

CICS Transaction Server for z/  
OSバージョン 5 リリース 6

*CICSplex SM* アプリケーション・プログラ  
ミング・ガイド



## 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、[製品の特記事項](#)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 6 (製品番号 5655-Y305655-BTA)、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  
Version 5 Release 5  
CICSplex SM Application Programming Guide

### 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 International Business Machines Corporation 1974, 2020.

# 目次

PDF について.....	vii
<b>第 1 章 CICSplex SM API の概要.....</b>	<b>1</b>
サポートされている環境と言語.....	1
使用可能なインターフェース.....	2
CICSplex SM への接続.....	2
接続プロセス.....	4
セキュリティーに関する考慮事項.....	5
環境間での互換性.....	6
CICSplex SM のリリース間の互換性.....	6
REXX アプリケーションについての特別な考慮事項.....	8
新規リリースのリソース・テーブルへのアクセス.....	8
以前のリリースのリソース・テーブルへのアクセス.....	9
サンプル・プログラム.....	10
<b>第 2 章 CICSplex SM API の使用方法.....</b>	<b>11</b>
CICSplex SM 管理対象オブジェクト.....	11
管理対象オブジェクトのタイプ.....	11
CICSplex SM リソース・テーブル.....	13
カスタマイズされたリソース・テーブル・レコードの作成.....	14
カスタマイズしたリソース・テーブル・レコードに対するコピーブックの作成方法.....	15
管理対象オブジェクトの選択.....	16
コンテキストおよびスコープの設定.....	16
フィルター式の使用.....	17
結果セットの操作.....	20
結果セット・コマンドの概要.....	21
結果セットからのレコードの取得.....	24
結果セット内のレコード・ポインターの配置.....	27
結果セット内の選択されたレコードの処理.....	28
結果セット内のレコードの集計.....	30
結果セット内のレコードのソート.....	34
管理対象リソースの変更.....	35
リリース属性の変更.....	35
リソースに対するアクションの実行.....	36
CICSplex SM と CICS の定義の操作.....	37
非同期処理.....	45
LISTEN コマンドの使用.....	46
NOWAIT オプションの使用.....	46
トークンを使用した要求の識別.....	46
ADDRESS コマンドの使用.....	47
RECEIVE コマンドの使用.....	47
CICSplex SM トークンの使用.....	48
メタデータ・リソース・テーブルの使用.....	48
ATTR.....	49
ATTRAVA.....	57
METADESC.....	57
METANAME.....	59
METAPARM.....	60
OBJACT.....	62
OBJECT.....	64



INSTALLAGENT、INSTALLTIME、 INSTALLUSRID、および BASDEFINEVER の各属性.....	103
FEEDBACK 属性の処理.....	103
<b>第 6 章 REXX エラー処理.....</b>	<b>105</b>
変換エラー.....	105
ランタイム・エラー.....	106
TPARSE と TBUILD のエラー.....	106
内のテキスト.....	106
EYU_TRACE データ.....	107
<b>付録 A BINCONRS、BINCONSC、および BINSTERR のエラー・コード.....</b>	<b>109</b>
<b>付録 B CICSplex SM API サンプル・プログラム.....</b>	<b>111</b>
EYU#API1.....	111
EYUCAPI2.....	113
EYUAAPI3.....	119
EYULAPI4 .....	131
<b>特記事項.....</b>	<b>141</b>
<b>索引.....</b>	<b>147</b>



## PDF について

---

この PDF では、CICS Transaction Server for z/OS の CICSplex SM エLEMENTのサービスにアクセスするアプリケーションを記述する方法について説明します。使用されているコマンドに関する参照情報は、「*CICSplex SM アプリケーション・プログラミング・リファレンス*」という PDF を参照してください。

本書で使用されている用語や表記について詳しくは、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、2020 年 5 月 28 日に作成されました。





# 第 1 章 CICSplex SM API の概要

CICSplex® SM アプリケーション・プログラミング・インターフェース (API) を使用すると、CICS システム管理情報にアクセスでき、外部プログラムから CICSplex SM サービスを呼び出すことができます。

API は、企業内の CICS システムをモニターおよび制御するように設計されたプログラム用の単一インターフェースを提供できます。さらに、この API は CICSplex SM 自体へのインターフェースも提供します。したがって、CICSplex SM の動作方法を制御する管理機能にアクセスするためのプログラムを作成することもできます。

この API の一般的な用途は次のとおりです。

- CICS 環境内の主要なリソースをモニターします。
- 企業内の他の条件を基準にして CICS リソースのステータスを変更します。
- CICS 環境に対する変更の流れを制御します。
- CICSplex SM によって提供される情報を自動化製品に渡します。
- CICS データと CICSplex SM データの代替の表示形式と報告形式を開発します。
- 次のようなイベントに関する CICSplex SM 通知を処理します。
  - リアルタイム分析しきい値に達した
- ビジネス・アプリケーション・サービス、ワークロード管理、リアルタイム分析、およびリソース・モニターの CICSplex SM 定義を作成および維持します。
- CICSplex SM データ・リポジトリ内の CICS リソース定義を作成および維持します。

照会構造に応じて、CICSplex SM API コマンドを CMAS で実行してそのコマンドの目的を達成したり、CICSplex SM API コマンドをその範囲内のすべての MAS に経路指定したりできます。要求が MAS に経路指定されると、その要求は、CONL トランザクション ID を使用するシステム・タスクによって処理されるか、CONA トランザクション ID を使用するシステム・タスクのインスタンスにオフロードされます (そのように構成されている場合)。

注：デフォルトでは、MAS に転送された CICSplex SM API 要求は 255 という優先度で実行されるため、多くの CICSplex SM API 要求を頻繁に発行すると、それより低い優先度で実行されているワークロードが影響を受ける可能性があります。ただし、CICSplex SM API 要求の優先度を下げる方法があります。

- CICSplex SM トランザクション CONL は優先度 255 (変更不可) で実行されます。デフォルトでは、このトランザクションは API、WUI、および RTA を通じて MAS に転送されたほとんどの要求を処理します。
- CICSplex SM API 要求をトランザクション CONA にオフロードでき、EYUPARM で **MASALTLRTCNT** を指定することで、これらの要求の優先度を変更できます。詳しくは、[MAS における長時間実行タスクの数の制御](#)を参照してください。

RTA の EVALDEF で別個のタスクを要求することで、COIR トランザクション ID を使用するシステム・タスクに一部の RTA 処理をオフロードすることもできます。**COIRTASKPRI** という CICSplex SM システム・パラメーターを使用して、COIR システム・タスクの優先度を調整します。詳しくは、[CICSplex SM システム・パラメーター](#)を参照してください。

CICSplex SM では、いくつかのサンプル・プログラムが提供されています。各サポート言語のサンプル・プログラムがソース形式で配布されています。これらのサンプルの目的は、ユーザーが作成できるプログラムのタイプと、それらのプログラムで使用する必要のあるコマンドを示すことです。詳しくは、[CICSplex SM API サンプル・プログラム](#)を参照してください。

## サポートされている環境と言語

この API は、さまざまな環境で実行されているプログラムから呼び出すことができます。

- z/OS バッチ
- TSO
- IBM Tivoli® NetView®

- CICS Transaction Server for z/OS の CICS 要素

注: CICSplex SM API を NetView RODM メソッド内から呼び出すことはできません。RODM メソッド・サービスに適用される制約事項について詳しくは、[IBM Tivoli NetView for z/OS リソース・オブジェクト・データ・マネージャーおよび GMFHS プログラマーズ・ガイド](#)を参照してください。

## 使用可能なインターフェース

CICSplex SM では、API ユーザー用に 2 つのインターフェースが提供されています。

### コマンド・レベル・インターフェース

このインターフェースは CICS 変換プログラムを使用して EXEC CPSM ステートメントを受け入れ、これらのステートメントをソース言語の適切な命令シーケンスに変換します。これらの命令はその後、CICSplex SM によって提供されるインターフェース・スタブ・ルーチンにリンクされます。

コマンド・レベル・インターフェースは、以下の言語で記述されたプログラムに使用できます。

- Assembler バージョン 2 以降
- OS PL/I Optimizing Compiler バージョン 2.3 以降
- COBOL Compiler バージョン 1.3.2 以降
- C バージョン 2.1. 以降

[2 ページの表 1](#) は、各環境のコマンド・レベル・インターフェースでサポートされている言語を示しています。

表 1. コマンド・レベル・インターフェースでサポートされているプログラミング言語				
環境	アセンブラ	COBOL	PL/I	C
CICS TS	はい	はい	はい	はい
MVS™ Batch	はい	はい	はい	はい
MVS TSO	はい	はい	はい	はい
MVS NetView	はい		はい	はい

### ランタイム・インターフェース

ランタイム・インターフェースは、以下の MVS 環境で REXX EXEC として作成されたプログラムをサポートしています。

- バッチ
- TSO
- NetView

このインターフェースは、CICSplex SM によって提供されている REXX 関数パッケージで構成されます。この関数パッケージはテキスト・ストリング形式のコマンドを受け入れ、適切な API 呼び出しを生成します。

## CICSplex SM への接続

CICSplex SM API プログラムは、3 つの環境に存在しているプログラムとして、または 3 つの環境にアクセスできるプログラムとして見なすことができます。

### ユーザー環境

プログラム自体とその実行環境 (MVS や CICS など)。

### CICSplex SM 環境

CICSplex SM によって維持されるデータと、プログラムに提供されるサービス。

### 管理対象リソース環境

CICSplex SM によって管理されるリソースと、プログラムがアクセスできるリソース。

プログラムから CICSplex SM 環境およびその環境で管理されるリソースにアクセスするには、まず CICSplex SM への接続を確立する必要があります。この接続は API 処理スレッドと呼ばれ、次の 2 つの基本的な役割を果たします。

- スレッドが作成されると、ユーザーが特定されて、プログラム動作のセキュリティ検証とセキュリティ監査を透過的に実行できるようになります。
- 一部の API 関数の間には暗黙的な関係があり、それらの関係はスレッド・レベルで維持されます。各スレッドは固有の API ユーザーと見なされ、どのリソースもスレッドの境界を超えることはできません。

スレッドが作成されると、プログラムはローカル CMAS のコンテキスト内でコマンドを発行できます。ローカル CMAS は、接続コマンドの発行場所と発行方法によって決まります。

- CICS システム内で発行された場合は、その CICS システムに対するローカル CMAS がローカル CMAS となります。
- バッチ・ジョブとして発行されて、CMAS が明示的に指定されていない場合は、最後に開始された CMAS がローカル CMAS です。
- バッチ・ジョブとして発行されて、MVS イメージ内の CMAS が CONNECT コマンドに含まれている場合は、その CMAS がローカル CMAS です。

ローカル CMAS 以外の CMAS のデータを参照できますが、それらを直接指すようにコンテキストを変更することはできません。

単純な API プログラムでは、単一スレッドのみが確立されます。ユーザーがそのスレッドを確立して、操作を実行してから、そのスレッドを終了できます。複雑なプログラムでは複数の同時スレッドが維持されることがあり、それらのスレッドによって単一スレッド上では禁止される並列操作が実行されたり、コマンドと結果の相関関係が簡素化されたりします。

以下のコマンドを使用して API スレッドを管理できます。

#### **CONNECT**

CICSplex SM への接続を確立して、API 処理スレッドを定義し、このスレッドのデフォルト設定を提供します。このスレッドは、API セッションをサポートしている CMAS によって維持されます。

#### **DISCONNECT**

CICSplex SM から API 処理スレッドを切断して、このスレッドに関連付けられたすべてのリソースを解放します。

#### **QUALIFY**

当該スレッドによって発行される後続コマンドの CICSplex SM コンテキストと CICSplex SM スコープを定義します。

#### **TERMINATE**

このコマンドを発行する CICS タスクまたは MVS タスクによって作成されたすべてのスレッド上のすべての API 処理を終了します。

これらのコマンドは、ユーザー環境 (プログラム) と CICSplex SM の間の接続を管理します。これらのコマンドは管理対象リソースには影響を与えません。[4 ページの図 1](#) は、これらのコマンドが API 環境に与える影響を示しています。

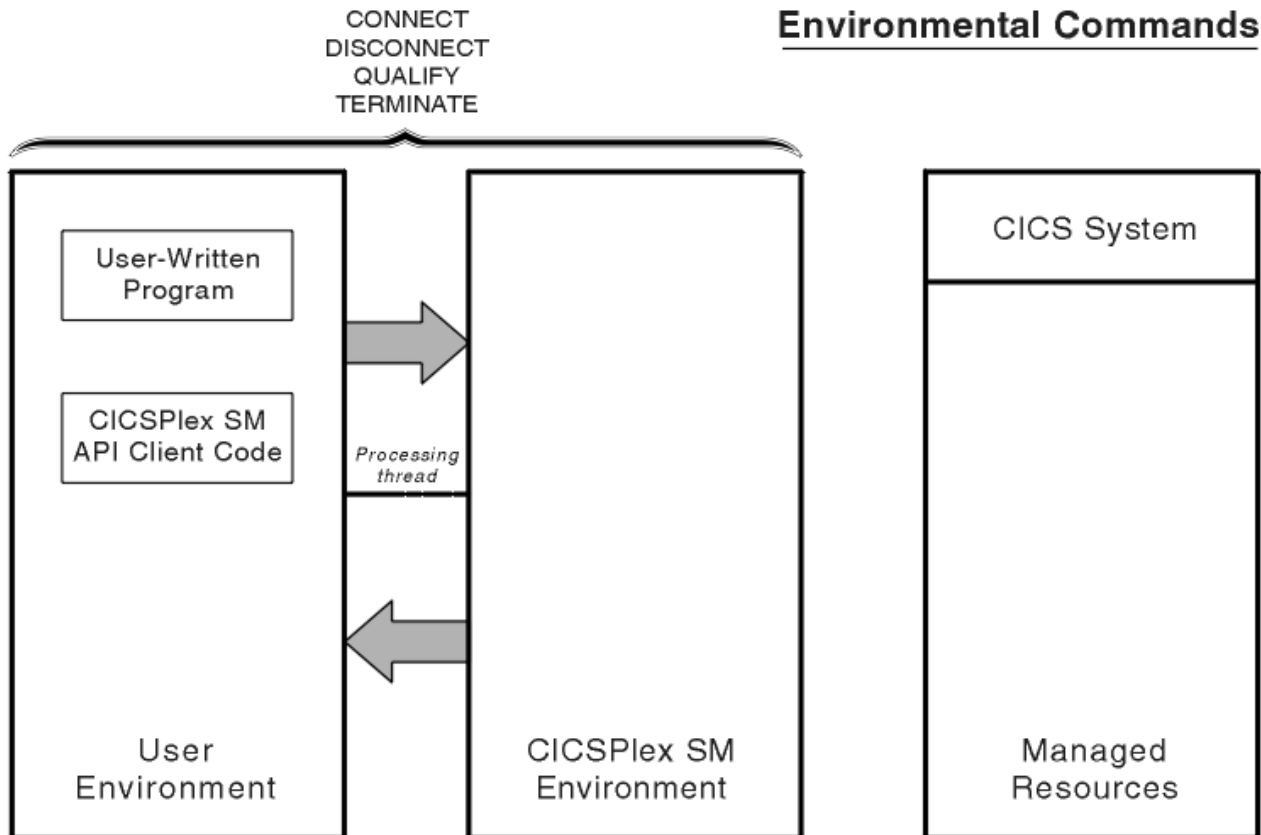


図 1. スレッドの管理に使用される API コマンド

これらのコマンドの詳細な説明は、[CICSplex SM API コマンド](#)を参照してください。

## 接続プロセス

CICSplex SM への接続のプロセスは、作成するプログラムのタイプとそれを実行する場所によって異なります。

コマンド・レベル・インターフェースを使用して作成したプログラムの場合、以下の要件に注意してください。

### CICS

CICS アプリケーションとして実行するように作成されたプログラムは、適切なスタブ・ルーチンとリンクされていなければならない、CICSplex SM によってローカル MAS としてアクティブに管理されている CICS システムで実行する必要があります。

まず、CICS システム内にある MAS エージェント・コードとの接続が確立し、その後この MAS を制御する CMAS への接続が確立されます。CONNECT コマンドにローカル CMAS の CONTEXT を指定する必要があります。

### Batch、NetView、または TSO

バッチ・ジョブとして、または NetView か TSO を使用して実行するように作成されたプログラムは、適切なスタブ・ルーチンとリンクされていなければならない、接続する CMAS と同じ MVS イメージ内で実行する必要があります。

このような環境で、MVS イメージに複数の CMAS がある場合、API はふさわしい CMAS を選択して接続を確立します。CMAS の選択には以下のルールが適用されます。

- CMAS はランタイム・モジュールのバージョン (EYU9AB00) と同じバージョンの CICSplex SM を実行しなければならない。
- CONNECT コマンドに指定されたコンテキストが CMAS である場合、CICSplex SM はその CMAS に接続する。その CMAS がアクティブでないか、適切なバージョンの CICSplex SM を実行中でない場合、CONNECT コマンドは失敗します。

- CONNECT に指定されたコンテキストが CICSplex である場合、CICSplex SM は CICSplex の管理の一部を担う適切なバージョンを実行している CMAS を選択する。
- CONNECT に指定されたコンテキストがない場合、CICSplex SM は最後に開始された CMAS に接続する (その CMAS が適切なバージョンの CICSplex SM を実行している場合)。

CICSplex SM API は別のタイプのバッチ環境もサポートします。プログラムは、それ自体が CICS トランザクションでなくても、CICS システムを実行しているアドレス・スペースから API コマンドを発行できます。言い換えると、そのプログラムは CICS システムと同じアドレス・スペースで別個の MVS タスクを実行できます。このタイプのプログラムはバッチ環境のスタブ・ルーチンとリンクしていなければならない、接続プロセスは他のバッチ・プログラムの場合と同じです。

**注:** CICS トランザクションであるプログラムは、CICSplex SM のローカル MAS である CICS システムで実行する必要があります。

上記の各環境で必要とされるスタブ・ルーチンの詳細については、[82 ページの『プログラムのリンク・エディット』](#)を参照してください。

**注:** REXX で作成されたプログラムの場合、接続プロセスは同じ環境 (バッチ、TSO、または NetView) で実行するコマンド・レベル・プログラムの場合と同じです。スタブ・ルーチンは必要ありませんが、CICSplex SM によって提供されている REXX 関数パッケージが正しくインストールされている必要があります。

**MVS の制限:** CONNECT 要求が正常に完了すると、すぐにスレッド・トークンがユーザーに返されます。このスレッド・トークンを参照するそれ以降のすべてのコマンドは、接続要求を出したのと同じ MVS TCB から発行する必要があります。

## セキュリティに関する考慮事項

API プログラムが CICSplex SM への接続を要求すると、接続先の CMAS は環境からユーザー権限データを抽出しようとします。この接続の確立方法は、このような権限データが存在するかどうか、およびその CMAS 内でセキュリティがアクティブになっているのかどうかに応じて異なります。

### セキュリティが存在する場合

CMAS セキュリティがアクティブになっているかどうかにかかわらず、API プログラムの実行場所にセキュリティ環境が存在する場合:

- API セキュリティ・ルーチン EYU9XESV は呼び出されません。
- CONNECT コマンドの USER オプションと SIGNONPARM オプションは無視されます。
- API プログラムは、アクセサー環境要素 (ACEE) から取得された呼び出し元ユーザーのユーザー ID を使用して接続されます。

**注:** CMAS セキュリティがアクティブでない場合は、ACEE のユーザー ID は CICSplex SM によって検証されません。

このタイプのセキュリティ環境は、CICS セキュリティがアクティブになっている TSO、バッチ、NetView、またはローカル MAS でプログラムが実行される場合に存在する可能性があります。セキュリティ検査は、API プログラムが実行されている環境によって実行されます。

### セキュリティが存在しておらず、CMAS セキュリティがアクティブでない場合

- API セキュリティ・ルーチン EYU9XESV は呼び出されません。
- CONNECT コマンドの USER オプションと SIGNONPARM オプションは無視されます。
- サインオンが実行されていません。ただし、セキュリティ・ルーチン・パラメーター・ブロック EYUBXESV の XESV\_CONN\_USERID フィールドで指定されたユーザー ID は接続と関連付けられています。

このタイプのセキュリティ環境は、CICS セキュリティがアクティブになっていないローカル MAS でプログラムが実行される場合に存在する可能性があります。CMAS セキュリティはアクティブになっていないため、セキュリティ検査は実行されません。

### セキュリティが存在しておらず、CMAS セキュリティがアクティブである場合

- API セキュリティ・ルーチン EYU9XESV が呼び出されます。
- CONNECT コマンドの USER と SIGNONPARM の値が EYU9XESV に渡されます。



- EYU9XESV によって戻されるユーザー ID を使用してサインオンが実行されますが、パスワード検査は実行されません。デフォルトでは、EYU9XESV は CMAS 用のデフォルト CICS ユーザー ID を戻します (DFLT\_UID の値)。

このタイプのセキュリティー環境は、CICS セキュリティーがアクティブになっていないローカル MAS でプログラムが実行される場合に存在する可能性があります。CMAS セキュリティーがアクティブになっているため、セキュリティー検査が EYU9XESV によって実行されます。

6 ページの表 2 は、API セキュリティーのレベルと、これらのレベルが実装される条件の要約を示しています。

表 2. 考えられる API セキュリティー環境

環境セキュリティー	CMAS セキュリティー	CMAS セキュリティーなし
YES	EYU9XESV は呼び出されません。 CONNECT オプションは無視されます。 ユーザー ID は ACEE。	EYU9XESV は呼び出されません。 CONNECT オプションは無視されます。 ユーザー ID は ACEE (検査なし)。
NO	EYU9XESV は呼び出されます。 CONNECT のオプションは渡されます。 ユーザー ID は EYU9XESV によって戻される値 (パスワード検査なしでサインオン)。	EYU9XESV は呼び出されません。 CONNECT オプションは無視されます。 ユーザー ID は XESV_CONN_USERID (サインオンなし)。

USER オプションと SIGNONPARM オプションについて詳しくは、API CONNECT コマンドの説明を参照してください。CONNECT コマンドを参照してください。EYU9XESV についておよびこのセキュリティー・ルーチンのカスタマイズについて詳しくは、[CICS TS セキュリティー](#)を参照してください。

## 環境間での互換性

ある環境で実行するための CICSplex SM API プログラムを作成したら、わずかな変更を加えるだけでそのプログラムを別の環境で実行できます。

例えば、EXEC CPSM コマンドを使用して作成された CICS アプリケーションを MVS バッチ・プログラムに変換するには、以下の作業を実行する必要があります。

- 次のような適切なコード変更を加えます。
  - 含まれる可能性のある EXEC CICS コマンドを削除します。
  - 必要な MVS 呼び出しを追加します。
- バッチ環境のスタブ・ルーチンを使用してそのプログラムを再リンク・エディットします。

注：環境固有の関数を一切使用していない場合は、REXX プログラムを変更することなく現在の MVS 環境 (バッチ、TSO、または NetView) から別の MVS 環境に移動できます。

EXEC CPSM プログラムを、そのプログラムが作成された対象である環境以外の環境に移動しようとする前に、以下のセクションを参照してください。

- 78 ページの『[言語と環境に関する考慮事項](#)』
- 80 ページの『[プログラムの変換](#)』
- 82 ページの『[プログラムのリンク・エディット](#)』。

## CICSplex SM のリリース間の互換性

特定の CICSplex SM リリースで実行するための API プログラムを作成したら、そのリリースで提供されるデータに引き続きアクセスすることも、製品の後続のリリースで提供されるデータにアクセスすることもできます。

一般に、CICSplex SM API の複数のリリースにアクセスする予定の場合は、以下に留意してください。

## ランタイム環境

CICSplex SM API プログラムのランタイム・バージョンは、そのプログラムが接続する CMAS のレベルと等しいバージョンになります。

- CICS アプリケーションとして実行するために作成されたプログラムの場合は、ランタイム・バージョンは MAS の接続先である CMAS のバージョンです。
- バッチ・ジョブとして実行するために、または NetView や TSO で実行するために作成されたプログラムの場合は、ランタイム・バージョンは CICSplex SM ランタイム・モジュール (EYU9AB00) のバージョンによって決まります (このモジュールはそのバージョンの SEYUAUTH ライブラリーで配布されます)。

プログラムのランタイム・バージョンは、次に示すバージョン以上でなければなりません。

- そのプログラムをリンク・エディットするために使用されたスタブ・ルーチン・モジュール (EYU9AxSI) のバージョン。
  - CICS プログラムの場合は、スタブ・モジュールは EYU9AMSI という名前であり、該当バージョンの SEYULOAD ライブラリーで配布されます。
  - バッチ、TSO、または NetView プログラムの場合は、モジュールは EYU9ABSI という名前であり、該当バージョンの SEYUAUTH ライブラリーで配布されます。

また、別個にリンク・エディットされて呼び出されたプログラムのスタブ・モジュールのバージョンは、CONNECT コマンドを発行したプログラムをリンク・エディットするために使用されたバージョンと同じである必要があります。

- CONNECT コマンドの VERSION オプションで指定された値。

注: REXX で記述されたプログラムの場合は、ランタイム・バージョンは、関数パッケージ (EYU9AR00) のバージョン以上である必要があります (この関数パッケージは該当バージョンの SEYUAUTH ライブラリーで配布されます)。

## VERSION オプション

CONNECT コマンドの VERSION オプションによって、CICSplex SM リソース・テーブルのどのリリースをユーザーのプログラムで利用できるのかが制御されます (リソース・テーブルは CICSplex SM データの外部表現です)。

- API プログラムは、リリース 2 より前の CICSplex SM リリースから提供されるデータにアクセスすることはできません (リリース 2 は API が導入されたリリースです)。VERSION の値は 0120 以上に設定する必要があります。
- API プログラムは、指定したランタイム・モジュールより後の CICSplex SM リリースから提供されるデータにアクセスすることはできません。VERSION の値は、そのランタイム・モジュールのリリース番号以下に設定する必要があります。
- 以下の条件が満たされる場合は、API プログラムは、そのプログラムが元々作成された対象のリリースより後の CICSplex SM リリースから提供されるデータにアクセスできます。
  - 指定したバージョンの適切なコピーブックを使用してそのプログラムをコンパイルすること。
  - そのプログラムが指定したバージョンのコピーブックと 互換性があること。

## CONTEXT オプション

各種の API コマンドでサポートされている CONTEXT オプションによって、プログラムがデータを受け取る元の CICS システムが決定されます。CONTEXT の値は、現在サポートされているリリースの CICSplex SM を実行している任意の CMAS または CICSplex に設定できます。ただし、その CMAS または CICSplex のリリース・レベルはランタイム・モジュールのリリースと同じである必要があります。

## CURRENT オプション

CURRENT オプションを指定する場合は、レコード・ポインターは移動しません (したがって、後続の FETCH によって同じレコードが取得されます)。以前は、レコード・ポインターは次のレコードに移動していました。詳しくは、[27 ページの『結果セット内のレコード・ポインターの配置』](#)を参照してください。

## REXX アプリケーションについての特別な考慮事項

REXX アプリケーション・プログラムを使用している場合は、CICSplex の一部のメンバーにのみ PTF を適用して、その PTF および REXX プログラムによって導入された新しいテーブル・フィールドに値を入力するように REXX API プログラムに変更を加えた後、その PTF が適用されず、したがってその新規フィールドの定義がない CMAS に接続すると、CICSplex SM がどのように動作するかを認識する必要があります。

この場合は、以下のようになります。

1. その CMAS は新規フィールドの値を保守ポイント CMAS に転送しません。
2. 保守ポイント CMAS はレコード領域を変換して、新規フィールドにデフォルト値を割り当てます。新しい値は、REXX プログラムによって最初に指定された値とは異なる可能性があります。
3. その後保守ポイント CMAS は、レコードを発信元の CMAS にブロードキャストして戻しますが、レコードを変換し戻して新規フィールドを削除します。この時点で、保守ポイント・リポジトリには目的の値は格納されず (デフォルト値が格納されます)、保守ポイント・リポジトリによってレコードが発信元の CMAS にブロードキャストされたときには、このリポジトリから目的の値が削除されています。
4. 同じ REXX プログラムによってレコードの TPARSE が発行された場合は、このフィールドの値は作成されたときと同じままになり、この TPARSE によって変更されません。この結果として、このフィールドに目的の値が含まれているという間違った内容がプログラムによって示される一方で、保守ポイント・リポジトリ内では、このフィールドにはデフォルト値が割り当てられ、バックレベル CMAS リポジトリ内では、このフィールドは存在しないという状況が発生することがあります。
5. その後 REXX API プログラムがバックレベル CMAS に接続して、レコードの TPARSE を発行した場合は、この TPARSE によって新規フィールドに値が格納されることはありません。この場合は、このフィールドには通常の REXX デフォルト値が格納されます (このフィールドの値はこのフィールドの名前と同じになります)。

上記の一連の状況が該当し、問題が発生する可能性がある場合は、レコード長を取得して確認する QUERY を発行するためのコードを REXX プログラムに含める必要があります。

## 新規リリースのリソース・テーブルへのアクセス

新規リリースの API で既存のプログラムを実行することで、最新の CICSplex SM リソース・テーブルにアクセスできます。

**注:** ただし、CICSplex SM の新機能 (ビジネス・アプリケーション・サービスなど) を最大限に活用するには、既存のプログラムを変更するか新しいプログラムを作成する必要があります。

既存の API プログラムを新規リリースの CICSplex SM で実行するには、以下の指示に従ってください。

- プログラムで以下を使用できることを確認します。
  - 新規リリース用のランタイム・モジュール (新規リリースの SEYUAUTH ライブラリーで提供される EYU9AB00)
  - 新規リリースを実行している CMAS
- CONNECT コマンドの VERSION 値を新規リリースの CICSplex SM に合わせて変更して、新規リリースで提供されているスタブ・モジュールを使用してそのプログラムを再リンク・エディットします。
- CICSplex SM リソース・テーブルに対する変更による可能性のある影響を検証します。

新規リリース内のリソース・テーブルに属性を追加できます。その結果として、そのテーブルに対するそのプログラム側の参照が影響を受ける可能性があります。属性の追加や変更によって、特定のリソース・テーブルの長さがリリース間で変更されることがあります。新規リリースで提供されるリソース・テーブル・コピーブックは、このような変更に関する情報源として役立ちます。

**注:** 当該リリースの新機能を活用する必要がない場合は、使用されたリンク・エディット・レベルに合わせて CONNECT コマンドの VERSION 値が設定されている場合は、既存の API プログラムを変更することなく継続的に実行できます。

プログラムを新規リリースの CICSplex SM で実行したときに、RESPONSE と REASON の値として INVALIDPARM LENGTH を受け取った場合は、テーブルの長さが増大している可能性があり、データ・バッファが新しいリソース・テーブル・レコードを格納するのに十分な長さでない可能性があります。



- リソース・テーブルのカスタマイズしたビューを使用している場合は、いずれかの新規リソース・テーブルの名前が、カスタマイズしたビューの名前と重複していないことを確認してください。重複している場合は、処理が影響を受ける可能性があります。詳しくは、[14 ページの『カスタマイズされたリソース・テーブル・レコードの作成』](#)を参照してください。

特定リリースの新規リソース・テーブルと変更されたリソース・テーブルの完全なリストについては、[CICSplex SM リソース・テーブル](#)を参照してください。

## 以前のリリースのリソース・テーブルへのアクセス

旧リリースの CICSplex SM で提供されているリソース・テーブルに引き続きアクセスできます。

そのためには、以下のことを行う必要があります。

- CONNECT コマンドの VERSION オプションで、アクセスする CICSplex SM データのリリースを指定します。
- アクセスするリリースで提供されているランタイム・モジュール (EYU9AB00) を使用するか、このランタイム・モジュールをサポートしている後続リリースで提供されているこのランタイム・モジュールを使用します。
- ランタイム・モジュール以下のバージョンのスタブ・モジュール (EYU9AxSI) を使用します。

9 ページの表 3 は、さまざまなリリースの CICSplex SM のデータにアクセスするための VERSION オプション、スタブ・モジュール、およびランタイム・モジュールの有効な組み合わせを示しています。

表 3. さまざまなリリースのデータにアクセスするための有効な方法

VERSION の値	スタブ・モジュール (EYU9AxSI)	ランタイム・モジュール (EYU9AB00)	使用可能な CMAS	使用される CMAS	使用可能なデータ
0220	V2R2	V2R2	V1R2 V1R3 V1R4 V2R1 V2R2	V2R2	V2R2
0120	V1R2	V1R2	V1R2	V1R2	V1R2
0120	V1R2	V1R3	V1R3	V1R3	V1R2
0120	V1R3	V1R3	V1R2 V1R3	V1R3	V1R2
0120	V1R4	V1R4	V1R2 V1R3 V1R4	V1R2	V1R2
0130	V1R3	V1R3	V1R2 V1R3	V1R3	V1R3
0130	V1R3	V1R4	V1R2 V1R3 V1R4	V1R4	V1R3
0130	V1R4	V1R4	V1R2 V1R3 V1R4	V1R4	V1R3
0140	V1R4	V1R4	V1R2 V1R3 V1R4	V1R4	V1R4
0210	V2R1	V2R1	V1R2 V1R3 V1R4 V2R1	V2R1	V2R1

10 ページの表 4 は、VERSION オプション、ランタイム・モジュール、およびスタブ・モジュールの無効な組み合わせと、これらの組み合わせでエラーが発生する理由を示しています。

表 4. さまざまなリリースにアクセスする際の一般的なエラー

VERSION の値	スタブ・モジュール (EYU9AxSI)	ランタイム・モジュール (EYU9AB00)	使用可能な CMAS	エラーの説明
0140	V2R1	V1R4	V1R4 V2R1	スタブ・モジュールのリリース・レベルがランタイム・モジュールより大きい。
0210	V2R1	V1R4	V1R4 V2R1	スタブ・モジュールのリリース・レベルがランタイム・モジュールより大きい。
0210	V1R4	V1R4	V1R4	VERSION の値がランタイム・モジュールより大きい。
0210	V2R1	V2R1	V1R4	必要なランタイム・レベルで使用可能な CMAS がない。

注: REXX で記述されたプログラムの場合、互換性の問題はほぼ同じです。リリース 2 の関数パッケージ (必要なスタブ・モジュールが含まれています) は、リリース 2 またはリリース 3 のランタイム・モジュールと組み合わせて正常に実行できます。ただしリリース 3 の関数パッケージは、リリース 2 のランタイム・モジュールと組み合わせて実行できず、代わりにリリース 3 のモジュールが必要です。

## サンプル・プログラム

各サポート言語のサンプル・プログラムが CICSplex SM とともにソース形式で配布されています。これらのサンプルの目的は、ユーザーが作成できるプログラムのタイプと、それらのプログラムで使用する必要のあるコマンドを示すことです。

これらのサンプル・プログラムは、EYUxAPI<sub>n</sub> という名前のメンバーに格納されて配布されます (x は 1 文字の言語 ID であり、n は連番のプログラム ID です)。例えば、EYUCAPI1 は C 言語で記述されたサンプル・プログラム番号 1 です。これらはすべて SEYUSAMP ライブラリーに配置されています。

次の表は、サンプル・プログラムの詳細を示しています。

表 5. CICSplex SM で提供されているサンプル・プログラム

言語	Programs (プログラム)	ライブラリー
アセンブラー	EYUAAPI1 EYUAAPI2 EYUAAPI3	SEYUSAMP
COBOL	EYULAPI1 EYULAPI2 EYULAPI4	SEYUSAMP
PL/I	EYUPAPI1 EYUPAPI2	SEYUSAMP
C	EYUCAPI1 EYUCAPI2	SEYUSAMP
REXX	EYU#API1 EYU#API2 EYU#API3	SEYUSAMP

CICSplex SM API サンプル・プログラムには、各サンプル・プログラム (いずれかのサポート言語で記述) のリストが掲載されています。

注: 追加のサンプル CICSplex SM API プログラムについては、次のアドレスで IBM CICS SupportPacs システムを通じて入手できます。

<http://www-01.ibm.com/support/docview.wss?uid=swg27007241>

## 第 2 章 CICSplex SM API の使用方法

CICSplex SM API を使用する前に、管理対象オブジェクト、リソース・テーブル、結果セットなどのいくつかの概念を理解しておく必要があります。

### CICSplex SM 管理対象オブジェクト

CICSplex SM はオブジェクト指向システムです。したがって、CICSplex SM 環境内の各リソースはオブジェクトのインスタンスです。各オブジェクトは特定のタイプであると見なされ、正式に定義された固有名を持っています。

#### 管理対象オブジェクトのタイプ

CICSplex SM 環境には、さまざまなタイプのオブジェクトがあります。一部のオブジェクト (CICS システム、プログラム、トランザクションなど) は、CICSplex SM によって管理される実世界のリソースです。定義オブジェクト (モニター仕様やワークロード定義など) は CICSplex SM 内専用として作成されるリソースです。

イベントは、CICSplex SM 処理の結果として生成されるランタイム・オブジェクトの例です。

CICSplex SM の管理対象オブジェクトは、次のカテゴリーにグループ化できます。

- 管理対象 CICS リソース
  - CICS リソース
  - モニターされた CICS リソース
- CICS リソース定義
- CICSplex SM 定義
- CICSplex SM マネージャー・リソース
- CICSplex SM 通知
- CICSplex SM メタデータ。

#### 管理対象の CICS リソース

これらのオブジェクトは、CICSplex SM によって管理されている CICS システム内に存在する実際の CICS リソースを表します。

このタイプの各オブジェクトは、CICSplex SM が報告および操作できる CICS リソースを記述しています。標準の CICS インターフェースを使用している CICSplex SM で使用可能なすべてのリソースについて、管理対象オブジェクトが存在しています。一部のケースでは、CICSplex SM の管理対象オブジェクトは CICS による表現よりも確定的なリソース表現を提供します。例えば、LOCTRAN オブジェクトと REMTRAN オブジェクトは CICS によってトランザクションとして結合されます。これらのオブジェクトは、ローカル・トランザクションとリモート・トランザクションを区別するために CICSplex SM によって使用されます。

標準の CICS リソースに加えて、CICSplex SM はリソース・モニター活動の結果として管理対象オブジェクトを作成します。モニター対象 CICS リソースにはリソース属性のサブセットが含まれており、通常これらの属性はそのリソースの状態特性と消費特性を反映する属性です。また、CICSplex SM によって、リソース使用状況を平均、率、またはパーセンテージの形式で示す派生属性が提供されることがあります。MLOCTRAN と MREMTRAN はモニター対象 CICS リソース・オブジェクトの例であり、これらは LOCTRAN と REMTRAN という CICS リソース・オブジェクトから派生しています。モニター対象 CICS リソース・オブジェクトは、関連する CICS リソース・オブジェクトが CICS システムから削除された後も存続できるとともに、このシステム自体がシャットダウンされた後も存続できます。

## CICS リソース定義

これらのオブジェクトは、CICSplex SM が CICS システムに割り当てて (場合によっては) インストールできる CICS リソースの定義を表しています。

実際の定義は、CICSplex SM のデータ・リポジトリに定義レコードとして保管されています。例えば TRANDEF オブジェクトは、CICSplex 全体にわたる複数の CICS システムにローカルとリモートの両方で割り当て可能な CICS トランザクションを表しています。

CICS リソースを CICS システムに割り当てると、CICSplex SM はそれらのリソースを 1 つの論理グループ (アプリケーションなど) として管理できるようになります。また、CICSplex SM は、EXEC CICS CREATE コマンドをサポートしている CICS システムにリソースのインスタンスをインストールできます。

## CICSplex SM 定義

これらのオブジェクトは、CICSplex SM の管理アプリケーションによって使用される定義を表しています。

実際の定義は、CICSplex SM のデータ・リポジトリに定義レコードとして保管されています。例えば MONSPEC オブジェクトは、CICS システム内にリソース・モニターを確立するために CICSplex SM によって使用されるユーザー定義のモニター仕様を表しています。

CICSplex SM 定義に加えた変更内容はすべて、CICSplex 全体に自動的に配信されます。また、一部の定義は、参照整合性のために他の定義にバインドされています。これらの定義のいずれかを削除した場合は、すべての関連する定義も削除されます。例えば CPLEXDEF オブジェクトを削除すると、その CICSplex 用のすべての定義オブジェクトが、その CICSplex を管理しているすべての CMAS から自動的に削除されます。

## CICSplex SM マネージャー・リソース

これらのオブジェクトは、CICSplex SM 定義から作成されたランタイム・リソース、または処理時に CICSplex SM 管理アプリケーションによって作成されたランタイム・リソースを表しています。

基盤の定義に必ずしも影響を与えることなく、CICSplex SM マネージャー・リソースを操作できます。RTAACTV オブジェクトは CICSplex SM マネージャー・リソースの例であり、現在インストールされている RTADEF および STATDEF 定義オブジェクトを記述しています。

どの定義にも直接関連していない他の CICSplex SM マネージャー・リソースが存在します。例えば CRESCONN オブジェクトは、アクティブな MAS 内の CICS 接続を記述するトポロジー・サービス・リソース・マップです。

## CICSplex SM 通知

CICSplex SM 通知は実際には、CICSplex SM 管理対象オブジェクトによって非同期で生成されるメッセージです。

通知では、該当オブジェクトに関連する、関心のあるイベントが記述されています。CICSplex SM マネージャー・リソースは、これらのイベントの 1 つ以上に対する関心を登録できます。通知が生成されると、マネージャー・リソースは発生したイベントに基づいて必要な処理を実行します。

API プログラムは、CICSplex SM 通知を生成するイベントに対する関心を登録することもできます。EMSTATUS、EMASSICK、および EMASWELL の各オブジェクトは、CICSplex SM MAS エージェントによって生成される通知メッセージの例です。これらの通知では、MAS の現在の状態が記述されています。

ERMCxxxx オブジェクトは、トポロジー・リソース・マップが変更されたときに CICSplex SM によって生成されます。CICSplex SM では、MAS 内の特定 CICS リソースのトポロジーを記述しているリソース・マップが維持されます。リソース・マップが維持されている CICS リソースには、対応する ERMCxxxx 通知オブジェクトがあります。CICSplex SM エージェントは、これらの CICS リソースのインストールと破棄を検知して、トポロジー・リソース・マップを更新させます。例えば、ファイル定義が MAS 内にインストールされた場合は、トポロジー・リソース・マップが変更されて、ERMCFILE 通知が生成されます。ERMCFILE 通知の ACTION 属性は、インストールが実行されたことを示します。さらに、ローカル MAS の場合は、CICSplex SM MAS エージェントによってこれらの CICS リソースの更新が検知されます。例えば、いずれかのプログラムが無効化された場合は、更新を示す ACTION 属性を有する ERMCPRGM 通知が生成されます。

## **CICSplex SM メタデータ**

これらのオブジェクトは、CICSplex SM 管理対象リソースの構造を記述します。この情報は、それぞれのアクティブな CMAS に存在するオブジェクト・ディレクトリーに保持されます。

API プログラムは、オブジェクト・ディレクトリーに格納されている以下のタイプのメタデータを要求できます。

### **OBJECT**

オブジェクトの一般特性

### **OBJECT**

オブジェクトの有効なアクション

### **METADESC**

オブジェクト属性の基本的な説明

### **ATTR**

オブジェクト属性の詳細な説明

### **ATTRAVA**

属性の有効な EYUDA 値または CVDA 値

### **METANAME**

CVDA、CVDAT、および EYUDA のすべての情報

### **METAPARM**

アクションのパラメーターについての説明

### **PARMAVA**

パラメーターで許可される値の説明

## **CICSplex SM リソース・テーブル**

各 CICSplex SM 管理対象オブジェクトは、リソース・テーブルによって外部で表されます。

リソース・テーブルでは、オブジェクトのすべての属性が定義されます。これらの属性は、そのオブジェクトで利用できるデータの集合を表します。

正式なオブジェクト名は、そのオブジェクトの属性を記述するリソース・テーブルの名前として使用されます。API プログラム内でオブジェクトを特定するには、そのオブジェクトのリソース・テーブル名を指定します。例えば、1 つ以上の CICS システム内のプログラムに関する情報を取得するには、PROGRAM オブジェクトにアクセスします。PROGRAM は、CICS プログラムを記述する CICSplex SM リソース・テーブルの名前です。

オブジェクトの各インスタンスは、CICSplex SM 環境内の実リソースを記述するリソース・テーブル・レコードとしてフォーマット設定されます。オブジェクトの属性は、リソース・テーブル・レコードの個別フィールドに表示されます。リソース・テーブル自体はオブジェクトではないことに注意することが重要です。リソース・テーブル・レコードは、CICSplex SM によって戻される管理対象オブジェクト情報の単なるフォーマットとなります。この情報に含まれている内容は、現在の属性値、そのオブジェクトでサポートされているアクション、およびそのオブジェクトが有効である CICS リリースです。

CICSplex SM 管理対象オブジェクトの各タイプについて、1 つのリソース・テーブル・タイプがあります。

### **リソース・テーブル・タイプ**

オブジェクト・タイプ

### **CICS 定義**

CICS リソース定義

### **CICS リソース**

CICS リソース

### **CICS モニター対象**

モニターされた CICS リソース

### **CPSM 定義**

CICSplex SM の定義

## **CPSM マネージャー**

CICSplex SM マネージャー・リソース

## **CPSM 通知**

CICSplex SM 通知

## **CPSM メタデータ**

CICSplex SM メタデータ

## **CPSM 構成**

CICSplex SM 構成定義

タイプ別の CICSplex SM リソース・テーブルの要約と、個別のリソース・テーブルの詳細な説明は、[CICSplex SM リソース・テーブル](#)を参照してください。

## **制限付きのリソース・テーブル属性**

CICSplex SM リソース・テーブル内の特定の属性は内部使用専用であるため、API プログラムによって変更したり操作したりできません。

CICS リソースおよび CICS モニター対象のテーブル内では、CICSplex SM は以下の属性を使用して、該当リソースが格納されている CICS システムを一意的に識別します。

- EYU\_CICSNAME
- EYU\_CICSREL.

