレジスター。バイナリー値で表されます。

INQUIRE_TCLASS の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
DISASTER	LOGIC_ERROR
INVALID	なし
EXCEPTION	UNKNOWN_CLASS

INQUIRE_TRANDEF 呼び出し

INQUIRE_TRANDEF 関数は、DFHXMXDX マクロ呼び出しで提供されます。目的は、指定のトランザクション定義に関する情報を取得できるようにすることです。一般的には、この関数呼び出しは、多少の違いはありますが EXEC CICS INQUIRE TRANSACTION コマンドと等価です。

INQUIRE_TRANDEF

```
DFHXMXDX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION(INQUIRE_TRANDEF),
       INQ_TRANSACTION_ID(name4 | string | 'string'),]
       [OUT,
       [BREXIT(name8),]
       [CMDSEC(name1),]
       [DTIMEOUT(name4 | (Rn)),]
       [DUMP(name1),]
       [DYNAMIC(name1),]
       [INDOUBT(name1),]
       [INDOUBT_WAIT(name1),]
```

```
[INDOUBT_WAIT_TIME(name4),]
[INITIAL_PROGRAM(name8),]
[ISOLATE(name1),]
[LOCAL_QUEUING(name1),]
[OTSTIMEOUT(name4 | (Rn)),]
[PARTITIONSET(name1),]
[PARTITIONSET_NAME(name8),]
[PROFILE_NAME(name8),]
[REMOTE(name1),]
[REMOTE_NAME(name8),]
[REMOTE_SYSTEM(name4),]
[RESSEC(name1),]
[RESTART(name1),]
[ROUTABLE_STATUS(ROUTABLE|NOT_ROUTABLE),]
[RUNAWAY_LIMIT(name4 | (Rn)),]
[SHUTDOWN(name1),]
[SPURGE(name1),]
[STATUS(name1),]
[STORAGE_CLEAR(name1),]
[STORAGE_FREEZE(name1),]
[SYSTEM_ATTACH(name1),]
[SYSTEM_RUNAWAY(name1),]
[TASKDATAKEY(name1),]
[TASKDATALOC(name1),]
[TCLASS(name1),[TCLASS_NAME(name8),]]
[TPURGE(name1),]
[TRACE(name1),]
[TRAN_PRIORITY(name4 | (Rn)),]
[TRAN_ROUTING_PROFILE(name8),]
[TRANSACTION_ID(name4),]
[TWASIZE(name4 | (Rn)),]
RESPONSE (name1 | *),
REASON (name1 | *)]
```

このコマンドはスレッド・セーフです。

以下のパラメーターの説明では、INQUIRE_TRANDEF 呼び出しで返される指定可能な値を簡単に説明します。これらの一部のパラメーターの詳細については、TRANSACTION 属性に記載の TRANSACTION リソース定義の対応するパラメーターの説明を参照してください。

BREXIT(name8)
指定のトランザクションに対して指定されたデフォルトのブリッジ出口プログラムの名前を返します。ブリッジ出口が指定されていない場合には、ブランクが返されます。

name8
ブリッジ出口プログラムの名前を受け取る 8 バイトの位置の名前。

CMDSEC(name1)
トランザクションに対するコマンド・セキュリティー検査が必要かどうかを示す等価の値を 1 バイトの位置 (*name1*) に返します。

XMXD_YES
コマンド・セキュリティー検査が必要です。

XMXD_NO
コマンド・セキュリティー検査は必要ではありません。

DTIMEOUT(name4)
トランザクションのデッドロック・タイムアウト値を返します。

name4
デッドロック・タイムアウト値を受け取る 4 バイトの位置の名前。バイナリー値で表されます。

(Rn) デッドロック・タイムアウト値を受け取るレジスター。バイナリー値で表されます。

値がゼロの場合は、トランザクション・リソース定義により DTIMOUT(NO) が指定されていることを示します。

DUMP(name1)
トランザクションが異常終了した場合に、CICS でトランザクション・ダンプを取得するかしないかを示す等価の値を 1 バイトの位置 (*name1*) に返します。

XMXD_YES
トランザクション・ダンプが必要です。

XMXD_NO
トランザクション・ダンプは必要ではありません。

DYNAMIC(name1)
動的トランザクション・ルーティングに対してトランザクションを定義するかしないかを示す等価の値を 1 バイトの位置 (*name1*) に返します。

XMXD_YES
トランザクションはリモート CICS に動的にルーティングされます。

XMXD_NO
トランザクションは動的にルーティングされません。

INDOUBT(name1)
CICS 領域がコーディネーターとの接続に失敗するかまたはそれを失い、作業単位が未確定期間にある場合に実行されるアクションを示す等価の値を 1 バイトの位置 (*name1*) に返します。(このアクションは、TRANSACTION リソース定義の ACTION 属性に基づいています。)

アクションは、INDOUBT_WAIT および INDOUBT_WAIT_TIME で返される値によって異なります。INDOUBT_WAIT が XMXD_YES を返した場合、INDOUBT_WAIT_TIME で返される時間が満了するまでアクションは取られません。

XMXD_BACKOUT
トランザクションによってリカバリー可能リソースに対して行われたすべての変更がバックアウトされます。

XMXD_COMMIT
トランザクションによってリカバリー可能リソースに対して行われたすべての変更がコミットされます。

INDOUBT_WAIT(name1)
作業単位 (UOW) が未確定状態の間に障害が発生した場合、その作業単位がどのように応答するかを示す等価の値を 1 バイトの位置 (*name1*) に返します。

XMXD_NO
UOW は障害からのリカバリーを保留状態にして待機しません。 CICS はただちに TRANSACTION 定義の ACTION 属性で指定されたアクションを実行します。

XMXD_YES
UOW は、障害からのリカバリーを保留状態にしながら待機して、リカバリー可能リソースをバックアウトするかコミットするかを決定します。

INDOUBT_WAIT_TIME(name4)
未確定期間中の障害が発生してから、トランザクションが INDOUBT フィールドで返されたアクションをとるまでの時間の長さ (分) を返します。戻り値は、作業単位が未確定であり、INDOUBT_WAIT が XMXD_YES を返す場合にのみ有効です。

name4
遅延時間を受け取る 4 バイトの位置の名前。バイナリー値で表されます。

INDOUBT および INDOUBT_WAIT も参照してください。

INITIAL_PROGRAM(name8)
トランザクションの制御を与えられた初期プログラムの名前を返します。

name8
初期プログラムの名前を受け取る 8 バイトの位置の名前。

INQ_TRANSACTION_ID(name4 | string | 'string')
このトランザクション定義の照会用のトランザクション ID を指定します。

name4
トランザクション ID の名前が入る 4 バイトの位置の名前。

string トランザクション ID の名前を示す、間にブランクのない文字ストリング。

'string'
トランザクション ID の名前を示す、引用符で囲まれた文字ストリング。ストリングの長さは、引用符内にブランクを埋め込むことにより 4 に設定されます。

ISOLATE(name1)
トランザクションのタスク存続期間ユーザー・キー・ストレージにトランザクション分離が必要かどうかを示す等価の値を 1 バイトの位置 (*name1*) に返します。

XMXD_NO
タスク存続期間ユーザー・キー・ストレージに対するトランザクション分離は必要ではありません。

XMXD_YES
タスク存続期間ユーザー・キー・ストレージに対するトランザクション分離が必要です。

LOCAL_QUEUING(name1)
トランザクションが別のシステムで開始され、リモート・システムが使用不可の場合、このトランザクションに関する開始済み要求がローカルで待機できるかどうかを示す等価の値を 1 バイトの位置 (*name1*) に返します。

XMXD_NO
要求はローカルで待機しません。

XMXD_YES
要求はローカルで待機できます。

OTSTIMEOUT(name4)
Enterprise JavaBeans (EJB) 環境で作成され、この CICS トランザクションの下で実行される Object Transaction Service (OTS) トランザクションが、同期点をとる (または OTS トランザクションをロールバックする) OTS トランザクションのイニシエーターなしに実行することを許される期間 (秒単位) を返します。

name4
タイムアウト設定を受け取る 4 バイトの位置の名前。バイナリー値で表されます。

(Rn) タイムアウト設定を受け取るレジスター。バイナリー値で表されます。

値がゼロの場合は、トランザクション・リソース定義により OTSTIMEOUT(NO) が指定されていることを示します。

PARTITIONSET(name1)
トランザクション定義で指定された区分セットを示す等価の値を 1 バイトの位置 (*name1*) に返します。

XMXD_KEEP
区分セットに予約名 KEEP が指定されます。つまり、このトランザクション定義下で実行するタスクは、トランザクションに関連付けられている端末のアプリケーション区分セットを使用するということです。

XMXD_NAMED
区分セットは、トランザクション定義で厳密に指定されます。名前は PARTITIONSET_NAME パラメーターで返されます。

XMXD_NONE
トランザクション定義に指定された区分セットはありません。

XMXD_OWN
区分セットに予約名 OWN が指定されます。つまり、このトランザクション定義下で実行するタスクは、独自の区分セット管理を実行するということです。

PARTITIONSET_NAME(name8)
トランザクション定義で定義されている区分セットの名前を返します。

name8
区分セットの名前を受け取る 8 バイトの位置の名前。

PROFILE_NAME(name8)
トランザクション定義に関連付けられているプロファイル定義の名前を返します。

name8
トランザクション定義に関連付けられているプロファイル定義の名前を受け取る 8 バイトの位置の名前。

REMOTE(name1)
トランザクションがリモートとして定義されているかどうかを示す等価の値を 1 バイトの位置 (*name1*) に返します。

XMXD_NO
トランザクションはリモート・トランザクションではありません。

XMXD_YES
トランザクションはリモート・トランザクションです。

REMOTE_NAME(name8)
リモート・システムでトランザクションを認識するための名前を返します。

name8
トランザクションのリモート名を受け取る 8 バイトの位置の名前。

REMOTE_SYSTEM(name4)
トランザクション定義で指定されているリモート・システムの名前を返します。

DYNAMIC パラメーターが XMXD_YES を返した場合、REMOTE_SYSTEM はデフォルト名を返します。デフォルト名は、動的ルーティング・プログラムで変更できます。

DYNAMIC パラメーターが XMXD_NO を返した場合、これは、トランザクションのルーティング先である実際のリモート・システムです。

name4
リモート・システムの定義名を受け取る 4 バイトの位置の名前。

RESSEC(name1)
トランザクションに対するリソース・セキュリティー検査が必要かどうかを示す等価の値を 1 バイトの位置 (*name1*) に返します。

XMXD_NO
リソース・セキュリティー検査は必要ではありません。

XMXD_YES
リソース・セキュリティー検査が必要です。

RESTART(name1)
トランザクションをトランザクション再始動の対象とするかしないかを示す等価の値を 1 バイトの位置 (*name1*) に返します。

XMXD_NO
トランザクションを再始動できません。

XMXD_YES
トランザクションを再始動できます。

ROUTABLE_STATUS(ROUTABLE|NOT_ROUTABLE)
トランザクションが適格な EXEC CICS コマンドの対象である場合に、それが拡張ルーティング方式によってルーティングされるかどうかを示す値を返します。

NOT_ROUTABLE
トランザクションが START コマンドの対象である場合に、「従来の」メソッドを使用してルーティングされます。

ROUTABLE
トランザクションが適格なSTARTコマンドの対象である場合に、拡張方式を使用してトランザクションをルーティングします。

EXEC CICS STARTコマンドが呼び出すトランザクションの拡張方式 および「従来の」方式でのルーティングについて詳しくは、CICS トランザクション・ルーティングを参照してください。

RUNAWAY_LIMIT(name4 | (Rn))
トランザクション定義で指定されたランナウェイ・タスクの時間制限を返します。SYSTEM_RUNAWAY が XMXD_YES である場合、返される値は、**ICVR** システム初期設定パラメーターで定義されている値になります。

name4
タスク・ランナウェイ制限を受け取る 4 バイトの位置の名前。バイナリー値で表されます。

(Rn) タスク・ランナウェイ制限を受け取るレジスター。バイナリー値で表されます。

SHUTDOWN(name1)
CICS シャットダウン中にトランザクションを実行できるかどうかを示す等価の値を 1 バイトの位置 (*name1*) に返します。

XMXD_DISABLED
CICS シャットダウン中に、トランザクションを実行できなくなります。

XMXD_ENABLED
CICS シャットダウン中に、トランザクションを実行できます。

SPURGE(name1)
トランザクションがシステムによりパージ可能として定義されているかどうかを示す等価の値を 1 バイトの位置 (*name1*) に返します。

XMXD_NO
トランザクションはシステムによりパージできません。

XMXD_YES
トランザクションはシステムによりパージ可能です。

STATUS(name1)
トランザクション定義の状況を示す等価の値を 1 バイトの位置 (*name1*) に返します。

XMXD_DISABLED
トランザクション定義は使用不可です。

XMXD_ENABLED
トランザクション定義は使用可能です。

STORAGE_CLEAR(name1)
このトランザクション定義に関連付けられているタスクのタスク存続期間ストレージを FREEMAIN コマンドで解放する前にクリアするかどうかを示す等価の値を 1 バイトの位置 (*name1*) に返します。

XMXD_NO
タスク存続期間ストレージを解放する前にクリアする必要はありません。

XMXD_YES
タスク存続期間ストレージを解放する前にクリアする必要があります。

STORAGE_FREEZE(name1 | (Rn))