これらの属性は、CICS リソースおよび CICS モニター対象のすべてのリソース・テーブル・レコードに含まれています。これらの属性を GROUP コマンドで指定して、結果セット内のレコードを集計できます。ただし、ORDER、SPECIFY FILTER、および SPECIFY VIEW の各コマンドでは、これらの属性を指定しないでください。

CPSM 定義テーブルと CICS 定義テーブルには、その定義が最後に変更された日時を表す CHANGETIME 属性が含まれています。CICS 定義テーブルには、その定義が作成された日時を表す CREATETIME 属性も含まれています。属性フィールド (CHANGEAGENT、CHANGEAGREL、CHANGEUSRID) は BAS リソース定義用です。これらの属性で示されるのは、そのリソースが定義または最後に変更された方法の詳細、そのリソースが定義または最後に変更されたときに実行されていた CICS システムのレベル、およびそのリソース定義を作成または変更したユーザーの ID です。CHANGEAGENT、CHANGEAGREL、CHANGETIME、CREATETIME、および CHANGEUSRID の各属性は CICSplex SM によって単独で維持されているため、これらの属性値を変更しようとししないでください。

## **カスタマイズされたリソース・テーブル・レコードの作成**

通常は、結果セットを作成したとき、各リソース・テーブル・レコードには、CICSplex SM によって定義された形式で属性セット全体が含まれています。ただし、場合によっては、それらの属性のサブセットを操作することや、それらの属性を異なる順序で操作することが必要となる場合があります。SPECIFY VIEW コマンドを使用すると、レコードに含めるリソース・テーブル属性とそれらの表示順序を指定できます。実際には、一時的なカスタムのリソース・テーブルを作成することになります。

CICS リソースというタイプのリソース・テーブルのビューのみを作成でき、他のタイプのリソース・テーブルのビューを作成することはできません。また、一度に 1 つのみのリソース・テーブルの属性からビューを作成できます。別々のリソース・テーブルの属性を組み合わせ、単一のビューを作成することはできません。配列式で EYU\_CICSNAME 属性と EYU\_CICSREL 属性を指定して、ビュー・レコードの収集元となった CICS システムを特定できます。

リソース・テーブル・ビューを作成する際は、そのビューに名前を割り当てる必要があります。ビューに割り当てた名前は、既存のリソース・テーブル名より優先されます。したがって、既存のリソース・テーブル名を再定義することで、元のテーブルとは出現順序が異なる属性のサブセットを表現することができます。

プログラムの保守を簡易化するために、カスタマイズされたリソース・テーブル・ビューには固有の名前を割り当ててください。固有の名前を使用しない場合は、同じ処理スレッド内の同じ名前の別のビューにアクセスするには、元のビューを破棄するというプログラミングのオーバーヘッドが必要になります。使

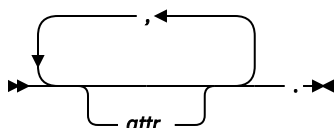


用しているバージョンの CICSplex SM をアップグレードするときには、いずれの新規リソース・テーブルの名前も、カスタマイズされたビュー名と重複しないことを確認してください。

どのリソース・テーブル属性をどのような順序で含めるかを CICSplex SM に対して定義するには、SPECIFY VIEW コマンドの FIELDS オプションで配列式を指定します。この式は、ORDER コマンドを使用して結果セット内のレコードをソートする際に使用する式と似ています。配列式は、ビューに含める属性のリストで構成されます。

ビューを作成するための配列式の構文は次のとおりです。

#### 配列式 - ビューの作成



#### attr

リソース・テーブルの属性の名前。

必要な数の属性名を指定できますが、配列式の合計長はコンマやブランク・スペースを含めて 255 文字を超えてはなりません。属性名を指定しない場合は、配列式にはリソース・テーブル内の最初の属性の名前が含まれます (例: CICSRRGN リソース・テーブル内の JOBNAME 属性)。

例えば、LOCTRAN リソース・テーブルの制限付きビューを作成するには、次のように指定します。

```
TRANID,STATUS,USECOUNT,PROGRAM,PRIORITY,TRANCLASS.
```

配列式の後ろから指定されたバッファの末尾までは、ブランク・スペースまたはヌル文字にする必要があります。つまり、(LENGTH オプションを使用して) 指定するバッファ長には、配列式以外のデータを含めてはなりません。ビューを作成したら、GET コマンドの OBJECT オプションでそのビューを指定できます。GET によって戻されるリソース・テーブル・レコードには、SPECIFY VIEW コマンドの配列式で指定した属性のみが含まれます。

作成するすべてのビューは、それらを作成するときに基礎とする特定の処理スレッドと関連付けられます。したがって、それらのビューを他の処理スレッドと共用することはできません。処理スレッドを終了すると、それに基づいて作成したすべてのビューが破棄されます。DISCARD コマンドを使用して、任意の時点でビューを破棄することもできます。

## カスタマイズしたリソース・テーブル・レコードに対するコピーブックの作成方法

カスタマイズしたビューの構造を構築するには、SPECIFY VIEW、GET、および FETCH の各コマンドを使用して構造にデータを移動します。

以下に例を示します。

```
*****
*          SPECIFY VIEW          *
*****
STRING 'POOLNAME,MINITEMLEN,QUELENGTH,NUMITEMS,'
'RECOVSTATUS,MAXITEMLEN,LASTUSEDINT,'
'NAME,TRANID,LOCATION.'
DELIMITED BY SIZE INTO BUFFERA.
MOVE 96 TO BUFFERL.
EXEC CPSM SPECIFY
      VIEW('VTSQSHR')
      FIELDS(BUFFERA)
      LENGTH(BUFFERL)
      OBJECT('TSQSHR')
      THREAD(TTKN(1))
      RESPONSE(SMRESP)
      REASON(SMRESP2)
END-EXEC.
```

図 2. 構造を構築する SPECIFY VIEW コマンド

関連する構造は、SPECIFY VIEW FIELDS キーワードで指定された各属性で構成されます。これについては、[16 ページの図 3](#) に示します。

```

01 VTSQSHR.
* Shared Temporary Storage Queue
  02 POOLNAME          PIC X(0008).
* TS Pool Name
  02 MINITEMLEN        PIC S9(0004) USAGE BINARY.
* Smallest item Length in bytes
  02 QUELENGTH         PIC S9(0008) USAGE BINARY.
* Total length in bytes . FLENGT
  02 NUMITEMS          PIC S9(0004) USAGE BINARY.
* Number items in queue
  02 RECOVSTATUS       PIC S9(0008) USAGE BINARY.
* Recovery Status
  02 MAXITEMLEN        PIC S9(0004) USAGE BINARY.
* Largest item length in bytes
  02 LASTUSEDINT       PIC S9(0008) USAGE BINARY.
* Interval since last use
  02 NAME-R            PIC X(0016).
* Queue Name
  02 TRANSID           PIC X(0004).          -- RESERVED WORD --
* Trans that created tsqueue
  02 LOCATION          PIC S9(0008) USAGE BINARY.
* Queue Location

```

図 3. カスタマイズしたビューの構造

この構造では、EYU-CICSNAME、EYU-CICSREL、および EYU-RESERVED の各属性、フィールドの位置合わせ属性、または埋め込み属性は使用されません。

## 管理対象オブジェクトの選択

API プログラムの関心対象は、一般的に、CICSplex SM 管理対象オブジェクトのサブセットのみとなります。

以下の方法で、操作する管理対象オブジェクトを特定できます。

- プログラムのコンテキストとスコープを設定します。
- 個別のコマンドでフィルター式を使用します。

## コンテキストおよびスコープの設定

API プログラムで操作できる一連の管理対象オブジェクトを決定する主な要因は、処理スレッドに関連付けられたコンテキストとスコープです。

CICSplex SM のすべての操作と同様、API プログラムのコンテキストとスコープによって、そのプログラムを実行できる CICS システムが特定されます。

一般に、コンテキストとスコープの値は次のように設定できます。

### CONTEXT

CICSplex 内のほとんどの操作については、コンテキストはその CICSplex の名前です。CMAS 構成に関連する操作については (CICSplex や CMAS 通信リンクの定義など)、コンテキストは CMAS 名である必要があり、CICS のローカル MAS で実行されているアプリケーションについては、CMAS 名はローカル CMAS 名である必要があります。

### SCOPE

コンテキストが CICSplex である場合は、可能なスコープは以下のとおりです。

- CICSplex 自体
- その CICSplex 内の CICS システムまたは CICS システム・グループ
- CICSplex SM リソース記述 (RESDESC) で定義されている論理スコープ

コンテキストが CMAS の場合は、スコープ値は無視されます。スコープ値が無視されるいくつかのリソースも存在します。これらについては、[CICSplex SM リソース・テーブル](#)のリソース・テーブルの記述の「**SCOPE applies**」フィールドで示されています。

プログラムのデフォルトのコンテキストとスコープを設定するには、以下のいずれかのコマンドを使用します。



## CONNECT

API 処理スレッドが確立されたときのデフォルトのコンテキストとスコープを定義します。

## QUALIFY

スレッド上で発行される後続コマンドのデフォルトのコンテキストとスコープを変更します。

これらのいずれかのコマンドに設定する値は、コンテキストとスコープを使用するすべての API コマンドに対して有効になります。

代わりに、個別の API コマンドのコンテキストとスコープの値を指定することもできます。以下のコマンドでは、CONTEXT オプションと SCOPE オプションの一方または両方がサポートされています。

- CREATE
- GET
- LISTEN
- PERFORM OBJECT
- REMOVE
- UPDATE

これらのすべてのコマンドに設定するコンテキストとスコープの値は、そのコマンドに対してのみ有効になります。スレッドのデフォルトのコンテキストとスコープを指定した場合、これらのいずれかのコマンドに設定する値によって、一時的にデフォルト値が指定変更されます。デフォルトのコンテキストとスコープを指定しなかった場合に、これらの値を必要とするコマンド (GET など) を発行するには、そのコマンドにコンテキストとスコープを指定する必要があります。

## フィルター式の使用

特定のプログラムにのみ関心がある場合は、フィルター式を使用して、特定の PROGRAM 属性の現行値に基づき、戻されるレコードの数を制限できます。

CICSplex SM 管理下オブジェクト・データ要求では、大量のリソース・テーブル・レコードが生成されることがあります。デフォルトでは、現在のコンテキストとスコープ内の指定したオブジェクトについて存在するすべてのリソース・テーブル・レコードが戻されます。例えば、PROGRAM オブジェクト・データを要求した場合は、現在のコンテキストとスコープ内のすべての CICS システム内のすべてのプログラムのリソース・テーブル・レコードを受け取ります。

### フィルター式の使用法

フィルター式は、次の 2 つの方法で使用できます。

- GET または PERFORM OBJECT コマンドの CRITERIA オプションを使用して、そのコマンドによって戻されるリソース・テーブル・レコードをフィルタリングします。フィルター式は 1 回のみ使用され、そのフィルター式を使用したコマンドの処理が完了すると破棄されます。
- SPECIFY FILTER コマンドを使用して、繰り返し使用できるフィルターを定義します。

フィルターを定義したら、以下のコマンドでそのフィルターを使用して処理対象のリソース・テーブル・レコードを限定できます。

- COPY
- DELETE
- FETCH
- GET
- GROUP
- LISTEN
- LOCATE
- MARK
- PERFORM OBJECT
- PERFORM SET
- REFRESH

- SET
- UNMARK

SPECIFY FILTER コマンドを使用して定義したフィルター式は、そのフィルター式を破棄するか (DISCARD コマンドを使用) 処理スレッドを終了するまで、プログラムで使用できます。

### フィルター式の作成方法

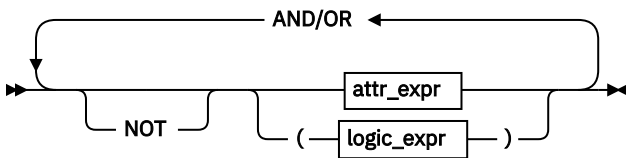
フィルター式は、リソース・テーブル・レコードのフィルターで使用される論理式を定義する文字ストリングです。

フィルター式は、次の形式の 1 つ以上の属性式で構成できます。

#### フィルター式

➡ **logic\_expr** — .->

#### logic\_expr



#### attr\_expr

➡ **attr** — oper — **value** ➡

説明:

#### attr

リソース・テーブルの属性の名前。

フィルター式では、同じ属性を複数回指定できます。

**注:** フィルター式では、EYU\_CICSREL 属性や最大長が 256 バイトを超える属性は指定できません。

#### oper

以下の比較演算子のいずれか。

- <      より小さい
- <=    より小か等しい
- =      等しい
- >=    より大か等しい
- >      より大きい
- ≠    等しくない

**重要:** 保管クロック・ベースのフィールドを扱う際は、等号 (=) の代わりに CRITERIA 式でより小 (<) 演算子またはより大 (>) 演算子を使用してください。CICSplex SM API または CMCI を直接使用している場合に、特定の保管クロック値に基づいたレコードを選択する必要がある場合は、複合 CRITERIA 式を使用できます。例えば、"RESPTIME>='0000:00:00.008351' AND RESPTIME<'0000:00:00.008352'." を使用できます。

#### 値は

属性のテスト対象となる値。これは、リソース・テーブルの属性に対して有効な値でなければなりません。

## 総称値

属性が文字データを受け入れる場合には、総称値の指定が可能です。総称値には、次のものを含めることができます。

- アスタリスク (\*)。任意の数の文字 (0 文字を含む) を表します。アスタリスクは指定値の中で最後に置く必要があります。これを 1 文字だけで使用することもできます。例えば、次のとおりです。

```
TRANID=PAY*.
```

- 正符号 (+)。単一文字を表します。+ は、指定値の中で 1 回以上使用できます。例えば、次のとおりです。

```
TRANID=P++9.
```

### 注:

1. 総称値検査は、フィルター値に対してのみ適用されます。例えば、フィルター値 `USERID=S*` は `S` で始まるユーザー ID を持つリソース・テーブル・レコードを返しますが、フィルター値 `USERID=SMITH` は総称文字を含むようなリソース・テーブル・レコード (例えば `S*` のユーザー ID を持つレコード) を返しません。
2. 16 進データ・タイプの場合、総称検索のためにアスタリスク (\*) を付加する前にデータを 16 進数に変換する必要があります。正符号 (+) は、16 進データ・タイプではサポートされません。
3. Web ユーザー・インターフェースでは、WLM アクティブ・ビュー (EYUSTARTWLMATAFF など) の属性フィルターでの組み込み総称文字の使用がサポートされていません。単一のアスタリスク (\*) を使用してすべての値を要求できます。

## 組み込みブランクまたは特殊文字

値に組み込みブランクまたは特殊文字 (ピリオド、コンマ、等号など) が含まれる場合、値ストリング全体を単一引用符で囲む必要があります。例えば、次のとおりです。

```
TERMID='Z AB'.
```

値に単一引用符またはアポストロフィーを組み込む場合は、文字を繰り返す必要があります。例えば、次のようにします。

```
DESCRIPTION='October''s Payroll'.
```

**注:** CICSplex SM の値ストリングで単一引用符を使用する際は、必ずプログラミング言語の引用規則を考慮してください。

## 16 進データ

属性のデータ・タイプが HEX である場合には、値を 16 進表記にする必要があります。

例えば、REQID リソース・テーブルの NAME 属性は HEX データ・タイプです。01234567 と同等の名前を 16 進表記で指定すると、次のような値になります。

```
NAME=F0F1F2F3F4F5F6F7.
```

## AND/OR

論理演算子 AND および OR を使用して、属性式を複合論理式に結合します。例えば、次のようにします。

```
attr_expr AND attr_expr.
```

フィルター式は、左から右に評価されます。括弧を使用すると、フィルター式の意味を変えることができます。例えば、次のような式があるとします。

```
attr_expr AND (attr_expr OR attr_expr).
```

これは、次の式とは意味が異なります。

```
(attr_expr AND attr_expr) OR attr_expr.
```

## NOT

1 つ以上の属性式を否定します。

単一の属性式を否定することができます。例えば、次のようにします。

```
NOT attr_expr.
```

複数の属性式を否定することもできますし、次のようにしてフィルター式全体を否定することさえできます。

```
NOT (attr_expr OR attr_expr).
```

否定する複数の属性式 (またはフィルター式) の前後を括弧で囲む必要がある点に注意してください。

**注:** フィルター式では、指定されたバッファの末尾に空白・スペースまたはヌル文字を置く必要があります。つまり、(LENGTH オプションを使用して) 指定するバッファ長には、フィルター式以外のデータを含めてはなりません。

例えば、以下の簡易フィルター式を使用すると、ゼロより大きい記憶保護違反回数を持つ、使用可能なローカル・トランザクションを表す LOCTRAN オブジェクトが選択されます。

```
STATUS=ENABLED AND STGVCNT>0.
```

より複雑なフィルター式を作成することにより、非常に具体的な属性の組み合わせを持つオブジェクトを選択できます。例えば、次の条件を持つ LOCTRAN オブジェクトを選択する場合を考えます。

- P で始まるトランザクション ID を持つ
- PAY で始まるプログラム名を持つ
- 使用可能である
- ゼロ以外の使用回数および記憶保護違反を持つ、または再始動済み

この場合、次のようなフィルター式を指定できます。

```
((TRANID=P* AND PROGRAM=PAY* AND STATUS=ENABLED) AND  
((USECOUNT>0 AND STGVCNT>0) OR NOT RESTARTCNT=0)).
```

この例の RESTARTCNT 属性では、NOT 演算子の代わりに「より大」演算子を使用して指定することもできる点に注意してください。

## パラメーター式

パラメーター式は、結果セットのレコード数を減らすために、いくつかのオブジェクトに対し、フィルターによって提供される機能に加えて追加の機能をサポートしています。

パラメーター式に使用する構文は、36 ページの『リソースに対するアクションの実行』で説明されているものと同じです。例えば、2006 年 7 月 17 日の午後 5 時から午後 5 時 5 分までに完了したタスクの履歴 (HTASK) レコードを含む結果セットを構築するには、次のパラメーター式を使用します。

```
PARM('STARTDATE(07/17/2006) STARTTIME(17:00)  
INTERVAL(300).')
```

GET のパラメーターについて詳しくは、[CICSplex SM リソース・テーブル](#)を参照してください。

## 結果セットの操作

CICSplex SM は、結果セットで選択したリソース・テーブル・レコードを配置します。結果セットは、API プログラムによるアクセス、レビュー、操作が可能なリソース・テーブル・レコードの論理グループです。

結果セットは、以下の 2 つの方法で作成できます。

- リソース・データを取得するための直接 API 要求によって作成します。GET コマンドは、リソース・データを収集して結果セットを作成するための主要な手段です。

- ある結果セットを操作して別の結果セットを作成する API 要求によって作成します。COPY は、既存の結果セット内のレコードから新しい結果セットを作成できるコマンドの例です。レコードのコピー元となる結果セットは、ソース結果セットと呼ばれます。コピー先の結果セットは、ターゲット結果セットと呼ばれます。

同じ結果セット内のすべてのリソース・テーブル・レコードは、同一タイプの管理対象オブジェクトを表している必要があります。つまり、PROGRAM リソース・テーブル・レコードが格納されている結果セットには、LOCTRAN リソース・テーブル・レコードを追加で格納することはできません。これらのリソース・テーブル・レコードは、同じ CICSplex SM コンテキストから収集される必要もあります。このため、いずれかの CICSplex のレコードが含まれる結果セットは、他の CICSplex のレコードを保持するために使用できません。結果セットが作成された後は、その結果セットのリソース・タイプとコンテキストは固定されます。結果セットのタイプやコンテキストを変更するための唯一の方法は、その結果セットの内容を新しいリソース・テーブル・レコードに完全に置き換えることです。

結果セット内のリソース・テーブル・レコードは、実際の管理対象オブジェクトではなく、データが収集された時点の管理対象オブジェクトの属性のレポートであることに留意してください。これは重要な違いです。そのリソース・テーブル・レコードがプログラムに返されたときには既に、実際の管理対象オブジェクトは変化しているか存在しなくなっている可能性があるからです。管理対象オブジェクトが追加または削除されるのに伴い、返されるレコードの数は変わることがありますが、結果セット内のレコードの構造は一定のままです。

単純な API プログラムは、一度に 1 つの結果セットのみを処理する場合があります。データが新たに要求されるたびに、直前の結果セットを置き換える結果セットが作成される可能性があります。複雑なプログラムでは、複数の結果セットが同時に維持されて、これらの結果セットの保持がより直接的に制御されることがあります。

## 結果セット・コマンドの概要

以下のコマンドを使用して、結果セットを作成して、これらの結果セットによって表されるリソースを管理できます。

### GET

管理対象リソースのインスタンスを表す、選択されたリソース・テーブル・レコードを含む結果セットを戻します。

### PERFORM

1 つ以上の管理対象リソースに対して操作を実行します。PERFORM SET は、既存の結果セット内のリソース・テーブル・レコードに対して実行されます。PERFORM OBJECT は、結果セットが存在していても使用できます。このコマンドによって結果セットが暗黙的に作成されます。

### REFRESH

結果セット内のリソース・テーブル・レコードによって表される一部またはすべての管理対象リソースのデータをリフレッシュします。

### SET

結果セット内のリソース・テーブル・レコードによって表される 1 つ以上の管理対象リソースの属性を変更します。

これらのコマンドは、結果セット内のリソース・テーブル・レコードに影響を与えるだけでなく、それらのレコードによって表される管理対象リソースにも影響を与えます。[22 ページの図 4](#) は、これらのコマンドと API 環境の関係を示しています。

## Managed Resource Commands

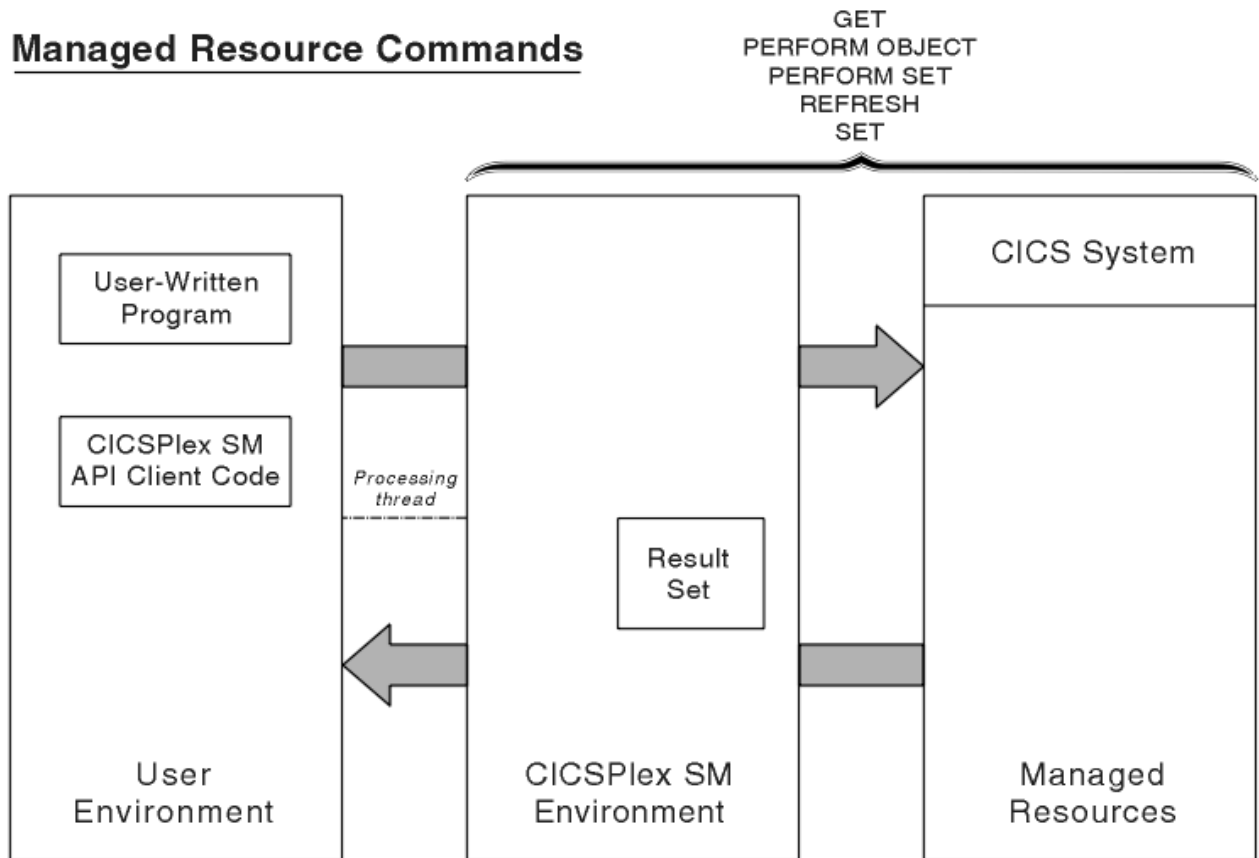


図 4. 管理対象リソースを操作する API コマンド

結果セットが作成されたら、その結果セットに含まれるレコードに対して各種の操作を実行できます。結果セット内のレコードのソート、マーク付け、コピー、削除、および集計を実行できます。おそらく最も重要なこととして、結果セットからローカル・ストレージにレコードを取得して、そこでこれらのレコードをプログラムによって処理できます。

以下のコマンドを使用して、結果セット内の 1 つ以上のレコードを操作できます。

### **COPY**

結果セット内の一部またはすべてのリソース・テーブル・レコードを別の結果セットにコピーします。

### **DELETE**

結果セットから 1 つ以上のリソース・テーブル・レコードを削除します。

### **EXPAND**

1 つの集計レコード内に集計されたすべてのレコードが含まれる結果セットを戻します。

### **FETCH**

結果セット内の 1 つ以上のリソース・テーブル・レコードのデータと状況情報を取得します。

### **GROUP**

結果セット内の一部またはすべてのリソース・テーブル・レコードをグループ化することで、集計された結果セットを戻します。

### **LOCATE**

結果セット内のレコード・ポインターを配置します。

### **MARK**

結果セット内の選択されたリソース・テーブル・レコードにマークを付けます。

### **ORDER**

結果セット内のリソース・テーブル・レコードをソートします。

## UNMARK

以前に実行された MARK コマンドによってリソース・テーブル・レコードに付けられたマークを削除します。

これらのコマンドは、結果セットの現在の内容のみに影響を与え、その結果セットによって表される管理対象リソースには影響を与えません。23 ページの図 5 は、これらのコマンドと API 環境の関係を示しています。

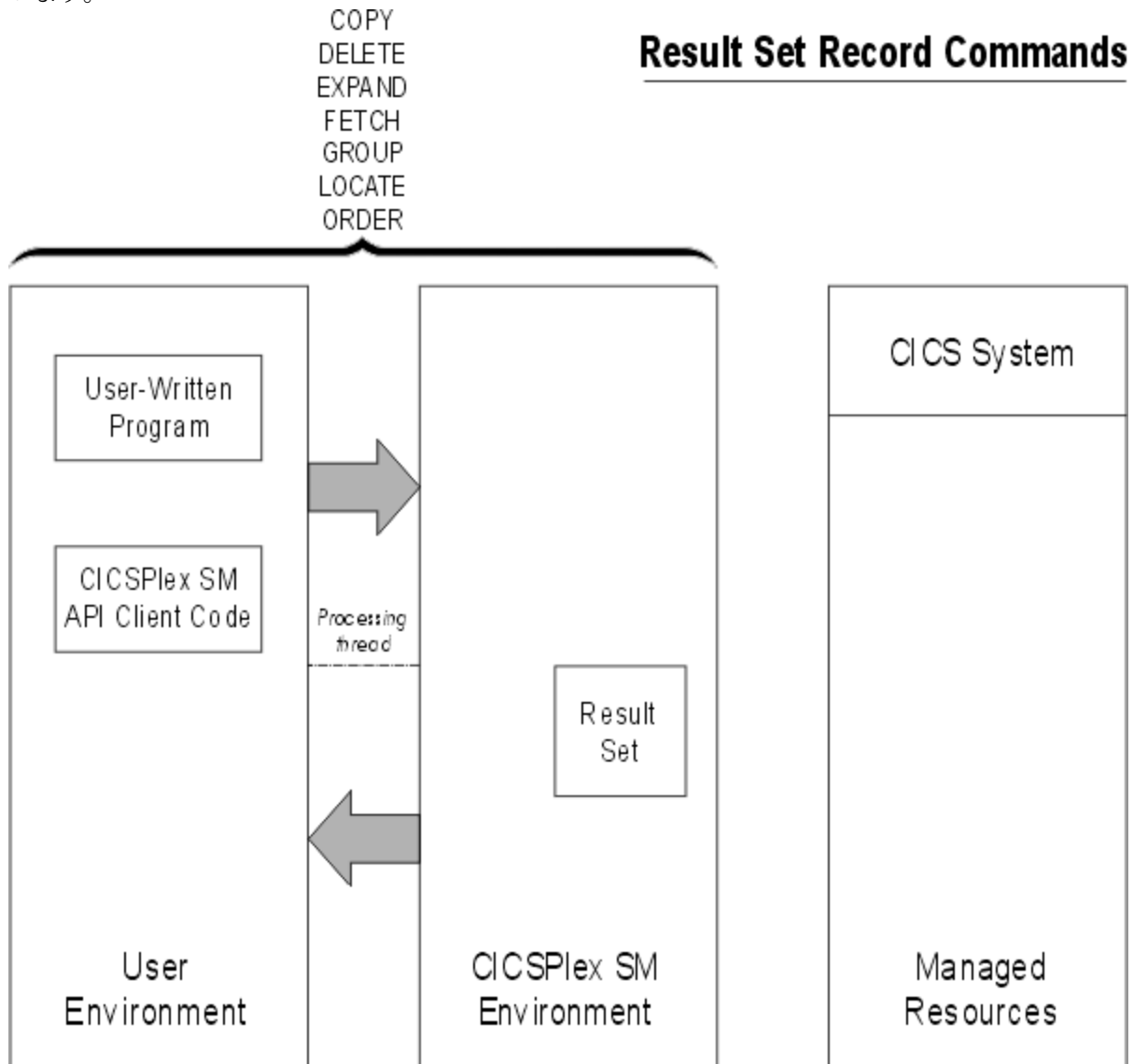


図 5. 結果セット・レコードを操作する API コマンド

CICSplex SM では、複数の結果セットを一まとまりとして管理するためのツールも提供されています。これらのツールは、結果セットの内容を制御するためのフィルターやビュー、および結果セットを確認および破棄するためのコマンドです。

以下のコマンドを使用して、結果セットとその内容を管理できます。

## DISCARD

結果セットを破棄します。

## QUERY

結果セットに関する情報とその結果セットに含まれるリソース・テーブル・レコードに関する情報を取得します。

## SPECIFY FILTER

結果セットの内容を制御するために使用できる属性フィルターまたは値フィルターを定義します。

## SPECIFY VIEW

結果セットの内容を制御するために使用できる、リソース・テーブルのカスタマイズしたビューを作成します。

これらのコマンドは、既存のまたは新規作成された結果セットのみに影響を与え、その結果セットによって表される管理対象リソースには影響を与えません。24 ページの図 6 は、これらのコマンドと API 環境の関係を示しています。

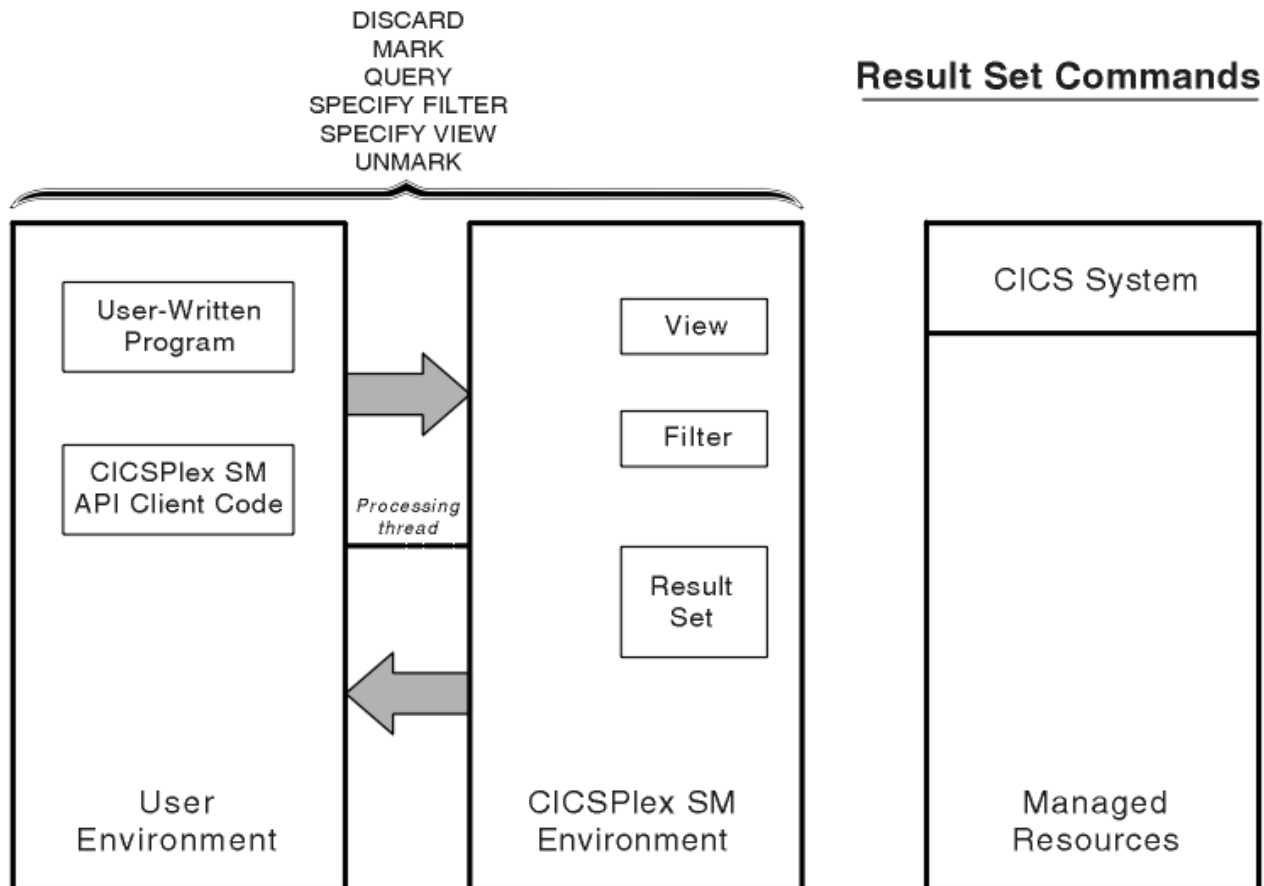


図 6. 結果セットを操作する API コマンド

## 結果セットからのレコードの取得

結果セットを作成したら (GET コマンドを使用)、その結果セットに含まれている一部またはすべてのレコードを処理するためにローカル・ストレージに転送できます。

FETCH コマンドを使用して、単一のリソース・テーブル・レコード、選択した複数のレコード、または結果セット全体を一度に取得できます。

取得する各リソース・テーブル・レコードには、それによって表される管理対象リソースに関する現在のデータが含まれています。各レコードには、CICSplex SM によって維持されている特定の状況情報も含まれています。

この状況情報は、OBJSTAT というリソース・テーブルとして表示されます。OBJSTAT リソース・テーブルの内容を以下で説明しています。

実際には、結果セット内の各レコードには、一対のリソース・テーブル (OBJSTAT リソース・テーブルのインスタンスと、それに続く要求されたリソース・テーブルのインスタンス) が含まれています。管理対象リソースのデータと OBJSTAT 状況情報は、FETCH コマンドで指定するオプションに応じて、対で取得することも別個に取得することもできます。

## DATA

指定されたリソース・テーブル・データのみを取得します。



## STATUS

OBJSTAT 状況情報のみを取得します。

## BOTH

リソース・テーブル・データと OBJSTAT 状況情報の両方を取得します。

25 ページの図 7 は、結果セット・レコードに含まれている情報と、その情報を取得するために使用できる FETCH コマンドを示しています。

GET OBJECT(LOCTRAN) RESULT(TOKENA) ...

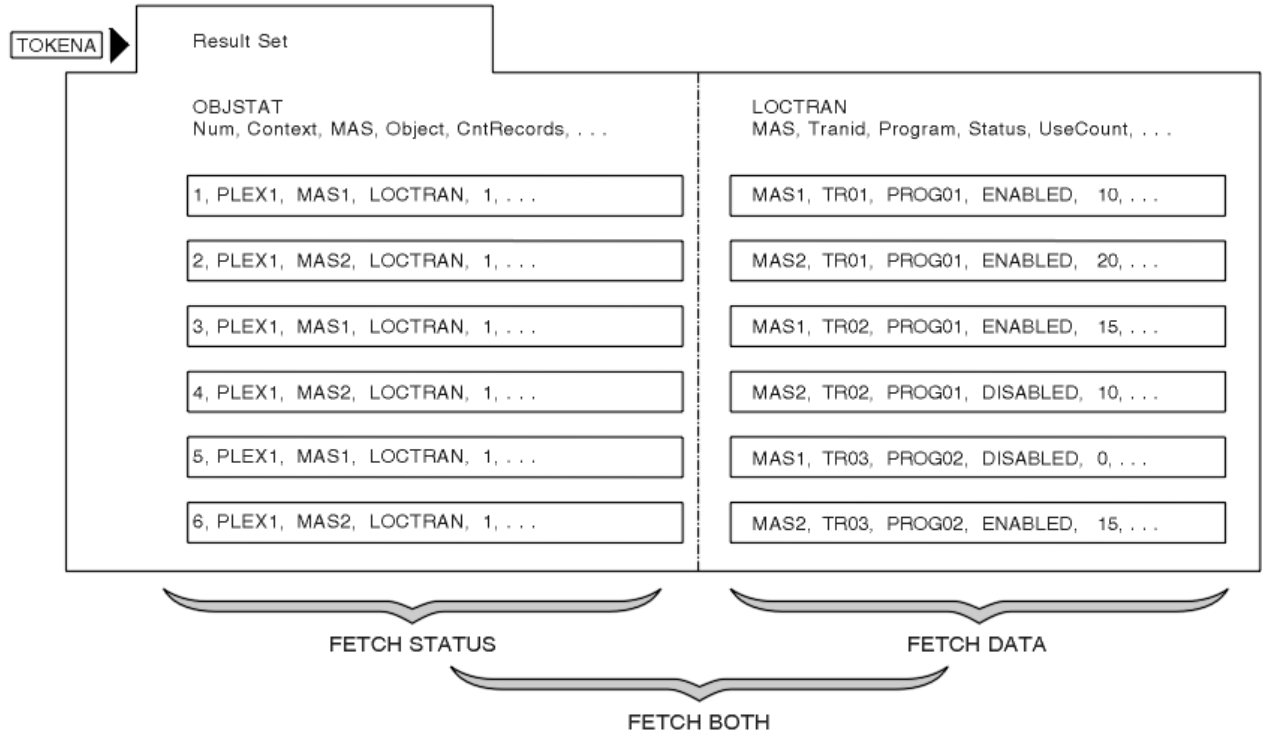


図 7. FETCH を使用した結果セット・レコードの取得

TOKENA によって参照される結果セットは、LOCTRAN レコード向けの GET コマンドを発行することによって作成されました。結果セット内の各レコードは、LOCTRAN データと OBJSTAT データで構成されています。

25 ページの図 7 に示す FETCH コマンドを使用して、一部またはすべてのデータを選択的に取得できます。例えば、25 ページの図 8 は、FETCH DATA コマンドの出力を示しています。

FETCH DATA ALL RESULT(TOKENA) INTO(AREA1) ...

MAS1, TR01, PROG01, ENABLED, 10, ...
MAS2, TR01, PROG01, ENABLED, 20, ...
MAS1, TR02, PROG01, ENABLED, 15, ...
MAS2, TR02, PROG01, DISABLED, 10, ...
MAS1, TR03, PROG02, DISABLED, 0, ...
MAS2, TR03, PROG02, ENABLED, 15, ...