CICS 提供の技術員トランザクションで STGFRZ オプションを使用して、トランザクションにストレージの凍結を定義するかしないかを示す等価の値を 1 バイトの位置 (*name1*) に返します。

XMXD_NO

ストレージは、トランザクションの実行中に正常に解放されます。

XMXD_YES

トランザクションの実行中に正常に解放されたストレージが凍結します。

SYSTEM_ATTACH(name1)

この tranid に接続されているタスクを常にシステム・タスクとして接続するかどうかを示す等価の値を 1 バイトの位置 (*name1*) に返します。

XMXD_NO

このトランザクションには、ユーザー・タスクが接続されています。

XMXD_YES

このトランザクションには、システム・タスクが接続されています。

SYSTEM_RUNAWAY(name1)

トランザクション定義によりシステム・デフォルトのランナウェイ・タスクの時間制限を指定するかしないかを示す等価の値を 1 バイトの位置 (*name1*) に返します。ランナウェイ・タスクの時間制限は、**ICVR** システム初期設定パラメーターで指定されます。

XMXD_NO

トランザクションはシステム・ランナウェイ制限によって支配されません。

XMXD_YES

トランザクション定義はシステム・デフォルトのランナウェイ制限を指定します。

TASKDATAKEY(name1)

このトランザクション定義に関連付けられているタスクに関するタスク存続期間ストレージのストレージ・キーを示す等価の値を 1 バイトの位置 (*name1*) に返します。

XMXD_CICS

タスク存続期間ストレージに CICS キーが指定されます。

XMXD_USER

タスク存続期間ストレージに USER キーが指定されます。

TASKDATALOC(name1)

このトランザクション定義に関連付けられているタスクに関するタスク存続期間ストレージのデータ位置を示す等価の値を 1 バイトの位置 (*name1*) に返します。

XMXD_ANY

タスク存続期間ストレージは仮想ストレージの 16 MB を超える位置に配置できます。

XMXD_BELOW
タスク存続期間ストレージは仮想ストレージの 16 MB より下に配置しなければなりません。

TCLASS(name1)
トランザクションがトランザクション・クラスに属するかどうかを示す等価の値を 1 バイトの位置 (*name1*) に返します。

XMXD_NO
トランザクションはトランザクション・クラスのメンバーではありません。

XMXD_YES
トランザクションは、TCLASS_NAME パラメーターで指定されたトランザクション・クラス名のメンバーです。

TCLASS_NAME(name8)
トランザクションが属するトランザクション・クラスの名前を返します。

name8
トランザクションが属するトランザクション・クラスの名前を受け取る 8 バイトの位置の名前。

TPURGE(name1)
z/OS Communications Server 端末エラーが発生した場合に、トランザクションがパージ可能として定義されているかどうかを示す等価の値を 1 バイトの位置 (*name1*) に返します。

XMXD_NO
端末エラーが発生した場合にトランザクションをパージできません。

XMXD_YES
端末エラーが発生した場合にトランザクションをパージできます。

TRACE(name1)
トランザクションに対して定義されたトレースのレベルを示す等価の値を 1 バイトの位置 (*name1*) に返します。

XMXD_SPECIAL
CICS 特殊レベル・トレース。これは、EXEC CICS SET TRANSACTION コマンドを使用して設定されている特殊トレースの結果です。

XMXD_STANDARD
CICS 標準レベル・トレース。これは、TRANSACTION リソース定義の TRACE(YES) と等価です。

XMXD_SUPPRESSED
トランザクションに対するトレースを抑止します。これは、TRANSACTION リソース定義の TRACE(NO) と等価です。

TRAN_PRIORITY(name4 | (Rn))
トランザクション定義で指定されたトランザクション優先順位を返します。

name4
トランザクション優先順位を受け取る 4 バイトの位置の名前。バイナリー値で表されます。

(Rn) トランザクション優先順位を受け取るレジスター。バイナリー値で表されます。

TRAN_ROUTING_PROFILE(name8)
トランザクションをリモート・システムにルーティングするするために CICS が使用するプロファイルの名前を返します。

name8
トランザクション・ルーティング・プロファイル を受け取る 8 バイトの位置の名前。

TRANSACTION_ID(name4)
このトランザクション定義の照会用の 1 次トランザクション ID を返します。

name4
トランザクション ID の名前が入る 4 バイトの位置の名前。

TWASIZE(name4 | (Rn))
トランザクション定義で指定されたトランザクション作業域のサイズを返します。

name4
トランザクション作業域のサイズを受け取る 4 バイトの位置の名前。バイナリー値で表されます。

(Rn) トランザクション作業域のサイズを受け取るレジスター。バイナリー値で表されます。

INQUIRE_TRANDEF の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	UNKNOWN_TRANSACTION_ID
INVALID	なし
DISASTER	LOGIC_ERROR
PURGED	なし

INQUIRE_TRANSACTION 呼び出し

INQUIRE_TRANSACTION 関数は、DFHXMIQX マクロ呼び出しで提供されます。目的は、接続されるトランザクション (タスク) に関する情報を取得できるようにすることです。一般的には、この呼び出しは、多少の違いはありますが EXEC CICS INQUIRE TASK コマンドと等価です。

INQUIRE_TRANSACTION

```
DFHXMIQX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION(INQUIRE_TRANSACTION),
       [TRANSACTION_TOKEN(name8),]]
       [OUT,
       [ATTACH_TIME(name8),]
       [CICS_UOW_ID(name8),]
       [DTIMEOUT(name4 | (Rn)),]
       [DYNAMIC(name1),]
       [FACILITY_NAME(name4),]
       [FACILITY_TYPE(name1),]
```

```
[INITIAL_PROGRAM(name8),]
[NETNAME(name8),]
[ORIGINAL_TRANSACTION_ID(name4),]
[OUT_TRANSACTION_TOKEN(name8),]
[RE_ATTACHED_TRANSACTION(name1),]
[REMOTE(name1),]
[REMOTE_NAME(name8),]
[REMOTE_SYSTEM(name4),]
[RESOURCE_NAME(name16),]
[RESOURCE_TYPE(name8),]
[RESTART(name1),]
[RESTART_COUNT(name2 | (Rn)),]
[SPURGE(name1),]
[START_CODE(name1),]
[STATUS(name1),]
[SUSPEND_TIME(name4 | (Rn)),]
[SYSTEM_TRANSACTION(name1),]
[TASK_PRIORITY(name1),]
[TCLASS(name1),[TCLASS_NAME(name8),]]
[TERMINATE_PROTECTED(name1),]       [TPURGE(name1),]
[TRANNUM(name4 | string | 'string'),]
[TRAN_PRIORITY(name1),]
[TRAN_ROUTING_PROFILE(name8),]
[TRANSACTION_ID(name4),]
[USERID(name8),]
RESPONSE (name1 | *),
REASON (name1 | *)]
```

このコマンドはスレッド・セーフです。

以下のパラメーターの説明は、INQUIRE_TRANDEF 関数呼び出しの対応するパラメーターと同じです。

DTIMEOUT
DYNAMIC
INITIAL_PROGRAM
REMOTE
リモート名
REMOTE_SYSTEM
RESTART
SPURGE
STATUS
TCLASS
TRAN_ROUTING_PROFILE
TRANSACTION_ID

以下のパラメーターの説明では、INQUIRE_TRANSACTION 呼び出しで返される指定可能な値を簡単に説明します。これらのパラメーターの詳細については、以下に記載の TRANSACTION リソース定義の対応するパラメーターの説明を参照してください。TRANSACTION 属性.

ATTACH_TIME(name8)
タスクが接続されてから経過した時間をミリ秒単位で返します。

name8
時刻を パック 10 進数 ABSTIME 形式で受け取る 8 バイトの位置の名前。

CICS_UOW_ID(name8)
タスクの CICS 作業単位 ID を返します。

name8
作業単位 ID を受け取る 8 バイトの位置の名前。

FACILITY_NAME(name4)
タスクに関連付けられている基本機能の名前を返します。

name4
基本機能の名前を受け取る 4 バイトの位置の名前。

FACILITY_TYPE(name1)
タスクに関連付けられている基本機能のタイプを示す等価の値を 1 バイトの位置 (*name1*) に返します。

XMIQ_NONE
基本機能はありません。

XMIQ_START
基本機能はインターバル制御エレメントです。

XMIQ_TD
基本機能は一時データ・キューです。

XMIQ_TERMINAL
基本機能は端末です。

NETNAME(name8)
タスクに関連付けられている基本機能のネットワーク名を返します。

name8
ネットワーク名を受け取る 8 バイトの位置の名前。

ORIGINAL_TRANSACTION_ID(name4)
トランザクションの接続に使用されたトランザクション ID を返します。例えば、端末で別名が使用された場合、このフィールドはその別名を返します。

name4
元のトランザクション ID の名前を受け取る 4 バイトの位置の名前。

OUT_TRANSACTION_TOKEN(name8)
タスクを表すトークンを返します。

name8
タスクのトランザクション・トークンを受け取る 8 バイトの位置の名前。

RE_ATTACHED_TRANSACTION(name1)
トランザクションが再接続されたかどうかを示す等価の値を 1 バイトの位置 (*name1*) に返します。

XMIQ_NO
トランザクションは再接続されず、グローバル・ユーザー出口プログラムは、元のトランザクション接続と同じ環境で呼び出されます。

XMIQ_YES
トランザクションが再接続され、グローバル・ユーザー出口プログラムは、元のトランザクション接続とは別の環境で呼び出されます。

RESOURCE_NAME(name16)
(中断状態の) トランザクションが待機しているリソースの名前を返します。

name16
トランザクションが待機しているリソースの名前を受け取る 16 バイトの位置の名前。

RESOURCE_TYPE(name8)
(中断状態の) トランザクションが待機しているリソースのタイプを返します。

name8
トランザクションが待機しているリソースのタイプを受け取る 8 バイトの位置の名前。

RESTART_COUNT(name2 | (Rn))
このトランザクションのインスタンスが再始動された回数を返します。

name2
トランザクションが再始動された回数を受け取る 2 バイトの位置の名前。ハーフワードのバイナリー値で表されます。

(Rn) トランザクションが再始動された回数を受け取るレジスター。ハーフワードのバイナリー値で表されます。

START_CODE(name1)
タスクの開始方法を示す等価の値を 1 バイトの位置 (*name1*) に返します。

C CICS 内部接続。

XMIQ_DF
開始コードがまだ判明していないため、後で設定されます。

XMIQ_QD
一時データ・トリガー・レベルの接続。

XMIQ_S
データを伴わない START コマンド。

XMIQ_SD
データを伴う START コマンド。

XMIQ_SZ
フロントエンド・プログラミング・インターフェース (FEPI) 接続。

XMIQ_T
端末入力接続。

XMIQ_TT
永続トランザクション端末接続。

SUSPEND_TIME(name4 | (Rn))
タスクが現在中断状態にある時間の長さを返します。

name4
タスクが中断状態にある間の秒数 (端数切り捨て) を受け取る 4 バイトの位置の名前。バイナリー値で表されます。

(Rn) タスクが中断状態にある間の秒数 (端数切り捨て) を受け取るレジスター。バイナリー値で表されます。

SYSTEM_TRANSACTION(name1)
タスクが CICS システム・タスクであるかどうかを示す等価の値を 1 バイトの位置 (*name1*) に返します。

XMIQ_NO
タスクは CICS システム・タスクではありません。

XMIQ_YES
タスクは CICS システム・タスクです。

TASK_PRIORITY(name1)
結合されたタスク優先順位 (端末、トランザクション、およびオペレーターに対して定義された優先順位の合計) を返します。

name1
タスク優先順位を受け取る 1 バイトの位置の名前。バイナリーで表されます。

TERMINATE_PROTECTED(name1)
トランザクションの強制終了が可能かどうかを示す等価の値を 1 バイトの位置 (name1) に返します。

XMIQ_NO
トランザクションを強制終了できます。

XMIQ_YES
トランザクションを強制終了できません。

TRANNUM(name4)
トランザクションのタスク番号を返します。

name4
タスク番号を受け取る 4 バイトの位置の名前。

TRANSACTION_TOKEN(name8)
照会対象のタスクのトランザクション・トークンを指定します。このパラメーターはオプションであり、省略された場合は、現行タスクが想定されます。

XXMATT グローバル・ユーザー出口プログラム内でこの呼び出しを実行すると、現行タスクは CICS システム・タスクである場合があります。XXMATT を呼び出すユーザー・タスクを照会するには、XXMATT 出口固有のパラメーター・リストで渡されるトランザクション・トークンを指定する必要があります。

name8
トランザクション・トークンが入る 8 バイトの位置の名前。

USERID(name8)
このタスクと関連付けられたユーザー ID を返します。

name8
ユーザー ID を受け取る 8 バイトの位置の名前。

INQUIRE_TRANSACTION の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
DISASTER	ABEND
	LOOP
INVALID	なし
EXCEPTION	NO_TRANSACTION_ENVIRONMENT
	BUFFER_TOO_SMALL
	INVALID_TRANSACTION_TOKEN

RESPONSE	*REASON*
KERNERROR	なし

SET_TRANSACTION 呼び出し

SET_TRANSACTION 関数は、DFHXMIQX マクロ呼び出しで提供されます。目的は、現行タスクのタスク優先順位およびトランザクション・クラスを変更できるようにすることです。