図 8. FETCH DATA の出力例

## OBJSTAT

OBJSTAT リソース・テーブルでは、結果セット内の特定のレコードの状況情報が示されます。

名前

説明

**RECORDNUM**

結果セット内のレコードの番号。

**CONTEXT**

レコードのデータが収集されたときの有効なコンテキスト。

**CICSNAME**

データの収集元の CICS システムの名前。

**CICSREL**

データの収集元の CICS システムのリリース・レベル。

**OBJECT**

データの参照先の管理対象オブジェクトの名前。

**OBJTYPE**

管理対象オブジェクトのデータ型:

- 1 CICSplex SM リソース
- 2 論理ビュー

**RECTYPE**

レコード・データのタイプ:

- 1 詳細データ
- 2 要約データ

**LASTOPER**

オブジェクトに対して実行された最後の操作:

- 1 COPY 操作
- 2 DELETE 操作
- 3 GET 操作
- 4 MARK 操作
- 5 REFRESH 操作
- 6 PERFORM OBJECT 操作
- 7 PERFORM SET 操作
- 8 SET 操作
- 9 UNMARK 操作

**STATUS**

現在のレコードの状況:

1... ..	X'80'	The record is MARKED
.... ..1	X'01'	Operation error

このビットがオンに設定されている場合、この属性はそのバージョンの CICS では有効ではありません。

## CNTRECORDS

レコード・カウント。RECTYPE=1 の場合、レコード・カウントはゼロです。RECTYPE=2 の場合、明細レコードの数がレコード・カウントに反映されます。

## KEYLEN

キー・データの長さ。

## KEYDATA

ネイティブ・キー・データ

## RESERVE1

将来使用するための予約域。

## 結果セット内のレコード・ポインターの配置

CICSplex SM では、各結果セット内に現行レコード・ポインターが維持されています。結果セットを最初で作成すると (GET コマンドなどを使用して)、ポインターはその結果セットの先頭に配置されます。その結果セットに対して最初に発行するコマンドは、最初のレコードに影響を与えます。

ほとんどの場合は、FETCH コマンドを発行して結果セットからレコードを取得すると、レコード・ポインターは結果セット内の次のレコードに配置されます (つまり、最後にフェッチされたレコードの直後のレコード)。ただし、特定の API コマンドは常に最後にフェッチされたレコードに対して実行されます。これらのコマンドのいずれかを FETCH コマンドの後に発行すると、レコード・ポインターは次のレコードに進みません。

- COPY
- DELETE
- MARK
- UNMARK
- PERFORM SET CURRENT
- REFRESH CURRENT
- SET CURRENT

結果セット内のレコード・ポインターは、レコードを取得している方向に応じて前方または後方に移動する可能性があります。FETCH コマンドを発行し、指定した基準に一致するレコードが見つからない場合、レコードは取得されません。その場合、ポインターは、ポインターが移動していた方向に応じて結果セットの先頭または末尾に配置されます。

FETCH コマンドを発行し、すべてのレコードを取得するためのストレージが不足している場合、ポインターは、十分なスペースがあれば最後に取得されていたはずのレコードに配置されます。ポインターは、最後に取得されたレコードには配置されていません。ポインターの位置を明確にするには、LOCATE コマンドを使用して結果セット内で明示的に配置する必要があります。

GET コマンドと FETCH コマンドは、レコード・ポインターを事前定義された特定の位置に保持しますが、他の API コマンドの場合は保持しません。多くの API コマンドは、結果セット内のレコードを操作したり結果セット内のデータを更新したりします。いずれかのコマンドの後のレコード・ポインターの位置は、そのコマンドで指定したオプションを含め、複数の要因の組み合わせによって異なります。ポインターは、1 つ以上のレコード分だけ前方または後方に移動したり、結果セットの先頭または末尾に配置されたりする可能性があります。CURRENT オプションを指定した場合は、レコード・ポインターは移動せずに、コマンドの完了後も現在のレコードに配置されたままになります。

このため、CICSplex SM では、結果セット内で明示的にレコード・ポインターを配置できる LOCATE コマンドが用意されています。以下のいずれかのコマンドを発行した後にレコード・ポインターを使用する必要がある場合は、まず LOCATE コマンドを使用して再配置してください。

- COPY
- DELETE
- GETDEF
- GROUP
- MARK

- ORDER
- PERFORM OBJECT
- PERFORM SET
- REFRESH
- SET
- UNMARK.

## 結果セット内の選択されたレコードの処理

結果セット内のリソース・テーブル・レコードのサブセットを処理する場合は、関心のあるレコードを特定できます。

これは、次のように行うことができます。

- SPECIFY FILTER コマンドを使用して、[17 ページの『フィルター式の使用』](#)の説明に従って、レコードを選択するためのフィルターを定義します。
- MARK と UNMARK の各コマンドを使用してレコードのマークを付けます。

### MARK と UNMARK の使用

MARK コマンドを使用して、将来参照するために結果セット内の一部またはすべてのリソース・テーブル・レコードにマークを付けることができます。

UNMARK コマンドを使用して、選択したレコードから既存のマークを削除できます。結果セット内のレコードにマークを付けたら、以降のコマンドで、マークありのレコードやマークなしのレコードを参照できます。以下の API コマンドでは、MARKED オプションと NOTMARKED オプションがサポートされています。

- COPY
- DELETE
- FETCH
- GROUP
- LOCATE
- PERFORM SET
- REFRESH
- SET

例えば [29 ページの図 9](#) は、選択されたリソース・テーブル・レコードにマークが付けられた結果セットを示しています。その後 FETCH コマンドで MARKED オプションを使用して、マークが付けられたレコードのみを取得しています。

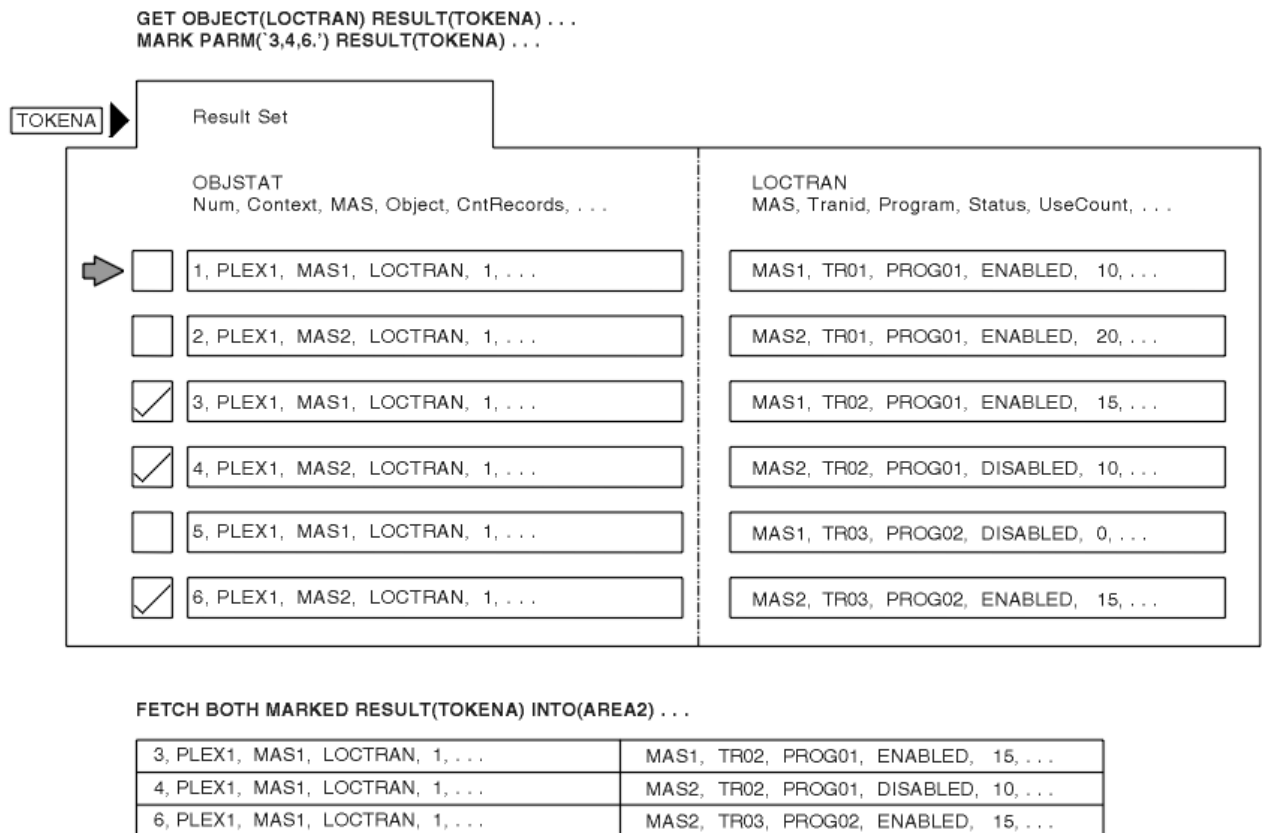


図 9. 結果セット内のレコードのマーク付けと取得

### マークを付けるレコードの特定

デフォルトでは、MARK または UNMARK コマンドを発行すると、現在のリソース・テーブル・レコードのみにマークが付けられるか、マークが解除されます。

ただし、マークを付けるレコードを特定するためのさまざまな方法があります。

- 現行レコード以外の特定期間レコードにマークを付けるには、POSITION オプションを使用して、結果セット内の相対位置によってそのレコードを特定します。
- 以前定義したフィルター基準に一致する 1 つ以上のレコードにマークを付けるには、FILTER オプションまたは NOTFILTER オプションを使用します。
- 結果セット内のすべてのレコードにマークを付けるには、ALL オプションを使用します。

これらのオプションに加えて、PARM オプションを使用してマーク対象のレコードのリストを指定できます。PARM オプションを使用するには、パラメーター式でレコード番号の文字ストリングを指定します。パラメーター式には、以下を含めることができます。

- 個別のレコード番号 (コンマで区切る)。
- レコード番号の範囲 (範囲の両端の番号をコロンで区切る)。

パラメーター式全体は、ピリオドで終わらなければなりません。

例えば、結果セット内のレコード 1、3、6 から 9、および 24 にマークを付けるには、次のように指定します。

```
PARM('1,3,6:9,24.')
```

PARM オプションを使用する際は、PARMLEN オプションも使用して、パラメーター式を格納するバッファの長さを指定する必要があります。

注：

1. 負の値と 0 は有効なレコード番号ではありません。無効なレコード番号を指定した場合は、MARK (または UNMARK) コマンドは RESPONSE と REASON の値として INVALIDPARM PARM を戻します。
2. 範囲の最初に大きい値を間違えて指定した場合 (例: 9:6)、CICSplex SM は、2 つの値が反転されて有効な範囲が生成されます。
3. 単一の値の前または後ろにコロンを間違えて指定した場合 (例: 6:), このコロンは無視されます。CICSplex SM によって、指定したレコードのみにマークが付けられます。

### マークを付けることができなかったレコードの特定

レコードのマークを付けるか解除する際は、特定したすべてのレコードが正常に処理されたかどうかかわかると便利です。

例えば、CICSplex SM に結果セットから以前に削除されたレコードにマークを付けるか解除するように間違えて指定する場合があります。または、特定したレコード番号が結果セットにとって範囲外である可能性もあります。

MARK または UNMARK コマンドで COUNT オプションを使用すると、マークを付けるか解除できなかったレコードの数を確認できます。INTO オプションと LENGTH オプションを使用して、マークを付けることができなかったレコードのリストを受け取るバッファを指定することもできます。INTO バッファの長さを決定する際の留意事項として、MARK 要求の結果として生じる可能性のある最大数のレコード番号を保持するのに十分な長さにする必要があります (どのレコードにもマークを付けることができなかった場合)。さらに、すべてのレコード番号は個別に (範囲ではなく) INTO バッファ内に、コンマで区切られてリストされます。したがって、例えば次のように PARM オプションを指定したとします。

```
PARM('1,3:6,12,15.')
```

この場合、INTO バッファは次の文字ストリングを保持するのに十分な長さである必要があります。

```
1,3,4,5,6,12,15
```

指定した INTO バッファの長さが、マークを付けることができなかったレコードの完全なリストを保持するのに十分でない場合は、RESPONSE の値として WARNING AREATOOSMALL を受け取ります。その場合は、INTO バッファからはレコード・リストの一部が戻されて、LENGTH の値は完全なリストに必要なバッファ長に設定されます。その場合は、適切な LENGTH 値を指定した MARK コマンドを再実行して、マークを付けることができなかったレコードを特定できます。

### 結果セット内のマークを削除する方法

UNMARK コマンドを使用して、以前に実行された MARK コマンドによってリソース・テーブル・レコードに付けられた一部のまたはすべてのマークを削除できます。ただし、それと同時に他のレコードにマークを付ける場合は、MARK コマンドの RESET オプションを使用することで手順を 1 つ減らすことができます。

デフォルトでは、結果セット内で既にマークが付けられているレコードに加えて、MARK コマンドで指定したレコードにマークが付けられます。つまり、RESET オプションを使用しない限り、以前にマークが付けられたリソース・テーブル・レコードはマークが付いたままになります。RESET オプションを使用すると、以前に付けられたすべてのマークが結果セットからワイプされます。このため、処理の完了後は、現在の MARK コマンドで特定されたレコードのみにマークが付いた状態になります。MARK コマンドで RESET オプションを使用する方法は、UNMARK コマンドを使用してから MARK コマンドを使用する方法の代替手段となります。

**注:** COPY コマンドを使用して、マークが付けられているリソース・テーブル・レコードを現在の結果セットから別の結果セットにコピーすると、それらのレコードに付けられたすべてのマークは削除されます。

## 結果セット内のレコードの集計

結果セット内の多数のレコードを分析したり変更したりする場合は、それらのレコードを集計すると便利です。

GROUP コマンドを使用すると、特定のリソース・テーブル属性の値に基づいて結果セット内のレコードを集計できます。

**注:** 長さが 251 以下の属性のみに基づいて集計できます。

GROUP コマンドを発行すると、CICSplex SM によって特定の結果セット内のレコードが集計され、新しい集計結果セットが作成されます。集計結果セットは、特別なタイプの結果セットです。これには、ソース結果セット内の 1 つ以上のレコードに対応する集計リソース・テーブル・レコードが含まれています。

例えば、GROUP コマンドを使用して、LOCTRAN リソース・テーブル・レコードが含まれた結果セットを集計できます。これらのレコードを STATUS 属性の値に従ってグループ化する場合、集計結果セットには最大で 2 つのレコードが含まれます。1 つは、STATUS の値が ENABLED であるレコードを表すもので、もう 1 つは、STATUS の値が DISABLED であるレコードを表すものです。31 ページの図 10 は、GROUP コマンドをこのように使用する例を示しています。

GROUP BY(STATUS) FROM(TOKENA) TO(TOKENB) ...

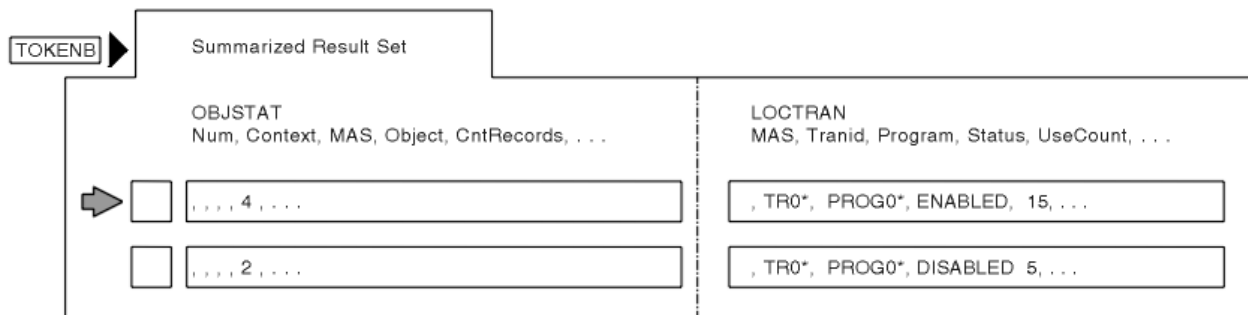


図 10. GROUP を使用した結果セット・レコードの集計

一般に、集計結果セットの操作は、通常の結果セットの操作と同じ方法で行うことができます。FETCH コマンドを使用して、集計結果セットからレコードを取得できます。集計のベースとなっているソース結果セットの個別レコードを取得することもできます。FETCH コマンドの DETAIL オプションを使用すると、ソース結果セット内のレコードのうち、特定の集計レコードに対応するレコードのサブセットを取得できます。

31 ページの図 11 では、集計レコードと関連付けられた明細レコードをフェッチする例を示しています。この例では、集計レコードは、すべての有効化されたトランザクションを表していた LOCTRAN レコードでした。

FETCH DETAIL RESULT (TOKENB) INTO (AREA3) ...

MAS1, TR01, PROG01, ENABLED, 10, ...
MAS2, TR01, PROG01, ENABLED, 20, ...
MAS1, TR02, PROG01, ENABLED, 15, ...
MAS2, TR03, PROG02, ENABLED, 15, ...

図 11. FETCH DETAIL の出力例

PERFORM コマンドまたは SET コマンドを使用して、集計結果セット内のレコードを変更できます。このことは、ソース結果セット内のレコードのうち、特定の集計レコードによって表されているすべてのレコードを変更することと同じです。ただし、集計結果セット内の各レコードに単一の OBJSTAT レコードが関連付けられる (変更対象の各ソース・レコードに 1 つの OBJSTAT レコードが関連付けられるのではなく) ため、FETCH DETAIL コマンドを使用して集計操作の結果を確認すると役立つ場合があります。

集計結果セット内の個別レコードを操作するためのもう 1 つの方法は、EXPAND コマンドを使用することです。このコマンドは DETAIL オプションを指定した FETCH コマンドと似ていますが、EXPAND コマンドでは、個別の集計レコード内の GROUP 別に集計された各レコードについて 1 つのレコードが含まれた新しい結果セットが作成されます。これにより、GROUP または FETCH コマンドの追加の使用など、結果セットに対するアクションをさらに実行できます。EXPAND コマンドには、操作対象の集計レコードを選択するためにレコード・カウンターを操作するためのいくつかのオプションが用意されています。このコマンドは、MARK コマンドや UNMARK コマンドと組み合わせて使用することもできます。

GROUP コマンドを発行したときは、ソース結果セット内の OBJSTAT レコードは集計されません。このため、集計結果セット内の OBJSTAT レコードはすべてのソース・レコードの OBJSTAT 情報を表しません。

ただし、集計結果セット内の OBJSTAT レコードに含まれている集計カウントによって、各集計レコードを作成するために組み合わせられたソース・レコードの数が示されます。

集計結果セットとそのソース結果セットは、一緒に使用されるペアとして考える必要があります。これらの結果セットは特定の属性を共用しているため、集計結果セットはソース・レコードに対する以下のような一定の依存関係を持ちます。

- 集計結果セットは、その作成元となったソース結果セットなしでは存在できません。ソース結果セットを破棄した場合は、そのソース結果セットから作成されたすべての集計結果セットも破棄されます。
- 集計結果セットを再利用できるのは、同じソース結果セット内のレコードを再集計する場合のみです。既存の集計結果セットを、異なるソース結果セット向けの GROUP コマンドのターゲットとして使用することはできません。
- 集計結果セットを COPY コマンドのソースとして使用することはできません。
- ソース結果セットまたは集計結果セットを何らかの方法で変更した場合、そのソース結果セットから作成されたすべての集計結果セットは再作成されます。

注：このことを防止するには、PERFORM コマンドまたは SET コマンドで NOREFRESH オプションを指定します。

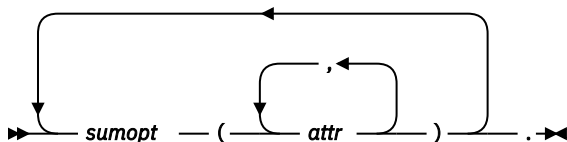
### 集計式の指定

集計レコードの属性は、その属性のデータ・タイプに適した集計オプションに従って設定されます。各リソース・テーブル属性について、CICSplex SM によってデフォルトの集計オプションが定義されます。これらのデフォルトを明示的に指定変更しない限り、CICSplex SM でレコードを集計するときこれらのデフォルトが使用されます。

レコード内の属性の集計方法を CICSplex SM に指示するには、GROUP コマンドの SUMOPT オプションで集計式を指定します。集計式は、1 つ以上の集計オプションと、それらの集計オプションが適用されるリソース・テーブル属性で構成される文字ストリングです。

集計式の構文は、以下のとおりです。

#### 集計式



説明:

#### sumopt

指定されたリソース・テーブル属性用に使用される集計オプションです。

#### AVG

平均の属性値を返します。数値フィールドに対してのみ有効です。

#### DIF

すべての基盤レコードに共通する文字を返し、共通ではない文字についてはアスタリスク (\*) を表示します。文字フィールドに対してのみ有効です。

#### LIKE

すべてのレコードに共通の値が含まれている場合は、CVDA または EYUDA の値を返します。そうでない場合は、N/A と表示します。CVDA フィールドと EYUDA フィールドに対してのみ有効です。

#### MAX

最大の属性値を返します。

#### MIN

最小の属性値を返します。

#### SUM

属性値の合計を返します。数値フィールドに対してのみ有効です。

同一の集計式で同じ集計オプションを複数回指定できます。



**attr**

リソース・テーブルの属性の名前。

**注:** 長さが 251 以下の属性のみに基づいて集計できます。

各集計オプションについて、必要な数の属性名を指定できます。

**注:** 集計式では、指定されたバッファの末尾にブランク・スペースまたはヌル文字を置く必要があります。つまり、(LENGTH オプションを使用して) 指定するバッファ長には、集計式以外のデータを含めてはなりません。

例えば、LOCTRAN レコードをグループ化する際に次のような集計式を使用できます。

```
SUM(USECOUNT) MAX(PRIORITY,TWASIZE) .
```

デフォルトでは、これらの属性の値の平均が算出されます。ただし、この集計式では、各集計レコードには全 USECOUNT 値の合計および最大の PRIORITY 値と TWASIZE 値を含めることを指定しています。

33 ページの表 6 では、各種データ・タイプの有効な集計オプションを示しています。

表 6.						
属性データ・タイプ別の有効な集計オプション						
	AVG	DIF	LIKE	MAX	MIN	SUM
ADDRESS				X	X	
AVG	X			X	X	X
AVG3	X			X	X	X
BIN	X			X	X	X
BIT				X	X	
CHAR		X		X	X	
CODEBIN	X			X	X	X
COMPID				X	X	
CVDAS			X	X	X	
CVDAT			X	X	X	
DATE				X	X	
DATETIME				X	X	
DEC				X	X	
DECSTP				X	X	
EYUDA			X	X	X	
HCHAR		X		X	X	
HEX		X		X	X	
HHMM				X	X	
INTVMSEC	X			X	X	X
INTVSEC	X			X	X	X
INTVSTCK	X			X	X	X
INTVUSEC	X			X	X	X
INTV16US	X			X	X	X

表 6. (続き)						
属性データ・タイプ別の有効な集計オプション						
PCT	X			X	X	X
PCT3	X			X	X	X
RATE	X			X	X	X
RATE3	X			X	X	X
RATIO	X			X	X	X
RESTYPE				X	X	
SCLOCK	X			X	X	X
SCLOCK12	X			X	X	X
SUM	X			X	X	X
SUM3	X			X	X	X
TEXT		X		X	X	
TIMESTP				X	X	

## 結果セット内のレコードのソート

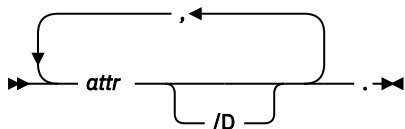
結果セット内のレコードは、リソース・テーブルのキー属性によって正常にソートされます。

CICS リソース・テーブルと CICS モニター対象テーブルの場合は、レコードはその収集元となった CICS システムに基づいてソートされます。結果セットを操作するとき、レコードが任意の論理的な順序になっている方がレコードを処理しやすいことがあります。CICSplex SM API の **ORDER** コマンドを使用すると、特定のリソース・テーブル属性の値に従って結果セット内のレコードをソートできます。CICS 管理クライアント・インターフェース (CMCI) を使用している場合は、代わりに **ORDERBY** パラメーターを使用してください。

CICSplex SM API を使用している場合は、**ORDER** コマンドの **BY** オプションで配列式を指定することで、レコードのソート方法を選択できます。CICS 管理クライアント・インターフェース (CMCI) を使用している場合は、**ORDERBY** パラメーターを使用してレコードのソート方法を指定できます。配列式は、リソース・テーブル・レコードのソートで使用される 1 つ以上の属性名で構成される文字ストリングです。

レコードをソートするための配列式の構文は次のとおりです。

### 配列式 - レコードのソート



説明:

#### **attr**

リソース・テーブルの属性の名前。

必要な数の属性名を指定できますが、配列式の合計長はコンマやブランク・スペースを含めて 255 文字を超えてはなりません。

#### **/D**

属性値を降順でソートすることを指定します。デフォルトでは、値は昇順にソートされます。

注: 配列式の後ろから指定されたバッファの末尾までは、ブランク・スペースまたはヌル文字にする必要があります。つまり、(LENGTH オプションを使用して) 指定するバッファ長には、配列式以外のデータを含めてはなりません。

例えば、トランザクション ID と有効化状況に基づいて LOCTRAN レコードの結果セットをソートするには、次のように指定します。

```
TRANID, STATUS.
```

この例では、トランザクション ID は 1 次ソート・キーであり、有効化状況は 2 次ソート・キーです。使用回数の降順でレコードをソートするには、次のように属性名の後ろに /D を追加します。

```
USECOUNT/D
```

## 管理対象リソースの変更

CICSplex SM によって管理されているリソースをさまざまな方法で変更できます。ここで説明するアクションは、結果セット内のリソース・テーブル・レコードに対して発行されます。ただし、要求した変更は、それらのレコードによって表される実際のリソースに加えられます。

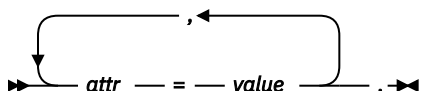
### リリース属性の変更

SET コマンドまたは UPDATE コマンドを使用してリソース属性の現行値を変更できます。

SET では CICS リソースの属性が変更されることに対して、UPDATE では CICSplex SM 定義と CICS 定義が変更されます。これらのコマンドの MODIFY オプションでは変更式を指定できます。変更式は、属性に加える変更内容を定義する文字ストリングです。

変更式は、次の形式の 1 つ以上の属性式で構成できます。

#### 変更式



説明:

#### attr

リソース・テーブル内の変更可能な属性の名前。

#### 値は

属性の値として設定する値です。次の制限が適用されます。

- これは、属性に対して有効な値でなければなりません。
- 値に組み込みブランクまたは特殊文字 (ピリオド、コンマ、等号など) が含まれる場合、以下のよう  
に、値ストリング全体を単一引用符で囲む必要があります。

```
DESCRIPTION='Payroll.OCT'
```

- 値に単一引用符またはアポストロフィーを組み込む場合は、文字を繰り返す必要があります。例  
えば、次のようにします。

```
DESCRIPTION='October''s Payroll'
```

注: CICSplex SM の値ストリングで単一引用符を使用する際は、必ずプログラミング言語の引用規則を考慮してください。

注: 変更式では、指定されたバッファの末尾にブランク・スペースまたはヌル文字を置く必要があります。つまり、(LENGTH オプションを使用して) 指定するバッファ長には、変更式以外のデータを含めてはなりません。

例えば、1 つ以上のローカル・トランザクション (LOCTRAN) を無効にするには、次のように指定します。

```
STATUS=DISABLED.
```

上記は、SET コマンドの MODIFY オプションで指定します。

要求された変更をサポートしていない CICS システムに対して SET コマンドを発行した場合、要求はそれらの CICS システムで無視されます。コンテキストとスコープが、変更をサポートしない CICS システムのみで構成される場合、NOTAVAILABLE SCOPE の RESPONSE 値と REASON 値を受け取ります。

CICS トランザクション定義 (TRANDEF) のタスク保管場所を変更するには、次のように指定します。

```
TASKDATALOC=ANY
```

上記は、UPDATE コマンドの MODIFY オプションで指定します。

UPDATE の MODIFY オプションは、CICS 定義リソース・テーブルに対してのみ有効です。

各リソースの属性とそれらの有効値のリストについては、[CICSplex SM リソース・テーブル](#)を参照してください。

## リソースに対するアクションの実行

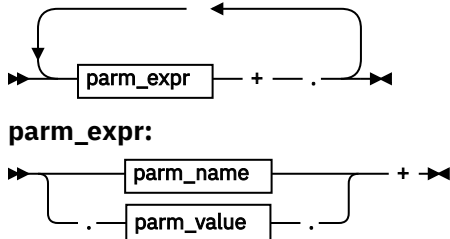
個々の属性を変更することに加えて、PERFORM コマンドの PERFORM OBJECT または PERFORM SET のいずれかを使用することによって多数のリソースに対してアクションを実行することもできます。

これら 2 つコマンドの間の違いは、PERFORM SET が既存の結果セット内のリソース・テーブル・レコードに対してアクションを実行するのにに対し、PERFORM OBJECT はまず結果セットを作成し、その後、要求されたアクションを実行するという点です。

アクションの中には、自己完結型および自己説明型のものがあります。その場合、アクションを指定するだけで、リソースに対してどのような変更を行うかが示されます。例えば、LOCFILE リソース・テーブル・レコードに対して DISCARD アクションを発行することで、ローカル・ファイルを破棄できます。

その他のアクションでは、追加パラメーターを指定する必要があります。この種のアクションでは、必要な機能を取得するためにパラメーター式を必要とする場合があります。パラメーター式は、次の形式の 1 つ以上の parm 式で構成することができます。

パラメーター式



ここで、

- parm\_name は、アクションに関連したパラメーターの名前です。
- parm\_value は、指定のパラメーター名に関連した値です (該当する場合)。

parm\_exp の複数インスタンスが存在する場合は、スペースで区切る必要があります。パラメーター式バッファーは、ピリオド (.) で終了します。

### 例

- ローカル・ファイル (LOCFILE) を使用不可にするには、現在使用中のファイルを処理する方法を指示する必要があります。これは、次のパラメーター式を指定することによって行えます。

```
PARM('BUSY(cvda).')
```

ここで cvda は、ファイル使用中状態に関する有効な CVDA 値 (例えば WAIT、NOWAIT、または FORCE) です。

- ダンプ・コードとダンプのタイトルを指定して CICS 領域 (CICSRGN) のダンプを要求するには、次のパラメーター式を使用できます。

```
PARM('DUMPCODE(PMR12345) TITLE('Doc for PMR12345').')
```

parm\_value にスペースやピリオドなどの特殊文字が含まれる場合、値を単一引用符で囲む必要がある点に注意してください。また、パラメーター値はすべて大文字に変換されます。

要求されたアクションをサポートしていない CICS システムに対して PERFORM コマンドを発行した場合、要求はそれらの CICS システムで無視されます。コンテキストとスコープが、アクションをサポートしない CICS システムのみで構成される場合、NOTAVAILABLE SCOPE の RESPONSE 値と REASON 値を受け取ります。

PERFORM OBJECT コマンドでは、既存の結果セットは必要ありません。実質的には、GET コマンドに続いて PERFORM SET が実行されるからです。この場合、以下で示すように、パラメーター式が GET と PERFORM SET のどちらで有効かどうかに応じて、コマンドの GET または PERFORM SET フェーズでパラメーター式を渡すことができます。

- パラメーター式が GET で有効な場合、PERFORM OBJECT コマンドの GET フェーズで使用されます。
- パラメーター式が PERFORM で有効な場合、PERFORM OBJECT コマンドの PERFORM SET フェーズで使用されます。
- パラメーター式が GET と PERFORM で有効な場合、PERFORM OBJECT コマンドの GET および PERFORM SET フェーズで使用されます。この規則は、PERFORM OBJECT では実行できないアクションが存在する可能性があることを意味しています。意図した結果を得るために、GET コマンドと PERFORM SET コマンドを別個に使用することが必要になる場合があります。
- パラメーターが GET または PERFORM で有効でない場合、INVALIDPARM PARM 条件が発生します。

各リソースの有効なアクションとそれらの必須パラメーターのリストについては、[CICSplex SM リソース・テーブル](#)を参照してください。

## CICSplex SM と CICS の定義の操作

CICSplex SM と CICS の定義を操作する場合は、いくつかの特別な API コマンドとコマンド・オプションを使用できます。

### 定義の作成、更新、および削除

特定の API コマンドを使用して、データ・リポジトリ内の CICSplex SM と CICS の定義を保守できます。

#### CREATE

指定した属性値を使用して、新しい CICSplex SM 定義または CICS 定義を作成します。新しい定義は、データ・リポジトリに保管されます。

#### UPDATE

指定した属性値に従って、既存の CICSplex SM 定義または CICS 定義を更新します。更新された定義によって、データ・リポジトリ内の既存の定義が置き換えられます。

#### REMOVE

CICSplex SM 定義または CICS 定義をデータ・リポジトリから削除します。

注：

1. 定義を更新または削除する前に、FETCH コマンドを使用して適切なリソース・テーブル・レコードの結果セットから取得する必要があります。
2. CICSplex をコンテキストとして持つ CICSplex SM 定義の場合は (ワークロード管理定義やリアルタイム分析定義など)、加えた変更は、その CICSplex の管理に関与しているすべての CMAS に自動的に配布されます。

これらの各コマンドで FROM オプションを使用して、操作対象の定義の CICSplex SM 定義リソース・テーブル・レコードまたは CICS 定義リソース・テーブル・レコードを指定します。このレコードには、その定義のリソース・テーブル内のすべての属性が含まれている必要があります。特定のオプション属性を指定しない場合は、それらのフィールドをヌル (ゼロ) 値に設定する必要があります。

別の方法として、CICS 定義を更新する際に、UPDATE コマンドの RESULT オプションと MODIFY オプションを使用できます。これらのオプションを使用すると、複数の定義を同時に変更できます (これは CICSplex SM エンド・ユーザー・インターフェースから ALTER アクション・コマンドを発行することと同じです)。

CICS 定義を更新するには、CICS 定義リソース・テーブル・レコードが含まれた結果セットを RESULT オプションで指定します。次に、MODIFY オプションを使用して、それらの定義に加える変更を指定します。MODIFY オプションでは変更式を指定できます (35 ページの『リリース属性の変更』を参照)。

#### **CHANGEAGENT、CHANGEAGREL、CHANGETIME、CHANGEUSRID、および CREATETIME の各属性**

既存の CICSplex SM または CICS 定義を使用して作業する場合、それぞれのレコードの最初の 8 バイトには CHANGETIME という属性が含まれ、この属性はレコードの最終変更日時を示していることを銘記しておいてください。CICS 定義レコードには CREATETIME 属性も含まれていて、これは定義の作成日時を示します。

既存の CHANGETIME 属性と CREATETIME 属性を組み合わせた、リソース・テーブル定義レコード内の CHANGEAGENT、CHANGEAGREL、CHANGEUSRID の各属性フィールドは、リソース定義シグニチャーに由来し、BAS リソース定義に関してのみ有効です。

CHANGEAGENT は、リソースの定義、または最終変更について表示します。CHANGEAGREL には、リソース定義を作成または最終変更した CICS システムのレベルが含まれます。CHANGEUSRID には、リソース定義を作成または最終変更したユーザー ID が含まれます。

CHANGEAGENT、CHANGEAGREL、CHANGETIME、CHANGEUSRID、および CREATETIME の各属性は CICSplex SM によって内部的に維持されます。これらの属性値は変更しないでください。定義リソース・テーブル・レコードを更新または除去する場合、CICSplex SM に戻す CHANGETIME 値と CREATETIME 値は受信した値と同じでなければなりません。

#### **PARM オプションの使用**

ほとんどの CICSplex SM 定義と CICS 定義については、API 要求を処理するために必要なすべての情報がリソース・テーブルの属性に含まれています。

ただし、一部の定義では、オプションのデータといくつかの必須追加データを指定できます。これらの定義については、以下の適切な API コマンドで PARM オプションを指定する必要があります。

- CREATE
- UPDATE
- REMOVE
- GET

PARM オプションにはパラメーター式を指定できます。パラメーター式は、処理する定義に必要なパラメーターを定義する文字ストリングです。

例えば、LNKSMSCG 定義を作成するとします。この定義は、CICS システム・グループとモニター仕様 (MONSPEC) の間の関連付けを記述する CICSplex SM 定義です。CICSplex SM で要求を処理できるようにするには、その変更の影響を受ける可能性のある他のリンクを処理する方法を認識させる必要があります。このため、CREATE コマンドを発行するときには、PARM オプションで、次のようなパラメーター式を指定する必要があります。

```
PARM('FORCE.')
```

これにより、CICS システム・グループ内のすべての CICS システムが新しい仕様を継承するように CICSplex SM に対して指示されます。

PARM オプションが特に便利なのは、CICS 定義を操作する場合です。各 CICS 定義リソース・テーブルについて、リソース・グループ (RESGROUP) に対する定義の関連付けを記述する別のリソース・テーブルが存在します (この関連付けが存在する場合)。例えば、CONNDEF リソース・テーブルは接続定義を表し、CONINGRP リソース・テーブルは接続定義とリソース・グループの間の関連付けを表します。CICS 定義向けの CREATE コマンドと GET コマンドに用意されている RESGROUP パラメーターを使用すると、これらのレコードの処理が簡素化されます。



CICS 定義レコードを作成する際は、定義を追加する既存のリソース・グループを指定できます。そのためには、次のように PARM オプションを使用してリソース・グループを指定します。

```
PARM('RESGROUP(resgroup).')
```

RESGROUP パラメーターを使用すると、CICS 定義とそのリソース・グループの間の関連付けを記述する xxxINGRP レコード (CONINGRP レコードなど) が自動的に作成されます。

GET コマンドを使用してデータ・リポジトリ内の CICS 定義レコードを要求する際は、定義が属するリソース・グループに従ってそれらの定義を選択できます。PARM オプションを使用してリソース・グループを指定します。

```
PARM('RESGROUP(resgroup).')
```

CICSplex SM は、指定されたリソース・グループのみから CICS 定義を選択します。PARM オプションを使用しない場合は、CICSplex SM によって、GET コマンドで指定された、別の基準に従ってすべてのリソース・グループから定義が選択されます。

注：特定のリソース・テーブルで必要な (またはサポートされている) CREATE、UPDATE、REMOVE、および GET のパラメーターの完全なリストについては、[CICSplex SM リソース・テーブル](#)を参照してください。

### CSD リソースについての特別な考慮事項

CICSplex SM でサポートされる CSD リソース定義の管理には、いくつかの制限があります。

互換モードは CICSplex SM API ではサポートされません。

CSD 要求の場合、個別の CICS システムとしてスコープを指定する必要があります。論理スコープ、CICS システム・グループ、および CICSplex 名は使用できません。個別の CICS システムのスコープを指定するため、複数のシステムに対してコマンドを同時に発行することはできません。

>

CSD では無効な BAS 固有の属性を **CREATE** または **UPDATE** 要求で指定した場合、それらは無視されます。

**CSDLOCK** および **CSDUNLOCK** アクションで生成される要求は常に同じ値を使用して認証を受けるので、CICSplex SM API でのそれらの機能は制限されます。これらのアクションで使用される値は、PLTPIUSR の OPIDENT、および SCOPE で指定した CICS システムの APPLID です。これらのアクションを使用して不意な更新を防止することができますが、セキュリティの目的で使用することは推奨されていません。

以下の状況では、EXEC CPSM COPY コマンドを使用できません。

- リソース・テーブル・レコードを CSD 結果セットから非 CSD 結果セットにコピーする場合。
- リソース・テーブル・レコードを非 CSD 結果セットから CSD 結果セットにコピーする場合。
- リソース・テーブル・レコードを CSD 結果セットから別のスコープ内の CSD 結果セットにコピーする場合。

### ファイル定義をインストールする場合の CICSplex SM API の使用例

CICSplex SM API では、CICS リソース定義を CSD からインストールしたり、BAS を使用してインストールしたりできます。

シンプルな例にするために、タスクに関連する属性のみを含めています。例えば、CRITERIA の THREAD 属性および RESULT 属性、PARM の LENGTH 属性および PARMLEN 値は省略しています。

### BAS を使用したファイルのインストール

この例では、単一の CICS ファイル定義 filedef\_name を、CICSplex cicsplex\_name 内の CICS システム cics\_system\_name にインストールする方法を示しています。

```
CONNECT CONTEXT(cicsplex_name) SCOPE(cicsplex_name) 1
GET OBJECT(FILEDEF)
  CRITERIA(NAME=filedef_name AND DEFVER=def_ver.) 2
```

```
PERFORM SET ACTION(INSTALL)
  PARM(TARGET(cics_system_name) USAGE(LOCAL).) 3
```

## BAS を使用して調整したファイルのインストール

この例では、単一の CICS ファイル定義 `filedef_name` のインストールされる属性 (名前など) を調整する方法を示しています。これを使用して、テンプレート定義を活用したり、領域に応じて属性を変えたりできます。この例で、CICS ファイル定義 `filedef_name` は、CICSplex `cicsplex_name` 内の CICS システム `cics_system_name` に、CICS ファイル `cics_file_name` およびステータス `cics_file_status` としてインストールされます。

```
CONNECT CONTEXT(cicsplex_name) SCOPE(cicsplex_name) 1
GET OBJECT(FILEDEF)
  CRITERIA(NAME=filedef_name AND DEFVER=def_ver.) 2
PERFORM SET ACTION(INSTALL)
  PARM(TARGET(cics_system_name) USAGE(LOCAL) 3
    OVERRIDE(NAME=cics_file_name,STATUS=cics_file_status)
    OVERTYPE)TARGET.) 5
```

**3** TARGET パラメーターは、BAS がリソースをインストールする 1 つまたは複数の CICS システムを指定します。

**5** OVERRIDE パラメーターは、インストール時にリソースで使用する `attribute_name=attribute_new_value` のペアを指定します。OVERTYPE パラメーターは、OVERRIDE の適用場所を示します。

## CSD からのファイルのインストール

この例では、単一の CICS ファイル定義 `filedef_name` を、CICS システム `cics_system_name` の CSD グループ `csd_group_name` 内にインストールする方法を示しています。

```
CONNECT CONTEXT(cicsplex_name) SCOPE(cicsplex_name) 1
GET OBJECT(FILEDEF)
  SCOPE(cics_system_name) 1
  PARM(CSDGROUP(csd_group_name).) 4
  CRITERIA(NAME=filedef_name.) 2
PERFORM SET ACTION(CSDINSTALL)
```

**1** SCOPE 値は BAS インストールに使用されません。CSD インストールの場合、アクティブな SCOPE は、リソースを取り出す CSD を所有していて、定義がインストールされる CICS システムの名前である必要があります。

**2** DEFVER 属性は、リソースの定義バージョンを指定します。これは同じ名前の複数のリソースがある場合に役に立ちます。BAS を使用したリソースのインストール時に、同じ名前の複数のリソースが存在する場合は、リソース名と定義バージョンの両方を指定する必要があります。CSD に定義されたリソースには DEFVER 属性を使用しないでください。

**3** TARGET パラメーターは、BAS がリソースをインストールする 1 つまたは複数の CICS システムを指定します。

**4** CSDGROUP パラメーターは、リソース定義オブジェクトを SCOPE 内の CICS システムに関連付けられた CSD から取り出すことを示します。

## CICS 接続定義をインストールする場合の CICSplex SM API の使用例

CICSplex SM API では、CICS 接続定義を CSD からインストールしたり、BAS を使用して CICSplex SM データ・リポジトリからインストールしたりできます。

CICS 接続定義のインストールは、関連付けられた 1 つ以上のセッション定義と共にインストールする必要があるという点で、他の CICS リソースのインストールとは異なります。BAS では、CICS 接続定義をインストールするには、リソース割り当て (RASGNDEF) を使用します。CSD では、CICS 接続定義をインストールするには、グループから接続とセッションをインストールします。



## BAS を使用した接続定義のインストール

この例では、CICS 接続定義 `conndef_name` を、CICSplex `cicsplex_name` 内の CICS システム `cics_system_name` にインストールする方法を示しています。

```
CONNECT CONTEXT(cicsplex_name) SCOPE(cicsplex_name)

GET OBJECT(CONNDEF)
  CRITERIA(NAME=conndef_name AND DEFVER=def_ver.))

PERFORM SET ACTION(INSTALL)
  PARM(TARGET(cics_system_name)
        USAGE(LOCAL)
        REFASSIGN(rasgndef_name).) 1
```

## CSD からの接続定義のインストール

この例では、CSD グループ `csd_group_name` に定義された CICS 接続定義 `condef_name` を、CICS システム `cics_system_name` にインストールする方法を示しています。

```
CONNECT CONTEXT(cicsplex_name) SCOPE(cicsplex_name)

GET OBJECT(CSDGROUP)
  SCOPE(cics_system_name)
  CRITERIA(NAME=csd_group_name.) 2

PERFORM SET ACTION(CSDINSTALL) 3
```