この呼び出しを使用して TCLASS_NAME を変更できるのは、呼び出しが XXMATT グローバル・ユーザー出口プログラムから呼び出された場合のみであることに注意してください。

SET_TRANSACTION

```
DFHXMIQX  [CALL,]
       [CLEAR,]
       [IN,
       FUNCTION(SET_TRANSACTION),
       [TASK_PRIORITY(name4),]
       [TCLASS_NAME(name8),]
       [TRANSACTION_TOKEN(name8),]]
       [OUT,
       RESPONSE (name1 | *),
       REASON (name1 | *)]
```

このコマンドはスレッド・セーフです。

TASK_PRIORITY(name4)
TRANSACTION_TOKEN によって識別されるタスクに設定する新規のタスク優先順位を指定します。

name4
新規のタスク優先順位番号が入る 4 バイトの位置の名前。バイナリー値で表されます。

TCLASS_NAME(name8)
このタスクを関連付ける新規トランザクション・クラス名を指定します。タスクがトランザクション・クラスに含まれないように指定するには、特殊なデフォルト・システム名 DFHTCL00 を指定します。

name8
新規トランザクション・クラスの名前が入る 8 バイトの位置の名前。トランザクション・クラスがない場合は、このフィールドを DFHTCL00 に設定します。

TRANSACTION_TOKEN(name8)
変更対象のタスクを示すトランザクション・トークンを指定します。このパラメーターを省略すると、この呼び出しではデフォルトで現行タスクが使用されます。

name8
トランザクション・トークンが入る 8 バイトの位置の名前。

SET_TRANSACTION の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	NO_TRANSACTION_ENVIRONMENT
	UNKNOWN_TCLASS
	INVALID_TRANSACTION_TOKEN
DISASTER	ABEND
	LOOP
INVALID	なし
KERNERROR	なし

第 17 章 ユーザー・ジャーナリング XPI 関数

XPI は 1 つのユーザー・ジャーナリング関数 (DFHJCJCX 呼び出し WRITE_JOURNAL_DATA) を提供します。

制約事項: 以下のグローバル・ユーザー出口点から呼び出された出口プログラムでは、DFHJCJCX 呼び出しを使用することはできません。

- 統計ドメイン
- モニター・ドメイン
- ダンプ・ドメイン
- ディスパッチャー・ドメイン
- 一時データ・プログラム

WRITE_JOURNAL_DATA 呼び出し

WRITE_JOURNAL_DATA は、ジャーナル名に一致するジャーナル・モデル定義で指定されたジャーナルに単一のジャーナル・レコードを書き込みます (MVS システム・ロガー・ログ・ストリーム上のジャーナルか SMF データ・セット上のジャーナルです。定義で DUMMY が定義されている場合、レコードは書き込まれません)。

WRITE_JOURNAL DATA

```
DFHJCJCX [CALL,]
      [CLEAR,]
      [IN,
      FUNCTION(WRITE_JOURNAL_DATA),
      FROM(block-descriptor),
      JOURNALNAME(name8 | string | 'string' ) |
      JOURNAL_RECORD_ID(name2 | string | 'string'),
      WAIT(YES|NO),
      [RECORD_PREFIX(block-descriptor),]]
      [OUT,
      RESPONSE(name1 | *),
      REASON(name1 | *)]
```

このコマンドはスレッド・セーフです。

重要

初期設定段階での XPI の使用には制限があります。PLTPI の第 2 フェーズまでは、XPI 関数の TRANSACTION_DUMP、WRITE_JOURNAL_DATA、MONITOR、および INQUIRE_MONITOR_DATA を使用する出口プログラムを開始しないでください。PLTPI について詳しくは、Writing initialization and shutdown programsを参照してください。

FROM(block-descriptor)
ジャーナル・レコードのアドレスおよび長さを指定します。

© Copyright IBM Corp. 1977, 2018

ブロック記述子は、8 バイトのデータで構成されます。最初の 4 バイトに書き込み対象のデータのアドレスが保持されます。2 番目の 4 バイトにデータの長さが保持されます。ブロック記述子は、DFHJCJCX マクロ呼び出しによって、DFHJCJCY DSECT でマップされる場所 JCJC_FROM に移動されます。

JOURNALNAME(name8 | string | "string")
FROM データの書き込み先の CICS ジャーナルまたはログの名前を指定します。

JOURNAL_RECORD_ID(name2 | string | "string")
発信元の識別のためにジャーナル・レコードに書き込む 2 文字の値を指定します。

name2
2 バイトの位置の名前

string 生成されたコード内での長さが 2 に制限される文字ストリング

"string"
生成されたコード内での長さが 2 に制限される、引用符で囲まれた文字ストリング

RECORD_PREFIX(block-descriptor)
オプションのユーザー接頭部を指定します。

WAIT(YES|NO)
出口プログラムに制御を返す前に、レコードがジャーナルまたはログに書き込まれるまで CICS が待機するかどうかを指定します。

WRITE_JOURNAL_DATA の RESPONSE 値および REASON 値

RESPONSE	*REASON*
OK	なし
EXCEPTION	IO_ERROR
	JOURNAL_NOT_FOUND
	JOURNAL_NOT_OPEN
	LENGTH_ERROR
	STATUS_ERROR
DISASTER	なし
INVALID	なし
KERNERROR	なし
PURGED	なし

注: 詳しくは、XPI 呼び出しの実行 にある RESPONSE および REASON の説明を参照してください。

第 18 章 スレッド・セーフ XPI コマンド

大抵の場合 (すべての場合ではなく)、XPI コマンドはスレッド・セーフです。非スレッド・セーフ・コマンドを発行すると、CICS は QR TCB を使用して逐次化を確保します。

コマンド構文図では、スレッド・セーフの XPI コマンドに「このコマンドはスレッド・セーフです」というコメントを付けています。ここでは、スレッド・セーフのコマンドの一覧を示します。

スレッド・セーフ・コマンド

- DFHAPIQX INQ_APPLICATION_DATA
- DFHBRIQX INQUIRE_CONTEXT
- DFHDDAPX BIND_LDAP
- DFHDDAPX END_BROWSE_RESULTS
- DFHDDAPX FLUSH_LDAP_CACHE
- DFHDDAPX FREE_SEARCH_RESULTS
- DFHDDAPX GET_ATTRIBUTE_VALUE
- DFHDDAPX GET_NEXT_ATTRIBUTE
- DFHDDAPX GET_NEXT_ENTRY
- DFHDDAPX SEARCH_LDAP
- DFHDDAPX START_BROWSE_RESULTS
- DFHDDAPX UNBIND_LDAP
- DFHDSATX CHANGE_PRIORITY
- DFHDSSRX ADD_SUSPEND
- DFHDSSRX DELETE_SUSPEND
- DFHDSSRX RESUME
- DFHDSSRX SUSPEND
- DFHDSSRX WAIT_MVS
- DFHDUDUX SYSTEM_DUMP
- DFHJCJCX WRITE_JOURNAL_DATA
- DFHKEDSX START_PURGE_PROTECTION
- DFHKEDSX STOP_PURGE_PROTECTION
- DFHLDLDX ACQUIRE_PROGRAM
- DFHLDLDX DEFINE_PROGRAM
- DFHLDLDX DELETE_PROGRAM
- DFHLDLDX IDENTIFY_PROGRAM
- DFHLDLDX RELEASE_PROGRAM
- DFHLGPAX INQUIRE_PARAMETERS
- DFHLGPAX SET_PARAMETERS
- DFHMNMNX INQUIRE_MONITORING_DATA
- DFHMNMNX MONITOR
- DFHNQEDX DEQUEUE
- DFHNQEDX ENQUEUE
- DFHPGAQX INQUIRE_AUTOINSTALL
- DFHPGAQX SET_AUTOINSTALL

© Copyright IBM Corp. 1977, 2018

- DFHPGISX END_BROWSE_PROGRAM
- DFHPGISX GET_NEXT_PROGRAM
- DFHPGISX INQUIRE_CURRENT_PROGRAM
- DFHPGISX INQUIRE_PROGRAM
- DFHPGISX SET_PROGRAM
- DFHPGISX START_BROWSE_PROGRAM
- DFHSAIQX INQUIRE_SYSTEM
- DFHSAIQX SET_SYSTEM
- DFHSMMCX GETMAIN
- DFHSMMCX FREEMAIN
- DFHSMMCX INQUIRE_ELEMENT_LENGTH
- DFHSMMCX INQUIRE_TASK_STORAGE
- DFHSMSRX INQUIRE_ACCESS
- DFHSMSRX INQUIRE_SHORT_ON_STORAGE
- DFHSMSRX SWITCH_SUBSPACE
- DFHTRPTX TRACE_PUT
- DFHXMCLX INQUIRE_TCLASS
- DFHXMIQX INQUIRE_TRANSACTION
- DFHXMIQX SET_TRANSACTION
- DFHXMSRX INQUIRE_DTRTRAN
- DFHXMSRX INQUIRE_MXT
- DFHXMXDX INQUIRE_TRANDEF

非スレッド・セーフ・コマンド

- DFHDUDUX TRANSACTION_DUMP

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。この資料の他の言語版を IBM から入手できる場合があります。ただし、これを入手するには、本製品または当該言語版製品を所有している必要がある場合があります。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。 IBM 製品、プログラムまたはサービスに代えて、IBM の知的所有権を侵害することのない機能的に同等のプログラムまたは製品を使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒*103-8510*
東京都中央区日本橋箱崎町*19*番*21*号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様自身の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

© Copyright IBM Corp. 1977, 2018

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119 Armonk,
NY 10504-1785
United States of America

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名前はすべて架空のものであり、類似する個人や企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。これらのサンプル・プログラムは特定物として現存するままの状態で提供されるものであり、いかなる保証も提供されません。 IBM は、お客様の当該サンプル・プログラムの使用から生ずるいかなる損害に対しても一切の責任を負いません。

プログラミング・インターフェース情報

CICS には、プログラミング・インターフェースと見なすことのできる資料と、プログラミング・インターフェースと見なすことのできない資料があります。

オンライン製品資料の以下のセクションには、CICS Transaction Server for z/OS, バージョン 5 リリース 5 のサービスを取得するプログラムをお客様が作成するためのプログラミング・インターフェースが含まれています。

- アプリケーションの開発
- Developing system programs
- 保護の概要
- 外部インターフェースに向けた開発
- リファレンス: アプリケーション開発h
- リファレンス: システム・プログラミング
- リファレンス: 接続

オンライン製品資料の以下のセクションには、CICS Transaction Server for z/OS, バージョン 5 リリース 5 のプログラミング・インターフェースとして意図されていない (プログラミング・インターフェースと誤解される可能性のある) 情報が含まれています。

- Troubleshooting and support
- リファレンス: 診断

PDF 形式のマニュアルで CICS 資料にアクセスする場合は、CICS Transaction Server for z/OS, バージョン 5 リリース 5 のサービスを取得するプログラムをお客様が作成するためのプログラミング・インターフェースが以下のマニュアルに含まれています。

- アプリケーション・プログラミング・ガイドおよびアプリケーション・プログラミング・リファレンス
- Business Transaction Services
- Customization Guide
- C++ OO Class Libraries
- Debugging Tools Interfaces Reference
- Distributed Transaction Programming Guide
- External Interfaces Guide
- Front End Programming Interface Guide
- IMS Database Control Guide
- インストール・ガイド
- セキュリティー・ガイド
- Supplied Transactions
- CICSPlex® SM Managing Workloads
- CICSPlex SM Managing Resource Usage
- CICSPlex SM アプリケーション・プログラミング・ガイドおよび CICSPlex SM アプリケーション・プログラミング・リファレンス
- Java™ Applications in CICS

PDF 形式のマニュアルで CICS 資料にアクセスする場合は、CICS Transaction Server for z/OS, バージョン 5 リリース 5 のプログラミング・インターフェース

として意図されていない (プログラミング・インターフェースと誤解される可能性のある) 情報が以下のマニュアルに含まれています。

- Data Areas
- Diagnosis Reference
- Problem Determination Guide
- CICSPlex SM Problem Determination Guide

商標

IBM、IBM ロゴおよび ibm.com® は、世界の多くの国で登録された International Business Machines Corporation の商標または登録商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、 http://www.ibm.com/legal/copytrade.shtml をご覧ください。

Adobe、Adobe ロゴ、PostScript、PostScript ロゴは、Adobe Systems Incorporated の米国およびその他の国における登録商標または商標です。

インテル、Intel、Intel ロゴ、Intel Inside、Intel Inside ロゴ、Intel Centrino、Intel Centrino ロゴ、Celeron、Intel Xeon、Intel SpeedStep、Itanium、および Pentium は、Intel Corporation または子会社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。

Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社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。

Linux は、Linus Torvalds の米国およびその他の国における登録商標です。

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

UNIX は The Open Group の米国およびその他の国における登録商標です。

製品資料に関するご使用条件

これらの資料は、以下のご使用条件に同意していただける場合に限りご使用いただけます。

適用範囲
: IBM Web サイトの「ご利用条件」に加えて、以下のご使用条件が適用されます。

個人使用
: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、非商業的な個人による使用目的に限り複製することができます。ただし、IBM の明示的な承諾をえずに、これらの資料またはその一部について、二次的著作物を作成したり、配布 (頒布、送信を含む) または表示 (上映を含む) することはできません。