**1** CONNDEF インストールには REFASSIGN パラメーターが必要です。その値は、接続定義と共にインストールする、リソース・グループ (RESGROUP) 内のセッション定義を 1 つ以上示すリソース割り当て (RASGNDEF) です。

**2** CSD から接続定義をインストールするには、CSDGROUP、CSDINGRP、CSDINLST、CSDLIST のいずれかを使用する必要があります。また、インストールされる結果セットには、接続とセッションのペアが 1 つ以上含まれている必要があります。

**3** OBJECT は CSD 限定のリソースであるので、CSDINSTALL アクションにはパラメーターは必要ありません。

## リモート CICS トランザクション定義をインストールする場合の CICSplex SM API の使用例

CICSplex SM API では、リモート CICS トランザクション定義を CSD からインストールしたり、BAS を使用してインストールしたりできます。

機能シップをサポートするリソースの場合、BAS は、リソースのローカル定義とリモート定義を同時にインストールできます。CSD からインストールする場合は、別個のローカル定義とリモート定義を別々にインストールする必要があります。

## BAS を使用したリモート CICS トランザクション定義のインストール

この例では、ローカルの CICS トランザクション定義 `trandef_name` をルーティング CICS システム `cics_system_local` に、対応する同じ名前のリモート定義をターゲット CICS システム `cics_system_remote` にインストールする方法を示しています。どちらも、CICSplex `cicsplex_name` に存在します。

```
CONNECT CONTEXT(cicsplex_name) SCOPE(cicsplex_name)

GET OBJECT(TRANDEF)
  CRITERIA(NAME=randef_name AND DEFVER=def_ver.))

PERFORM SET ACTION(INSTALL)
  PARM(TARGET(cics_system_remote)
        USAGE(REMOTE)
        MODE(DYNAM)
        RELATED(cics_system_local).)
```

## CSD からのリモート CICS トランザクション定義のインストール

この例では、ローカルの CICS トランザクション定義 `trandef_name` を、ローカルの CICS システム `cics_system_local` の CSD グループ `csd_group_local` にインストールする方法を示しています。そして、対応するリモート定義を、ターゲット CICS システム `cics_system_remote` の CSD グループ `csd_group_remote` に別途インストールしています。

```
CONNECT CONTEXT(cicsplex_name) SCOPE(cicsplex_name)

GET OBJECT(TRANDEF)
  SCOPE(cics_system_local)
  PARM(CSDGROUP(csd_group_local).)
  CRITERIA(NAME=trandef_name.)

PERFORM SET ACTION(CSDINSTALL)

GET OBJECT(TRANDEF)
  SCOPE(cics_system_remote)
  PARM(CSDGROUP(csd_group_remote).)
  CRITERIA(NAME=trandef_name.)

PERFORM SET ACTION(CSDINSTALL)
```

## ATOM サービス定義を作成する場合の CICSplex SM API の使用例

CICSplex SM API を使用して、CICS CSD および CICSplex SM BAS の両方に CICS ATOM サービス定義を作成できます。

### BAS を使用した ATOM サービス定義の作成

この例では、BAS を使用して CICS ATOM サービス定義 `atomdef_name` を作成する方法を示しています。

```
CONNECT CONTEXT(cicsplex_name) 1

CRERESG_RESGROUP      = resgroup_name
CRERESG_DESCRIPTION    = "Sample BAS Resource Group"

CREATE OBJECT(RESGROUP)
  FROM(CRERESG)
  LENGTH(resgroup_tbl_len) 2

CREATOM_DEFVER        = "1"; 3
CREATOM_NAME          = atomdef_name;
CREATOM_DESCRIPTION    = "Dummy FILE ATOM Service";
CREATOM_STATUS        = "ENABLED";
CREATOM_ATOMTYPE      = "FEED";
CREATOM_RESOURCECETYPE = "FILE";
CREATOM_RESOURCECENAME = atomdef_file_name;
CREATOM_BINDFILE      = atomdef_bindfile_name;
CREATOM_CONFFILE      = atomdef_configfile_name;

CREATE OBJECT(ATOMDEF)
  FROM(CREATOM)
  LENGTH(atomdef_tbl_len)
  PARM(RESGROUP(resgroup_name).) 4
```

### CSD からの ATOM サービス定義の作成

この例では、CICS ATOM サービス定義 `atomdef_name` を、CICS システム `cics_system_name` の CSD グループ `csd_group_name` 内に作成する方法を示しています。

```
CONNECT CONTEXT(cicsplex_name)

CREATOM_CSDGROUP      = csd_group_name; 5
CREATOM_NAME          = atomdef_name;
CREATOM_DESCRIPTION    = "Dummy FILE ATOM Service";
CREATOM_STATUS        = "ENABLED";
CREATOM_ATOMTYPE      = "FEED";
CREATOM_RESOURCECETYPE = "FILE";
CREATOM_RESOURCECENAME = atomdef_file_name;
CREATOM_BINDFILE      = atomdef_bindfile_name;
CREATOM_CONFFILE      = atomdef_configfile_name;
```

```
CREATE OBJECT(ATOMDEF)
  SCOPE(cics_system_name)
  FROM(CREATOM)
  LENGTH(atomdef_tbl_len)
  PARM(CSD.)
```

6

7

**1** BAS リソース定義は、**CONTEXT** パラメーターで指定した CICSplex に保管されます。 **SCOPE** パラメーターを BAS リソース定義に指定しないでください。

**2** **RESGROUP** パラメーターを **CREATE** コマンドに指定する場合、その RESGROUP は CICSplex に既に定義されている必要があります。

**3** BAS リソースを定義する場合、**DEFVER** パラメーターの値を指定する必要があります。 **DEFVER** パラメーターおよび **CSDGROUP** パラメーターの両方を指定した場合、**CSDGROUP** パラメーターは無視されます。

**4** BAS CICS リソース定義のリソース・グループへの追加はオプションです。 **RESGROUP** パラメーターを指定した場合、CICS リソース定義が BAS リソース・グループに関連付けられます。アクションを **ACTION=ADDTOTGRP** に設定した **PERFORM** コマンドを使用して、リソース定義を 1 つ以上のリソース・グループに明示的に追加することもできます。

**5** **CSDGROUP** パラメーターは、リソース定義を作成する CSD グループを指定します。すべての CSD リソース定義が 1 つのグループに属する必要があります。CSD 内にグループがまだ存在していなければ、グループが動的に作成されます。

**6** **SCOPE** パラメーターは、定義を作成する CSD を使用している CICS システムの名前を指定します。

**7** **PARM** パラメーターに CSD 属性を指定して、リソース定義が CSD リソース定義であることを示します。

### リストに CSD グループを追加する場合の CICSplex SM API の使用例

CICSplex SM API を使用して、リストに CSD グループを追加できます。

#### リストへの CSD グループの追加

```
CONNECT CONTEXT(cicsplex_name)
  SCOPE(cics_system_name)

GET OBJECT(CSDGROUP)
  CRITERIA(NAME=csd_group_new.)

PERFORM SET ACTION(CSDADD)
  PARM(TO_LIST(csd_list_name)
    ADD_CSDGROUP(csd_group_old)
    ADD_LOCATION(AFTER))
```

1

2

**1** **SCOPE** パラメーターの値は、グループを追加する CSD (グループを追加するリストが定義されている CSD) を使用している CICS システムの名前でなければなりません。 **PERFORM SET** コマンドのために、**SCOPE** パラメーターをスレッドのアクティブ・スコープとして指定する必要があります。 **PERFORM** コマンドのスコープは **GET** コマンドと同じでなければならず、コマンドで明示的に指定することができないからです。 **SCOPE** パラメーターは、**CONNECT** または **QUALIFY** コマンドによって指定されます。

**2** この例では、**ADD\_LOCATION(AFTER)** の指定により、グループ **csd\_group\_new** がリスト **csd\_list\_name** 内の **csd\_group\_old** の後に追加されます。

### グループから CSD リソースを削除する場合の CICSplex SM API の使用例

CICSplex SM API を使用して、グループからリソースを削除できます。

#### グループからの CSD リソースの削除

この例では、CICS システム **cics\_system\_name** で使用される CSD の CSD グループ **csd\_group\_name** に定義されている CICS トランザクション定義 **trandef\_name** を削除する方法を示します。

```
CONNECT CONTEXT(cicsplex_name)
  SCOPE(cics_system_name)

GET OBJECT(TRANDEF)
  CRITERIA(NAME=trandef_name.)
```

1

```

    PARM(CSDGROUP(csd_group_name).)
    RESULT(result_set_token)

    2

    FETCH INTO(trandef_record_buffer)
          LENGTH(trandef_record_length)
          RESULT(result_set_token)

    REMOVE OBJECT(TRANDEF)
           FROM(trandef_record_buffer)
           LENGTH(trandef_record_length)
           PARM(CSD.)
    3

```

**1** GET および REMOVE の両方のコマンドの **SCOPE** パラメーターの値は、CSD を使用している CICS システムの名前でなければなりません。

**2** トランザクション定義 TRANDEF の **CSDGROUP** パラメーターは、リソースが CSD に定義されていることを示し、そのリソースが定義されている CSD グループを指定します。

**3** **CSD** パラメーターは、**FETCH** コマンドで取り出した CICS トランザクション定義レコードを CSD リソースとして示しています。 トランザクション定義レコードには、トランザクション定義の名前、それが現在定義されている (そしてそれが削除される) CSD グループの両方が含まれています。

### リストから CSD グループを削除する場合の CICSplex SM API の使用例

CICSplex SM API を使用して、CSD グループを CSD リストから削除できます。

#### リストからの CSD グループの除去

この例では、CICS システム `cics_system_name` によって使用される CSD で、CSD グループ `csd_group_name` を CSD リスト `csd_list_name` から削除する方法を示します。

```

CONNECT CONTEXT(cicsplex_name)
        SCOPE(cics_system_name)
    1

GET  OBJECT(CSDINLST)
     CRITERIA(CSDLIST=csd_list_name
             AND CSDGROUP=csd_group_name.)
     RESULT(result_set_token)

FETCH INTO(csdinlst_record_buffer)
        LENGTH(csdinlst_record_length)
        RESULT(result_set_token)

REMOVE OBJECT(CSDINLST)
        FROM(csdinlst_record_buffer)
        LENGTH(csdinlst_record_length)

```

**1** GET および REMOVE の両方のコマンドの **SCOPE** パラメーターの値は、CSD を使用している CICS システムの名前でなければなりません。

### CSD グループを削除する場合の CICSplex SM API の使用例

CICSplex SM API を使用して、CSD グループを CSD から削除できます。

#### CSD グループの削除

この例では、CICS システム `cics_system_name` によって使用される CSD で、CSD グループ `csd_group_name`、このグループ内のすべての CICS リソース定義、および CSD リストに含まれているこのグループに対するすべての参照を削除する方法を示します。

```

CONNECT CONTEXT(cicsplex_name)
        SCOPE(cics_system_name)
    1

GET  OBJECT(CSDGROUP)
     CRITERIA(NAME=csd_group_name.)
     RESULT(result_set_token)

FETCH INTO(csdgroup_record_buffer)
        LENGTH(csdgroup_record_length)
        RESULT(result_set_token)

```

```
REMOVE OBJECT(CSDGROUP)
FROM(csdgroup_record_buffer)
LENGTH(csdgroup_record_length)
PARM(LISTREMOVE.)
```

**1** GET および REMOVE の両方のコマンドの **SCOPE** パラメーターの値は、CSD を使用している CICS システムの名前でなければなりません。

**2 LISTREMOVE** パラメーターは、CSD グループを CSD 内のすべてのリストから削除します。このパラメーターを指定しない場合、CSD グループは削除されますが、グループに対する参照は、グループが属していた CSD リスト内に残されます。

## 非同期処理

ほとんどの CICSplex SM API コマンドは、通常、同期的に機能します。つまり、プログラムは要求を発行した後、コマンドの処理が完了するまで待機します。

CANCEL コマンドは未処理の LISTEN 要求を取り消します。他のコマンドは同期的または非同期的に使用できます。これらのいずれかのコマンドに NOWAIT オプションを指定した場合、要求は非同期的に処理されます。

非同期処理の結果をモニターおよび受信するために使用できる API コマンドは以下のとおりです。

- ADDRESS
- RECEIVE

45 ページの図 12 は、これらのコマンドと API 環境の関係を示しています。

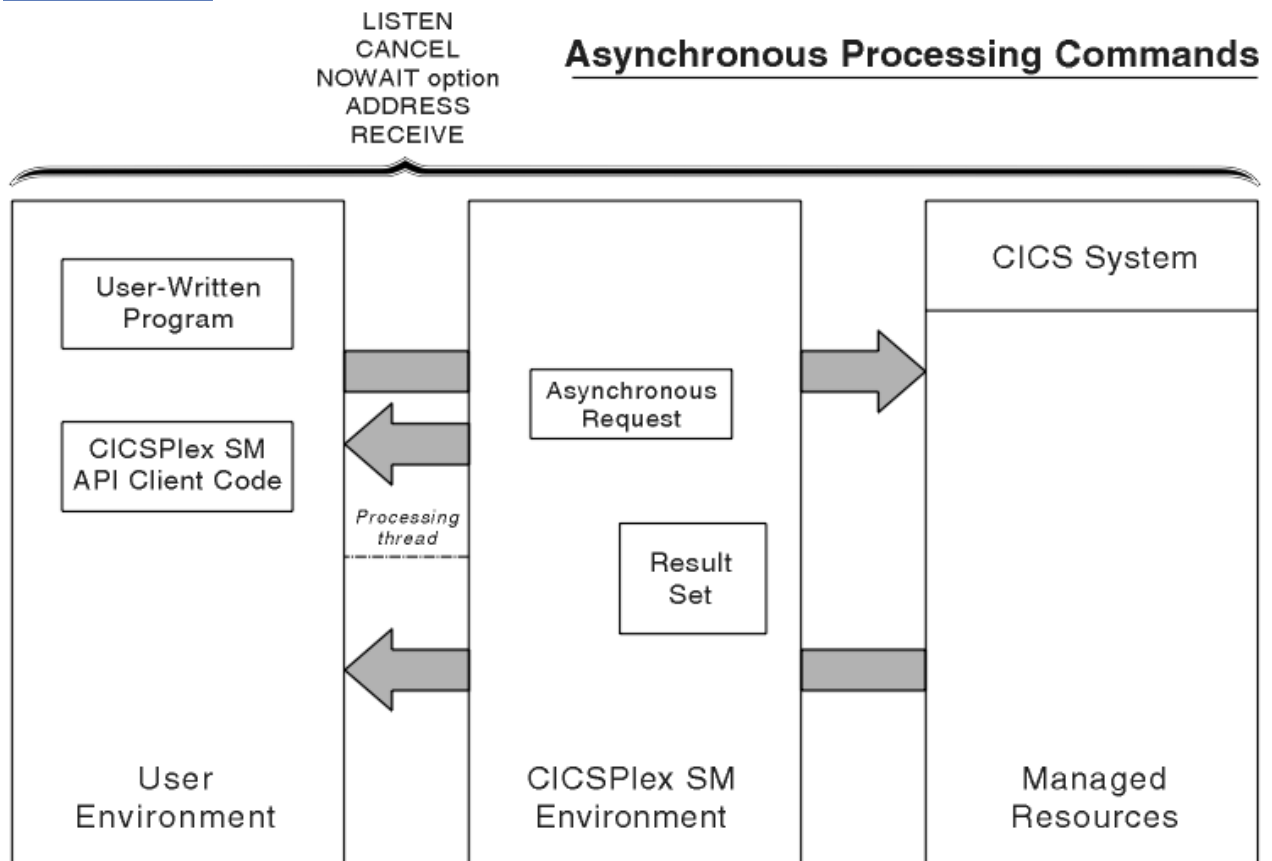


図 12. 非同期処理の API コマンド

## LISTEN コマンドの使用

CICSplex SM によって管理されるリソースの多くは、CICSplex にとって重要と見なされるイベントが発生したときにシステムに通知できます。

このようなイベントはスケジュールされず、予期できないため、これらの通知を処理するように設計されたプログラムは非同期的に処理する必要があります。LISTEN コマンドを使用すると、関心のあるイベント通知を識別できます。

listen できるイベントは、通知タイプのリソース・テーブルによって表されます。例えば、CICS システムの正常性に悪影響を与える条件が発生すると、EMASSICK 通知が MAS によって作成されます。通知リソース・テーブルのリストと他のリソース・テーブルの詳細な説明は、[CICSplex SM リソース・テーブル](#)を参照してください。

LISTEN コマンドを発行すると、その結果として生成された通知が API 処理スレッド用の未処理データ・キューに追加されます。ADDRESS コマンドの SENTINEL オプションを指定すると、完了済みの非同期要求の数が報告されます (NOWAIT オプションを使用して発行された要求およびイベント通知を含む)。これらのイベント通知を取得するには、RECEIVE コマンドを発行します。

## NOWAIT オプションの使用

GET、PERFORM OBJECT、PERFORM SET、REFRESH、または SET のいずれかのコマンドで NOWAIT オプションを指定した場合、要求の処理は即時に完了しません。代わりに、要求は処理対象としてスケジュールされて、コマンドは RESPONSE の値として SCHEDULED を戻して、制御はプログラムに戻されます。

非同期要求の実行中に、プログラムは、別の CICSplex SM API コマンドの発行を含む他の処理を実行できます。ただし、コマンドがアクティブである間は、処理対象としてそのコマンドに割り当てられた結果セットを使用することはできません。非同期要求によってまだ処理中の結果セットにアクセスしようとした場合は、RESPONSE の値として INUSE が戻されます。

非同期要求が完了すると、ASYNCREQ リソース・テーブル・レコードが作成されます。ADDRESS コマンドの SENTINEL オプションを指定すると、完了済みの非同期要求の数が報告されます (NOWAIT オプションを使用して発行された要求を表す ASYNCREQ レコードを含む)。ASYNCREQ レコードを取得するには、RECEIVE コマンドを発行します。

ASYNCREQ リソース・テーブルには、通常はコマンド自体によって戻される情報の多くが含まれています。コマンドの処理が完了する前に制御がプログラムに戻されるため、その情報をコマンドでを使用することはできません。ASYNCREQ リソース・テーブル内に戻される情報には、以下のようなものがあります。

- 発行されたコマンド。
- 関連付けられた結果セット・トークン。
- コマンドによって戻された RESPONSE と REASON の値。
- 通常は FEEDBACK リソース・テーブル・レコード内に戻される診断データ (RESPONSE の値が OK でない場合)。
- 非同期要求を識別するユーザー定義トークン (指定された場合)。

注: REXX プログラムから ASYNCREQ データにアクセスするには、ASIS オプションを指定した CICSplex SM TPARSE コマンドを使用するか、REXX SUBSTR 関数を使用します。

## トークンを使用した要求の識別

プログラムによって発行される非同期要求を追跡管理するために、各要求に固有の識別トークンを割り当てることができます。

これにより、プログラムは、LISTEN 要求および NOWAIT オプションを使用して発行された要求を後続の RECEIVE コマンドの結果と関連付けることができます。CICSplex SM API では、ユーザー定義のトークンは使用されません。ユーザー・トークンの値は、関連する要求が完了してから RECEIVE コマンドによってプログラムに戻されるまで保持されます。任意の 1 文字から 4 文字の値を識別トークンとして使用できます。例えば、以下のように指定できます。

- リテラル定数
- サービス・ルーチンのオフセット

- データ構造のアドレス

## ADDRESS コマンドの使用

CONNECT コマンドを発行して、API 処理スレッドが確立されると、プログラムが実行されている MVS アドレス・スペースまたは CICS システム内に 2 つの制御フィールドが作成されます。

これらのスレッド制御フィールドのアドレスを要求することで、処理を中断したりポーリングしたりすることなく、非同期出力が得られるかどうかを確認できます。

ADDRESS ECB() SENTINEL() コマンドを使用して、以下のフィールドのアドレスを要求できます。

### ECB

非同期要求が完了するたびに、ECB が API によって送信されて、スレッドの未処理データ・キューに追加されます。ECB アドレスを使用して、以下のことを実行できます。

- 適切な MVS ポスト・ビットをテストして、出力が得られるかどうかを確認します。
- MVS WAIT コマンドをバッチ、TSO、または NetView のプログラムで発行します。
- EXEC CICS WAITCICS または WAIT EXTERNAL コマンドを CICS プログラムで発行します。

SENTINEL フィールドのカウンター値が 0 に達するたびに、ECB フィールドはクリアされます。

### SENTINEL

標識は、スレッドに関連付けられた完了済み非同期要求の 4 バイト・カウンタです。

非同期要求が完了するたびに、標識値は大きくなります。完了済み非同期要求の例としては、以下が挙げられます。

- LISTEN コマンドで指定されたイベントが発生する。
- NOWAIT オプションを指定して発行されたコマンドの処理が完了する。

RECEIVE コマンドが発行されると、標識値は小さくなります。

注：

1. ADDRESS コマンドは、RECEIVE コマンドを発行する前に使用する必要があります。標識値が 0 の場合は、受信される完了済み非同期要求がないということです。
2. 非同期処理の特性上、標識値はいずれかの時点で未処理要求の実数を下回ることがあります。複数の非同期要求を処理しているときは、NODATA の応答が戻されることですべての出力が受信されたことが確認されるまで、RECEIVE IMMEDIATE コマンドを発行することが必要になります。

## RECEIVE コマンドの使用

RECEIVE コマンドを使用して、発行した非同期要求のいずれかが完了済みかどうかを確認できます。

RECEIVE は、これらの要求の出力を戻します。戻された出力は以下のようになります。

- 以前に実行された LISTEN コマンドで指定されたイベントを表すリソース・テーブル・レコード
- 非同期の GET、PERFORM、REFRESH、または SET 要求の完了を表す ASYNCREQ リソース・テーブル・レコード

注：RECEIVE コマンドを発行する前に、ADDRESS コマンドを発行して SENTINEL の値を確認して、受信される未処理の非同期要求が存在するかどうかを確認する必要があります。SENTINEL の値が 0 の場合は、受信される未処理の非同期要求はありません。

例えば、同じ API スレッド上で NOWAIT オペランドを指定した GET コマンドと LISTEN コマンドがプログラムによって発行されることがあります。その場合は、RECEIVE コマンドへの応答として、その GET コマンドに対する ASYNCREQ リソース・テーブル・レコードを受信するか、listen していたイベントに関連付けられたリソース・テーブル・レコードを受信します。

代わりの方法として、複数の API スレッドを使用して後続の複数の RECEIVE コマンドによって戻される出力を分離できます。例えば、1 つのスレッドを作成し、LISTEN コマンドからイベント通知を受信するためにのみ使用することが考えられます。他の API 関数で使用するために、別のスレッドを作成することもできます。このようにして、各スレッドに対して発行された RECEIVE コマンドによって戻される出力の内容を制御できます。

複数の API スレッドを作成することが望ましいもう 1 つの理由は、各スレッドは一度に最大で 256 件の未処理非同期要求のみを保持できるためです。プログラムによって単一の API スレッド上で多数の非同期要求が発行される場合は、RECEIVE コマンドを定期的に発行してください。いずれかの処理スレッドが上限である 256 件に達した場合は、非同期要求は破棄されて処理されなくなります。

デフォルトでは、RECEIVE コマンドは非同期出力が得られるまで待機してから、プログラムに制御を戻します。したがって、非同期要求が完了するまで処理は中断されます。無制限に待機させる代わりに、以下のいずれかのオプションを RECEIVE コマンドで指定できます。

#### **DELAY(data-value)**

非同期出力の有無を確認して、出力が得られるまで指定された秒数だけ待機した後、出力の有無に関係なく、処理スレッドに制御を戻します。

#### **IMMEDIATE**

非同期出力の有無を確認した直後に、出力が得られるかどうかに関係なく、処理スレッドに制御を戻します。

## **CICSplex SM トークンの使用**

CICSplex SM API コマンドの多くは相互に関連しています。プログラムの目的を達成するために、これらを互いに組み合わせで使用します。例えば、GET コマンドを発行して結果セットを作成した後、その結果セット内のリソース・テーブル・レコードにアクセスするために、FETCH コマンドを発行します。

さまざまな操作の結果をその後に発行する要求と相関付けるために、CICSplex SM によって以下の API 環境オブジェクトに 4 バイトのトークンが割り当てられます。

- 処理スレッド
- 結果セット
- フィルター
- ビュー
- LISTEN 要求。

したがって、例えば、各処理スレッドには 4 バイトの固有識別トークンが割り当てられます。プログラムによって発行される各 API コマンドには、スレッド・トークンを指定して、それを処理するスレッドを指定する必要があります。同様に、結果セットまたはフィルターが作成された後、後続のコマンドでその結果セットまたはフィルターを参照するには、CICSplex SM によってその結果セットまたはフィルターに割り当てられたトークン値を指定します。各 LISTEN 要求にはトークンが割り当てられるため、CANCEL コマンドを使用してその要求を取り消すことができます。

注：

1. CICSplex SM によって、内部使用専用のトークンがビューに割り当てられます。外部では、ユーザーが割り当てたビュー名によってビューを参照します。
2. 各処理スレッドで使用できる CICSplex SM トークンの数には制限があります。一般に、単一の処理スレッド上で作成される結果セット、フィルター、ビュー、および LISTEN 要求の数は 255 を超えることはできません。

トークン値が個別のオブジェクトについて固有であるだけでなく、トークンの構造がオブジェクト・タイプによって異なります。このため、CICSplex SM によって、スレッド・トークンが他のいずれかのタイプのトークンとして誤認識されることはありません。無効なトークンを指定した場合は (FILTER オプションで結果セット・トークンを指定した場合など)、RESPONSE の値として INVALIDPARM を受け取ります。

## **メタデータ・リソース・テーブルの使用**

GETDEF コマンドは、CICSplex SM 管理対象オブジェクトの一般的な特性、有効なアクション、オブジェクト属性などの構造を示すレコードを取得するために使用します。

GETDEF コマンドの OBJECT オプションにより、取得するメタデータのタイプが指定されます。次のメタデータ・リソース・テーブルの内容が示されます。

- ATTR
- ATTRAVA



- METADESC
- METANAME
- METAPARM
- OBJACT
- OBJECT
- PARMAVA

## ATTR

ATTR リソース・テーブルには、管理対象オブジェクトの特定の属性に関する詳細情報が含まれています。

### 属性

#### 説明

#### OBJECT

この特定の属性が属する管理対象オブジェクトの名前。

#### TABLEVER

OBJECT 属性によって識別されるテーブルのバージョン。

#### NAME

特定の属性の名前。長さは 1 文字から 12 文字までです。

#### ID

属性の ID。

#### LENGTH

この属性に関連付けられたデータの長さ。ATTR 属性 NAME の長さと混同しないでください。

#### OFFSET

この属性データの開始位置となるリソース・テーブル内のオフセット。

#### DATATYPE

この属性データのデータ・タイプ。

#### COMPID

CICSplex SM コンポーネント ID

#### BINARY

Binary

#### RATE

小数点以下 1 位まで比率を算出する

#### PERCENT

小数点以下 1 位までパーセンテージを算出する

#### SUM

値の合計、小数点以下 1 桁まで

#### RATIO

比率

#### AVERAGE

小数点以下 1 位まで平均を算出する

#### TIMESTP

Time stamp (タイム・スタンプ)

#### BIT

ビット・ストリング

#### TEXT

テキスト

#### CHAR

文字

#### EYUDA

EYUDA

**CVDAS**

標準 CVDA

**CVDAT**

端末 CVDA

**RESTYPE**

リソース・タイプ

**DECIMAL**

Packed Decimal (パック 10 進)

**DECDATE**

10 進数形式の日付

**ILABEL**

内部ラベル

**HHMM**

バイナリーの時間/分

**SCLOCK**

CMF 8 バイトの間隔保管クロック

**SCLOCK12**

CMF 12 バイトの間隔保管クロック

**INTUSEC**

マイクロ秒単位の間隔

**INTMSEC**

ミリ秒単位の間隔

**INT16US**

16 マイクロ秒単位の間隔

**INTSEC**

秒単位の間隔

**INTTSTP**

間隔タイム・スタンプ差分

**DATETIME**

日時グループ

**DECTSTP**

10 進数のタイム・スタンプ

**ADDRESS**

Address (アドレス)

**CNUMERIC**

コード化された数値

**HIDCHAR**

非表示文字

**HEX**

Hexadecimal (16 進)

**TBLVER**

リソース・テーブルのバージョン

**RATE3**

小数点以下 3 位まで比率を算出する

**PERCENT3**

小数点以下 3 位までパーセンテージを算出する

**SUM3**

値の合計、小数点以下 3 桁まで

**AVERAGE3**

小数点以下 3 位まで平均を算出する

**DECTIME**

10 分の 1 秒単位の時間

**DECTIMES**

秒単位の時間

**SUMOPT**

この属性に使用されるデフォルトの集計オプション

**AVG**

平均

**DIFF**

差異

**MIN**

最小

**MAX**

最大

**SUM**

要約

**LIKE**

共通

**IDATATYPE**

内部データ・タイプを表す数値

**0**

コンポーネント

**4**

数値

**8**

率

**12**

パーセント

**16**

合計

**20**

比率

**24**

平均

**28**

Timestamp

**32**

ビット

**36**

テキスト

**40**

文字

**44**

EYUDA

**48**

CVDA 標準

**52**

CVDA 端末

<b>56</b>	リソース・タイプ
<b>60</b>	パック 10 進数
<b>64</b>	パック 10 進数日付
<b>68</b>	内部ラベル・フィールド
<b>72</b>	HHMM
<b>76</b>	間隔保管クロック、8 バイト
<b>80</b>	間隔のマイクロ秒数
<b>84</b>	間隔のミリ秒数
<b>88</b>	間隔の 16 マイクロ秒数
<b>92</b>	間隔の秒数
<b>96</b>	保管クロック差分
<b>100</b>	日時グループ
<b>104</b>	パック 10 進数のタイム・スタンプ、10 分の 1 秒単位まで
<b>108</b>	Address (アドレス)
<b>112</b>	コード数値
<b>116</b>	非表示文字
<b>120</b>	Hexadecimal (16 進)
<b>124</b>	テーブル・バージョン
<b>128</b>	バイナリーの導出された率、小数点以下 3 桁まで
<b>132</b>	バイナリーの導出されたパーセント、小数点以下 3 桁まで
<b>136</b>	バイナリーの導出された合計、小数点以下 3 桁まで
<b>140</b>	バイナリーの導出された平均、小数点以下 3 桁まで
<b>144</b>	パック 10 進数の時間、秒単位まで
<b>148</b>	パック 10 進数の時間、10 分の 1 秒単位まで
<b>152</b>	間隔保管クロック、12 バイト

**SETVALID**

この属性を設定/変更できるかどうか: Y または N

**REQUIRED**

この属性が CREATE に必要かどうか: Y または N

**AVAAVAIL**

この属性について属性値アサーション情報が得られるかどうか: Y または N。得られる場合は、ATTR 属性を参照してください。

- AVACOUNT

ATTRAVA リソース・テーブルを使用して属性値アサーション情報を取得します。

**CICSVALAVAIL**

CICS 妥当性データが得られるかどうか: Y または N。得られる場合は、ATTR 属性を参照してください。

- VALCICSESA
- VALCICSVSE
- VALCICSOS2
- VALCICSWNT
- VALCICES2

**HDRTXTAVAIL**

属性ヘッダー・テキストが得られるかどうか: Y または N。得られる場合は、ATTR 属性を参照してください。

- HDRTEXT

**VALSETAVAIL**

値設定情報が得られるかどうか: Y または N。得られる場合は、ATTR 属性を参照してください。

- VALCOUNT

ATTRAVA リソース・テーブルを使用して値設定情報を取得します。

**SOURCE**

この属性データのソース。

**V**

CICSplex SM によって作成された

**I**

CICS INQ から取得された

**S**

CICS STATS から取得された

**P**

CICS CMF データから取得された

**KEY**

この属性が管理対象オブジェクトのキーに参加しているかどうか: 0 または n。0 の場合、属性はキーの一部ではなく、n はキーのパーツ・ナンバーを示します。

**AVACOUNT**

この属性の属性値アサーションの数。この値は、属性の AVA という LISTTYPE 値を使用して得られる ATTRAVA リソース・テーブル・レコードの数に対応しています。AVAAVAIL 属性が Y の場合にのみ存在します。

**VALCOUNT**

この属性の値設定値の数。この値は、属性の VALUE という LISTTYPE 値を使用して得られる ATTRAVA リソース・テーブル・レコードの数に対応しています。VALSETAVAIL 属性が Y の場合にのみ存在します。

## VALCICSESA

この属性が、異なるバージョンの CICS で有効かどうかを示すフラグの第 1 バイト。

1... ..	X'80'	CICS/MVS 2.1.2
.1... ..	X'40'	CICS/ESA 3.3.0
..1... ..	X'20'	CICS/ESA 4.1.0
...1 ...	X'10'	CICS Transaction Server for OS/390 Release 1
.... 1...	X'08'	CICS Transaction Server for OS/390 Release 2
.... .1..	X'04'	CICS Transaction Server for OS/390 Release 3
.... ..1.	X'02'	CICS Transaction Server for z/OS Version 2 Release 1
.... ...1	X'01'	CICS Transaction Server for z/OS Version 2 Release 2

このビットがオンに設定されている場合、この属性はそのバージョンの CICS では有効ではありません。  
VALCICSES2 にはフラグの第 2 バイトが格納されています。

## VALCICSVSE

この属性が、異なるバージョンの CICS/VSE で有効かどうかを示すフラグ。

1... ..	X'80'	CICS/VSE 2.2.0
.1... ..	X'40'	CICS/VSE 2.3.0
..1... ..	X'20'	CICS/VSE 4.1.0
...1 1111		Reserved

このビットがオンに設定されている場合、この属性はそのバージョンの CICS では有効ではありません。

## VALCICSOS2

この属性が、異なるバージョンの CICS OS/2 で有効かどうかを示すフラグ。

1... ..	X'80'	CICS OS/2 2.0.1
.1... ..	X'40'	CICS OS/2 3.0.0
..1... ..	X'20'	CICS OS/2 3.1.0
...1 1111		Reserved

このビットがオンに設定されている場合、この属性はそのバージョンの CICS では有効ではありません。

## VALCICSWNT

この属性が、異なるバージョンの TXSeries® で有効かどうかを示すフラグ。

1... ..	X'80'	CICS for TXSeries 4.3.0
.1... ..	X'40'	CICS for TXSeries 5.0.0
..11 1111		Reserved

このビットがオンに設定されている場合、この属性はそのバージョンの CICS では有効ではありません。

## VALCICSES2

このアクションが、異なるバージョンの CICS Transaction Server for z/OS で有効かどうかを示すフラグの第 2 バイト。

.1... ..	X'40'	CICS Transaction Server for z/OS, Version 3 Release 1
..1... ..	X'20'	CICS Transaction Server for z/OS, Version 3 Release 2
...1 ...	X'10'	CICS Transaction Server for z/OS, Version 4 Release 1
.... 1...	X'08'	CICS Transaction Server for z/OS, Version 4 Release 2
.... .1..	X'04'	CICS Transaction Server for z/OS, Version 5 Release 1
.... ..1.	X'02'	CICS Transaction Server for z/OS, Version 5 Release 2
.... ...1	X'01'	CICS Transaction Server for z/OS, Version 5 Release 3

このビットがオンに設定されている場合、このアクションはそのバージョンの CICS では有効ではありません。フラグの第 1 バイトは VALCICSESA に格納されています。

## VALCICSES3

このアクションが、異なるバージョンの CICS Transaction Server for z/OS で有効かどうかを示すフラグの第 3 バイト。

1... ..	X'80'	CICS Transaction Server for z/OS, Version 5 Release 4
---------	-------	---

## SETCICSESA

この属性が、異なるバージョンの CICS で変更可能かどうかを示すフラグの第 1 バイト。

1... ..	X'80'	CICS/MVS 2.1.2
.1... ..	X'40'	CICS/ESA 3.1.0
..1... ..	X'20'	CICS/ESA 4.1.0
...1 ...	X'10'	CICS Transaction Server for OS/390 Release 1
.... 1...	X'08'	CICS Transaction Server for OS/390 Release 2
.... .1..	X'04'	CICS Transaction Server for OS/390 Release 3
.... ..1.	X'02'	CICS Transaction Server for z/OS Version 2 Release 1
.... ...1	X'01'	CICS Transaction Server for z/OS Version 2 Release 2

このビットがオンに設定されている場合、この属性はそのバージョンの CICS では変更可能ではありません。フラグの第 2 バイトは SETCICSES2 に格納されています。

## SETCICSES2

この属性が、異なるバージョンの CICS Transaction Server for z/OS で変更可能かどうかを示すフラグの第 2 バイト。

.1... ..	X'40'	CICS Transaction Server for z/OS, Version 3 Release 1
..1... ..	X'20'	CICS Transaction Server for z/OS, Version 3 Release 2
...1 ...	X'10'	CICS Transaction Server for z/OS, Version 4 Release 1
.... 1...	X'08'	CICS Transaction Server for z/OS, Version 4 Release 2
.... .1..	X'04'	CICS Transaction Server for z/OS, Version 5 Release 1
.... ..1.	X'02'	CICS Transaction Server for z/OS, Version 5 Release 2
.... ...1	X'01'	CICS Transaction Server for z/OS, Version 5 Release 3

このビットがオンに設定されている場合、この属性はそのバージョンの CICS では変更可能ではありません。

## SETCICSES3

この属性が、異なるバージョンの CICS Transaction Server for z/OS で変更可能かどうかを示すフラグの第 3 バイト。

1... ..	X'80'	CICS Transaction Server for z/OS, Version 5 Release 4
---------	-------	---

## SETCICSVSE

この属性が、異なるバージョンの CICS/VSE で変更可能かどうかを示すフラグ。

1... ..	X'80'	CICS/VSE 2.2.0
.1... ..	X'40'	CICS/VSE 2.3.0
..1... ..	X'20'	CICS/VSE 4.1.0
...1 1111		Reserved

このビットがオンに設定されている場合、この属性はそのバージョンの CICS では変更可能ではありません。

## SETCICSOS2

この属性が、異なるバージョンの CICS OS/2 で変更可能かどうかを示すフラグ。

1... ..	X'80'	CICS OS/2 2.0.1
.1... ..	X'40'	CICS OS/2 3.0.0
..1... ..	X'20'	CICS OS/2 3.1.0
...1 1111		Reserved

このビットがオンに設定されている場合、この属性はそのバージョンの CICS では変更可能ではありません。

## SETCICSWNT

この属性が、異なるバージョンの TXSeries で変更可能かどうかを示すフラグ。

1... ..	X'80'	CICS for TXSeries 4.3.0
.1... ..	X'40'	CICS for TXSeries 5.0.0
..11 1111		Reserved

このビットがオンに設定されている場合、この属性はそのバージョンの CICS では変更可能ではありません。

**IGNVALUE**

この属性について、適用外または無視を示す値。

**LOWVALUE**

この属性の有効な値範囲内の最小許容値。

**HIGHVALUE**

この属性の有効な値範囲内の最大許容値。

**HDRTEXT**

属性のヘッダー・テキスト。HDRTXTAVAIL 属性の値が Y の場合にのみ存在します。

**DESC**

属性の説明。

**DEFAULT**

この属性のデフォルト値 (存在する場合)。

**UCHAR**

この属性値が大文字かどうか: Y または N。

**CICSSETAVAIL**

SET コマンドがいずれかの属性に対して有効かどうかを示します (**Y** または **N**)。Y に設定されている場合、以下の ATTR 属性により、そのコマンドが有効である各種 CICS 製品のレベルが示されます。

- SETCICSESA
- SETCICSVSE
- SETCICSOS2
- SETCICSWNT
- SETCICSES2

**SORT**

この属性が ORDER に参加しているかどうかを示します。

**Y**

この属性は ORDER に参加しています。

**N**

この属性は ORDER に参加していません。

**FILTER**

この属性が SPECIFY FILTER に参加しているかどうかを示します。

**Y**

この属性は SPECIFY FILTER に参加しています。

**N**

この属性は SPECIFY FILTER に参加していません。

**SUMMARISE**

この属性が集計に適格かどうかを示します。

**Y**

この属性を集計できます。

**N**

この属性を集計できません。

**VIEWMOD**

この属性が表示サポートに適格かどうかを示します。

**Y**

この属性は表示サポートに適格です。

**N**

この属性は表示サポートに適格ではありません。



## **INHERIT**

この属性が継承に参加しているかどうかを示します。

**Y**

この属性は継承に参加しています。

**N**

この属性は継承に参加していません。

## **ATTRAVA**

このリソース・テーブル内の情報は、ATTR リソース・テーブルの AVAAVAIL 属性または VALSETAVAIL 属性の値が Y である場合にのみ使用することができます。

ATTRAVA リソース・テーブルでは、管理対象オブジェクトの特定の属性の許容値が提供されます。特定の属性の ATTRAVA 基本テーブル・セットによって、すべての許容値のリストが提供されます。

ただし、属性は一定範囲の値をサポートしている可能性があり (0 から 999 など)、それらの範囲値の ATTRAVA 基本テーブルは存在しません。属性のデフォルト値の ATTRAVA 基本テーブルが存在しない可能性もあります。デフォルト値、および範囲値の最大値と最小値は、属性の ATTR 基本テーブルに含まれています。

### **属性**

#### **説明**

### **OBJECT**

この特定の属性が属する管理対象オブジェクトの名前。

### **TABLEVER**

OBJECT 属性によって識別されるテーブルのバージョン。

### **NAME**

特定の属性の名前。長さは 1 文字から 12 文字までです。

### **AVAAVALUE**

この属性の値。

### **LISTTYPE**

AVAAVALUE データが属性値アサーションであるか、その属性の他の許容値であるかを示します。

### **AVA**

属性値アサーションから派生した値。

### **VALUE**

現時点では、「無視」を意味する特別な値を戻すためにのみ使用されます。

### **IOTYPE**

この属性値が入力、出力、または入出力のいずれの操作に使用されるかを示します。

**I**

入力

**O**

出力

**B**

入力および出力

## **METADESC**

METADESC リソース・テーブルでは、管理対象オブジェクトの特定の属性の基本的な構造とレイアウトの情報が提供されます。

### **属性**

#### **説明**

### **NAME**

特定の属性の名前。長さは 1 文字から 12 文字までです。

**LENGTH**

この属性に関連付けられたデータの長さ。METADESC 属性 NAME の長さと混同しないでください。

**OFFSET**

この属性データの開始位置となるリソース・テーブル内のオフセット。

**DATATYPE**

この属性データのデータ・タイプ。

<b>0</b>	コンポーネント ID
<b>4</b>	2 進数値
<b>8</b>	バイナリーの導出された率
<b>12</b>	バイナリーの導出されたパーセント
<b>16</b>	バイナリーの導出された合計
<b>20</b>	バイナリーの導出された比率
<b>24</b>	バイナリーの導出された平均
<b>28</b>	オペレーティング・システムのタイム・スタンプ
<b>32</b>	ビット
<b>36</b>	テキスト
<b>40</b>	文字
<b>44</b>	EYUDA
<b>48</b>	CVDA 標準
<b>52</b>	CVDA 端末
<b>56</b>	リソース・タイプ (Resource Type)
<b>60</b>	Packed Decimal (パック 10 進)
<b>64</b>	パック 10 進数日付
<b>68</b>	内部ラベル・フィールド
<b>72</b>	バイナリーの HHMM
<b>76</b>	間隔保管クロック、8 バイト
<b>80</b>	間隔のマイクロ秒数

<b>84</b>	間隔のミリ秒数
<b>88</b>	間隔の 16 マイクロ秒数
<b>92</b>	間隔の秒数
<b>96</b>	間隔保管クロック差分
<b>100</b>	日時グループ
<b>104</b>	パック 10 進数のタイム・スタンプ、10 分の 1 秒単位まで
<b>108</b>	Address (アドレス)
<b>112</b>	コード化された数値
<b>116</b>	非表示文字
<b>120</b>	Hexadecimal (16 進)
<b>124</b>	テーブル・バージョン
<b>128</b>	バイナリーの導出された率、小数点以下 3 桁まで
<b>132</b>	バイナリーの導出されたパーセント、小数点以下 3 桁まで
<b>136</b>	バイナリーの導出された合計、小数点以下 3 桁まで
<b>140</b>	バイナリーの導出された平均、小数点以下 3 桁まで
<b>144</b>	パック 10 進数のタイム・スタンプ、秒単位まで
<b>148</b>	パック 10 進数のタイム・スタンプ、10 分の 1 秒単位まで
<b>152</b>	間隔保管クロック、12 バイト

#### **INHERIT**

この属性値が継承可能かどうかを示します: Y または N。CICSplex の継承に参加している CICSplex SM 定義リソース・テーブルに対してのみ有効です。

#### **METANAME**

METANAME リソース・テーブルには、すべての CVDAS、CVDAT、および EYUDA に関する情報が含まれています。

#### **属性**

##### **説明**

#### **NAMETYPE**

データのタイプ

##### **1**

CVDAS

**2**

CVDAT

**3**

EYUDA

**VALUE**

CVDA または EYUDA の数値

**NAME**

CVDA または EYUDA の名前

**DESCRIPTION**

CVDA または EYUDA の説明

**METAPARM**

METAPARM リソース・テーブルには、アクションのパラメーターに関する情報が含まれています。

**属性**

説明

**TABLE**

テーブル名

**ACTION**

アクション名

**NAME**

API の PARM スtring で示されるパラメーター名

**ID**

パラメーター番号

**GROUP\_ID**

グループの 1 つのみを指定できるという意味で、複数のパラメーターを相互に関連付けることができます。このようにして関連付けられたパラメーターには同じグループ ID が割り当てられます。

**REQUIRED**

このパラメーターが必須かどうかを示します。

**Y**

パラメーターが必要な場合

**N**

パラメーターが不要な場合

**WORKLOAD**

このパラメーターがワークロード名かどうかを示します。

**Y**

このパラメーターはワークロード名です。

**N**

このパラメーターはワークロード名ではありません。

**WRKLOWNER**

このパラメーターがワークロード所有者の名前かどうかを示します。

**Y**

このパラメーターはワークロード所有者の名前です。

**N**

このパラメーターはワークロード所有者の名前ではありません。

**VALUE**

パラメーター値

**MODE**

パラメーターの適用方法

- 1 ベース・テーブルからコピー
- 2 値の配列
- 3 ビット設定
- 4 API パラメーター・ストリング内のキーワード
- 5 フィルター・ストリング
- 6 値を持つ API キーワード
- 7 存在ビット付きの基本テーブル・フィールド
- 8 API 変更ストリング

モード 3 と 4 は独立キーワードとして API パラメーター・ストリング内に出現します。モード 2、5、6、および 8 は値を持つキーワードとして API パラメーター・ストリング内に出現します。モード 1 と 7 は API パラメーター・ストリング内に出現しません。

## DESCRIPTION

説明

### CICSVALAVAIL

CICS 妥当性データが得られるかどうかを示します。

**Y**

CICS 妥当性データが得られます。

**N**

CICS 妥当性データが得られません。

### VALCICSESA

このパラメーターが、異なるバージョンの CICS で有効かどうかを示すフラグの第 1 バイト。

1... ..	X'80'	CICS/MVS 2.1.2
.1... ..	X'40'	CICS/ESA 3.3.0
..1... ..	X'20'	CICS/ESA 4.1.0
...1... ..	X'10'	CICS Transaction Server for OS/390 Release 1
.... 1... ..	X'08'	CICS Transaction Server for OS/390 Release 2
.... .1... ..	X'04'	CICS Transaction Server for OS/390 Release 3
.... ..1... ..	X'02'	CICS Transaction Server for z/OS, Version 2 Release 1
.... ...1... ..	X'01'	CICS Transaction Server for z/OS, Version 2 Release 2

このビットがオンに設定されている場合、このパラメーターはそのバージョンの CICS では有効ではありません。フラグの第 2 バイトは VALCICSES2 に格納されています。

### VALCICSES2

このパラメーターが、異なるバージョンの CICS Transaction Server for z/OS で有効かどうかを示すフラグの第 2 バイト。

1... ..	X'80'	CICS Transaction Server for z/OS, Version 2 Release 3
.1... ..	X'40'	CICS Transaction Server for z/OS, Version 3 Release 1
..1... ..	X'20'	CICS Transaction Server for z/OS, Version 3 Release 2
...1... ..	X'10'	CICS Transaction Server for z/OS, Version 4 Release 1
.... 1... ..	X'08'	CICS Transaction Server for z/OS, Version 4 Release 2
.... .1... ..	X'04'	CICS Transaction Server for z/OS, Version 5 Release 1
.... ..1... ..	X'02'	CICS Transaction Server for z/OS, Version 5 Release 2
.... ...1... ..	X'01'	CICS Transaction Server for z/OS, Version 5 Release 3

このビットがオンに設定されている場合、このパラメーターはそのバージョンの CICS では有効ではありません。フラグの第 1 バイトは VALCICSESA に格納されています。

### VALCICSES3

このパラメーターが、異なるバージョンの CICS Transaction Server for z/OS で有効かどうかを示すフラグの第 3 バイト。

```
1... .... X'80' CICS Transaction Server for z/OS, Version 5 Release 4
```

### VALCICSVSE

このパラメーターが、異なるバージョンの CICS Transaction Server for VSE で有効かどうかを示すフラグ。

```
1... .... X'80' CICS/VSE 2.2.0
.1.. .... X'40' CICS/VSE 2.3.0
..1. .... X'20' CICS/VSE 4.1.0
...1 1111 Reserved
```

このビットがオンに設定されている場合、このパラメーターはそのバージョンの CICS では有効ではありません。

### VALCICSOS2

このパラメーターが、異なるバージョンの CICS OS/2 で有効かどうかを示すフラグ。

```
1... .... X'80' CICS OS/2 2.0.1
.1.. .... X'40' CICS OS/2 3.0.0
..1. .... X'20' CICS OS/2 3.1.0
...1 1111 Reserved
```

このビットがオンに設定されている場合、このパラメーターはそのバージョンの CICS では有効ではありません。

### VALCICSWNT

このパラメーターが、異なるバージョンの TXSeries で有効かどうかを示すフラグ。

```
1... .... X'80' CICS for TXSeries 4.3.0
.1.. .... X'40' CICS for TXSeries 5.0.0
..11 1111 Reserved
```

このビットがオンに設定されている場合、このパラメーターはそのバージョンの CICS では有効ではありません。

## OBJECT

OBJECT リソース・テーブルには、特定の管理対象オブジェクトのアクション情報が含まれています。

### 属性

#### 説明

### OBJECT

そのアクションが適用される管理対象オブジェクトの名前。

### TABLEVER

OBJECT 属性によって識別されるテーブルのバージョン。

### ACTION

アクションの名前。長さは 1 文字から 12 文字までです。

### VALCICSESA

このアクションが、異なるバージョンの CICS で有効かどうかを示すフラグの第 1 バイト。

```
1... .... X'80' CICS/MVS 2.1.2
.1.. .... X'40' CICS/ESA 3.3.0
..1. .... X'20' CICS/ESA 4.1.0
...1 .... X'10' CICS Transaction Server for OS/390 Release 1
.... 1... X'08' CICS Transaction server for OS/390 Release 2
.... .1.. X'04' CICS Transaction Server for OS/390 Release 3
.... ..1. X'02' CICS Transaction Server for z/OS, Version 2 Release 1
.... ...1 X'01' CICS Transaction Server for z/OS Version 2 Release 2
```

このビットがオンに設定されている場合、このアクションはそのバージョンの CICS では有効ではありません。フラグの第 2 バイトは VALCICSES2 に格納されています。

## VALCICSVSE

このアクションが、異なるバージョンの CICS Transaction Server for VSE で有効かどうかを示すフラグ。

1... ..	X'80'	CICS/VSE 2.2.0
.1... ..	X'40'	CICS/VSE 2.3.0
..1... ..	X'20'	CICS/VSE 4.1.0
...1 1111		Reserved

このビットがオンに設定されている場合、このアクションはそのバージョンの CICS では有効ではありません。

## VALCICSOS2

このアクションが、異なるバージョンの CICS OS/2 で有効かどうかを示すフラグ。

1... ..	X'80'	CICS OS/2 2.0.1
.1... ..	X'40'	CICS OS/2 3.0.0
..1... ..	X'20'	CICS OS/2 3.1.0
...1 1111		Reserved

このビットがオンに設定されている場合、このアクションはそのバージョンの CICS では有効ではありません。

## VALCICSWNT

このアクションが、異なるバージョンの TXSeries で有効かどうかを示すフラグ。

1... ..	X'80'	CICS for TXSeries 4.3.0
.1... ..	X'40'	CICS for TXSeries 5.0.0
..11 1111		Reserved

このビットがオンに設定されている場合、このアクションはそのバージョンの CICS では有効ではありません。

## VALCICSES2

このアクションが、異なるバージョンの CICS Transaction Server for z/OS で有効かどうかを示すフラグの第 2 バイト。

1... ..	X'80'	CICS Transaction Server for z/OS, Version 2 Release 3
.1... ..	X'40'	CICS Transaction Server for z/OS, Version 3 Release 1
..1... ..	X'20'	CICS Transaction Server for z/OS, Version 3 Release 2
...1 ....	X'10'	CICS Transaction Server for z/OS, Version 4 Release 1
.... 1...	X'08'	CICS Transaction Server for z/OS, Version 4 Release 2
.... .1..	X'04'	CICS Transaction Server for z/OS, Version 5 Release 1
.... ..1.	X'02'	CICS Transaction Server for z/OS, Version 5 Release 2
.... ...1	X'01'	CICS Transaction Server for z/OS, Version 5 Release 3

このビットがオンに設定されている場合、このアクションはそのバージョンの CICS では有効ではありません。フラグの第 1 バイトは VALCICSESA に格納されています。

## DESCRIPTION

アクションの説明

## ID

アクションの番号

## PARMCOUNT

このアクションのパラメーターの数

## APIPERFORM

EXEC CPSM の PERFORM、GET、SET、CREATE、UPDATE、および REMOVE に対してアクションが有効かどうかを示します。

## N

アクションは有効ではありません。

## Y

このアクションは有効です。

## OBJECT

OBJECT リソース・テーブルには、特定の管理対象オブジェクトの詳細情報が含まれています。

### 属性

#### 説明

### NAME

管理対象オブジェクトの名前。長さは 1 文字から 8 文字までです。

### ID

リソース・テーブル ID 番号。

### NUMTBLVER

存在することがわかっている、この管理対象オブジェクトの異なるバージョンの番号。

### HIGHTBLVER

この管理対象オブジェクトの最大バージョンの番号。

### RELTBLVER

現在の CICSplex SM リリースにおける、この管理対象オブジェクトのバージョン。

### OWNERNAME

この管理対象オブジェクトを所有しているコンポーネントの名前。

### CREATREL

この管理対象オブジェクトが導入された CICSplex SM リリース。

### QUERYREL

クエリー発行元 CMAS の CICSplex SM リリース。

### OBJTYPE

この管理対象オブジェクトのオブジェクト・タイプ。

#### C

CICS リソース

#### M

モニターされた CICS リソース

#### D

CPSM 定義

#### V

CPSM リソース

#### O

CPSM メタデータ

#### N

CPSM 通知

#### R

CICS リソース定義

#### L

CPSM 構成定義

### CURTBLVER

現在の CONNECT バージョンにおける、この管理対象オブジェクトのバージョン。

### CURNUMATTR

現在の CONNECT バージョンにおける、この管理対象オブジェクト内の属性の数。

### CURSTGSI

現在の CONNECT バージョンにおける、この管理対象オブジェクトの外部長さ。

### CURCPSMREL

現在の CONNECT バージョンにおける、この管理対象オブジェクトのバージョンが作成されたときの CICSplex SM リリース。

### CURVALRTA

この管理対象オブジェクトを RTA で有効に使用できるかどうかを示します: Y または N。



## CURVALUTL

この管理対象オブジェクトをバッチ・ユーティリティーで有効に使用できるかどうかを示します:Y または N。

## CURGETVAL

この管理対象オブジェクトが GET 要求に対して有効かどうかを示します:Y または N。

## CURSETVAL

この管理対象オブジェクトが SET 要求に対して有効かどうかを示します:Y または N。

## CURCREVAL

この管理対象オブジェクトが CREATE 要求に対して有効かどうかを示します:Y または N。

## CURUPDVAL

この管理対象オブジェクトが UPDATE 要求に対して有効かどうかを示します:Y または N。

## CURREMVAL

この管理対象オブジェクトが REMOVE 要求に対して有効かどうかを示します:Y または N。

## CURACTVAL

この管理対象オブジェクトにアクションが定義されているかどうかを示します:Y または N。

OBJECT リソース・テーブルを使用してアクション情報を取得します。

## CURVALESA

この管理対象オブジェクトが、異なるバージョンの CICS で有効かどうかを示すフラグの第 1 バイト。

1... ..	X'80'	CICS/MVS 2.1.2
.1... ..	X'40'	CICS/ESA 3.3.0
..1... ..	X'20'	CICS/ESA 4.1.0
...1... ..	X'10'	CICS Transaction Server for OS/390 Release 1
....1... ..	X'08'	CICS Transaction Server for OS/390 Release 2
.....1... ..	X'04'	CICS Transaction Server for OS/390 Release 3
.....1... ..	X'02'	CICS Transaction Server for z/OS Version 2 Release 1
.....1... ..	X'01'	CICS Transaction Server for z/OS Version 2 Release 2

このビットがオンに設定されている場合は、当オブジェクトはそのバージョンの CICS では有効ではありません。フラグの第 2 バイトは CURVALES2 に格納されています。

## CURVALES2

この管理対象オブジェクトが、異なるバージョンの CICS Transaction Server for z/OS で有効かどうかを示すフラグの第 2 バイト。

1... ..	X'80'	CICS Transaction Server for z/OS, Version 2 Release 3
.1... ..	X'40'	CICS Transaction Server for z/OS, Version 3 Release 1
..1... ..	X'20'	CICS Transaction Server for z/OS, Version 3 Release 2
...1... ..	X'10'	CICS Transaction Server for z/OS, Version 4 Release 1
....1... ..	X'08'	CICS Transaction Server for z/OS, Version 4 Release 2
.....1... ..	X'04'	CICS Transaction Server for z/OS, Version 5 Release 1
.....1... ..	X'02'	CICS Transaction Server for z/OS, Version 5 Release 2
.....1... ..	X'01'	CICS Transaction Server for z/OS, Version 5 Release 3

このビットがオンに設定されている場合、管理対象オブジェクトはそのバージョンの CICS では有効ではありません。フラグの第 1 バイトは CURVALESA に格納されています。

## CURVALES3

この管理対象オブジェクトが、異なるバージョンの CICS Transaction Server for z/OS で有効かどうかを示すフラグの第 3 バイト。

1... ..	X'80'	CICS Transaction Server for z/OS, Version 5 Release 4
---------	-------	---

## CURVALVSE

この管理対象オブジェクトが、異なるバージョンの CICS/VSE で有効かどうかを示すフラグ。

1... ..	X'80'	CICS/VSE 2.2.0
.1... ..	X'40'	CICS/VSE 2.3.0
..1... ..	X'20'	CICS/VSE 4.1.0
...1 1111		Reserved

このビットがオンに設定されている場合は、当オブジェクトはそのバージョンの CICS では有効ではありません。

#### **CURVALOS2**

この管理対象オブジェクトが、異なるバージョンの CICS OS/2 で有効かどうかを示すフラグ。

1... ..	X'80'	CICS OS/2 2.0.1
.1... ..	X'40'	CICS OS/2 3.0.0
..1... ..	X'20'	CICS OS/2 3.1.0
...1 1111		Reserved

このビットがオンに設定されている場合は、当オブジェクトはそのバージョンの CICS では有効ではありません。

#### **CURVALWNT**

この管理対象オブジェクトが、異なるバージョンの TXSeries で有効かどうかを示すフラグ。

1... ..	X'80'	CICS for TXSeries 4.3.0
.1... ..	X'40'	CICS for TXSeries 5.0.0
..11 1111		Reserved

このビットがオンに設定されている場合は、当オブジェクトはそのバージョンの CICS では有効ではありません。

#### **DESC**

この管理対象オブジェクトの説明。

#### **VIEWMOD**

この管理対象オブジェクトが表示サポートに適格かどうかを示します。

**Y**

この管理対象オブジェクトは表示サポートに適格です。

**N**

この管理対象オブジェクトは表示サポートに適格ではありません。

#### **APIPREFIX**

API 接頭部が必要かどうかを示します。

**Y**

API 接頭部は必要です。

**N**

API 接頭部は不要です。

#### **SCOPE SORT**

**Y**

API によって、スコープに基づいてソートされます。

**N**

API によって、スコープに基づいてソートされません。

#### **SCOPE REQ**

**Y**

スコープを指定する必要があります。

**N**

スコープを指定する必要はありません。

### **PARMAVA**

PARMAVA リソース・テーブルでは、パラメーターで指定できる値に関する情報が提供されます。

属性

説明

#### **PARMIDN**

パラメーター番号

## PARMAVAIDN

パラメーターの AVA 番号

## LITERAL

パラメーターのリテラル

## VALUE

数値形式のパラメーター値

## VALUENAME

文字ストリング形式のパラメーター値

## CRESxxxx リソース・テーブルの使用

CRESxxxx リソース・テーブルは、トポロジー・リソース・マップの外部化されたバージョンで、通常、リソースがインストール、追加、破棄、または削除されたときに更新されます。この情報は、CICS XRSINDI グローバル・ユーザー出口を介して、または CRESIPCN の場合は XMEOUT メッセージ出口を介して取り込まれます。

また、既存のリソースの特性が変更されたときも、少数の CRESxxxx リソース・テーブルが更新されます。MAS ハートビート時間で定期的に更新されるこれらの CRESxxxx リソース・テーブルは、次のとおりです。

- CRESDSNM データ・セット
- CRESFECO FEPI 接続
- CRESGLUE グローバル・ユーザー出口ルーチン
- CRESSDMP システム・ダンプ・コード
- CRESTDMP トランザクション・ダンプ・コード
- CRESTRUE タスク関連ユーザー出口ルーチン

CRESxxxx トポロジー・リソース・マップが変更されたことの通知を必要とするアプリケーション・プログラムでは、API LISTEN コマンドを使用して、対応する ERMCxxxx CPSM 通知リソース・テーブルに対するインタレストを登録できます。

注：接続の両側での状態の同期の結果、IPCONN が獲得または解放された場合、戻される実際の値は、実行側リソースとパートナー・リソースのそれぞれについて、**Obtaining** と **Acquired** の組み合わせ、または **Freeing** と **Released** の組み合わせになります。

## CICSplex SM API 出口の照会

CICS LMAS 環境では、タスク関連ユーザー出口を介して、CICSplex SM API 関数が実装されます。CICS アプリケーション・プログラムでは、EXEC CICS INQUIRE EXITPROGRAM コマンドを使用して、CICSplex SM API タスク関連ユーザー出口に関する情報を取得できます。

```
EXEC CICS INQUIRE EXITPROGRAM(EYU9XLAP)
                CONNECTST(cvda)
                QUALIFIER(data-area)
```

INQUIRE EXITPROGRAM コマンドの CONNECTST と QUALIFIER の各キーワードをサポートしている CICS システムでは、CONNECTST によって CICSplex SM API タスク関連ユーザー出口の状況を示す CVDA が戻されて、QUALIFIER によって LMAS の接続先の CICSplex の名前が戻されます。INQUIRE EXITPROGRAM コマンドについて詳しくは、[INQUIRE EXITPROGRAM](#) を参照してください。



## 第 3 章 EXEC CPSM プログラムの作成

CICSplex SM コマンド・レベル・インターフェースを使用して API プログラムを作成できます。CICSplex SM リソース・テーブルごとに、言語固有のコピーブックが提供されています。

### リソース・テーブルのコピーブックの使用

CICSplex SM API は、リソース属性が含まれたレコードの形式でリソース・データを受け取って返します。

例えば、LOCTRAN リソース・テーブル・レコードが含まれた結果セットに対して FETCH コマンドを発行した場合は、API からは、単一レコード内の特定のトランザクションのすべての属性が返されます。プログラム側では、これらのリソース・テーブル・レコードを受け取るためのストレージ域を指定する必要があります。

**注：**このようにしてデータを返す方法とは異なり、EXEC CICS システム・プログラミング・インターフェースでは、リソースの各属性を個別に取り出す必要があります。

これらのリソース・テーブル・レコードの使用を簡素化するために、CICSplex SM では、API プログラムからアクセスできる各リソース・テーブルの一連のコピーブックが提供されています。これらのコピーブックをプログラムに含めることにより、使用している言語に適した構造と形式のリソース・テーブル・データにアクセスできます。

### コピーブックのアクセス方法

コピーブックは、CICSplex SM インストール・プロセスの一部としてインストールされます。

コピーブックは、CICSplex SM インストール・プロセスの一部としてインストールされます。これらのコピーブックは以下のライブラリーに格納されます。

#### アセンブラー

CICSTS56.CPSM.SEYUMAC

#### COBOL

CICSTS56.CPSM.SEYUCOB

#### PL1

CICSTS56.CPSM.SEYUPL1

#### C

CICSTS56.CPSM.SEYUC370

コピーブックをプログラムに含めるには、アセンブルまたはコンパイルのステップで該当するライブラリーを使用する必要があります。

**注：**CICSplex SM API では、EYU で始まる変数名が使用されます。プログラムによって定義される変数や構造体で使用する変数名が、変換プログラムによって生成される変数名やリソース・テーブルのコピーブックで宣言される変数名と重複しないようにしてください。また、プログラムでこのような変数名が暗黙的に生成されないように注意してください。

### コピーブック名と別名

それぞれの CICSplex SM リソース・テーブルには、当製品内で固有の名前が割り当てられています。

さらに、各言語のリソース・テーブルの各コピーブック・バージョンに対して、固有の名前が作成されます。コピーブック名の形式は次のとおりです。

```
EYUtnnnn
```

説明:

**t**

そのコピーブックでサポートされている言語を識別し、次のいずれかの値になります。

**A**  
アセンブラー  
**P**  
PL/I  
**L**  
COBOL  
**C**  
C

**nnnn**

4 文字の数字のリソース・テーブル識別子です。

例えば、次のとおりです。

**EYUA0001**

これは CICS RGN リソース・テーブルのアセンブラー DSECT です。

**EYUC2451**

これは CMAS リソース・テーブルの C 構造化データ型です。

プログラムでコピーブックを参照しやすくするために、CICSplex SM ではコピーブック名の別名がサポートされています。適切なデータ・セットには、各リソース・テーブルについて次の 2 つのエントリーが含まれています。

**EYUtnnnn**

リソース・テーブルのコピーブックの名前。

**formname**

フォーマット名の別名 (これは CICSplex SM リソース・テーブル で説明しているリソース・テーブル名です)。

したがって、上記の例を使用した場合、CICS RGN リソース・テーブルのアセンブラー DSECT を EYUA0001 として参照することも、別名の CICS RGN として参照することもできます。

## コピーブックのフォーマット

各コピーブックには、リソース・テーブルとその特性を記述する序文部分が含まれています。

コピーブックの序文部分には以下のものが含まれています。

- 有効な API 操作
- 操作に必要なパラメーター
- 有効な API アクション
- このリソース・テーブルをサポートしていない CICS リリース (存在する場合)

リソース・テーブルの各属性の説明が示されます。また、該当する場合には、属性に関する以下の情報が提供されます。

- その属性を SET コマンドで変更可能かどうか
- その属性をサポートしていない CICS リリース (存在する場合)
- その属性の変更を許可していない CICS リリース (存在する場合)

## コピーブック・データの特性

API プログラムによって処理できる各リソース・テーブルには、そのリソース・テーブルの各属性のデータ値が含まれています。

属性値は、プログラムが実行されている環境とデータ・タイプに適した内部形式で提示されます。

- 標準の System/390® データ・フォーマットが使用されます。属性値に対して変換やフォーマット設定の操作は実行されません。
- C で作成されたプログラムの場合は、可変長文字フィールドにはゼロバイトの終了区切り文字は含まれていません。

- すべてのリソース・テーブル・レコードの長さは 8 バイトの倍数です。各コピーブックには、リソース・テーブル長の定義が含まれています。
- System/390 の境界合わせがすべてのデータ・タイプについて順守されています。つまり、すべてのリソース・テーブル・レコードは、内部的に、ダブルワードで境界合わせされた保管場所を開始位置として維持されます。位置合わせフィールドは各コピーブック内で自動的に生成されます。これらの位置合わせフィールドにはバイナリー・ゼロが含まれ、次のような名前が付けられます。

EYU\_RSVnnnn

プログラムでリソース・テーブル・レコードを送受信するために使用されるデータ領域に適切な境界合わせが適用されていることを確認してください。

## 提供されるコピーブック

リソース・テーブルのコピーブックは、各言語に対して提供されています。

### アセンブラー・コピーブック

アセンブラーのコピーブックは、CICSTS56.CPSM.SEYUMAC で配布されます。

#### 分散:

DSECT

#### コピーブック名:

EYUAnnnn

アセンブラーのコピーブックを使用する際は、以下のことに留意してください。

- DSECT および DS ステートメントを使用してリソース・テーブルを記述します。
- DSECT 名は、リソース・テーブル・フォーマット名 (EMASSTRT など) です。
- 属性名は、リソース・テーブル・フォーマット名と属性名をアンダースコアで連結したものです (例: EMASSTRT\_CMASNAME)。
- EQU ステートメントを使用して、ビット値、バイナリー値、および文字値のインディケータ・フィールドの設定を記述します。
- テーブル長フィールドは、リソース・テーブル・フォーマット名と定数 TBL\_LEN をアンダースコアで連結したものです (例: EMASSTRT\_TBL\_LEN)。

リソース・テーブルのデータ・タイプは、DS ステートメントのデータ定義オペランドを使用して定義されます。以下のデータ・タイプ定義が使用されます。

```
DS   X   - Bit, binary values greater than 8 bytes
        - Odd number binary values less than 8 bytes
        - Mixed character and binary data

DS   H   - 2-byte binary numeric values

DS   F   - 4-byte binary numeric values
        - 4-byte intervals

DS   D   - Time stamps and 8-byte intervals
        - 8-byte numeric values

DS   P   - Packed decimal data

DS   C   - Character data
```

72 ページの図 13 では、アセンブラーのリソース・テーブル・コピーブックをサンプルとして抽出したものを示しています。

```

*-----*
* Name = EYUA2400                                     *
* Format Name = EMASSTRT                             *
* Version = 0001                                       *
* Status = CPSMREL(0310)                             *
* Function = Base Table Structure generator           *
* Format definition for this element = EMASSTRT       *
* Valid Operations = None                             *
* Valid Actions = None                               *
*-----*
EMASSTRT          DSECT      Notify CICS System Start Event
EMASSTRT_CMASNAME DS CL0008  CMAS Name
EMASSTRT_PLEXNAME DS CL0008  CICSplex Name
EMASSTRT_CSYSNAME DS CL0008  CICS System Name
EMASSTRT_MON_SPEC DS CL0008  Monitor Spec Name
EMASSTRT_RTA_SPEC DS CL0008  Real Time Analysis Spec Name
EMASSTRT_WLM_SPEC DS CL0008  Work Load Manager Spec Name
EMASSTRT_STATUS   DS XL0001  Status
EMASSTRT_STATUS_LOCAL EQU 128 Local MAS
EMASSTRT_STATUS_REMOTE EQU 64 Remote MAS
EMASSTRT_DYNROUTE DS XL0001  Dynamic Routing Mode
EMASSTRT_DYNROUTE_ACTIVE EQU 1 Routing ACTIVE
EMASSTRT_DYNROUTE_SUSPEND EQU 2 Routing SUSPENDED
EMASSTRT_DYNTYPE   DS CL0003  Dynamic Routing Type
EMASSTRT_DYNTYPE_WLMTOR EQU C'TOR' Routing TOR
EMASSTRT_DYNTYPE_WLMAOR EQU C'AOR' Routing AOR
EMASSTRT_DESC      DS CL0030  Description
EMASSTRT_CSYSAPPL  DS CL0008  CICS System VTAM APPLID
EMASSTRT_EYU_RSV0015 DS XL0005 Alignment Padding
EMASSTRT_MASSTART  DS D      MAS Start STCK Value
EMASSTRT_TMEZONE0  DS XL0001  Time Zone Offset
EMASSTRT_TMEZONE   DS CL0001  Time Zone
EMASSTRT_EYU_RSV0019 DS XL0002 Alignment Padding
EMASSTRT_DAYLGHTSV DS F      Daylight saving in effect
EMASSTRT_SYSID     DS CL0004  MAS System Id
EMASSTRT_OPSYSREL  DS CL0004  MAS Op Sys Release
EMASSTRT_MVSNAME   DS CL0004  MVS System Name
EMASSTRT_JOBNAME   DS CL0008  MAS Job Name
EMASSTRT_CECNAME   DS CL0008  CEC Name
EMASSTRT_SYSPLEX   DS CL0008  SYSPlex Name
EMASSTRT_EYU_RSV0257 DS XL0004 Alignment Padding
EMASSTRT_TBL_LEN   EQU 152   Current Table size

```

注：VTAM® は現在 z/OS Communications Server になっています。

図 13. サンプルのアセンブラー・コピーブック

## PL/I コピーブック

PL/I コピーブックは、CICSTS56.CPSM.SEYUPL1 で配布されます。

### 分散:

基底付き構造

### コピーブック名:

EYUPnnnn

PL/I コピーブックを使用する場合は、以下のことに注意してください。

- 変数 EYUPTPTR を次のように明示的に宣言する必要があります。

```
DCL EYUPTPTR POINTER;
```

- 構造レベル 1 の名前は、リソース・テーブル・フォーマット名 (EMASSTRT など) です。
- 属性名は従属レベル名として使用されます。
- ビット・インディケーターを記述する属性については、従属構造レベルが使用されます。各ビット・インディケーターには固有の名前が割り当てられています。
- 他のすべてのインディケーター属性については、リソース・テーブルの末尾に定数宣言が生成されます。これらの定数は、その属性の割り当てや評価のために使用できます。定数名は、リソース・テーブル名、属性名、およびインディケーター名をアンダースコアで連結したものです (例: EMASSTRT\_DYNROUTE\_ACTIVE)。



- テーブル長フィールドは、リソース・テーブル・フォーマット名と定数 TBL\_LEN をアンダースコアで連結したものです (例: EMASSTRT\_TBL\_LEN)。

リソース・テーブル・データ・タイプは、有効な PL/I データ・タイプ・セット内にマップされています。ただし、正確なマッピングが常に可能であるわけではありません。リソース・テーブル・データ・タイプは以下のようにマップされています。

BIT(8) ALIGNED	- 1-byte binary numeric values
FIXED BIN(15)	- 2-byte binary numeric values
FIXED BIN(31)	- 4-byte binary numeric values - 4-byte intervals
(2) FIXED BIN(31)	- Time stamps and 8-byte intervals - 8-byte binary numeric values (an array of two fullwords)
FIXED DEC(n)	- Packed decimal data
CHAR(nnnn)	- Character data - Binary values greater than 8 bytes - Odd number binary values less than 8 bytes

74 ページの図 14 では、PL/I リソース・テーブル・コピーブックをサンプルとして抽出したものを示しています。

```

/*-----*/
/* Name = EYUP2400 */
/* Format Name = EMASSTRT */
/* Version = 0001 */
/* Status = CPSMREL(0310) */
/* Function = Base Table Structure generator */
/* Format definition for this element = EMASSTRT */
/* Valid Operations = None */
/* Valid Actions = None */
/*-----*/

```

```

DCL 01 EMASSTRT BASED(EYUPTPTR), /* Notify CICS System Start Event*/
02 CMASNAME CHAR(0008),
/* CMAS Name */
02 PLEXNAME CHAR(0008),
/* CICSplex Name */
02 CSYSNAME CHAR(0008),
/* CICS System Name */
02 MON_SPEC CHAR(0008),
/* Monitor Spec Name */
02 RTA_SPEC CHAR(0008),
/* Real Time Analysis Spec Name */
02 WLM_SPEC CHAR(0008),
/* Work Load Manager Spec Name */

```

```

02 STATUS,
/* Status */
03 LOCAL BIT(1) UNALIGNED,
/* Local MAS */
03 REMOTE BIT(1) UNALIGNED,
/* Remote MAS */
03 RSVD0003 BIT(1) UNALIGNED,
/* Reserved */
03 RSVD0004 BIT(1) UNALIGNED,
/* Reserved */
03 RSVD0005 BIT(1) UNALIGNED,
/* Reserved */
03 RSVD0006 BIT(1) UNALIGNED,
/* Reserved */
03 RSVD0007 BIT(1) UNALIGNED,
/* Reserved */
03 RSVD0008 BIT(1) UNALIGNED,
/* Reserved */

```

```

02 DYNROUTE BIT(8) ALIGNED,
/* Dynamic Routing Mode */
02 DYNTYPE CHAR(0003),
/* Dynamic Routing Type */
02 DESC CHAR(0030),
/* Description */
02 CSYSAPPL CHAR(0008),
/* CICS System VTAM APPLID */
02 EYU_RSV0015 CHAR(0005),
/* Alignment Padding */
02 MASSTART(2) FIXED BIN(31),
/* MAS Start STCK Value */
02 TMEZONE0 BIT(8) ALIGNED,
/* Time Zone Offset */
02 TMEZONE CHAR(0001),
/* Time Zone */
02 EYU_RSV0019 CHAR(0002),
/* Alignment Padding */
02 DAYLGHTSV FIXED BIN(31),
/* DayLight saving in effect */
02 SYSID CHAR(0004),
/* MAS System Id */
02 OPSYSREL CHAR(0004),
/* MAS Op Sys Release */
02 MVSNAME CHAR(0004),
/* MVS System Name */
02 JOBNAME CHAR(0008),
/* MAS Job Name */
02 CECNAME CHAR(0008),
/* CEC Name */
02 SYSPLEX CHAR(0008),
/* SYSPlex Name */
02 EYU_RSV0257 CHAR(0004),
/* Alignment Padding */

```

注: VTAMは現在 z/OS Communications Server になっています。

```

/*-----*/
/*
/* EMASSTRT Constants for Table
/*
/*-----*/
DCL EMASSTRT_DYNROUTE_ACTIVE BIT(8) ALIGNED STATIC INIT('01'BX);
/* Routing ACTIVE */
DCL EMASSTRT_DYNROUTE_SUSPEND BIT(8) ALIGNED STATIC INIT('02'BX);
/* Routing SUSPENDED */
DCL EMASSTRT_DYNTYPE_WLMTOR CHAR(3) STATIC INIT('TOR');
/* Routing TOR */
DCL EMASSTRT_DYNTYPE_WLMAOR CHAR(3) STATIC INIT('AOR');
/* Routing AOR */
DCL EMASSTRT_TBL_LEN FIXED BIN(15) STATIC INIT(152);

```

図 14. サンプルの PL/I コピーブック

## COBOL コピーブック

COBOL コピーブックは、CICSTS56.CPSM.SEYUCOB で配布されます。

### 分散:

構造

## コピーブック名:

EYULnnnn

COBOL コピーブックを使用する場合は、以下のことに注意してください。

- 構造レベル 1 の名前は、リソース・テーブル・フォーマット名 (EMASSTRT など) です。
- 属性名は従属レベル名として使用されます。
- インディケーターを記述する属性については、従属 88 レベルが使用されます。各インディケーターには固有の名前が割り当てられています。インディケーター設定の内容は 16 進数リテラルを使用して記述されます。
- デフォルトでは、CICSplex SM の属性名は、WLM\_SPEC のように、アンダースコア文字で接続した構成になります。しかし、CICS でサポートされている初期バージョンの COBOL では、アンダースコアの使用はサポートされていません。このため、アンダースコアが含まれるすべての属性名は、コピーブック内では WLM-SPEC のようにハイフンを使用するように変換されます。属性名がこの API に渡されるときは、それらにはハイフンではなくアンダースコア文字が含まれている必要があります。
- すべてのリソース・テーブルでは、リテラル区切り文字としてアポストロフィ文字が使用されます。提供されているコピーブックを使用してプログラムを変換またはコンパイルするときは、APOST オプションを指定する必要があります。それ以外の場合、COBOL の警告メッセージを受け取ります。
- COBOL では、多くの語が COBOL 独自の用途のために予約されています。CICSplex SM の一部のリソース・テーブル名と属性名はこれらの予約語と競合しています。このような競合を防ぐために、COBOL の予約語と競合するすべての CICSplex SM アイテム名には、-R という接尾部を追加する変更が加えられます。例えば、CONNECT リソース・テーブルの名前は CONNECT-R に変換され、STATUS 属性の名前は STATUS-R に変換されます。COBOL の予約語と競合する名前コメント領域には「-- RESERVED WORD --」という説明が表示されます。リソース・テーブルや属性の名前がこの API に渡されるときは、それらの名前には -R 接尾部が含まれていない必要があります。
- COBOL では、同じデータ構造内の異なるレベルにおける重複名はサポートされていません。CICSplex SM の一部の属性名はリソース・テーブル名と同じです。重複名の問題を回避するために、リソース・テーブル名と同じである属性名は、-A という接尾部を追加する変更が加えられます。例えば、DSNAME という属性名は DSNAME-A に変換されます。DSNAME リソース・テーブルの名前は変更されないままです。リソース・テーブルと同じ名前属性のコメント領域には「-- RESERVED WORD--」という説明が表示されます。属性名がこの API に渡されるときは、それらの名前には -A 接尾部が含まれていない必要があります。
- テーブル長フィールドは、リソース・テーブル・フォーマット名と定数 TBL-LEN をハイフンで連結したものです (例: EMASSTRT-TBL-LEN)。

## リソース・テーブル・データ・タイプから COBOL データ・タイプへのマッピング

リソース・テーブル・データ・タイプは、有効な COBOL データ・タイプ・セット内にマップされています。ただし、正確なマッピングが常に可能であるわけではありません。リソース・テーブル・データ・タイプは以下のようにマップされています。

PIC S9(0004) USAGE BINARY	- 2-byte binary numeric values
PIC S9(0008) USAGE BINARY	- 4-byte binary numeric values - 4-byte intervals
PIC S9(0016) USAGE BINARY	- Time stamps and 8-byte intervals - 8-byte binary numeric values
PIC S9(nnnn) USAGE PACKED-DECIMAL	- Packed decimal data
PIC X(0001)	- 1-byte binary and bit indicators
PIC X(nnnn)	- Character data - Binary values greater than 8 bytes - Odd number binary values less than 8 bytes

## サンプルの COBOL コピーブック

このサンプルは、COBOL リソース・テーブル・コピーブックをサンプルとして 抽出したものを示しています。

```
* -----*
* Name = EYUL2400                                     *
* Format Name = EMASSTRT                             *
* Version = 0001                                       *
* Status = CPSMREL(0310)                             *
* Function = Base Table Structure generator           *
* Format definition for this element = EMASSTRT       *
* Valid Operations = None                             *
* Valid Actions = None                                *
* -----*
01 EMASSTRT.
* Notify CICS System Start Event
  02 CMASNAME      PIC X(0008).
* CMAS Name
  02 PLEXNAME      PIC X(0008).
* CICSplex Name
  02 CSYSNAME      PIC X(0008).
* CICS System Name
  02 MON-SPEC      PIC X(0008).
* Monitor Spec Name
  02 RTA-SPEC      PIC X(0008).
* Real Time Analysis Spec Name
  02 WLM-SPEC      PIC X(0008).
* Work Load Manager Spec Name
  02 STATUS-R      PIC X(0001).
* Status
  88 LOCAL        VALUE X'80'.
  88 REMOTE        VALUE X'40'.
* Remote MAS
  02 DYNROUTE      PIC X(0001).
* Dynamic Routing Mode
  88 ACTIVE        VALUE X'01'.
* Routing ACTIVE
  88 SUSPEND       VALUE X'02'.
* Routing SUSPENDED
  02 DYNTYPE       PIC X(0003).
* Dynamic Routing Type
  88 WLMTOR        VALUE 'TOR'.
* Routing TOR
  88 WLMAOR        VALUE 'AOR'.
* Routing AOR
  02 DESC          PIC X(0030).
* Description
  02 CSYSAPPL      PIC X(0008).
* CICS System VTAM APPLID
  02 EYU-RSV0015   PIC X(0005).
  -- RESERVED WORD --
```

注: ここで、VTAM は z/OS Communications Server です。

```
* Alignment Padding
  02 MASSTART      PIC S9(0016) USAGE BINARY.
* MAS Start STCK Value
  02 TMEZONE       PIC X(0001).
* Time Zone Offset
  02 TMEZONE       PIC X(0001).
* Time Zone
  02 EYU-RSV0019   PIC X(0002).
* Alignment Padding
  02 DAYLGHTSV     PIC S9(0008) USAGE BINARY.
* DayLight saving in effect
  02 SYSID         PIC X(0004).
* MAS System Id
  02 OPSYSREL      PIC X(0004).
* MAS Op Sys Release
  02 MVSNAME       PIC X(0004).
* MVS System Name
  02 JOBNAME       PIC X(0008).
* MAS Job Name
  02 CECNAME       PIC X(0008).
* CEC Name
  02 SYSPLEX       PIC X(0008).
* SYSPlex Name
  02 EYU-RSV0257   PIC X(0004).
```

```

* Alignment Padding
* -----*
*
* EMASSTRT Constants for Table
*
* -----*
01 EMASSTRT-TBL-LEN PIC S9(4) USAGE BINARY VALUE 152.