商用使用
: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、お客様の企業内に限り、複製、配布、および表示することがで

きます。ただし、IBM の明示的な承諾をえずにこれらの資料の二次的著作物を作成したり、お客様の企業外で資料またはその一部を複製、配布、または表示することはできません。

権利　ここで明示的に許可されているもの以外に、資料や資料内に含まれる情報、データ、ソフトウェア、またはその他の知的所有権に対するいかなる許可、ライセンス、または権利を明示的にも黙示的にも付与するものではありません。

資料の使用が IBM の利益を損なうと判断された場合や、上記の条件が適切に守られていないと判断された場合、IBM はいつでも自らの判断により、ここで与えた許可を撤回できるものとさせていただきます。

お客様がこの情報をダウンロード、輸出、または再輸出する際には、米国のすべての輸出入関連法規を含む、すべての関連法規を遵守するものとします。

IBM これらの資料の内容についていかなる保証もしません。これらの資料は、特定物として現存するままの状態で提供され、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任なしで提供されます。

IBM オンラインでのプライバシー・ステートメント

サービス・ソリューションとしてのソフトウェアも含めた IBM ソフトウェア製品 (「ソフトウェア・オファリング」) では、製品の使用に関する情報の収集、エンド・ユーザーの使用感の向上、エンド・ユーザーとの対話またはその他の目的のために、Cookie はじめさまざまなテクノロジーを使用することがあります。多くの場合、ソフトウェア・オファリングにより個人情報が収集されることはありません。IBM の「ソフトウェア・オファリング」の一部には、個人情報を収集できる機能を持つものがあります。ご使用の「ソフトウェア・オファリング」が、これらの Cookie およびそれに類するテクノロジーを通じてお客様による個人情報の収集を可能にする場合、以下の具体的事項をご確認ください。

CICSPlex SM Web ユーザー・インターフェース **(**メイン・インターフェース**)** の場合**:**　このソフトウェア・オファリングは、展開される構成に応じて、セッション管理、認証、お客様の利便性の向上、または利用の追跡または機能上の目的のために、それぞれのお客様のユーザー名、およびその他の個人情報を、セッションごとの Cookie および持続的な Cookie を使用して収集する場合があります。これらの Cookie を無効にすることはできません。

CICSPlex SM Web ユーザー・インターフェース **(**データ・インターフェース**)** の場合**:**　このソフトウェア・オファリングは、展開される構成に応じて、セッション管理、認証、または利用の追跡または機能上の目的のために、それぞれのお客様のユーザー名またはその他の個人情報を、セッションごとの Cookie を使用して収集する場合があります。これらの Cookie を無効にすることはできません。

CICSPlex SM Web ユーザー・インターフェース **(**「**Hello World**」ページ**)** の場合**:**　このソフトウェア・オファリングは、展開される構成に応じて、個人情報を収集しないセッションごとの Cookie を使用する場合があります。これらの Cookie を無効にすることはできません。

CICS Explorer® の場合:

このソフトウェア・オファリングは、展開される構成に応じて、セッション管理、お客様の利便性の向上、または利用の追跡または機能上の目的のために、それぞれのお客様のユーザー名、およびその他の個人情報を、セッションごとの設定および持続的な設定を使用して収集する場合があります。これらの設定を無効にすることはできませんが、ユーザー・パスワードの暗号化形式でのディスクへの保管は、サインオン中にチェック・ボックスにチェック・マークを付けることによるユーザーの明示的な操作によってのみ有効化することができます。

この「ソフトウェア・オファリング」が Cookie およびさまざまなテクノロジーを使用してエンド・ユーザーから個人を特定できる情報を収集する機能を提供する場合、 お客様は、このような情報を収集するにあたって適用される法律、ガイドライン等を遵守する必要があります。これには、エンドユーザーへの通知や同意の要求も含まれますがそれらには限られません。

このような目的での Cookie を含む様々なテクノロジーの使用の詳細については、『IBM オンラインでのプライバシー・ステートメント』(http://www.ibm.com/privacy/details/jp/ja/) の『クッキー、ウェブ・ビーコン、その他のテクノロジー』および「IBM Software Products and Software-as-a-Service Privacy Statement」(http://www.ibm.com/software/info/product-privacy) を参照してください。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

［ア行］

アプリケーション・コンテキスト・データ 57

［サ行］

システム初期設定パラメーター
  RENTPGM 46, 86
スレッド・セーフ XPI コマンド 141

［マ行］

モニター
  XPI の関数 57

D

DFHAPIQX マクロ 95
DFHBABRX マクロ 1
DFHBRIQX マクロ 117
DFHDDAPX マクロ 3, 5, 6, 7, 8, 9, 10, 12, 13
DFHDSATX マクロ 15
DFHDSSRX マクロ 15
DFHDUDUX マクロ 31
DFHJCJCX マクロ 139
DFHKEDSX マクロ 41
DFHLDLDX マクロ 43
DFHLGPAX マクロ 55
DFHMNMNX マクロ 57
DFHNQEDX マクロ 37
DFHOTCOX マクロ 69
DFHOTTRX マクロ 65, 66, 67, 68, 69
DFHPGAQX マクロ 73
DFHPGCHX マクロ 73
DFHPGISX マクロ 73
DFHSAIQX マクロ 97, 102
DFHSMMCX マクロ 105
DFHSMSRX マクロ 110
DFHTRPTX マクロ 115
DFHXMCLX マクロ 121
DFHXMIQX マクロ 131, 136
DFHXMSRX マクロ 119
DFHXMXDX マクロ 122