```

## C コピーブック

C コピーブックは、CICSTS56.CPSM.CPSM.SEYUC370 で配布されます。

### 分散:

構造化データ・タイプ

### コピーブック名:

EYUCnnnn

C コピーブックを使用する場合は、以下のことに注意してください。

- リソース・テーブルは **Typedef** ステートメントを使用して記述されます。
- 構造名は、リソース・テーブル・フォーマット名 (EMASSTRT など) です。
- 属性名は従属名として使用されます。
- ビット・インディケーターを記述する属性については、**#define** ステートメントがリソース・テーブルの末尾に生成されます。各 **#define** ステートメントによって単一のインディケーター値が指定されます。これらの定数は、その属性の割り当てや評価のために使用できます。定数名は、リソース・テーブル名、属性名、およびインディケーター名をアンダースコアで連結したものです (例: EMASSTRT\_DYNROUTE\_ACTIVE)。
- コピーブックでは、3 文字表記 (複数文字の組み合わせ) を使用して大括弧が表現されます。
- この API に送信する可変長データは、フィールドの末尾までブランクが埋め込まれている必要があります。API によってゼロバイト終了区切り文字は挿入されません。
- テーブル長フィールドは、リソース・テーブル・フォーマット名と定数 TBL\_LEN をアンダースコアで連結したものです (例: EMASSTRT\_TBL\_LEN)。

リソース・テーブル・データ・タイプは、有効な C データ・タイプ・セット内にマップされています。ただし、正確なマッピングが常に可能であるわけではありません。リソース・テーブル・データ・タイプは以下のようにマップされています。

```

char - 1-byte binary numeric values
short int - 2-byte binary numeric values
long - 4-byte binary numeric values
      - 4-byte intervals
long 2 - Time stamps and 8-byte intervals
      8- byte binary numeric values
      (an array of two fullwords)
char nnnn - Packed decimal data
char nnnn - Character data
          - Binary values greater than 8 bytes
          - Odd number binary values less than 8 bytes

```

78 ページの図 15 では、C リソース・テーブル・コピーブックをサンプルとして抽出したものを示しています。

```

/*-----*
 * Name = EYUC2400                                     *
 * Format Name = EMASSTRT                             *
 * Version = 0001                                     *
 * Status = CPSMREL(0310)                             *
 * Function = Base Table Structure generator           *
 * Format definition for this element = EMASSTRT       *
 * Valid Operations = None                             *
 * Valid Actions = None                               *
 *-----*/
typedef struct EMASSTRT {
char    CMASNAME??(8??);      /* CMAS Name                      */
char    PLEXNAME??(8??);     /* CICSplex Name                  */
char    CSYSNAME??(8??);     /* CICS System Name              */
char    MON_SPEC??(8??);     /* Monitor Spec Name             */
char    RTA_SPEC??(8??);     /* Real Time Analysis Spec Name  */
char    WLM_SPEC??(8??);     /* Work Load Manager Spec Name  */
char    STATUS;              /* Status                         */
char    DYNROUTE;            /* Dynamic Routing Mode           */
char    DYNTYPE??(3??);      /* Dynamic Routing Type           */
char    DESC??(30??);        /* Description                     */
char    CSYSAPPL??(8??);     /* CICS System VTAM APPLID       */
char    EYU_RSV0015??(5??);  /* Alignment Padding             */
long    MASSTART??(2??);     /* MAS Start STCK Value          */
char    TMEZONE0;            /* Time Zone Offset              */
char    TMEZONE;             /* Time Zone                      */
char    EYU_RSV0019??(2??);  /* Alignment Padding             */
long    DAYLGHSTV;           /* Daylight saving in effect     */
char    SYSID??(4??);        /* MAS System Id                 */
char    OPSYSREL??(4??);     /* MAS Op Sys Release            */
char    MVSNAME??(4??);      /* MVS System Name               */
char    JOBNAME??(8??);      /* MAS Job Name                   */
char    CECNAME??(8??);      /* CEC Name                      */
char    SYSPLEX??(8??);      /* SYSplex Name                  */
char    EYU_RSV0257??(4??);  /* Alignment Padding             */
} EMASSTRT;

```

注 : VTAM は現在 z/OS Communications Server になっています。

```

/*-----*
 *
 * EMASSTRT Defines for Table                         *
 *
 *-----*/
#define EMASSTRT_STATUS_LOCAL      128
#define EMASSTRT_STATUS_REMOTE    64
#define EMASSTRT_DYNROUTE_ACTIVE  1
#define EMASSTRT_DYNROUTE_SUSPEND 2
#define EMASSTRT_DYNTYPE_WLMTOR   "TOR"
#define EMASSTRT_DYNTYPE_WLMAOR   "AOR"
#define EMASSTRT_TBL_LEN          152

```

図 15. サンプルの C コピーブック

## 言語と環境に関する考慮事項

CICSplex SM API プログラムを作成する際は、言語と環境の両方を考慮する必要があります。

各種の環境 (CICS、MVS バッチ、TSO、および NetView) に当てはまる通常の言語関連の考慮事項はすべて、それらの環境で実行するために作成された CICSplex SM プログラムにも当てはまります。

## アセンブラーに関する考慮事項

MVS バッチ、TSO、または NetView の環境で実行されるアセンブラー・プログラムについては、いくつかの特別な考慮事項を認識する必要があります。

- プログラムは CICS 内では実行されないため、DFHEIENT マクロや DFHEIRET マクロを使用しないでください。代わりに、CICS 変換プログラム・オプションの NOEPILOG、NOPROLOG、および NOSYSEIB を使用します。
- DFHEISTG マクロと DFHEIEND マクロを明示的にコーディングして、EXEC CPSM コマンドに必要な作業域を提供する必要があります。プログラムでは、EXEC CPSM 呼び出しを発行する前に、DFHEISTG 領域用のストレージを確保して、必要な基底レジスターをセットアップする必要があります。ローカル GETMAIN サービスを使用してこのストレージを動的に確保できるとともに、そのプログラムが再入不可

の場合は、このストレージをプログラム領域内で直接定義することもできます。同じアドレス・スペースで同時に使用される可能性があるプログラムには、再入可能プログラムが推奨されます。

- アセンブラー・ステップの SYSLIB 連結で、適切な CICS マクロ・ライブラリーを使用可能にする必要があります。DFHEISTG、DFHEIEND、および DFHSCALL の各マクロは、このライブラリーから取り出されます。
- DFHEIPL のラベルとともに、長さが 64 のフルワードのストレージ宣言を追加します。これは、DFHSCALL マクロへの依存の結果として必要です。SEYUSAMP ライブラリーにある CICSplex SM バッチ・サンプル API プログラム EYUAAPI3 に、その例が示されています。

## PL/I に関する考慮事項

PL/I プログラムについては、変数 EYUPTPTR に関する以下の特別な考慮事項を認識する必要があります。

- 変数 EYUPTPTR を次のように明示的に宣言する必要があります。

```
DCL EYUPTPTR POINTER;
```

## NetView に関する考慮事項

C プログラムを NetView 環境で実行する予定の場合は、いくつかの特別な考慮事項を認識する必要があります。

- アクセスするリソース・テーブルによっては、リソース・テーブル属性の CICSplex SM #define ステートメントと標準の NetView #define ステートメントの間で、名前の競合が発生する可能性があります。例えば、NetView の #include "dsic.h" ステートメントによって、次の define ステートメントが生成されます。

```
#define COMMAND "COMMAND "
```

一部の CICSplex SM リソース・テーブルでは、COMMAND が属性名として使用されています。NetView 側で用意されている #include "dsic.h" をそのまま使用した場合は、リソース・テーブル属性名は変換されて、CICSplex SM によって処理できなくなります。

競合の可能性を処理する方法の 1 つは、次のように COMMAND 値を定義解除することです。

```
#include "dsic.h"
#undef COMMAND
#include "feedback.h"
.
.
.
```

任意で、どのリソース・テーブル属性名とも競合しない新しい名前を使用して、次のように COMMAND 値を再定義することもできます。

```
#include "dsic.h"
#undef COMMAND
#define XCOMMAND "COMMAND "
#include "feedback.h"
.
.
.
```

## ユーザーが置換可能なプログラム

CICSplex SM API は、ユーザー置き換え可能プログラムの EYU9XESV および EYU9WRAM 内から使用することはできません。

## CICS グローバル・ユーザー出口プログラム

CICS XICEREQ グローバル・ユーザー出口プログラム内から、CICSplex SM API を使用できます。CICSplex SM API プログラム内での再帰は回避してください。CICSplex SM 関連タスクによって発行された要求が出口によって遅延することはありません。

他の CICS グローバル・ユーザー出口点内からの CICSplex SM API の使用は、結果が予測不能になるため、推奨できません。

## 状況プログラム

CICSplex SM API は、STATDEF ビューを通じて呼び出されたプログラム内から使用することはできません。API にアクセスする必要がある場合は、別のタスクを始動して、その新しいタスクから API を呼び出す必要があります。

## プログラムの変換

個別変換は、コンパイラー (またはアセンブラー) が理解できる実行可能コードにプログラムを変換するプロセスです。

一部のコンパイラーで使用可能な**統合 CICS 変換プログラム**という手法では、コンパイル時にコンパイラーが CICS にアクセスして、CICS コマンドを解釈し、それらのコマンドを CICS サービス・ルーチンの呼び出しに自動的に変換します。統合 CICS 変換プログラムの手法を使用した場合は、変換タスクの多くが自動的に実行されます。統合 CICS 変換プログラムについて詳しくは、[統合 CICS 変換プログラム](#)を参照してください。

コマンド・レベル・インターフェースを使用して作成したプログラムについては、API のソース・プログラムを解釈するために言語変換プログラムを使用する必要があります。EXEC CPSM コマンドが含まれた外部プログラムは、適切なバージョンの CICS コマンド・レベル変換プログラムによって処理される必要があります。

CICS TS 変換プログラムは EXEC CPSM コマンドをサポートしています。ビジネス・アプリケーション・サービス (BAS) を使用して CICS リソース定義を作成する場合は、作成する定義に対応した適切なバージョンの変換プログラムを必ず使用してください。

**注:** CICS 変換プログラムで CPSM オプションの使用を許可されるユーザーを決定するには、RACF® を使用して、変換時に DFHSMTAB テーブルの読み込みを許可されるユーザーを制御します。RACF プログラム制御について詳しくは、[z/OS Security Server RACF セキュリティー管理者のガイド](#)を参照してください。DFHSMTAB は、CICSplex SM API のコマンドを定義する言語定義表です。これはオンデマンドでのみロードされます。

## CPSM 変換プログラム・オプションの指定

CICSplex SM によって CICS 変換プログラムが使用されるため、CICSplex SM API プログラムを変換するためのモデルとして CICS 変換 JCL を使用できます。

CICSplex SM プログラムを変換するためには、CPSM という追加の変換プログラム・オプションを指定する必要があります。CPSM オプションを指定するには、EXEC ステートメントの PARM オペランドを使用するか、言語固有の XOPTS オプション・ステートメントを使用します。

プログラムに EXEC CICS コマンドも含まれている場合は、これらのコマンドは同じ変換ステップで処理されます。CICS 変換プログラムによって、プログラムを適切に実行するために必要な変数定義と呼び出し定義が挿入されます。

非 CICS 環境で CICSplex SM API を使用している場合は、CICS や SP の変換プログラム・オプションを必ず削除して、CPSM 変換プログラム・オプションのみを指定してください。

変換プロセスの結果として、EXEC CPSM ステートメントが、言語固有の EXEC インターフェース・スタブ・プログラムの呼び出しに置き換えられます。

### アセンブラーの変換例

CPSM 変換プログラム・オプションを指定するには、EXEC ステートメントの PARM オペランドまたは XOPTS オペランドを使用するか、XOPTS オプション・ステートメントを使用します。

PARM を次のように使用します。

```
//TRANSLAT EXEC PGM=DFHEAP1$,PARM='CPSM',REGION=4096K
```

XOPTS の場合は次のようになります。



```
*ASM XOPTS(...CPSM)
```

### PL/I の変換例

CPSM 変換プログラム・オプションを指定するには、EXEC ステートメントの PARM オペランドを使用するか、XOPTS オプション・ステートメントを使用します。

PARM を次のように使用します。

```
//TRANSLAT EXEC PGM=DFHEPP1$,PARM='CPSM',REGION=4096K
```

XOPTS の場合は次のようになります。

```
*PROCESS XOPTS(...CPSM)
```

### COBOL の変換例

CPSM 変換プログラム・オプションを指定するには、次の 3 つの方法のいずれかを使用できます。

EXEC ステートメントの PARM オペランドを次のように使用します。

```
//TRANSLAT EXEC PGM=DFHECP1$,PARM='COBOL3,CPSM',REGION=4096K
```

または (個別の変換プログラム用に) XOPTS オプション・ステートメントを次のように使用します。

```
PROCESS XOPTS(...CPSM)
```

または (統合変換プログラム用に) CICS コンパイラー・オプションを次のように使用します。

```
CICS('opt1 opt2 optn ...')
```

COBOL プログラムを変換するときには、CPSM と COBOL3 の両方の変換プログラム・オプションを指定する必要があります。

### C の変換例

CPSM 変換プログラム・オプションを指定するには、EXEC ステートメントの PARM オペランドを使用するか、XOPTS オプション・ステートメントを使用します。

PARM を次のように使用します。

```
//TRANSLAT EXEC PGM=DFHEDP1$,PARM='CPSM',REGION=4096K
```

XOPTS の場合は次のようになります。

```
#pragma XOPTS(...CPSM)
```

## プログラムのコンパイル

CICSplex SM API プログラムのコンパイルは、CICS プログラムのコンパイルと似ています。CICS コンパイラ JCL をモデルとして使用し、使用する言語に従って、以下の変更を加えることができます。

### アセンブラに関する考慮事項

CICSplex SM プログラムをアセンブルするには、CICSTS56.CPSM.SEYUMAC マクロ・ライブラリーの SYSLIB ステートメントをコンパイル JCL に組み込む必要があります。

```
//ASM      EXEC PGM=ASMA90,REGION=4096K
           .
           .
           .
//SYSLIB   DD DSN=CICSTS56.CPSM.SEYUMAC,DISP=SHR
           .
           .
           .
```

## PL/I に関する考慮事項

CICSplex SM プログラムをコンパイルするには、CICSTS56.CPSM.SEYUPL1 マクロ・ライブラリーの SYSLIB ステートメントをコンパイル JCL に組み込みます。

```
//COMPILE EXEC PGM=IBMZPLI,REGION=1000K,  
//          PARM='OBJECT,MACRO,LIST'  
:  
:  
:  
//SYSLIB DD DSN=CICSTS56.CPSM.SEYUPL1,DISP=SHR  
:  
:  
:
```

サポートされている PL/I コンパイラー については、[アプリケーション・プログラミング言語に関する CICS サポートの変更点](#)を参照してください。

## COBOL に関する考慮事項

CICSplex SM プログラムをコンパイルするには、CICSTS56.CPSM.SEYUCOB マクロ・ライブラリーの SYSLIB ステートメントをコンパイル JCL に組み込みます。

```
//COMPILE EXEC PGM=IGYCRCTL,REGION=4096K  
:  
:  
:  
//SYSLIB DD DSN=CICSTS56.CPSM.SEYUCOB,DISP=SHR  
:  
:  
:
```

サポートされている COBOL コンパイラー については、[アプリケーション・プログラミング言語に関する CICS サポートの変更点](#)を参照してください。

## C および C++ 考慮事項

CICSplex SM プログラムをコンパイルするには、CICSTS56.CPSM.SEYUC370 マクロ・ライブラリーの SYSLIB ステートメントをコンパイル JCL に組み込みます。

```
//COMPILE EXEC PGM=EDCCOMP,REGION=4096K  
:  
:  
:  
//SYSLIB DD DSN=CICSTS56.CPSM.SEYUC370,DISP=SHR  
:  
:  
:
```

サポートされている C および C++ コンパイラーについて詳しくは、[アプリケーション・プログラミング言語に関する CICS サポートの変更点](#)を参照してください。

## プログラムのリンク・エディット

CICS 変換プログラムは、CICSplex SM EXEC インターフェース・スタブ・プログラムに呼び出しを挿入します。

スタブ入り口名は、オブジェクトやロード・モジュールの名前ではありません。CICSplex SM API プログラムはさまざまな環境で実行できるので、スタブ参照は用途と整合性のあるモジュールに解決される必要があります。この解決は、リンク・エディット時に、INCLUDE リンケージ・エディター制御ステートメントを使用して実行されます。

すべてのプログラム・ロード・モジュールを、そのプログラムを実行する環境にとって適切な CICSplex SM スタブ・モジュールでリンク・エディットしなければなりません。そのためには、以下のいずれかのスタブ・モジュールを INCLUDE ステートメントで指定します。

### EYU9AMSI

CICS TS プログラムの場合。EYU9AMSI は、CICSTS56.CPSM.SEYULOAD ライブラリーにあります。

## EYU9ABSI

バッチ、TSO、または NetView プログラムの場合。EYU9ABSI は、CICSTS56.CPSM.SEYUAUTH ライブラリーにあります。

これらの各スタブ・モジュールには、該当する入り口点 ID があります。入り口点によって提供されるサービスは、実行環境のタイプごとに固有です。リンク・エディット・ステップ中に適切なスタブ・モジュールと SYSLIB データ・セットを組み込むことができなかった場合は、シンボル EYUEI1 が未解決であることを報告するメッセージが表示されます。これは、後続の IEW2456E SYMBOL EYUEI1 UNRESOLVED と似ています。

**注：**CICS プログラムとして識別されるプログラムをバッチ環境で実行しようとししないでください。同様に、バッチ・プログラムは CICS での実行に適していません。

CICS リンク・エディット JCL を、CICSplex SM プログラムのリンク・エディットのモデルとして使用できます。このセクションの残りの部分にある言語固有の考慮事項を検討し、その情報に従って JCL を変更してください。

また、プログラムに EXEC CICS コマンドが含まれている場合は、リンク・エディットに関する考慮事項を確認する必要があります。同様に、ご使用のプログラムが NetView で実行されている場合は、ご使用のプログラミング言語に応じて「[IBM Tivoli NetView for z/OS プログラミング: アセンブラー](#)」または「[IBM Tivoli NetView for z/OS プログラミング: PL/I および C](#)」のいずれかの NetView 情報を参照してください。

## アセンブラーに関する考慮事項

アセンブラーのロード・モジュールは、24 ビットまたは 31 ビットのストレージに配置することができ、いずれかのアドレッシング・モードで開始できます。

CICSplex SM プログラムで実行するためのアセンブラー・モジュールをリンク・エディットするには、SEYULOAD ロード・ライブラリーの SYSLIB ステートメントをリンク・エディット・ステップに組み込む必要があります。これにより、リンク・エディット時に適切な CICSplex SM スタブ・モジュールを組み込むことが可能になります。例えば、次のとおりです。

```
//LKED      EXEC  PGM IEWL,
//          PARM='XREF,LET,LIST,AMODE=ANY,RMODE=31',
//          REGION=4096K,COND=(7,LT,ASM)
.
.
.
//SYSLIB    DD  DSN=CICSTS56.CPSM.SEYULOAD,DISP=SHR
.
.
.
INCLUDE     SYSLIB(userprog)
INCLUDE     SYSLIB(EYU9AMSI)
NAME        LMODNAME(R)
```

## PL/I、COBOL、および C に関する考慮事項

PL/I、COBOL、および C のロード・モジュールは、24 ビットまたは 31 ビットのストレージに配置することができ、いずれかのアドレッシング・モードで開始できます。

CICSplex SM プログラムで実行するためのモジュールをリンク・エディットするには、SEYULOAD ロード・ライブラリーの SYSLIB ステートメントをリンク・エディット・ステップに組み込む必要があります。これにより、リンク・エディット時に適切な CICSplex SM スタブ・モジュールを組み込むことが可能になります。例えば、次のとおりです。

```
//LKED      EXEC  PGM=IEWL,
//          PARM='XREF,LET,LIST,AMODE=ANY,RMODE=31',
//          REGION=4096K,COND=(8,LE,COMPILE)
.
.
.
//SYSLIB    DD  DSN=CICSTS56.CPSM.SEYULOAD,DISP=SHR
.
.
.
INCLUDE     SYSLIB(userprog)
INCLUDE     SYSLIB(EYU9AMSI)
NAME        LMODNAME(R)
```

## ランタイムの考慮事項

以下に示すいくつかのランタイム関連の考慮事項を認識する必要があります。

- CICSplex SM API プログラムのランタイム・バージョンは、そのプログラムが接続する CMAS のレベルと等しいバージョンになります。
  - CICS アプリケーションとして実行するために作成されたプログラムの場合は、ランタイム・バージョンは MAS の接続先である CMAS のバージョンです。
  - バッチ・ジョブとして実行するために、または NetView や TSO で実行するために作成されたプログラムの場合は、ランタイム・バージョンは CICSplex SM ランタイム・モジュール (EYU9AB00) のバージョンによって決まります。
- EYU9AB00 は CICSTS56.CPSM.SEYUAUTH で配布されます。実行時に CICSplex SM は、STEPLIB 内、MVS リンク・リスト内、または LPA ライブラリー連結内で EYU9AB00 を検出する必要があります。
- プログラムのランタイム・バージョンは、次に示すバージョン以上でなければなりません。
  - そのプログラムをリンク・エディットするために使用されたスタブ・ルーチン・モジュール (EYU9AxSI) のバージョン。
  - CONNECT コマンドの VERSION オプションで指定された値。
- PL/I、COBOL、または C で作成されたプログラムの場合は、一連のランタイム・ライブラリーが言語コンパイラに付属しています。これらのいずれかの言語で作成された CICSplex SM プログラムを実行するには、その言語の適切なランタイム・ライブラリーを参照するように現在の環境の始動プロシージャを変更する必要があります。
- いずれかの CICSplex SM プログラムを CICS で実行する前に、そのプログラムと関連トランザクションが CEDA に対して定義されていることを確認してください。このプログラムは、EXECKEY の値として User または CICS を指定して定義できます。関連するトランザクションは、TASKDATAKEY の値として User または CICS を指定して定義できます。

## 第 4 章 例外条件の取り扱い

CICSplex SM API プログラムのエラー状態の取り扱いには、いくつかのツールと手法を使用できます。

### デフォルトの CICSplex SM 例外処理

CICSplex SM API によって、ユーザー・トレース・レコードの形式で CICS トレース・データ・セットに例外トレースが書き込まれます。

CICSplex SM API を介して使用可能なリソースはリカバリーできません。そのため、例外の前に更新されたリソースは、リカバリーされることはなく、EXEC CICS SYNCPOINT および EXEC CICS SYNCPOINT ROLLBACK の各コマンドを使用したアプリケーションによるバックアウトに使用することもできません。

### RESPONSE オプションと REASON オプションの使用

各 API コマンドには、RESPONSE オプションと REASON オプションが必要です。

これらのオプションをユーザー定義の変数として指定して、コマンドによって戻される応答と理由の数値を受け取る必要があります。その後、この数値を、より意味のある同等の文字に変換できます。通常、RESPONSE はコマンド処理の結果を説明し、REASON は応答を特定のコマンドにさらに限定しています。

注: REXX ランタイム・インターフェースでのみ使用可能な TBUILD コマンドと TPARSE コマンドでは、RESPONSE オプションと REASON オプションは使用されません。これらの REXX 固有の処理の結果は、その STATUS オプションによって戻されます。詳しくは、[105 ページの『第 6 章 REXX エラー処理』](#)を参照してください。

### 応答のタイプ

API コマンドでは、通常、警告、またはエラーの応答コードが戻されます。

戻される RESPONSE と REASON の値の同等の文字については、各コマンドの説明で示します。コマンドごとの RESPONSE と REASON の文字値の概要については、[CICSplex SM API コマンド](#)を参照してください。また、RESPONSE と REASON の文字値とその同等の数値のリストについては、[RESPONSE 値および REASON 値](#)を参照してください。

#### 通常応答

通常応答は、API コマンドの処理が正常に完了したことを示します。

通常応答は、API コマンドの処理が正常に完了したことを示します。次の値は、通常応答を表します。

#### OK

コマンドが正常に処理されて、制御がプログラムに戻されました。OK の応答には、理由は関連付けられていません。

#### SCHEDULED

NOWAIT オプションを指定して発行されたコマンドの処理がスケジュールされました。コマンド処理の実際の結果は、RECEIVE コマンドによって、ASYNCREQ リソース・テーブル・レコードに戻されます。SCHEDULED の応答には、理由は関連付けられていません。

#### 警告応答

警告応答は、API コマンドは正常に処理されたが、調査が必要な状態が発生したことを示します。

状態を説明する REASON 値も戻されます。次の値は、警告応答を表します。

#### NODATA

通常ではプログラムにデータを戻すコマンドについて、処理は正常に行われましたが、戻されるデータがありませんでした。NODATA 応答の理由が、それを戻したコマンドとともに示されます。

#### WARNING

通常ではプログラムにデータを戻すコマンドについて、処理は正常に行われましたが、使用可能なデータの一部が戻されませんでした。プログラムによって提供される出力域のサイズが不十分で、一部のデ

ータが保持されなかったことが、この応答の典型的な理由です。WARNING 応答の実際の理由が、それを戻したコマンドとともに示されます。

### **エラー応答**

エラー応答は、API コマンドが成功しなかったことを示します。エラーを示す、1 つ以上の REASON の値も戻されます。

**注:** FAILED のエラー応答を除いて、これらの応答コードは、通常、ユーザーの API プログラムのエラー (例えば、不要になったリソースの破棄の失敗) または CICSplex SM 環境でのエラー (例えば、CMAS または MAS が使用できない) を示します。

次の値は、エラー応答を表します。

#### **BUSY**

コマンドによって参照されるリソースが、現在、別のコマンドによって処理されています。この状況は、以前に NOWAIT オプションを指定して発行されたコマンドが、現在のコマンドで必要になるリソースを処理しているときに発生することがあります。BUSY 応答の理由が、それを戻したコマンドとともに示されます。

#### **DUPE**

コマンドによって参照されるリソースが既に存在します。DUPE 応答の理由が、それを戻したコマンドとともに示されます。

#### **ENVIRONERROR**

環境条件 (ストレージ不足など) により、コマンドが処理されませんでした。ENVIRONERROR 応答の理由が、それを戻したコマンドとともに示されます。

#### **FAILED**

コマンドの処理中に予期しない問題が発生しました。FAILED 応答の理由が、それを戻したコマンドとともに示されます。

FAILED EXCEPTION 応答の場合は、状態に関連する情報について、次のソースを確認する必要があります。

- EYULOG
- ジョブ・ログ
- AUXTRACE データ・セット

#### **INCOMPATIBLE**

コマンドによって参照される複数のリソースに互換性がありません。INCOMPATIBLE 応答の理由が、それを戻したコマンドとともに示されます。

#### **INUSE**

コマンドによって参照されるリソースが使用中であるため、破棄できません。INUSE 応答の理由が、それを戻したコマンドとともに示されます。

#### **INVALIDATA**

コマンド・パラメーター・リストに無効なデータが含まれています。INVALIDATA 応答の理由は常に、無効なデータが含まれているパラメーターの名前になります。この応答を戻したコマンドとともに理由が示されます。

#### **INVALIDCMD**

理由コードで示されたとおりに、コマンドが無効です。

##### **Filter**

構築しているフィルターが大きすぎるか、複雑すぎます。

##### **長さ**

コマンドで使用されるすべての入力 of 全長が最大制限を超えています。

##### **N\_A**

コマンドが無効です。API コマンドの変換に使用された CICS 変換プログラムのバージョンを確認してください。使用するコマンドが、プログラムで使用している CICSplex SM リリースで使用可能であることも確認してください。

## INVALIDPARM

コマンド・パラメーター・リストが無効です。INVALIDPARM 応答は、さまざまな状況で発生します。例えば、次のとおりです。

### 構文エラー

入力パラメーターの構文が正しくありません (例えば、リソース・テーブル名が数字で始まるなど)。

### ヌル・パラメーター・アドレス

入力パラメーターに対して生成されたアドレスが 0 であるため、このパラメーターが見つかりませんでした。

INVALIDPARM 応答の理由は常に、無効なパラメーターの名前になります。この応答を戻したコマンドとともに理由が示されます。

## NOTAVAILABLE

必要な CMAS リソースまたは MAS リソースを使用できません。NOTAVAILABLE 応答の理由が、それを戻したコマンドとともに示されます。

## NOTFOUND

コマンドによって参照されるリソースが見つかりませんでした。NOTFOUND 応答の理由が、それを戻したコマンドとともに示されます。

## NOTPERMIT

API 要求が、企業の外部セキュリティ・マネージャー (ESM) によって許可されていません。NOTPERMIT 応答の理由が、それを戻したコマンドとともに示されます。

## SERVERGONE

処理スレッドが接続されていた CMAS がアクティブではなくなりました。SERVERGONE の応答には、理由は関連付けられていません。

## TABLEERROR

リソース・テーブル・レコード (結果セット・レコードまたは CICSplex SM 定義レコード) でエラーが検出されました。TABLEERROR 応答の理由が、それを戻したコマンドとともに示されます。

## VERSIONINVL

無効なバージョンの CICSplex SM が検出されました。VERSIONINVL 応答の理由が、それを戻したコマンドとともに示されます。

## RESPONSE と REASON のテスト

API コマンドの結果を評価するには、コマンドで RESPONSE オプションおよび REASON オプションをコーディングし、コマンドの直後に戻り値のテストを続けます。

RESPONSE オプションおよび REASON オプションによって、数値が戻されます。コマンド・レベル・インターフェースと REXX ランタイム・インターフェースでは、応答と理由の数値を変換およびテストするための、さまざまな組み込み関数が提供されています。

### コマンド・レベル・インターフェースの使用

CICSplex SM コマンド・レベル・インターフェースを使用している場合は、EYUVALUE 組み込み関数を使用して、API コマンドによって戻される RESPONSE と REASON の数値を変換およびテストできます。

例として、次のような API コマンドが考えられます。

```
EXEC CPSM CONNECT
      CONTEXT(WCONTEXT)
      SCOPE(WSCOPE)
      VERSION('0310')
      THREAD(WTHREAD)
      RESPONSE(WRESPONSE)
      REASON(WREASON)
      .
      .
```

それぞれのサポート言語で RESPONSE 値をテストする場合は、次のようにコーディングすることができます。

## COBOL または PL/I:

```
IF WRESPONSE NOT = EYUVALUE(OK) GO TO NOCONNECT.
```

## C:

```
if (WRESPONSE != EYUVALUE(OK)) { goto NOCONNECT; }
```

## アセンブラー言語:

```
CLC    WRESPONSE,EYUVALUE(OK)
BNE    NOCONNECT
```

これを組み込み関数が次のように変更します。

```
CLC    WRESPONSE,=F'1024'
```

RESPONSE により理由が戻される場合も、同じ方法で EYUVALUE を使用して、REASON 値をテストできます。

## REXX ランタイム・インターフェースの使用

REXX ランタイム・インターフェースを使用している場合は、EYURESP と EYUREAS の組み込み関数を使用して、API コマンドによって戻される RESPONSE と REASON の数値を変換およびテストできます。

例として、次のような API コマンドが考えられます。

```
var = EYUAPI('CONNECT'
             'CONTEXT('WCONTEXT')' ,
             'SCOPE('WScope')' ,
             'VERSION(0310)' ,
             'THREAD(WTHREAD)' ,
             'RESPONSE(WRESPONSE)' ,
             'REASON(WREASON)')
:
```

RESPONSE 値をテストする場合は、次のようにコーディングすることができます。

```
If WRESPONSE <> EYURESP(OK) Then Signal NOCONNECT
```

この場合、WRESPONSE で戻される RESPONSE の数値と OK と同等の数値が比較されます。

あるいは、次のようにコーディングすることができます。

```
If EYURESP(WRESPONSE) <> "OK" Then Signal NOCONNECT
```

この場合、まず RESPONSE の数値が同等の文字に変換されます。

**注:** RESPONSE オプションと REASON オプションでは、ランタイム・エラーのみが報告されます。API コマンドの解釈におけるエラーは、REXX RC 変数か、REXX 関数に割り当てられた変数で報告されます。

## FEEDBACK レコードの取り出し

CICSplex SM は、コマンドの RESPONSE および REASON オプションによって返される特定の値だけでなく、FEEDBACK リソース・テーブル・レコード形式の診断データも提供します。このデータは、特にコマンドが正常に完了しなかった場合に、API コマンドの結果を評価する上で役立ちます。

## FEEDBACK コマンドの使用

FEEDBACK コマンドを発行して、以前に発行された API コマンドに関する診断データを取得できます。

診断データを必要とするコマンドのタイプは、FEEDBACK コマンドの指定方法と、データの配置場所に影響を与えます。

### 結果セットを処理したコマンド

FEEDBACK コマンドの RESULT オプションを使用して、特定の結果セットを処理した前回のコマンドに関するデータを取得します。



結果セットを処理したコマンドで OK 以外の RESPONSE が戻された場合は、それに関連するエラーが発生した結果セットの各リソース・テーブル・レコードの最後に、FEEDBACK リソース・テーブル・レコードが追加されます。FEEDBACK コマンドの FIRST、NEXT、および COUNT の各オプションを使用すると、複数の FEEDBACK レコードを取得できます。

FEEDBACK コマンドで結果セットの診断データを取得できるのは、別のコマンドで同じ結果セットが処理されるまでです。その時点で、後続のコマンドの FEEDBACK レコードによってデータが置換されます。

**注：**結果セットを処理したコマンドで OK の RESPONSE が戻された場合、FEEDBACK レコードは生成されません。

#### 結果セットを処理しなかったコマンド

RESULT オプションを指定せずに FEEDBACK コマンドを使用して、FEEDBACK の直前に発行されたコマンドに関するデータを取得します。

FEEDBACK リソース・テーブル・レコードは、個別のフィードバック領域に戻されます。結果セット指向でないコマンドが実行されるたびに、このフィードバック領域のレコードがクリアされてリフレッシュされます。そのため、結果セットではなく、フィードバック領域に診断データを配置するコマンドの場合、FEEDBACK で取得できるのは、最後に発行されたコマンドのデータのみです。

コマンドの診断データを取得するために FEEDBACK コマンドを発行すると、フィードバック・レコードまたはフィードバック領域がクリアされます。同じ FEEDBACK リソース・テーブル・レコードを複数回要求することはできません。

## FEEDBACK レコードの評価

CICSplex SM API コマンドの診断データは、FEEDBACK リソース・テーブル・レコードに示されます。該当のリソース・テーブルの属性によって、API コマンドの完了状況に関するさまざまな情報が提供されます。

**注：**このセクションでは、FEEDBACK レコードに関する一般情報を示します。CICSplex SM によって提供される FEEDBACK リソース・テーブルのコピーブックでは、FEEDBACK レコードの内容と構造が詳細に記述されています。FEEDBACK コマンドを使用するプログラムを作成する場合は、[CICSplex SM リソース・テーブル](#)または提供されたコピーブックを参照してください。

FEEDBACK レコードが適用される API 操作を特定するには、次のフィールドの値を確認します。

#### COMMAND

この FEEDBACK レコードが適用されるコマンドを特定する数字コード。API コマンドとその同等の数値を [89 ページの表 7](#) に示します。

表 7. 数字コードと API コマンド		
数字コード	ニーモニック	コマンド
02	CANCEL	Cancel
03	CONNECT	Connect
04	COPY	Copy
05	CREATE	Create
06	DELETE	Delete
07	DISCARD	Discard
08	DISCONN	Disconnect
09	FETCH	Fetch
10	GET	Get
11	LOCATE	Locate
12	MARK	Mark

表 7. 数字コードと API コマンド (続き)		
数字コード	ニーモニック	コマンド
13	ORDER	Order
14	PERFSET	Perform Set
15	PERFOBJ	Perform Object
16	QUALIFY	Qualify
17	QUERY	Query
18	RECEIVE	Receive
19	REMOVE	Remove
20	FILTER	Specify Filter
21	UNMARK	Unmark
22	ADDRESS	Address
23	GETDEF	Getdef
24	LISTEN	Listen
25	REFRESH	Refresh
26	SET	Set
27	VIEW	Specify View
28	TERM	Terminate
29	TRANS	Translate
30	GROUP	Group by
31	UPDATE	Update
32	FEEDBACK	Feedback
33	EXPAND	Expand

## OBJECT

コマンドの発行対象の CICSplex SM オブジェクト。

## OBJECT\_ACT

CICSplex SM オブジェクトに対して実行されたアクション。

## RSLTRECID

FEEDBACK レコードが結果セットに適用される場合の、この FEEDBACK レコードに関連付けられた結果セット・レコードの数値 ID。

FEEDBACK レコードに示される問題のタイプを特定するには、次のフィールドの値を確認します。

## ATTRDATAVAL

コマンドに対して属性データが使用可能かどうかを示します。属性データは、コマンド自体が正常に完了しなかった場合にのみ含まれます。

ATTRDATAVAL 値が Y の場合は、FEEDBACK レコードに、エラーの原因となった 5 つの属性 (ATTR\_NM1 から ATTR\_NM5) が示されます。各属性は、リソース・テーブル・レコード内で、その名前とオフセット番号および相対番号によって特定されます。各属性のデータ型と長さも含まれます。

ATTRDATAVAL 値が N の場合は、ATTR\_ フィールドを無視することができます。

## CEIBDATAVAL

コマンドに対して CICS EIB データが使用可能かどうかを示します。EIB データは、コマンドで CICS エラーが発生した場合にのみ含まれます。

CEIBDATAVAL 値が Y の場合は、FEEDBACK レコードに、CICS によって指定される EIBFN、RESP、および RESP2 の値が含まれます。RESP 値が、CICSplex SM シミュレート・セキュリティーが原因で発生した NOTAUTH 条件を示している場合、EIBFN は設定されていません。

CEIBDATAVAL 値が N の場合は、CEIBFN、CEIBRESP、および CEIBRESP1 の各フィールドを無視することができます。

## ERRCODEVAL

コマンドに対して CICSplex SM エラー・コードが使用可能かどうかを示します。エラー・コードは、コマンド自体が正常に完了しなかった場合にのみ含まれます。

ERRCODEVAL 値が Y の場合は、FEEDBACK レコードに、ERROR\_CODE の数値が含まれます。リソース・テーブルのコピーブックには、それぞれそのオブジェクトのエラー・コードとその意味のリストが含まれます。

ERRCODEVAL 値が N の場合は、ERROR\_CODE、RESPONSE、および REASON の各フィールドを無視することができます。

BAS リソースに影響を及ぼす一部の API 操作では、FEEDBACK レコードが、エラー結果セットの追加の診断データを指している場合があります。エラー結果セットの診断データの使用について詳しくは、[92 ページの『BAS の追加処理』](#)を参照してください。

## FEEDBACK レコードを使用できるかどうか

通常、FEEDBACK レコードは、すべての API コマンドに対して、コマンドが成功したかどうかにかかわらず生成されます。ただし、一部の API コマンドおよび特定の状況では、有用な診断データが提供されないため、FEEDBACK レコードが生成されません。

次のコマンドでは、FEEDBACK レコードは使用できません。

### DISCONNECT および TERMINATE

CICSplex SM から API 処理スレッドを切断した場合は、残りの診断データがすべて破棄されます。

### FEEDBACK

FEEDBACK コマンドは、それ自体の処理について報告できません。

### TBUILD および TPARSE

これらの REXX 固有のコマンドでは、一連の API コマンドが内部的に発行されて、同じフィードバック領域が再利用されます。そのため、フィードバック領域で一連のイベントをすべて表すことはできません。

次の状況でも、FEEDBACK レコードは使用できません。

- コマンドによって結果セットが処理されて、OK の RESPONSE 値で完了し、CICS によって EIBRESP2 フィールドに追加情報が戻されなかった。
- コマンドが非同期で処理された (つまり、NOWAIT オプションを指定した)。非同期要求の診断データは、ASYNCREQ 通知リソース・テーブルに戻されます。

# 結果セットの FEEDBACK の例

FEEDBACK データを使用する方法の例として、SET コマンドの発行結果を以下に示します。このケースでは、TOKENC で参照される結果セットで CONNECT レコードのサービス状況を変更する、SET が発行されました。

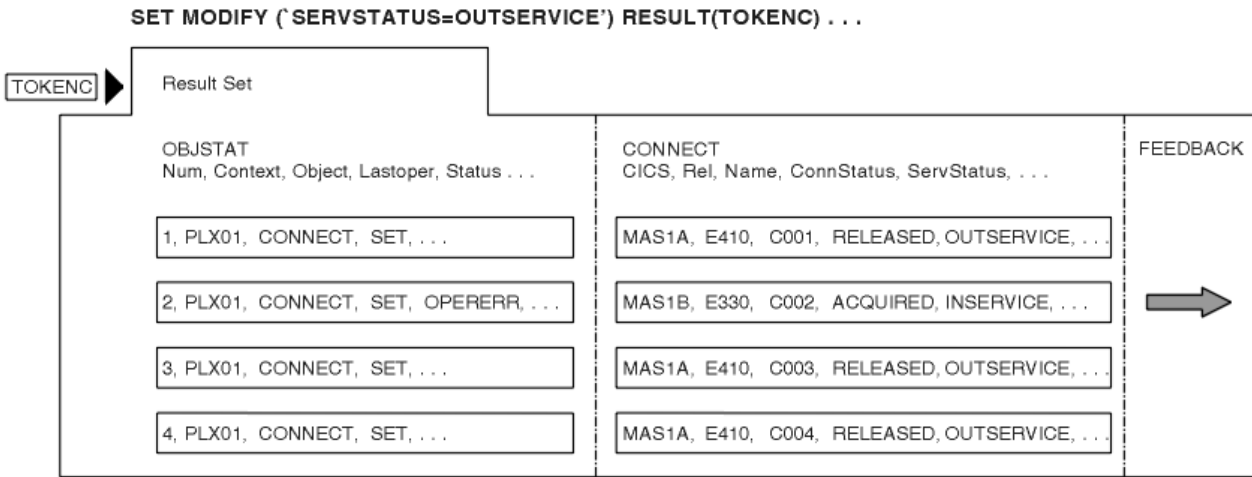


図 16. SET を使用した結果セット・レコードの変更

接続の 1 つ (MAS1B の C002) が SET コマンドで正常にサービス休止状態になりませんでした。ServStatus フィールドは依然として INSERVICE に設定されており、FEEDBACK データへのポインターが存在します。

92 ページの図 17 は、FEEDBACK コマンドを使用して、TOKENC で参照される結果セットに関連付けられた FEEDBACK レコードを取得する方法を示しています。

FEEDBACK RESULT(TOKENC) INTO(AREA5) ...