E

ENQUEUE 37

L

LIBRARYDSN オプション
  INQUIRE_CURRENT PROGRAM コマンド 84

R

RENTPGM、システム初期設定パラメーター 46, 86

X

XPI (出口プログラミング・インターフェース)
  エンキュー・ドメイン関数
    DEQUEUE 37
    ENQUEUE 37
  カーネル・ドメイン関数
    START_PURGE_PROTECTION 41
    STOP_PURGE_PROTECTION 41
  ジャーナル処理機能
    WRITE JOURNAL DATA 139
  状態データ・アクセス関数
    INQUIRE_SYSTEM 97
    INQ_APPLICATION_DATA 95
    SET_SYSTEM 102
  ストレージ管理機能
    FREEMAIN 108
    GETMAIN 105
    INQUIRE_ACCESS 109
    INQUIRE_ELEMENT_LENGTH 110
    INQUIRE_SHORT_ON_STORAGE 110
    INQUIRE_TASK_STORAGE 111
    SWITCH_SUBSPACE 112
  スレッド・セーフ・コマンド 141
  ダンプ管理機能
    トランザクション・ダンプ 32
    SYSTEM DUMP 31
  ディスパッチャー関数
    ADD SUSPEND 17
    CHANGE PRIORITY 19
    DELETE SUSPEND 20
    RESUME 21
XPI (出口プログラミング・インターフェース) (続き)
  ディスパッチャー関数 (続き)
    SUSPEND 22
    WAIT_MVS 26
  ディレクトリー・ドメイン関数
    BIND_LDAP 3
    END_BROWSE_RESULTS 5
    FLUSH_LDAP_CACHE 6
    FREE_SEARCH_RESULTS 6
    GET_ATTRIBUTE_VALUE 7
    GET_NEXT_ATTRIBUTE 8
    GET_NEXT_ENTRY 9
    INQUIRE_ACTIVATION 1
    SEARCH_LDAP 10
    START_BROWSE_RESULTS 12
    UNBIND_LDAP 13
  トランザクション管理関数
    COMMIT 68
    COMMIT_ONE_PHASE 66
    IMPORT_TRAN 65
    INQUIRE_CONTEXT 117
    INQUIRE_DTRTRAN 119
    INQUIRE_MXT 119
    INQUIRE_TCLASS 121
    INQUIRE_TRANDEF 122
    INQUIRE_TRANSACTION 131
    PREPARE 67
    ROLLBACK 68
    SET_COORDINATOR 69
    SET_ROLLBACK_ONLY 69
    SET_TRANSACTION 136
  トレース制御機能
    TRACE_PUT 115
  プログラム管理関数
    BIND_CHANNEL 94
    END_BROWSE_PROGRAM 91
    GET_NEXT_PROGRAM 89
    INQUIRE_AUTOINSTALL 91
    INQUIRE_CURRENT_PROGRAM 82
    INQUIRE_PROGRAM 73
    SET_AUTOINSTALL 93
    SET_PROGRAM 85
  モニター関数
    INQUIRE APP CONTEXT 57
    INQUIRE MONITORING DATA 59
    MONITOR 60
    SET_TRACKING_DATA 63
  ローダー関数
    ACQUIRE PROGRAM 43

© Copyright IBM Corp. 1977, 2018

XPI (出口プログラミング・インターフェース) (続き)
ローダー関数 (続き)
DEFINE PROGRAM 46
DELETE PROGRAM 49
IDENTIFY_PROGRAM 50
RELEASE PROGRAM 52
ログ・マネージャー機能
INQUIRE_PARAMETERS 55
SET_PARAMETERS 55
XPI の ACQUIRE PROGRAM 関数 43
XPI の ADD SUSPEND 関数 17
XPI の BIND_CHANNEL 関数 94
XPI の BIND_LDAP 関数 3
XPI の CHANGE PRIORITY 関数 19
XPI の COMMIT 関数 68
XPI の COMMIT_ONE_PHASE 関数 66
XPI の DEFINE PROGRAM 関数 46
XPI の DELETE PROGRAM 関数 49
XPI の DELETE SUSPEND 関数 20
XPI の DEQUEUE 関数 37
XPI の END_BROWSE_PROGRAM 関数 91
XPI の END_BROWSE_RESULTS 関数 5
XPI の FLUSH_LDAP_CACHE 関数 6
XPI の FREEMAIN 機能 108
XPI の FREE_SEARCH_RESULTS 関数 6
XPI の GETMAIN 関数 105
XPI の GET_ATTRIBUTE_VALUE 関数 7
XPI の GET_NEXT_ATTRIBUTE 関数 8
XPI の GET_NEXT_ENTRY 関数 9
XPI の GET_NEXT_PROGRAM 関数 89
XPI の IDENTIFY_PROGRAM 関数 50
XPI の IMPORT_TRAN 関数 65
XPI の INQUIRE APP CONTEXT 関数 57
XPI の INQUIRE MONITORING DATA 機能 59
XPI の INQUIRE_ACCESS 関数 109
XPI の INQUIRE_ACTIVATION 機能 1
XPI の INQUIRE_AUTOINSTALL 関数 91
XPI の INQUIRE_CONTEXT 関数 117
XPI の INQUIRE_CURRENT_PROGRAM 関数 82
XPI の INQUIRE_DTRTRAN 関数 119
XPI の INQUIRE_ELEMENT_LENGTH 関数 110
XPI の INQUIRE_MXT 関数 119
XPI の INQUIRE_PARAMETERS 関数 55
XPI の INQUIRE_PROGRAM 関数 73
XPI の INQUIRE_SHORT_ON_STORAGE 機能 110
XPI の INQUIRE_SYSTEM 関数 97
XPI の INQUIRE_TASK_STORAGE 関数 111
XPI の INQUIRE_TCLASS 関数 121
XPI の INQUIRE_TRANDEF 関数 122
XPI の INQUIRE_TRANSACTION 関数 131
XPI の INQ_APPLICATION_DATA 関数 95
XPI の MONITOR 関数 60
XPI の PREPARE 関数 67
XPI の RELEASE PROGRAM 関数 52
XPI の RESUME 関数 21
XPI の ROLLBACK 関数 68
XPI の SEARCH_LDAP 関数 10
XPI の SET_AUTOINSTALL 関数 93
XPI の SET_COORDINATOR 関数 69
XPI の SET_PARAMETERS 関数 55
XPI の SET_PROGRAM 関数 85
XPI の SET_ROLLBACK_ONLY 関数 69
XPI の SET_SYSTEM 機能 102
XPI の SET_TRACKING_DATA 関数 63
XPI の SET_TRANSACTION 関数 136
XPI の START_BROWSE_RESULTS 関数 12
XPI の START_PURGE_PROTECTION 関数 41
XPI の STOP_PURGE_PROTECTION 機能 41
XPI の SUSPEND 関数 22
XPI の SWITCH_SUBSPACE 関数 112
XPI の SYSTEM DUMP 関数 31
XPI の TRACE_PUT 機能 115
XPI の TRANSACTION DUMP 関数 32
XPI の UNBIND_LDAP 機能 13
XPI の WRITE JOURNAL DATA 機能 139
XPI のエンキュー・ドメイン関数 37
XPI のオブジェクト・トランザクション関数 65
XPI のカーネル・ドメイン関数 41
XPI の状態データ・アクセス関数 95
XPI のストレージ管理機能 105, 110
XPI のダンプ管理関数 31
XPI のディスパッチャー関数 15
XPI のディレクトリー・ドメイン関数 3
XPI のトランザクション管理関数 117
XPI のトレース制御関数 115
XPI のビジネス・アプリケーション・マネージャー・ドメイン関数 1
XPI のプログラム管理関数 73
XPI のユーザー・ジャーナリング関数 139
XPI のローダー関数 43
XPI のログ・マネージャー関数 55

日本アイ･ビー･エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21