図 17. FEEDBACK を使用した結果セットの診断データの取得

92 ページの図 17 に示す FEEDBACK レコードから、問題の原因がわかります。CICSplex SM から、TABLEERROR DATAERROR の RESPONSE と REASON の値が戻されました。これは、1 つ以上のリソース・テーブル属性に関連付けられた値が無効であることを意味します。さらに、CICS は、この接続の SET 要求に RESP(16) RESP2(2) で応答しました。CICS 応答コードのチェックにより、現在接続を獲得できないため、接続をサービス休止状態にする試みが無効だったことが示されます。

注: OBJSTAT リソース・テーブルの LASTOPER 属性と STATUS 属性、および一部の FEEDBACK 属性は、2 進数フィールドです (つまり、これらはオンまたはオフに設定されたビットで表されます)。特定のリソース・テーブルの属性値について詳しくは、CICSplex SM リソース・テーブル または提供されたコピーブックを参照してください。

## BAS の追加処理

BAS リソースに影響を及ぼす API 操作の場合は、FEEDBACK レコードの診断データが十分ではなく、エラー状態が完全に説明されない場合があります。

これらのケースでは、FEEDBACK レコードはエラー結果セットを指しています。エラー結果セットは、次のフィールドによって特定されます。

### ERR\_RESULT

エラー結果セットを識別する 4 バイトのトークン。

### ERR\_COUNT

ERR\_RESULT によって参照されるエラー結果セットのレコード数。

## ERR\_OBJECT

ERR\_RESULT によって参照されるエラー結果セットのレコードのタイプ。この値は、CICSplex SM リソース・テーブルの 1 から 8 文字の名前で、BINSTERR、BINCONRS、BINCONSC、または FEEDBACK になります。

注: BINSTERR、BINCONRS、および BINCONSC の各リソース・テーブルについて詳しくは、[CICSplex SM リソース・テーブル](#)を参照してください。

## エラー結果セット・レコードの評価

FEEDBACK レコードの ERR\_OBJECT フィールドに FEEDBACK が含まれている場合は、エラー結果セットに、CICSplex SM が CICS リソースを更新しようとしたときに発生したエラーが含まれます。

以下の API コマンドがあるとします。

```
UPDATE RESULT(token) MODIFY(string)
```

CICSplex SM は、指定した変更ストリングに従って、結果セットの複数の CICS 定義レコードの更新を試みます。変更できなかった CICS 定義ごとに、エラー結果セットにエラー・レコードが作成されます。返される RESPONSE 値と REASON 値は、TABLEERROR および DATAERROR です。

レコードは標準の FEEDBACK レコードです。エラー結果レコードにアクセスするには、FEEDBACK コマンドを使用して ERR\_RESULT 結果セットの各 CICS 定義に関する診断データを取得します。UPDATE コマンドの元の FEEDBACK レコード内の ERR\_COUNT 値は、ERR\_RESULT 結果セット内にあるレコードの数を示します。つまり、ERR\_RESULT 結果セットに対して FEEDBACK コマンドを発行する必要がある回数ということになります。

## BINSTERR リソース・テーブルのレコードの評価

FEEDBACK レコードの ERR\_OBJECT フィールドに BINSTERR が含まれる場合、それは、CICS リソースのインストール中にエラーが検出されたことを示しています。

次のような API コマンドがあるとします。

```
PERFORM OBJECT ACTION(INSTALL)  
PERFORM SET ACTION(INSTALL)
```

このいずれかに応答して、CICSplex SM は、1 つ以上のアクティブ CICS 領域での CICS リソースのインストールを試みます。インストールできない CICS リソースごとに BINSTERR レコードが作成されます。返される RESPONSE 値と REASON 値は、TABLEERROR および DATAERROR です。

受け取る BINSTERR レコードには、次の情報が含まれます。

### CMASNAME

指定された CICSplex を管理する CMAS の 1 文字から 8 文字の名前。

### PLEXNAME

指定された CICS システムが属する CICSplex の 1 文字から 8 文字の名前。

### CICSNAME

リソースをインストールできなかった CICS システムの 1 文字から 8 文字の名前。

### RESNAME

インストールできなかった CICS リソースの名前。

### RESVER

インストールされるリソースを表す CICS 定義のバージョン。

### ERRCODE

数値の CICSplex SM エラー・コード。[BINSTERR](#)を参照してください。BINSTERR リソース・テーブルのコピーブックには、エラー・コードとその意味のリストも含まれます。

### CRESP1

CICS によって返される RESP 値。

### CRESP2

CICS によって返される RESP2 値。

## CEIBFN

CICS によって返される EIBFN 値。

エラー結果セット・レコードにアクセスするには、FETCH コマンドを使用して ERR\_RESULT 結果セットから BINSTERR レコードを取り出します。PERFORM コマンドの FEEDBACK レコード内の ERR\_COUNT 値は、ERR\_RESULT 結果セット内にあるレコードの数を示します。つまり、ERR\_RESULT 結果セットに対して FETCH コマンドを発行する必要がある回数ということになります。

## BINCONRS リソース・テーブル・レコードの評価

FEEDBACK レコードの ERR\_OBJECT フィールドに BINCONRS が含まれている場合、それは、指定された定義の更新または作成を試みたときに不整合なリソース・セット・エラーが検出されたことを示しています。

次のような API コマンドがあるとします。

```
CREATE OBJECT(basdef)
UPDATE OBJECT(basdef)
```

このいずれかに応答して、CICSplex SM は、次のビジネス・アプリケーション・サービス定義のいずれかの作成または更新を試みます。

- RASGNDEF (リソース割り当て)
- RASINDSC (リソース記述のリソース割り当て)
- RESDESC (リソース記述)
- RESGROUP (リソース・グループ)
- RESINDSC (リソース記述のリソース・グループ)

BINCONRS リソース・テーブル・レコードは、不整合セット・エラーの原因となる CICS 定義ごとに作成されます。返される RESPONSE 値と REASON 値は、TABLEERROR および DATAERROR です。

受け取る BINCONRS レコードには、次の情報が含まれます。

### CMASNAME

指定された CICSplex を管理する CMAS の 1 文字から 8 文字の名前。

### PLEXNAME

指定された CICS システムが属する CICSplex の 1 文字から 8 文字の名前。

### CICSNAME

不整合リソース・セット・エラーが発生した CICS システムの 1 文字から 8 文字の名前。

### RESTYPE

CICS リソースのタイプ。

### ERROP

エラーが発生したときに実行される操作 (RASGNDEF の更新など) を識別する数値。BINCONRS を参照してください。BINCONRS リソース・テーブルのコピーブックには、ERROP 値とその意味のリストも含まれます。

### CANDNAME

候補リソースの名前

### CANDVER

候補リソースのバージョン

### CANDGRP

候補リソースのグループ

### CANDRASG

候補リソースの割り当て

### CANDRDSC

候補リソースの記述

### CANDUSAGE

候補の割り当て使用法

**CANDSGRP**

候補のシステム・グループ

**CANDTYPE**

候補のシステム・タイプ

**CANDASGOVR**

候補の割り当て指定変更

**EXISTNAME**

既存のリソースの名前

**EXISTVER**

既存のリソースのバージョン

**EXISTTRGRP**

既存のリソースのグループ

**EXISTRASG**

既存のリソースの割り当て

**EXISTRDSC**

既存のリソースの記述

**EXISTUSAGE**

既存の割り当て使用法

**EXISTSGRP**

既存のシステム・グループ

**EXISTTYPE**

既存のシステム・タイプ

**EXISTASGOVR**

既存の割り当て指定変更

エラー結果レコードにアクセスするには、FETCH コマンドを使用して ERR\_RESULT 結果セットから BINCONRS レコードを取り出します。CREATE または UPDATE コマンドの FEEDBACK レコード内の ERR\_COUNT 値は、ERR\_RESULT 結果セット内にあるレコードの数を示します。つまり、ERR\_RESULT 結果セットに対して FETCH コマンドを発行する必要がある回数ということになります。

**BINCONSC リソース・テーブル・レコードの評価**

ERR\_OBJECT フィールドに BINCONSC が含まれている場合、それは、指定された定義の更新または作成を試みたときにスコープ不整合エラーが検出されたことを示しています。

ERR\_OBJECT フィールドに BINCONSC が含まれている場合、それは、指定された定義の更新または作成を試みたときにスコープ不整合エラーが検出されたことを示しています。次のような API コマンドがあります。

```
CREATE OBJECT(basdef)
UPDATE OBJECT(basdef)
```

返される RESPONSE 値と REASON 値は、TABLEERROR および DATAERROR です。

BINCONSC レコードには、次の情報が含まれます。

**CMASNAME**

指定された CICSplex を管理する CMAS の 1 文字から 8 文字の名前。

**PLEXNAME**

指定された CICS システムが属する CICSplex の 1 文字から 8 文字の名前。

**CICSNAME**

スコープ不整合エラーが発生した CICS システムの 1 文字から 8 文字の名前。

**ERROP**

エラーが発生したときに実行される操作 (RASGNDEF の更新など) を識別する数値。BINCONSC を参照してください。BINCONSC リソース・テーブルのコピーブックには、ERROP 値とその意味のリストも含まれます。



## ERRCODE

数値の CICSplex SM エラー・コード。BINCONSC を参照してください。BINCONSC リソース・テーブルのコピーブックには、エラー・コードとその意味のリストが含まれます。

## TARGSCOPE

ターゲット・スコープの名前

## TARGRASG

ターゲット・スコープの割り当て

## TARGRDSC

ターゲットの記述

## RELSCOPE

関連スコープの名前

## RELRASG

関連スコープの割り当て

## RELRDSC

関連スコープの記述

エラー結果レコードにアクセスするには、FETCH コマンドを使用して ERR\_RESULT 結果セットから BINCONSC レコードを取り出します。CREATE または UPDATE コマンドの FEEDBACK レコード内の ERR\_COUNT 値は、ERR\_RESULT 結果セット内にあるレコードの数を示します。つまり、ERR\_RESULT 結果セットに対して FETCH コマンドを発行する必要がある回数ということになります。

## BAS エラー結果セットの例

FEEDBACK データで BAS エラー結果セット情報を取得する方法の例として、PERFORM OBJECT コマンドの発行結果を以下に示します。このケースでは、TOKENC で参照される結果セットに CONNDEF 定義をインストールする、PERFORM OBJECT ACTION(INSTALL) が発行されました。

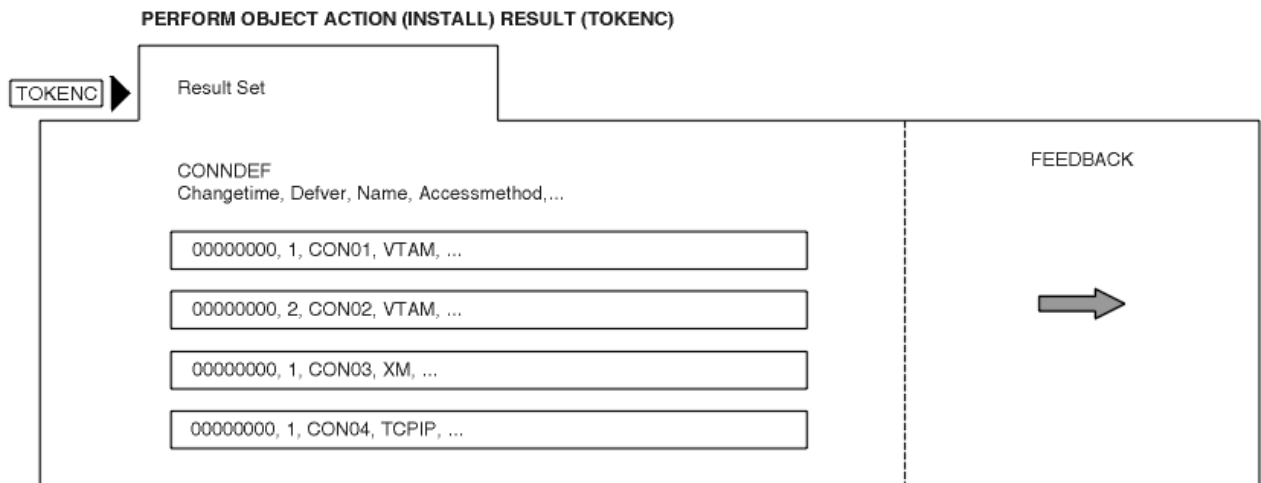


図 18. PERFORM OBJECT を使用した BAS 定義のインストール

接続定義の 1 つ (CON02、z/OS Communications Server) が PERFORM OBJECT コマンドで正常にインストールされませんでした。FEEDBACK データへのポインターが存在します。

96 ページの図 19 は、FEEDBACK コマンドを使用して、TOKENC で参照される結果セットに関連付けられた FEEDBACK レコードを取得する方法を示しています。

## FEEDBACK RESULT (TOKENC) INTO (AREA5) ...

```
PERFORM OBJECT, N, Y, N, TABLEERROR, DATAERROR, ..., 1, BINSTERR
```

図 19. FEEDBACK を使用した結果セットの診断データの取得



96 ページの図 19 に示す FEEDBACK データから、問題の原因がわかります。CICSplex SM から、TABLEERROR DATAERROR の RESPONSE と REASON の値が戻されました。これは、1 つ以上の接続定義が正常にインストールされなかったことを意味します。さらに、ERR\_RESULT 属性は、単一の BINSTERR リソース・テーブル・レコードを含むエラー結果セットを指しています。

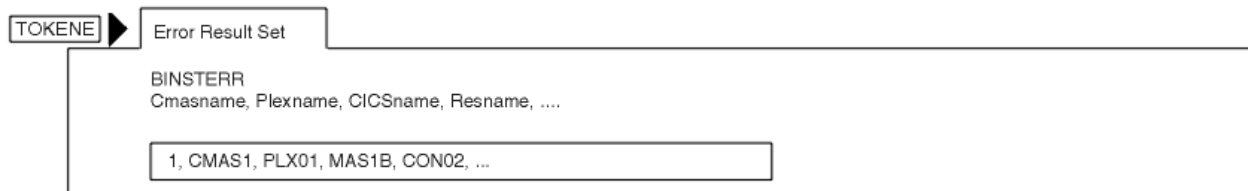


図 20. BINSTERR エラー結果セット

97 ページの図 20 に示す、TOKENED によって参照される BINSTERR エラー結果セットには、FETCH コマンドを使用してアクセスします。

**FETCH RESULT (TOKENED) INTO (AREA6) ...**



図 21. FETCH を使用した BINSTERR レコードの取得

97 ページの図 21 は、FETCH コマンドを使用して、TOKENED で参照されるエラー結果セットに関連付けられた BINSTERR レコードを取得する方法を示しています。

## MASQRYER レコードの取り出し

1 つ以上のターゲット・システムが適切なタイミングで応答しないことが原因で CICS 操作リソースで指定されたコマンドを完了できない場合は、CICSplex SM から MASQRYER リソース・テーブル・レコードの形式で詳細な診断データが提供されます。このデータは、応答に失敗した MAS または CMAS の特定と、失敗の理由の特定に役立ちます。

API コマンドによって生成された MASQRYER リソース・テーブル・レコードのセットは、以前の **GET**、**PERFORM**、または **SET** コマンドの対象の結果セットで指定された **QUERY** コマンドに **QUERYERROR** パラメーターを追加することにより、調べることができます。MASQRYER リソース・テーブル・レコードは、以前の **GET**、**PERFORM**、または **SET** コマンドの対象の結果セットで指定された **FETCH** コマンドに **QUERYERROR** パラメーターを追加することにより、取り出すことができます。

## MASQRYER レコードの評価

MASQRYER レコードには、API 要求に適時に応答できなかった CMAS または MAS を特定するフィールドが含まれています。

注：以下は MASQRYER レコードに関する一般情報です。CICSplex SM によって提供される MASQRYER リソース・テーブルのコピーブックでは、MASQRYER レコードの内容と構造が詳細に記述されています。MASQRYER リソースを使用するプログラムを記述する際は、CICSplex SM リソース・テーブルまたは提供されているコピーブックを参照してください。

要求に応答できなかった領域を特定するには、以下のフィールドの値を確認します。

### PLEXNAME

API 要求の CONTEXT。

### CICSNAME

応答できなかった CMAS または MAS の名前。

### ERRORTYPE

その領域が応答できなかった理由を特定する値。

### CICSTYPE

C = CMAS、M = MAS

## MASQRYER レコードの存在

MASQRYER リソース・テーブルのレコードは、CICS Operate リソース・テーブルで検索や操作を行う **GET**、**PERFORM OBJECT**、**PERFORM SET**、**REFRESH**、または **SET** コマンドによって生成されます。MASQRYER レコードは、要求のターゲット・スコープ内の 1 つ以上の領域から応答を受け取らなかった場合に生成されます。

MASQRYER レコードは、以下の状況では存在しません。

- コマンドが結果セットを処理して、OK の RESPONSE 値で完了する。
- コマンドが結果セットを処理して、FAILED の RESPONSE 値で完了する。

## 第 5 章 REXX プログラムの作成

REXX ランタイム・インターフェースを使用して API プログラムを作成できます。当 API には、CICSplex SM に付属している REXX 関数パッケージを通じてアクセスします。

### API 環境へのアクセス

REXX ランタイム・インターフェースでは、API コマンドの変換は必要ありません。これらのコマンドは、CICSplex SM に付属している REXX 関数パッケージによって解釈されます。

REXX 関数パッケージのインストール手順については、[REXX 機能パッケージのインストール](#)を参照してください。

プログラム内での CICSplex SM の最初の呼び出しは、EYUINIT または EYUAPI 関数でなければなりません。EYUINIT は、API 環境を初期化するための主要な手段です。ただし、EYUINIT が最初に発行されない場合は、EYUAPI 関数によって環境が初期化されます。

**注:** プログラムは、EYUINIT 関数または EYUAPI 関数である必要があります。EYUINIT は、API 環境を初期化するための主要な手段です。ただし、EYUINIT が最初に発行されない場合は、EYUAPI 関数によって環境が初期化されます。

例えば、サンプル・プログラム EYU#API1 (CICSTS22.CPSM.SEYUSAMP ライブラリーで配布されます) の開始部分は次のようになります。

```
Say 'Initializing API...'
XX = EYUINIT()
If XX <> 0 Then Signal UNEXPECTED
Say 'Establishing connection...'
XX = EYUAPI('CONNECT'
           'CONTEXT('W_CONTEXT')' ,
           'SCOPE('W_SCOPE')' ,
           'VERSION(0310)' ,
           'THREAD(W_THREAD)' ,
           'RESPONSE(W_RESPONSE)' ,
           'REASON(W_REASON)')
If XX <> 0 Then Signal UNEXPECTED
```

この例では、EYUINIT 関数が最初に発行されて API 環境を初期化します。次に、EYUAPI 関数を使用して API CONNECT コマンドが発行されます。

EYUINIT 関数または EYUAPI 関数を発行したら、次の操作を実行できます。

- 他の任意の CICSplex SM 関数を発行する。
- REXX ADDRESS コマンドを発行してホスト・サブコマンド環境にアクセスする。

いったん初期化された API 環境は、プログラムまたは REXX によって終了されるまで存在します。したがって、プログラム内の最後の CICSplex SM 呼び出しは、常に EYUTERM 関数にします。EYUTERM を発行しないと、ストレージなどの一部の REXX リソースが割り振られたままになる可能性があるため、REXX でそれらのリソースを解放することが必要になります。

例えば、サンプル・プログラム EYU#API1 の終了部分は次のようになります。

```
XX = EYUAPI('TERMINATE RESPONSE(W_RESPONSE) REASON(W_REASON)')
XX = EYUTERM()
```

この例では、EYUAPI 関数を使用して API の TERMINATE コマンドを発行しています。次に、EYUTERM を発行して API 環境を終了して、その割り振り済みリソースを解放しています。

EYUTERM 関数を使用することは、常に推奨されます。ただし、CICSplex SM ホスト・サブコマンド環境が企業内にインストールされている場合は (関数パッケージから呼び出されるのではなく)、各プログラムの末尾で EYUTERM 関数を使用する必要がない可能性があります。企業のプログラミング・ガイドラインによっては、割り振られたままの REXX リソースを、次にホスト・サブコマンド環境にアクセスする CICSplex SM API プログラムで再利用できます。

## API コマンドの指定

REXX でプログラムを作成するときには、CICSplex SM に付属している REXX 関数パッケージに、発行されるコマンドの文字イメージを渡します。

コマンド・ストリングには、必要に応じて、組み込み REXX 変数を含めることができます。次の 2 つのいずれかの方法でコマンドを指定することができます。

- そのコマンドの名前をパラメーターとして指定して EYUAPI 関数を呼び出します。
- REXX ADDRESS コマンドを使用して後続のステートメントを関数パッケージに渡します。

**注:** REXX PARSE VALUE コマンドを使用して API コマンドを関数パッケージに渡すこともできます。ただし、PARSE VALUE の処理オーバーヘッドは非常に高くなります。また、EYUAPI 関数の戻り値は単一の文字 (0 または 1) のみであるため、この関数の結果を解析する必要はありません。これらの理由から、PARSE VALUE の使用は推奨されません。

次の例では、EYUAPI 関数を使用して発行される GET コマンドの一部を示しています。

```
var = EYUAPI('GET OBJECT(LOCTRAN)...')
```

var は、EYUAPI 関数からの戻りコードを受け取るために割り当てられた変数です。

次の例では、REXX ADDRESS コマンドによって発行される同じ GET コマンドを示しています。

```
ADDRESS CPSM 'GET OBJECT(LOCTRAN)...'
```

REXX 変数内のデータが関数パッケージに渡される際は、API コマンドのテキスト部分を終了させて、REXX 変数を指定して、API コマンドの残り部分を完了する必要があります。次の例では、REXX 変数の組み込み使用を例示するための完全な GET コマンドを示しています。

```
var = EYUAPI('GET OBJECT(LOCTRAN)' ,  
            'RESULT(setvar) THREAD(THRD1)' ,  
            'RESPONSE(rspvar) REASON(reavar)')
```

この例では、LOCTRAN オブジェクトを受け取るための結果セットと、RESPONSE および REASON オプションはすべて REXX 変数として指定されています。

REXX が変数置換を処理する方法を考慮して、変数の用途が API へのデータ送信であるか、API からのデータ受信であるか、これらの両方であるかに留意する必要があります。次の例では、USER および VERSION オプションによって API にデータが送信される CONNECT コマンドを示しています。THREAD、RESPONSE、および REASON のオプションはすべて、API からデータを受信するための変数の名前を指定しています。データを受信するための変数の名前は、コマンドの一部として指定されていることに注意してください。

```
var = EYUAPI('CONNECT USER('userid') VERSION(0310)' ,  
            'THREAD(thdtkn) RESPONSE(rspvar) REASON(reavar)')
```

リソース・テーブルにアクセスする場合は、特別な処理が必要です。このことの例として FETCH コマンドを挙げることができ、この場合は、INTO オプションを使用して、プログラムによる処理のためにリソース・テーブル・データを配置する場所を定義する必要があります。REXX では、1 つ以上のリソース・テーブル・レコードを受信するためのステム変数の接頭部として INTO オプションを指定する必要があります。ステム変数のゼロ・エントリーは、戻されたレコードの数を示しています。

## リソース・テーブル・データへのアクセス

CICSplex SM が REXX にリソース・テーブル・データを提供する方法のために、2 つのコマンドが追加で REXX 関数パッケージの一部として提供されています。

コマンドは以下のとおりです。

### TPARSE

レコードから個別リソース・テーブル属性を抽出して、標準の REXX 変数に配置します。リソース・テーブル・レコード自体を、ステム変数などの有効な REXX 変数で指定できます。

**TPARSE** を使用すると、リソース・テーブル・レコードの属性データを分析したり、この属性データにアクセスしたりできます。

## TBUILD

指定した一連の変数から、CPSM 定義または CICS 定義のリソース・テーブル・レコードを構築します。各変数には、個別リソース・テーブル属性が含まれる必要があります。

**TBUILD** を使用すると、CICSplex SM データ・リポジトリで作成、更新、または削除する定義のリソース・テーブル・レコードを構築できます。

**注:** **TBUILD** では、指定した属性のみが使用されます。オプションの属性に対してデフォルト値が想定されません。オプションの属性に変数を指定しない場合、リソース・テーブル・レコードの対応するフィールドがそのデータ型に応じて初期化されます (つまり、文字フィールドはブランクに設定されて、バイナリー・データと EYUDA 値はゼロに設定されます)。

リソース・テーブル属性を表す変数は、**TPARSE** の場合は CICSplex SM によって作成され、**TBUILD** の場合はユーザー自身が作成します。変数名は、次のように、属性名に接頭部を追加することによって形成されます。

*prefix\_fieldname*

ここで、

### 接頭部

指定するテキスト・ストリング。接頭部の最大許容長は、REXX、およびプログラムが実行される環境によって決定します。

### fieldname

リソース・テーブルの属性の名前。

接頭部と属性名の間には下線文字 (   ) を挿入してください。

REXX で作成されたプログラムによってリソース・テーブル・レコードが API に渡される場合は、レコードの形式およびレイアウトを、CICSplex SM で定義されたとおりにしてください。

## TBUILD および TPARSE の使用のベスト・プラクティス

TBUILD と TPARSE の処理プログラムを実行するには、重要な一連の内部ストレージ構造が必要です。可能な場合は、連続する **TBUILD** コマンドおよび **TPARSE** コマンドの実行にこれらの構造が再利用されます。ただし、時間が経過するとストレージのフラグメント化が進み、結果的に、アドレス・スペース・ストレージの消費量が増加します。ストレージが使用不能であることが原因で TPARSE および TBUILD が失敗します。

単一論理パスの形式で実行されるアプリケーションには、この問題が悪影響を及ぼすことはありません。ただし、長時間にわたる長期実行プロセスの形式で実行されるアプリケーションは、TPARSE および TBUILD の失敗を報告し始める場合があります。

この問題を解決するには、これらの構造が解放されて、各プロセス・パスで再割り振りされるように、EYUINIT() コールおよび EYUTERM() コールを挿入します。

例えば、以下のコードについて注目してみます。

```
XX = EYUINIT();
XX = EYUAPI('CONNECT CONTEXT('my_context') THREAD(TTKN.1) .... ;
Do Forever;
/* perform application processing that includes TPARSE or TBUILD commands */
...
...
If Loop_Termination = True then
Signal EndJob;
End; /* Do Forever */
EndJob;
XX = EYUAPI('DISCONNECT THREAD(TTKN.1) .... ;
XX = EYUTERM();
Return;
```

Do Forever ループで TBUILD コールまたは TPARSE コールが実行される場合は、時間が経過するとストレージのフラグメント化が原因で実行が失敗します。推奨される解決方法は、次のようにループ内に EYUINIT() コールおよび EYUTERM() コールを挿入する方法です。これにより、これらの構造が解放され、各プロセス・パスで再割り振りされます。

```
Do Forever;
  XX = EYUINIT();
  XX = EYUAPI('CONNECT CONTEXT('my_context') THREAD(TTKN.1) .... ;
  /* perform application processing that includes TPARSE or TBUILD commands */
  ...
  ...
  XX = EYUAPI('DISCONNECT THREAD(TTKN.1) .... ;
  XX = EYUTERM();
  TTKN. = '00'X; /* Reset the Thread Token */
  If Loop_Termination = True then
    Signal EndJob;
  End; /* Do Forever */
EndJob;
Return;
```

## 属性値の変換

TBUILD コマンドおよび TPARSE コマンドでは、特定のリソース・テーブル属性を処理するときに、TRANSLATE API コマンドが使用されます。

例えば、リソース・テーブル・レコードでは、EYUDA と CVDA の値はその数字の形式で維持されます。デフォルトでは、TPARSE コマンドによって、これらの値が表示可能文字形式に変換されます。一方、TBUILD では、指定した EYUDA または CVDA のすべての文字値が同等の数値に変換されます。

ただし、これらのコマンドで ASIS オプションを使用すると、属性値は変換されません。TPARSE コマンドで ASIS を指定する場合は、API によって値の再変換が試行されないように、レコードを再構築するときに TBUILD コマンドでも ASIS を指定してください。

TPARSE コマンドで ASIS を指定した後で、属性値を変換することを決定した場合は、TRANSLATE API コマンドを使用できます。

## CHANGEAGENT、CHANGEAGREL、 CHANGETIME、CHANGEUSRID、および CREATETIME 属性の処理

CPSM 定義および CICS 定義のすべてのリソース・テーブル・レコードの最初の 8 バイトには、レコードの最終変更日時を示す CHANGETIME という属性が含まれます。CICS 定義レコードには、定義の作成日時を示す CREATETIME 属性も含まれます。

CICS 定義レコードには、定義の作成日時を示す CREATETIME 属性も含まれます。

既存の CHANGETIME 属性と CREATETIME 属性を組み合わせた、リソース・テーブル定義レコード内の CHANGEAGENT、CHANGEAGREL、CHANGEUSRID の各属性フィールドは、リソース定義シグニチャーに由来し、BAS リソース定義に関してのみ有効です。

CHANGEAGENT は、リソースの定義、または最終変更について表示します。CHANGEAGREL には、リソース定義を作成または最終変更した CICS システムのレベルが含まれます。CHANGEUSRID には、リソース定義を作成または最終変更したユーザー ID が含まれます。

運用基本テーブル・リソース名および BAS リソース定義バージョンを使用して、リソースのインストールに使用するリソース定義を特定できます。ただし、リソースのインストール以降に BAS 定義レコードが変更されている場合があります。運用基本テーブル・レコードの CHANGETIME 値と BAS リソース定義レコードの CHANGETIME 値を比較して、時間の値が対応しているかどうかを確認します。BAS レコードの CHANGETIME 値は完全なローカル STCK 形式ですが、運用基本テーブル・レコードの CHANGETIME 値は細分度の低い STCK 形式であるため、STCK 時間値の先頭のワードのみを比較できます。この制限は、BAS リソース定義の CREATETIME を対応する運用基本テーブルの DEFINETIME STCK 値と比較する場合にも適用されます。

CHANGEAGENT、CHANGEAGREL、CHANGETIME、CHANGEUSRID、および CREATETIME の各属性は CICSplex SM によって内部的に維持されるので、これらの属性値の変更を試みないでください。リソース・テーブル・レコードを更新する場合、TBUILD コマンドに渡す CHANGEAGENT、CHANGEAGREL、

CHANGETIME、CHANGEUSRID、および CREATETIME の値は、TPARSE から受け取った値と同じでなければなりません。

デフォルトでは、TPARSE コマンドによって、CHANGETIME 値と CREATETIME 値が表示可能な文字値に変換されます。ただし、文字形式のこれらの値を TBUILD に戻すことはできません。そのため、定義を更新してリソース・テーブル・レコードを再構築する予定の場合は、TPARSE コマンドおよび TBUILD コマンドで ASIS オプションを使用する必要があります。ASIS を使用する場合、CHANGETIME と CREATETIME の値は 16 バイトの 16 進値として表示されます。

## INSTALLAGENT、INSTALLTIME、 INSTALLUSRID、および BASDEFINEVER の各属性

INSTALLAGENT、INSTALLTIME、INSTALLUSRID、BASDEFINEVER という各属性フィールドは、多数のリソース・タイプの CICSplex SM 運用リソース・テーブルに含まれています。INSTALLAGENT、INSTALLTIME、および INSTALLUSRID フィールドの組み合わせによって、リソースのインストール・シグニチャーが形成されます。インストール・シグニチャーには、各リソースのインストール方法、時期、およびインストール実行者が示されます。

リソースがインストールされた時期に関する情報を表示できると問題判別に役立ちますし、詳細情報によってリソースの監査とトレースが向上します。

INSTALLAGENT はリソースのインストール方法について、INSTALLTIME はリソースがインストールされた時刻、および INSTALLUSRID はリソースをインストールしたユーザー ID をそれぞれ表示します。詳細については、[リソース・シグニチャー・フィールド値の要約](#) を参照してください。

インストール・シグニチャーをサポートしている運用基本テーブルは、以下のとおりです。ATOMSERV、BUNDLE、CONNECT、DB2CONN、DB2ENTRY、DB2TRN、DOCTEMP、EJCOSE、EJDJAR、ENQMODEL、EXTRATDQ、INDTDQ、INTRATDQ、IPCONN、JRNLMODL、LIBRARY、LOCFILE、LOCTRAN、PIPELINE、PROCTYP、PROFILE、PROGRAM、REMFIL、REMTDQ、REMTAN、RQMODEL、TCPIPS、TRANCLAS、TSMODEL、URIMAP、WEBSERV。

CICSplex SM DETAILED ビューを使用してインストール・シグニチャーを表示するため、画面下部には「リソース・シグニチャー (Resource signature)」ローカル・リンクが備えられています。これは、リソースのリソース・シグニチャー属性を表示する別のビューへのリンクです。

運用リソース・テーブルの BASDEFINEVER 属性には、インストールされる BAS リソース定義のバージョン値が表示されます。CICSplex SM API 要求で BASDEFINEVER 属性を使用すると、リソースのインストールに使用する BAS リソース定義を識別できます。この属性は 2 進数ハーフワードで、DREPAPI の CHANGEAGENT 値を持つリソースの DEFINESOURCE CPSMnn スtring の nn 部分が含まれています。

INSTALLAGENT、INSTALLTIME、INSTALLUSRID、および BASDEFINEVER の各属性は CICSplex SM によって内部的に維持されるので、これらの属性値を決して変更しないでください。

## FEEDBACK 属性の処理

TPARSE コマンドを使用して個別のリソース・テーブル属性を抽出した場合、そのデータを後続の API コマンドで使用するようにするには、追加の処理が必要になることがあります。

ERR\_RESULT エラー結果セット・トークンは 10 進数フォーマットで戻されるため、RESULT() オプションで使用可能にするためには文字形式に変換される必要があります。そのためには、次の例のように D2X() および X2C() という REXX 組み込み関数を使用できます。

```
var = X2C(RIGHT(D2X(FEEDBACK_ERR_RESULT),8,'0'))
```





## 第 6 章 REXX エラー処理

いくつかのタイプのエラーは、REXX ランタイム・インターフェースに関連付けられています。

### 変換エラー

REXX による CICSplex SM API コマンドの解釈の試行中にエラーが発生すると、REXX 戻りコードが生成されます。

REXX による CICSplex SM API コマンドの解釈の試行中にエラーが発生すると、REXX 戻りコードが生成されます。REXX が関数のコマンド・ストリングを処理できない場合は、ランタイム・インターフェースによって、次のいずれかの場所で REXX 戻りコードが設定されます。

#### RC 変数

ADDRESS CPSM コマンドが使用されたとき。

戻りコード値は、次のいずれかです。

**0**

コマンドは正常に処理されました。

**8**

コマンドに構文エラーが含まれており、REXX でコマンドを処理できませんでした。エラーを説明する EYUARnnnn メッセージが、IRXSAY WRITEERR 出力に対してシステム上で定義されている宛先に書き込まれます。

**16**

何らかのシステム障害 (ストレージの不足など) が原因でコマンドを処理できませんでした。エラーを説明する REXX メッセージが生成される場合があります。

**-3**

CICSplex SM API 環境を使用できません。この状態は、関数パッケージが適切にインストールされていない場合に発生します。関数パッケージがインストールされている場合は、ADDRESS CPSM コマンドを呼び出す前に 1 つ以上の EYUxxxx REXX 関数を発行しなかったことを意味する場合があります。

#### 関数変数

EYUxxxx REXX 関数が使用されたとき。

ほとんどの EYUxxxx 関数では、戻りコード値は次のいずれかです。

**0**

関数は正常に処理されました。

**1**

関数は失敗しました。エラーを説明する EYUARnnnn メッセージが、IRXSAY WRITEERR 出力に対してシステム上で定義されている宛先に書き込まれます。

EYURESP および EYUREAS 関数では、戻りコードは変換対象の値と同等の数値か、-1 (変換が失敗した場合) になります。

通常、REXX 戻りコードがこれと異なる場合は次のようになります。

**0**

EYUAPI、EYUINIT、または EYUTERM から

#### RESPONSE または REASON の有効な値

EYURESP または EYUREAS から

REXX によって API コマンドが正常に解釈されなかったため、CICSplex SM に渡されず、処理されませんでした。コマンドが処理されない場合は、RESPONSE と REASON の値は設定されないため、確認する必要はありません。

戻りコードが 0 の場合、REXX によって API コマンドが解釈されて、CICSplex SM に渡されました。0 の戻りコードは、コマンドが CICSplex SM によって正常に処理されたかどうかを示しているわけではありませ

ん。API コマンドの結果を確認するには、コマンドによって戻される RESPONSE と REASON の値を参照します。

## ランタイム・エラー

CICSplex SM による API コマンドの処理の試行中に発生したエラーは、コマンドの RESPONSE と REASON の値によって報告されます。

CICSplex SM による API コマンドの処理の試行中に発生したエラーは、コマンドの RESPONSE と REASON の値によって報告されます。詳しくは、[85 ページの『RESPONSE オプションと REASON オプションの使用』](#)を参照してください。

## TPARSE と TBUILD のエラー

TPARSE コマンドまたは TBUILD コマンドの結果は、これらのコマンドの必須指定のオプションである STATUS オプションによって戻されます。STATUS オプションの用途は、他の API コマンドの RESPONSE および REASON のオプションの用途と同様です。

STATUS オプションにより、次のいずれかの文字形式で REXX 状況値が戻されます。

### OK

コマンドは処理を正常に完了しました。

### SYNTAX ERROR

構文エラーのため、コマンドを処理できませんでした。エラーを説明する EYUARnnnn メッセージが、IRXSAY WRITEERR 出力に対してシステム上で定義されている宛先に書き込まれます。

### FAILURE

コマンドは、次のいずれかの原因で失敗しました。

- コマンドで処理しようとしたデータの一部が無効である。
- コマンドを処理するために必要なストレージを使用できない。

**注：TBUILD と TPARSE の各コマンドの処理では、時間が経過するとストレージのフラグメント化が進み、結果的に、アドレス・スペース・ストレージの消費量が増加することがあるため、ストレージが使用不能になるおそれがあります。この障害の原因の詳細と、この問題を回避するために従うベスト・プラクティスについては、[100 ページの『リソース・テーブル・データへのアクセス』](#)を参照してください。**

トレース・データが、EYU\_TRACE という REXX ステム変数に書き込まれます。また、障害を説明する EYUARnnnn メッセージが、IRXSAY WRITEERR 出力に対してシステム上で定義されている宛先に書き込まれます。さらに、一部の障害では、EYU#DUMP データ・セットへの領域ダンプの発行が試行されます。IBM サポートの要求がある場合のみ、EYU#DUMP DD ステートメントを実行 JCL に含めてください。

## 内のテキスト

REXX ランタイム・インターフェースの使用時に発生する多くのエラー状態に、エラーを説明するメッセージが伴います。

REXX ランタイム・インターフェースの使用時に発生する多くのエラー状態に、エラーを説明するメッセージが伴います。接頭部 EYUARnnnn で始まるこれらのメッセージが、IRXSAY WRITEERR 出力に対してシステム上で定義されている宛先に書き込まれます。デフォルトでは、このような出力は次のいずれかの場所に書き込まれます。

- TSO フォアグラウンドで実行されるプログラムの場合、出力は端末に書き込まれます。
- バックグラウンドで実行されるプログラムの場合、出力は SYSTSPRT DD の宛先に書き込まれます。

## EYU\_TRACE データ

エラーが発生した場合は常に、ランタイム・インターフェースによって EYU\_TRACE という REXX ステム変数が作成されて、トレースが保証されます。

エラーが発生した場合は常に、ランタイム・インターフェースによって EYU\_TRACE という REXX ステム変数が作成されて、トレースが保証されます。該当する状態を以下に示します。

- TBUILD または TPARSE のコマンドの STATUS が FAILURE である。
- EYUxxxx 関数の戻りコードが 0 以外である。

ステム配列のゼロのエントリーは、生成されたトレース・レコードの数を示しています。エントリー 1 から n には、実際のトレース・レコードが含まれます。

REXX プログラムまたはランタイム・インターフェースで問題が発生した場合、IBM サポートで EYU\_TRACE のトレース・レコードが必要になることがあります。CICSplex SM では、EYU\_TRACE レコードを形式設定および出力するための REXX EXEC が配布されており、IBM サポートでは、これを REXX プログラムに組み込むようお客様にお願いします。フォーマット設定ルーチンは EYU#TRCF と呼ばれ、SEYUCLIB ライブラリーで配布されます。EYU#TRCF は、IBM サポートの要求がある場合のみ使用してください。



---

## 付録 A BINCONRS、BINCONSC、および BINSTERR の エラー・コード

BINCONRS、BINCONSC、および BINSTERR の各コピーブックには、エラー・コードが含まれます。

これらのエラー・コードを含むフィードバック・エラー結果セットの解釈については、[フィードバック・レコードの取得](#)を参照してください。



## 付録 B CICSplex SM API サンプル・プログラム

CICSplex SM には、いくつかのサンプル・プログラムが用意されています。

ここでは、各サンプル・プログラムを、それが配布されている言語のいずれかで表示します。各言語で提供されるサンプル・プログラムのリストおよびそれらが配布されるライブラリーについては、[サンプル・プログラム](#)を参照してください。

注：その他のサンプル CICSplex SM API プログラムを、IBM CICS を通じて入手できます。

### EYU#API1

プログラム EYU#API1 は、TSO 環境向けに REXX で記述されています。

#### EYU#API1 について

このプログラムは、以下の処理を行います。

- API への接続を確立します。
- DFH、EYU、または IBM 以外で始まるすべての PROGRAM リソース・テーブル・レコードを含む結果セットを作成します。
- 結果セット内の各レコードを検索します。
- CICS CVDA 属性をすべて、意味のある文字値に変換します。
- 端末上の各レコードの、プログラム名、言語、使用可能状況、および CEDF 状況を表示します。
- API 接続を終了させます。

使用されているコマンド：CONNECT、FETCH、GET、TERMINATE、TRANSLATE

```
/* REXX */
/*****
/*
/* MODULE NAME = EYU#API1
/*
/* DESCRIPTIVE NAME = CPSM Sample API Program 1
/* (Sample REXX Version)
/*
/* 5695-081
/* COPYRIGHT = NONE
/*
/* STATUS = %CP00
/*
/* FUNCTION =
/*
/* To provide an example of the use of the following EXEC CPSM
/* commands: CONNECT, GET, FETCH, TRANSLATE, TERMINATE.
/*
/* When invoked, the program depends upon the values held in the
/* W_CONTEXT and W_SCOPE declarations when establishing a
/* connection with CICSplex SM. They must take the following
/* values:
/*
/* W_CONTEXT = The name of a CMAS or CICSplex. Refer to the
/* description of the EXEC CPSM CONNECT command
/* for further information regarding the CONTEXT
/* option.
/*
/* W_SCOPE = The name of a CICSplex, CICS system, or CICS
/* system group within the CICSplex. Refer to the
/* description of the EXEC CPSM CONNECT command
/* for further information regarding the SCOPE
/* option.
/*
/* This sample requires no parameters at invocation time.
/*
/* The sample establishes an API connection and issues a GET
/* command to create a result set containing program resource
/* table records which match the criteria.
/*
```

```

/* Using the FETCH command each record in the result set is */
/* retrieved. Once retrieved the TRANSLATE command is used to */
/* convert those attributes of each record which are EYUDA or */
/* CVDA values into meaningful character representations. A */
/* record is then displayed on the terminal showing the program */
/* name, language, program status, and CEDF status. */
/* */
/* Finally, the API connection is terminated. */
/* */
/*-----*/
/*NOTES : */
/* DEPENDENCIES = S/390, TSO */
/* RESTRICTIONS = None */
/* REGISTER CONVENTIONS = */
/* MODULE TYPE = Executable */
/* PROCESSOR = REXX */
/* ATTRIBUTES = Read only, Serially Reusable */
/* */

```

```

/*-----*/
/* */
/*ENTRY POINT = EYU#API1 */
/* */
/* PURPOSE = All Functions */
/* */
/* LINKAGE = From TSO as a REXX EXEC. */
/* */
/* INPUT = None. */
/* */
/*-----*/
/* */

```

```

Address 'TSO'
Parse Value 0 0 With W_RESPONSE W_REASON .
/*-----*/
/* CHANGE W_CONTEXT AND W_SCOPE TO MATCH YOUR INSTALLATION */
/*-----*/
W_CONTEXT = 'RTGA'
W_SCOPE = 'RTGA'
/*-----*/
/* OBTAIN A CPSM API CONNECTION. */
/* */
/* THE API WILL RETURN A TOKEN IDENTIFYING THE THREAD IN */
/* VARIABLE W_THREAD. */
/*-----*/
Say 'Initializing API...'
XX = EYUINIT()
If XX <> 0 Then Signal UNEXPECTED
Say 'Establishing connection...'
XX = EYUAPI('CONNECT' ,
            'CONTEXT('W_CONTEXT')' ,
            'SCOPE('W_SCOPE')' ,
            'VERSION(0310)' ,
            'THREAD(W_THREAD)' ,
            'RESPONSE(W_RESPONSE)' ,
            'REASON(W_REASON)')
If XX <> 0 Then Signal UNEXPECTED
If W_RESPONSE <> EYURESP(OK) Then Signal NO_CONNECT
/*-----*/
/* GET THE PROGRAM RESOURCE TABLE. */
/* */
/* CREATE A RESULT SET CONTAINING ENTRIES FOR ALL PROGRAMS */
/* WITH NAMES NOT BEGINNING DFH, EYU or IBM. */
/* THE NUMBER OF ENTRIES MEETING THE CRITERIA IS RETURNED IN */
/* VARIABLE W_RECCNT. */
/*-----*/
Say 'Get the PROGRAM resource table...'
W_CRITERIA = 'NOT (PROGRAM=DFH* OR PROGRAM=EYU* OR PROGRAM=IBM*)'
W_CRITERIALEN = 'LENGTH'(W_CRITERIA)
XX = EYUAPI('GET OBJECT(PROGRAM)' ,
            'CRITERIA(W_CRITERIA)' ,
            'LENGTH('W_CRITERIALEN')' ,
            'COUNT(W_RECCNT)' ,
            'RESULT(W_RESULT)' ,
            'THREAD(W_THREAD)' ,
            'RESPONSE(W_RESPONSE)' ,
            'REASON(W_REASON)')

```



```

If XX <> 0 Then Signal UNEXPECTED
If W_RESPONSE <> EYURESP(OK) Then Signal NO_GET

```

```

/*-----*/
/*    RETRIEVE INFORMATION ABOUT EACH PROGRAM.    */
/*-----*/
/*    FETCH EACH ENTRY AND USE TPARSE TO OBTAIN EACH ATTRIBUTE.    */
/*    DISPLAY DETAILS OF EACH PROGRAM TO THE USER.    */
/*-----*/
Say 'Fetching' W_RECcnt 'PROGRAM entries...'
Say 'Program Language      Status      CEDF Status'
W_INTO_OBJECTLEN = 136      /* LENGTH OF PROGRAM TABLE */
Do III = 1 To W_RECcnt
  XX = EYUAPI('FETCH INTO(W_INTO_OBJECT)' ,
              'LENGTH(W_INTO_OBJECTLEN)' ,
              'RESULT(W_RESULT)' ,
              'THREAD(W_THREAD)' ,
              'RESPONSE(W_RESPONSE)' ,
              'REASON(W_REASON)')
  If XX <> 0 Then Signal UNEXPECTED
  If W_RESPONSE <> EYURESP(OK) Then Signal NO_FETCH
  XX = EYUAPI('TPARSE OBJECT(PROGRAM)' ,
              'PREFIX(PGM)' ,
              'STATUS(W_RESPONSE)' ,
              'VAR(W_INTO_OBJECT.1)' ,
              'THREAD(W_THREAD)')
  If W_RESPONSE <> 'OK' Then Signal UNEXPECTED
  W_TEXT = PGM_PROGRAM
  W_TEXT = 'OVERLAY' (PGM_LANGUAGE,W_TEXT,10)
  W_TEXT = 'OVERLAY' (PGM_STATUS,W_TEXT,23)
  W_TEXT = 'OVERLAY' (PGM_CEDFSTATUS,W_TEXT,36)
  Say W_TEXT
End III
Signal ENDIT

```

```

/*-----*/
/*    PROCESSING FOR API FAILURES.    */
/*-----*/
UNEXPECTED:
  W_MSG_TEXT = 'UNEXPECTED ERROR.'
  Signal SCRNLG
NO_CONNECT:
  W_MSG_TEXT = 'ERROR CONNECTING TO API.'
  Signal SCRNLG
NO_GET:
  W_MSG_TEXT = 'ERROR GETTING RESOURCE TABLE.'
  Signal SCRNLG
NO_FETCH:
  W_MSG_TEXT = 'ERROR FETCHING RESULT SET.'
  Signal SCRNLG
SCRNLG:
  Say W_MSG_TEXT
  Say 'RESPONSE=||W_RESPONSE ,
      'REASON=||W_REASON 'RESULT=XX
ENDIT:
/*-----*/
/*    TERMINATE API CONNECTION.    */
/*-----*/
XX = EYUAPI('TERMINATE RESPONSE(W_RESPONSE) REASON(W_REASON)')
XX = EYUTERM()
Exit

```

EYUxAPI1 の C/370、COBOL、および PL/I の各バージョンは、CICS 環境用に作成されているため、**EXEC CICS SEND** コマンドをコメント化し、上記の言語固有の出力ステートメントをアンコメントすることにより、MVS バッチ環境で実行するよう変換することができます。

## EYUCAPI2

プログラム EYUCAPI2 は、CICS 環境用に C で作成されます。

### EYUxAPI2 について

このプログラムは、以下の処理を行います。

- API への接続を確立します。
- フィルターを定義して、言語属性がアセンブラの PROGRAM リソース・テーブル・レコードを識別します。
- DFH、EYU、または IBM 以外で始まるすべての PROGRAM リソース・テーブル・レコードを含む結果セットを作成します。
- 指定されたフィルター (LANGUAGE=ASSEMBLER) に一致する結果セット内のレコードにマークを付けます。
- マークされたレコードを、新しい結果セットにコピーします。
- マークされたレコードを、元の結果セットから削除します。
- 結果セット (LANGUAGE=ASSEMBLER and LANGUAGE≠ASSEMBLER) ごとに、次のようにします。
  - 各レコードを検索します。
  - CICS CVDA 属性をすべて変換します。
  - 端末装置にある各レコードを表示します。
- API 接続を終了させます。

使用されているコマンド: CONNECT、COPY、DELETE、FETCH、GET、LOCATE、MARK、SPECIFY FILTER、TERMINATE、TRANSLATE

```

/*****/
/*
/* MODULE NAME = EYUCAPI2
/*
/* DESCRIPTIVE NAME = CPSM Sample API Program 2
/* (Sample C Version)
/*
/*
/* 5695-081
/* COPYRIGHT = NONE
/*
/* STATUS = %CP00
/*
/* FUNCTION =
/*
/* To provide an example of the use of the following EXEC CPSM
/* commands: CONNECT, SPECIFY FILTER, GET, MARK, COPY, DELETE,
/* LOCATE, FETCH, TRANSLATE, TERMINATE.
/*
/* When invoked, the program depends upon the values held in the
/* W_CONTEXT and W_SCOPE declarations when establishing a
/* connection with CICSplex SM. They must take the following
/* values:
/*
/* W_CONTEXT = The name of a CMAS or CICSplex. Refer to the
/* description of the EXEC CPSM CONNECT command
/* for further information regarding the CONTEXT
/* option.
/*
/* W_SCOPE = The name of a CICSplex, CICS system, or CICS
/* system group within the CICSplex. Refer to the
/* description of the EXEC CPSM CONNECT command
/* for further information regarding the SCOPE
/* option.
/*
/* This sample requires no parameters at invocation time.
/*
/* The sample establishes an API connection and issues a SPECIFY
/* FILTER command to create a filter which will match only
/* specific program resource table records. The filter is used
/* later in the program by the MARK command.
/*
/* A GET command is issued to create a result set containing
/* program resource table records which match the criteria. The
/* result set is then used by the MARK command to flag records
/* meeting the previous filter specification. The marked records
/* are then COPIed to a new result set, and then DELETED from
/* the original result set. After this sequence of commands we
/* have two results sets; one containing records which did not
/* meet the filter specification (that is, records where the
/* LANGUAGE is not ASSEMBLER), and one containing records
/* which did match the filter (that is, records where the
/* LANGUAGE is ASSEMBLER).
/*

```

```

/*
/* Taking each of the two results sets in turn a LOCATE command
/* is used to ensure we start at the top of the result set
/* before a FETCH command is used to retrieve each record in
/* the result set. Once retrieved the TRANSLATE command is used
/* to convert those attributes of each record which are EYUDA
/* or CVDA values into meaningful character representations. A
/* record is then displayed on the terminal showing the program
/* name, language, program status, and CEDF status.
/*
/* Finally, the API connection is terminated.
*/
*/

```

```

* -----*
/*NOTES :
/* DEPENDENCIES = S/390, CICS
/* RESTRICTIONS = None
/* REGISTER CONVENTIONS =
/* MODULE TYPE = Executable
/* PROCESSOR = C
/* ATTRIBUTES = Read only, Serially Reusable
/*
/* -----*
/*
/* ENTRY POINT = EYUCAPI2
/*
/* PURPOSE = All Functions
/*
/* LINKAGE = From CICS either with EXEC CICS LINK or as a CICS
/* transaction.
/*
/* INPUT = None.
/*
/* -----*
/*
#include <PROGRAM>
void main()
{
/* -----*
/* CHANGE W_CONTEXT AND W_SCOPE TO MATCH YOUR INSTALLATION
/* -----*
char *W_CONTEXT = "RTGA ";
char *W_SCOPE = "RTGA ";
int W_RESPONSE;
int W_REASON;
int W_THREAD;
char *W_CRITERIA;
int W_CRITERIALEN;
int W_FILTER_TOKEN;
int W_RESULT = 0;
int W_COUNT;
int W_RESULT2 = 0;
int W_COUNT2;
int III;
int JJJ;
int W_RESULT_TOK;
int W_RECcnt;
PROGRAM W_INT0_OBJECT;
int W_INT0_OBJECTLEN;
char W_TRANSCVDA??(12??);
char W_TEXT??(81??);
char W_MSG_TEXT??(81??);
W_TEXT??(80??) = 0x13;
W_MSG_TEXT??(80??) = 0x13;
/* -----*
/* OBTAIN A CPSM API CONNECTION.
/*
/* THE API WILL RETURN A TOKEN IDENTIFYING THE THREAD IN
/* VARIABLE W_THREAD.
/* -----*
strcpy(W_TEXT,"Establishing connection...");
/* printf("Establishing connection...\n"); */
EXEC CICS SEND FROM(W_TEXT) LENGTH(81) ERASE;
EXEC CPSM CONNECT
CONTEXT(W_CONTEXT)
SCOPE(W_SCOPE)
VERSION("0310")
THREAD(W_THREAD)
RESPONSE(W_RESPONSE)

```

```

        REASON(W_REASON) ;
if (W_RESPONSE != EYUVALUE(OK)) { goto NO_CONNECT; }

```

```

/*-----*/
/*      CREATE A FILTER.                                     */
/*      */
/*      CREATE A FILTER WHICH WILL MATCH ONLY THOSE PROGRAMS WITH */
/*      A LANGUAGE OF ASSEMBLER.                                   */
/*      THE FILTER WILL BE USED IN A SUBSEQUENT MARK COMMAND.      */
/*-----*/
strcpy(W_TEXT,"Create a filter...");
/* printf("Create a filter...\n"); */
EXEC CICS SEND FROM(W_TEXT) LENGTH(81) WAIT;
W_CRITERIA = "LANGUAGE=ASSEMBLER.";
W_CRITERIALEN = strlen(W_CRITERIA);
EXEC CPSM SPECIFY FILTER(W_FILTER_TOKEN)
        CRITERIA(W_CRITERIA)
        LENGTH(W_CRITERIALEN)
        OBJECT("PROGRAM ")
        THREAD(W_THREAD)
        RESPONSE(W_RESPONSE)
        REASON(W_REASON) ;
if (W_RESPONSE != EYUVALUE(OK)) { goto NO_FILTER; }

```

```

/*-----*/
/*      GET THE PROGRAM RESOURCE TABLE.                     */
/*      */
/*      CREATE A RESULT SET CONTAINING ENTRIES FOR ALL PROGRAMS */
/*      WITH NAMES NOT BEGINNING DFH, EYU OR IBM.               */
/*      THE NUMBER OF ENTRIES MEETING THE CRITERIA IS RETURNED IN */
/*      VARIABLE W_COUNT.                                       */
/*-----*/
strcpy(W_TEXT,"Get the PROGRAM resource table...");
/* printf("Get the PROGRAM resource table...\n"); */
EXEC CICS SEND FROM(W_TEXT) LENGTH(81) WAIT;
W_CRITERIA = "NOT (PROGRAM=DFH* OR PROGRAM=EYU* OR PROGRAM=IBM*).";
W_CRITERIALEN = strlen(W_CRITERIA);
EXEC CPSM GET OBJECT("PROGRAM ")
        CRITERIA(W_CRITERIA)
        LENGTH(W_CRITERIALEN)
        COUNT(W_COUNT)
        RESULT(W_RESULT)
        THREAD(W_THREAD)
        RESPONSE(W_RESPONSE)
        REASON(W_REASON) ;
if (W_RESPONSE != EYUVALUE(OK)) { goto NO_GET; }
sprintf(W_TEXT,"Total number of entries: %d", W_COUNT);
/* printf(W_TEXT); */
EXEC CICS SEND FROM(W_TEXT) LENGTH(81) WAIT;
/*-----*/
/*      MARK SELECTED PROGRAM ENTRIES.                         */
/*      */
/*      USING THE FILTER WE MARK THOSE ENTRIES IN THE RESULT SET */
/*      WHICH MEET THE FILTER SPECIFICATION IE. THOSE ENTRIES WITH */
/*      A LANGUAGE OF ASSEMBLER.                                 */
/*-----*/
strcpy(W_TEXT,"Mark LANGUAGE=ASSEMBLER entries...");
/* printf("Mark LANGUAGE=ASSEMBLER entries...\n"); */
EXEC CICS SEND FROM(W_TEXT) LENGTH(81) WAIT;
EXEC CPSM MARK FILTER(W_FILTER_TOKEN)
        RESULT(W_RESULT)
        THREAD(W_THREAD)
        RESPONSE(W_RESPONSE)
        REASON(W_REASON) ;
if (W_RESPONSE != EYUVALUE(OK)) { goto NO_MARK; }

```

```

/*-----*/
/*      COPY MARKED ENTRIES TO ANOTHER RESULT SET.           */
/*      */
/*      HAVING MARKED ENTRIES IN THE RESULT SET WE CAN COPY THEM */
/*      TO A NEW RESULT SET.                                     */
/*      AFTER THIS COMMAND WE WILL HAVE TWO RESULT SETS. ONE     */
/*      CONTAINING ALL THE PROGRAM ENTRIES, AND THE OTHER CONTAINING */
/*      JUST THOSE ENTRIES WITH A LANGUAGE OF ASSEMBLER.        */
/*-----*/
strcpy(W_TEXT,"Copy marked entries...");
/* printf("Copy marked entries...\n"); */
EXEC CICS SEND FROM(W_TEXT) LENGTH(81) WAIT;
EXEC CPSM COPY FROM(W_RESULT)

```

```

        TO(W_RESULT2)
        MARKED
        COUNT(W_COUNT2)
        THREAD(W_THREAD)
        RESPONSE(W_RESPONSE)
        REASON(W_REASON) ;
if (W_RESPONSE != EYUVALUE(OK)) { goto NO_COPY; }
sprintf(W_TEXT,"Number of entries copied: %d", W_COUNT2);
/* printf(W_TEXT); */
EXEC CICS SEND FROM(W_TEXT) LENGTH(81) WAIT;
/*-----*/
/*      DELETE MARKED ENTRIES FROM RESULT SET.                                */
/*      */
/*      WE CAN NOW DELETE THE MARKED ENTRIES FROM THE ORIGINAL                */
/*      RESULT SET.                                                            */
/*      AFTER THIS COMMAND WE HAVE TWO RESULT SETS. ONE RESULT SET           */
/*      CONTAINING ENTRIES WITH LANGUAGE NOT ASSEMBLER, AND THE               */
/*      OTHER CONTAINING ENTRIES WITH A LANGUAGE OF ASSEMBLER.                */
/*-----*/
strcpy(W_TEXT,"Delete marked entries... ");
/* printf("Delete marked entries...\n"); */
EXEC CICS SEND FROM(W_TEXT) LENGTH(81) WAIT;
EXEC CPSM DELETE MARKED
        COUNT(W_COUNT)
        RESULT(W_RESULT)
        THREAD(W_THREAD)
        RESPONSE(W_RESPONSE)
        REASON(W_REASON) ;
if (W_RESPONSE != EYUVALUE(OK)) { goto NO_DELETE; }
sprintf(W_TEXT,"Number of entries remaining: %d", W_COUNT);
/* printf(W_TEXT); */
EXEC CICS SEND FROM(W_TEXT) LENGTH(81) WAIT;

/*-----*/
/*      RETRIEVE INFORMATION ABOUT EACH PROGRAM.                                */
/*      */
/*      FETCH EACH ENTRY, USE INCLUDED STRUCTURE TO OBTAIN EACH               */
/*      ATTRIBUTE AND USE TRANSLATE TO CONVERT CICS CVDAS.                    */
/*      DISPLAY DETAILS OF EACH PROGRAM TO THE USER.                          */
/*-----*/
W_INT0_OBJECTLEN = PROGRAM_TBL_LEN;
for (JJJ = 1; JJJ <= 2; JJJ++)
{
    if (JJJ == 1)
    {
        sprintf(W_TEXT,"Fetching %d non-ASSEMBLER PROGRAM entries...\n",
                W_COUNT);
        W_RESULT_TOK = W_RESULT;
        W_RECCNT = W_COUNT;
    }
    else
    {
        sprintf(W_TEXT,"Fetching %d ASSEMBLER PROGRAM entries...\n",
                W_COUNT2);
        W_RESULT_TOK = W_RESULT2;
        W_RECCNT = W_COUNT2;
    }
    /* printf(W_TEXT); */
    EXEC CICS SEND FROM(W_TEXT) LENGTH(81) WAIT;
    EXEC CPSM LOCATE TOP
        RESULT(W_RESULT_TOK)
        THREAD(W_THREAD)
        RESPONSE(W_RESPONSE)
        REASON(W_REASON) ;
    if (W_RESPONSE != EYUVALUE(OK)) { goto NO_LOCATE; }
    strcpy(W_TEXT,"Program Language      Status      CEDF Status");
    /* printf("Program Language      Status      CEDF Status\n"); */
    EXEC CICS SEND FROM(W_TEXT) LENGTH(81) WAIT;
    for (III = 1; III <= W_RECCNT; III++)
    {
        EXEC CPSM FETCH INTO(&W_INT0_OBJECT)
            LENGTH(W_INT0_OBJECTLEN)
            RESULT(W_RESULT_TOK)
            THREAD(W_THREAD)
            RESPONSE(W_RESPONSE)
            REASON(W_REASON) ;
        if (W_RESPONSE != EYUVALUE(OK)) { goto NO_FETCH; }
        memcpy(W_TEXT,W_INT0_OBJECT.PROGRAM,8);
        EXEC CPSM TRANSLATE OBJECT("PROGRAM ")
            ATTRIBUTE("LANGUAGE ")
            FROMCV(W_INT0_OBJECT.LANGUAGE)

```

```

        TOCHAR(W_TRANSCVDA)
        THREAD(W_THREAD)
        RESPONSE(W_RESPONSE)
        REASON(W_REASON) ;
if (W_RESPONSE != EYUVALUE(OK)) { goto NO_TRANSLATE; }
memcpy(W_TEXT+9,W_TRANSCVDA,12);
EXEC CPSM TRANSLATE OBJECT("PROGRAM ")
        ATTRIBUTE("STATUS ")
        FROMCV(W_INTD_OBJECT.STATUS)
        TOCHAR(W_TRANSCVDA)
        THREAD(W_THREAD)
        RESPONSE(W_RESPONSE)
        REASON(W_REASON) ;

if (W_RESPONSE != EYUVALUE(OK)) { goto NO_TRANSLATE; }
memcpy(W_TEXT+22,W_TRANSCVDA,12);
EXEC CPSM TRANSLATE OBJECT("PROGRAM ")
        ATTRIBUTE("CEDFSTATUS ")
        FROMCV(W_INTD_OBJECT.CEDFSTATUS)
        TOCHAR(W_TRANSCVDA)
        THREAD(W_THREAD)
        RESPONSE(W_RESPONSE)
        REASON(W_REASON) ;
if (W_RESPONSE != EYUVALUE(OK)) { goto NO_TRANSLATE; }
memcpy(W_TEXT+35,W_TRANSCVDA,12);
/* printf("%s\n",W_TEXT); */
EXEC CICS SEND FROM(W_TEXT) LENGTH(81) WAIT;
}
}
goto ENDIT;
/*-----*/
/*      PROCESSING FOR API FAILURES.      */
/*-----*/
NO_CONNECT:
    strcpy(W_MSG_TEXT,"ERROR CONNECTING TO API.\n");
    goto SCRNL0G;
NO_FILTER:
    strcpy(W_MSG_TEXT,"ERROR CREATING FILTER.\n");
    goto SCRNL0G;
NO_GET:
    strcpy(W_MSG_TEXT,"ERROR GETTING RESOURCE TABLE.\n");
    goto SCRNL0G;
NO_MARK:
    strcpy(W_MSG_TEXT,"ERROR MARKING RESULT SET.\n");
    goto SCRNL0G;
NO_COPY:
    strcpy(W_MSG_TEXT,"ERROR COPYING RESULT SET.\n");
    goto SCRNL0G;
NO_DELETE:
    strcpy(W_MSG_TEXT,"ERROR DELETING FROM RESULT SET.\n");
    goto SCRNL0G;
NO_LOCATE:
    strcpy(W_MSG_TEXT,"ERROR LOCATING TO TOP OF RESULT SET.\n");
    goto SCRNL0G;
NO_FETCH:
    strcpy(W_MSG_TEXT,"ERROR FETCHING RESULT SET.\n");
    goto SCRNL0G;
NO_TRANSLATE:
    strcpy(W_MSG_TEXT,"ERROR TRANSLATING ATTRIBUTE\n");
    goto SCRNL0G;
SCRNL0G:
    /* printf(W_MSG_TEXT); */
    EXEC CICS SEND FROM(W_MSG_TEXT) LENGTH(81) WAIT;
    sprintf(W_MSG_TEXT,"RESPONSE=%d REASON=%d\n",W_RESPONSE,W_REASON);
    /* printf(W_MSG_TEXT); */
    EXEC CICS SEND FROM(W_MSG_TEXT) LENGTH(81) WAIT;
ENDIT:
/*-----*/
/*      TERMINATE API CONNECTION.      */
/*-----*/
EXEC CPSM TERMINATE RESPONSE(W_RESPONSE) REASON(W_REASON);
EXEC CICS RETURN;
}

```

EYUxAPI2 の C、C++、COBOL、および PL/I の各バージョンは、CICS 環境用に作成されているため、**EXEC CICS SEND** コマンドをコメント化し、上記の言語固有の出力ステートメントをアンコメントすることにより、MVS バッチ環境で実行するよう変換することができます。

## EYUAAPI3

プログラム EYUAAPI3 は、MVS バッチ環境用にアセンブラーで作成されます。

### EYUAAPI3 について

このプログラムは、以下の処理を行います。

- 既存の CICSplex に設定されているコンテキストで API への接続を確立します。
- 推奨された新規 CICSplex 名が CICSplex SM に対して CICSplex、CMAS、CICS システム、または CICS システム・グループに対してまだ定義されていないことを検証します。
- 既存の CICSplex 定義の CPLEXDEF リソース・テーブル・レコードを含む結果セットを作成し、そのレコードを取得します。
- 既存のレコードをモデルとして使用して、新しい CPLEXDEF リソース・テーブル・レコードを作成します。
- 既存の CICSplex に関連付けられた CICSplex リソース・テーブル・レコードを含む結果セットを作成し、それらのレコードを取得します。
- 既存のレコードをモデルとして使用して、新しい CICSplex リソース・テーブル・レコードを作成します。
- CICS システム、CICS システム・グループ、ワークロード管理定義、リアルタイム分析定義、およびリソース・モニター定義を含め、既存の CICSplex に関連付けられたすべてのリソース・テーブル・レコードを順次取り出します。
- 既存のレコードをモデルとして使用して、新しい CICSplex に必要なすべてのリソース・テーブル・レコードを作成します。
- 必要なリソース・テーブル・レコードがすべて作成される前にエラーが発生する場合、新しい CICSplex 定義を削除します。
- API 処理スレッドを切断します。

使用されているコマンド: CONNECT、CREATE、DISCARD、DISCONNECT、  
FETCH、GET、PERFORM OBJECT、QUALIFY、QUERY、REMOVE

```
*
EYUAAPI3 TITLE 'EYUAAPI3 - CPSM SAMPLE API PROGRAM 3 - ASSEMBLER'
*****
*
* MODULE NAME = EYUAAPI3
*
* DESCRIPTIVE NAME = API sample program 3 ASSEMBLER Version
*
*      5695-081
*      COPYRIGHT = NONE
*
* STATUS = %CP00
*
* FUNCTION =
*
*   To mirror an existing PLEX to a new PLEX.
*
*   When invoked, the program depends upon the values held in the
*   OLDPLEX, NEWPLEX, and MPCMAS variables. They must be set to
*   the following values:
*
*   OLDPLEX    = The name of an existing PLEX that will be mirrored.
*
*   NEWPLEX    = The name that will be given to the new PLEX.
*
*   MPCMAS     = The maintenance point CMAS of the OLDPLEX. This
*               will also be the MP for the NEWPLEX.
*
*   This sample requires no parameters at invocation time.
*
*   The sample processes as follows:
*
*   - a CONNECTION is established to CPSM, with the CONTEXT and
*     SCOPE of the OLDPLEX.
*
```

```

*   - since a PLEX can be either a CONTEXT or SCOPE, we verify
*   that the NEWPLEX is not already a valid CONTEXT (i.e, an
*   existing CICSplex or CMAS) or SCOPE in the OLDPLEX (i.e,
*   an existing CICS system or CICS system group).
*
*   - we GET the CPLEXDEF record for the OLDPLEX, and use this as
*   a module to CREATE the NEWPLEX.
*
*   - we GET the CICSplex records for the OLDPLEX, and use these
*   to add the CMASs in the OLDPLEX to the NEWPLEX.
*
*   - using a list that contains CICSplex definitions including
*   CICS systems, CICS system groups, workload management
*   definitions, real-time analysis definitions and resource
*   monitoring definitions, we GET and FETCH the records from
*   the OrigPlex, and CREATE them in the NewPlex.
*
*   - we then DISCONNECT from CPSM.

```

```

* -----*
*
* NOTES :
*   DEPENDENCIES = S/370
*   RESTRICTIONS = None
*   REGISTER CONVENTIONS =
*       R0          Workarea / external call parameter pointer
*       R1          Workarea / external call parameter pointer
*       R2          Resource Table record pointer
*       R3          Loop counter
*       R4          List pointer
*       R5          Loop counter
*       R6          Unused
*       R7          Unused
*       R8          Unused
*       R9          Subroutine linkage
*       R10         Subroutine linkage
*       R11         Base register
*       R12         Base register
*       R13         Workarea pointer
*       R14         External call linkage
*       R15         External call linkage
*
*   MODULE TYPE   = Executable
*   PROCESSOR     = Assembler
*   ATTRIBUTES    = Read only, Serially Reusable
*                  AMODE(31), RMODE(ANY)
*
* -----*
*
* ENTRY POINT = EYUAAPI3
*
*   PURPOSE = All Functions
*
*   LINKAGE = Executed as a batch program.
*
*   INPUT   = None
*
*   OUTPUT  = File for messages.
*             DDNAME = SYSPRINT
*             DSORG   = PS
*             RECFM   = FB
*             LRECL   = 80
*             BLKSIZE = a multiple of 80
*
* -----*
*
*   EJECT
* EYUAAPI3 CSECT
*   STM R14,R12,12(R13)
*   LR  R12,R15
*   USING EYUAAPI3,R12
*
* -----*
*   GETMAIN working storage and set up SA chain.
*
* -----*
*   GETMAIN R,LV=WORKLEN
*   ST  R13,4(,1)
*   ST  R1,8(,R13)
*   L   R1,24(,R13)
*   L   R13,8(,R13)
*   USING SAVEAREA,R13

```



```

*-----*
*      Preset return code to error - will change to 0 if all ok.      *
*-----*
MVC      RETCODE,=F'8'
*-----*
*      OPEN file for error messages.                                  *
*-----*
OPEN      (SYSPRINT,OUTPUT)
*-----*
*      Specify variables: OLDPLEX, NEWPLEX, MPCMAS                    *
*-----*
*      Ensure that the values specified are valid NAME type (i.e.,   *
*      valid member name) or following code will fail.                *
*-----*
MVC      OLDPLEX,=CL8'plexold'      *** SPECIFY AS DESIRED ***
MVC      NEWPLEX,=CL8'plexnew'      *** SPECIFY AS DESIRED ***
MVC      MPCMAS,=CL8'mpcmas'        *** SPECIFY AS DESIRED ***
*-----*
*      Connect to CPSM API via OLDPLEX.                                *
*-----*
MVC      CONTEXT,OLDPLEX
EXEC      CPSM CONNECT
          CONTEXT(CONTEXT)
          VERSION(=CL4'0130')
          THREAD(THREAD)
          RESPONSE(RESPONSE)
          REASON(REASON)
          CLC      RESPONSE,EYUVALUE(OK)      RESPONSE OK?
          BNE      ERRCON                    No - msgs and out
*-----*
*      Verify that the desired NEWPLEX name is not already a         *
*      PLEX or CMAS. We do this by trying to set the CONTEXT         *
*      to the NEWPLEX name. If successful (NEWPLEX already exists    *
*      as a CONTEXT) issue messages and get out.                      *
*-----*
EXEC      CPSM QUALIFY
          CONTEXT(NEWPLEX)
          THREAD(THREAD)
          RESPONSE(RESPONSE)
          REASON(REASON)
          CLC      RESPONSE,EYUVALUE(OK)      RESPONSE OK?
          BE       ERRNISC                    Yes - already a CONTEXT
*-----*

*-----*
*      Verify that the NEWPLEX name is not already a                 *
*      CSYSDEF or CSYSGRP in the old, soon to be new, CICSplex.      *
*-----*
*
*      Here we will start issuing EXEC CPSM GET requests, to         *
*      get result sets of different Resource Tables. We make         *
*      the call through the GETOBJ subroutine. Variable OBJECT        *
*      must be set with the Resource Table name. If we only want     *
*      a subset of the records for a given Resource Table, we also    *
*      set variable CRITERIA with a selection criteria string.         *
*      This string can contain references to any fields in the        *
*      Resource Table, connected by logical operators, and must       *
*      end with a period - . -. Variable CRITLEN must be loaded       *
*      with the length of the criteria string.                         *
*
*      We will check the RESPONSE from GET calls inline, instead     *
*      of in the subroutine. The reason for this is that sometimes   *
*      a RESPONSE of OK will mean that we have a problem (e.g.,      *
*      the NEWPLEX name already exists as a CICS System name).        *
*-----*
*
*      Ask for a CSYSSYS record equal to the NEWPLEX name.
*
MVC      OBJECT,=CL8'CSYSDEF'
MVC      CRITERIA(5),=CL5'NAME='
MVC      CRITERIA+5(8),NEWPLEX
MVI      CRITERIA+13,C'.'
MVC      CRITLEN,=F'14'
BAS      R10,GETOBJ                      Build result set
CLC      RESPONSE,EYUVALUE(OK)      RESPONSE OK?
BE       ERRNISC                    Yes - already a CICS system
CLC      RESPONSE,EYUVALUE(NODATA)  No CSYSDEF with NEWPLEX name?
BE       NOTCSYS                    Yes - continue
B        ERRGETO                    No - some error - msgs and out
NOTCSYS  DS      OH
*
*      Ask for a CSYSGRP record equal to the NEWPLEX name.

```

```

*
MVC OBJECT,=CL8'CSYSGRP'
MVC CRITERIA(6),=CL6'GROUP='
MVC CRITERIA+6(8),NEWPLEX
MVI CRITERIA+14,C'.'
MVC CRITLEN,=F'15'
BAS R10,GETOBJ          Build the result set
CLC RESPONSE,EYUVALUE(OK) RESPONSE OK?
BE ERNRIS              Yes - already a system group
CLC RESPONSE,EYUVALUE(NODATA) No CSYSGRP with NEWPLEX name?
BE NOTCGRP             Yes - continue
B ERRGETO              No - some error - msgs and out
NOTCGRP DS OH
*-----*
* If we have gotten this far, we know that NEWPLEX is not *
* already the name of a CICSplex, CMAS, CICS System, or *
* CICS System group - so we can start building the NEWPLEX. *
* *
* Switch CONTEXT to MPCMAS to build NEWPLEX and add CMASs. *
*-----*
MVC CONTEXT,MPCMAS

```

```

*-----*
* Build new plex using OLDPLEX as a model. *
* *
* The record that defines a CICSplex is the CPLEXDEF Resource *
* Table. We will GET the OLDPLEX CPLEXDEF record, modify *
* it as needed, and then CREATE the NEWPLEX CPLEXDEF records. *
* This creates the NEWPLEX. *
*-----*
MVI PLEXBLT,C'N'          Indicate NEWPLEX not built yet
*
* First GET CPLEXDEF record for the OLDPLEX.
*
MVC OBJECT,=CL8'CPLEXDEF'
MVC CRITERIA(9),=CL9'CICSPLEX='
MVC CRITERIA+9(8),OLDPLEX
MVI CRITERIA+17,C'.'
MVC CRITLEN,=F'18'
BAS R10,GETOBJ          Build result set
CLC RESPONSE,EYUVALUE(OK) RESPONSE OK?
BNE ERRGETO              No - msgs and out
*
* Here we start using the GETBUF subroutine. This subroutine
* GETMAINS a buffer into which we can FETCH the records of the
* result set that we last issued a GET for.
*
BAS R10,GETBUF          Get storage to receive recs
*
* Here we start using the FETCH subroutine. This subroutine
* reads all the records from the result set into the buffer.
* On return to mainline, R2 points to the first record in
* the buffer.
*
BAS R10,FETCH          Sets R2 to fetched record
*
* Change the OLDPLEX CPLEXDEF record into the NEWPLEX
* CPLEXDEF record.
*
USING CPLEXDEF,R2          Map the record
MVC CPLEXDEF_CICSPLEX,NEWPLEX X
* Set CICSplex name to NEWPLEX
MVC CPLEXDEF_DESC,=CL30'API cloned from' X
* Modify CICSplex ....
MVC CPLEXDEF_DESC+16(8),OLDPLEX X
* .... description
MVC NEWPLXD(CPLEXDEF_TBL_LEN),0(R2) X
* Save NEWPLEX def and len ....
MVC NEWPLXDL,=A(CPLEXDEF_TBL_LEN) X
* .... for possible later REMOVE

```

```

*
* Here we start using the CREATE subroutine. This subroutine
* will cause a CPSM Resource Table record to be built. 可変
* OBJECT needs to be preset to the Resource Table name, the
* Resource Table record to be built must be pointed to by R2
* and must be filled out before called CREATE.
*
BAS R10,CREATE          CREATE NEWPLEX
MVI PLEXBLT,C'Y'          Indicate NEWPLEX now built

```

```

*
* Here we start using the FREEBUF subroutine. This subroutine
* FREEMAINS the buffer into which we FETCHed the records.
*
* BAS R10,FREEBUF Free record storage
*
* When a result set is built (in our program by either GET or
* PERFORM) an id is associated with the result set and placed
* into the variable pointed to by keyword RESULT (for GET we
* are using variable RESULT - for PERFORM, RESULT2). This is
* done so that subsequent calls can reference the result set
* built (e.g, FETCH can retrieve records for GET). When we
* are done using a result set, we must DISCARD it, so that
* CPSM frees us resources allocated for the result set.
* Note that we have not done this with the 2 previous GETs
* we did since the object of them was to NOT get a result set.
* If any of the previous GETs caused a result set to get built,
* we DISCONNECT from CPSM - which causes all our resources to
* be released - and exit.
*
* MVC RESULTD,RESULT Copy GET result set id for X
* DISCARD
* BAS R10,DISCARD Discard the GET result set
* DROP R2 Drop mapping to CPLEXDEF rec
*
*-----*
* Add CMASs in OLDPLEX to NEWPLEX. *
*
* There is a CICSplex Resource Table record for each CMAS *
* that participates in the management of a plex. We first *
* ask for all the CICSplex records for OLDPLEX, and use *
* this info to add the CMASs to the NEWPLEX. *
*-----*
* Ask for the CICSplex records from the OLDPLEX.
*
* MVC OBJECT,=CL8'CICSplex'
* MVC CRITERIA(9),=CL9'PLEXNAME='
* MVC CRITERIA+9(8),OLDPLEX
* MVI CRITERIA+17,C'.'
* MVC CRITLEN,=F'18'
* BAS R10,GETOBJ Build result set
* CLC RESPONSE,EYUVALUE(OK) RESPONSE OK?
* BNE ERRGETO no - msgs and out
* BAS R10,GETBUF Get storage for records
* BAS R10,FETCH Points R2 to first record
* USING CICSplex,R2 Map the Resource Table
* L R5,COUNT Will loop the number of X
* returned CMASs
*
* The MP CMAS is added to the CICSplex when the CPLEXDEF
* record was CREATED. To add any other CMASs to the CICSplex
* we issue a PERFORM against the CPLEXDEF record for NEWPLEX,
* with a parm = CICSplex(newplex) CMAS(cmasname).
*
* MVC ADDCPARM(ADDCLLEN),ADDC Build most of parm
* MVC PARMLLEN,=A(ADDCLLEN) Set its length
* MVC ADDCPLEX,NEWPLEX Add CICSplex name to parm
* MVC OBJECT,=CL8'CPLEXDEF' PERFORM against CPLEXDEF
* ADDCMAS DS 0H
* CLC CICSplex_CMASNAME,MPCMAS CMAS = MPCMAS?
* BE NOADDMP Yes - don't add it then
* MVC ADDCCMAS,CICSplex_CMASNAME X
* Add CMAS name to PARM X
* This comes from the CICSplex X
* records.
*
*
* Note that we already have the CICSplex result set active,
* with the id in RESULT. So here we will use RESULT2 for
* result set that is built for each PERFORM.
*
* MVC RESULT2,=F'0' Always build new result set
* EXEC CPSM PERFORM X
* OBJECT(OBJECT) X
* ACTION(=CL12'ASSIGN') X
* PARM(ADDCPARM) X
* PARMLLEN(PARMLLEN) X
* RESULT(RESULT2) X
* CONTEXT(CONTEXT) X
* THREAD(THREAD) X
* RESPONSE(RESPONSE) X
* REASON(REASON)
* CLC RESPONSE,EYUVALUE(OK) RESPONSE OK?

```

```

        BNE  ERRPERF          no - msgs and out
        MVC  RESULTD,RESULT2   Copy PERFORM result set id for X
                                DISCARD
NOADDMP  BAS  R10,DISCARD      Discard the PERFORM result set
        DS   0H
*
*      We need to get to the next CICSplex record for the next CMAS.
*      The GETBUF subroutine places into variable RECLen the length
*      of the Resource Table record. We now add this to the address
*      of the current record to point to the next record.
*
        A    R2,RECLen
        BCT  R5,ADDCMAS        Add the next CMAS
*
*      No more CICSplex records - discard the CICSplex result set
*      and continue on.
*
        BAS  R10,FREEBUF      Free FETCHed record storage
        MVC  OBJECT,=CL8'CICSplex' For possible DISCARD error msg
        MVC  RESULTD,RESULT   Copy GET result set id for X
                                DISCARD
        BAS  R10,DISCARD      Discard the GET result set
        DROP R2               Drop mapping to CICSplex rec

```

```

*-----*
*      Take all defs in OLDplex and put into NEWplex.
*
*      We have a list of all CICSplex Resource Table names. We
*      loop through this list, getting all the records for a
*      specific Resource Table from the OLDplex and adding them
*      to the NEWplex.
*-----*
        MVC  CRITLEN,=F'0'      Want all records from each X
                                Resource Table - so we don't X
                                want a CRITERIA for GET.
        LA   R3,DEFNUM          Get number of Resource Tables
        LA   R4,DEFLIST         Point R4 to first Resource X
                                Table in list
BLDLOOP  DS   0H
        MVC  OBJECT,0(R4)      Move in Resource Table name
*
*      Get old data - set CONTEXT to OLDplex.
*
        MVC  CONTEXT,OLDplex
        MVC  SCOPE,OLDplex
        BAS  R10,GETOBJ         Build result set
        CLC  RESPONSE,EYUVALUE(OK) RESPONSE OK?
        BE   GOTDEFS           Yes - FETCH and add
        CLC  RESPONSE,EYUVALUE(NODATA) No records returned?
        BE   NODATA            Yes - on to next Resource Tab
        B    ERRGETO           GET error - msgs and out
GOTDEFS  DS   0H
        BAS  R10,GETBUF        Get storage for records
        BAS  R10,FETCH         Point R2 to first record
        L    R5,COUNT          Load number of records for loop
*
*      Add new data - set CONTEXT to NEWplex.
*
        MVC  CONTEXT,NEWplex
CRELOOP  DS   0H
*
*      We need to check if the object being created is a RTAINAPS
*      table. If it is, we need to check if the SCOPE is the
*      OLDplex name - and if so, change it to the NEWplex name.
*      The RTAINAPS table is the only resource table in our list
*      that may have the OLDplex specified as a SCOPE.
*
        CLC  OBJECT,=CL8'RTAINAPS' Creating an RTAINAPS?
        BNE  CRELOOP2          No, just CREATE it
        USING RTAINAPS,R2      May to the record
        CLC  RTAINAPS_SCOPE,OLDplex Is SCOPE equal to OLDplex?
        BNE  CRELOOP2          No, don't change record
        MVC  RTAINAPS_SCOPE,NEWplex Alter SCOPE to NEWplex
        DROP R2               Drop mapping to RTAINAPS rec
CRELOOP2 DS   0H
        BAS  R10,CREATE        CREATE record in NEWplex
        A    R2,RECLen         Point to next record
        BCT  R5,CRELOOP        Loop
        BAS  R10,FREEBUF      Release record storage
        MVC  RESULTD,RESULT   Copy GET result set id for X
                                DISCARD

```

NODATA	BAS	R10,DISCARD	Discard the GET result set	
	DS	0H		
	LA	R4,8(,R4)	Point to next Resource Table	
	BCT	R3,BLDLOOP	Do next Resource Table	
*				
*			We have gone through all the Resource Tables ok. 設定	
*			the return code to 0.	
*				
	MVC	RETCODE,=F'0'		
*-----*				
*			Disconnect the connection and exit the program.	*
*-----*				
EXITDISC	DS	0H		
	EXEC	CPSM DISCONNECT		X
		THREAD(THREAD)		X
		RESPONSE(RESPONSE)		X
		REASON(REASON)		
EXIT	DS	0H		
	CLOSE	(SYSPRINT)		
*-----*				
*			Unchain save area, FREEMAIN working storage, and restore	*
*			registers.	*
*-----*				
	L	R2,RETCODE	Retrieve return code	
	L	R13,4(,R13)		
	L	R1,8(,R13)		
	FREEMAIN	R,A=(R1),LV=WORKLEN		
	L	R14,12(,R13)		
	LR	R15,R2		
	LM	R0,R12,20(R13)		
	LA	R15,0		
	BR	R14		
*-----*				
*			Error routines.	*
*-----*				
ERRCON	DS	0H		
	MVC	OUTLINE,=CL80'Error: Connecting to the API'		
	BAS	R9,PUTMSG		
	BAS	R10,DORR	Format and msg RESPONSE/REASON	
	B	EXIT	Exit	
ERRNISPC	DS	0H		
	MVC	OUTLINE,=CL80'Error: NEWPLEX is already defined as a CICX		
		Splex or CMAS'		
	BAS	R9,PUTMSG		
	B	EXITDISC	DISCONNECT and exit	
ERRNISC	DS	0H		
	MVC	OUTLINE,=CL80'Error: NEWPLEX is already defined as a CICX		
		S system in the OLDPLEX'		
	BAS	R9,PUTMSG		
	B	EXITDISC	DISCONNECT and exit	
ERRNISS	DS	0H		
	MVC	OUTLINE,=CL80'Error: NEWPLEX is already defined as a CICX		
		S system group in the OLDPLEX'		
	BAS	R9,PUTMSG		
	B	EXITDISC	DISCONNECT and exit	
ERRPERF	DS	0H		
	MVC	OUTLINE,=CL80'Error: Adding a CMAS to the NEWPLEX'		
	BAS	R9,PUTMSG		
	MVC	OUTLINE,=CL80' '		
	MVC	OUTTXT1,=CL10'CMASNAME:'		
	MVC	OUTDAT1,ADDCCMAS		
	BAS	R9,PUTMSG		
	BAS	R10,DORR	Format and msg RESPONSE/REASON	
	B	EXITERR		
ERRGETO	DS	0H		
	MVC	OUTLINE,=CL80'Error: GETting an object'		
	BAS	R9,PUTMSG		
	B	DOOBJMSG		
ERRQUERY	DS	0H		
	MVC	OUTLINE,=CL80'Error: QUERYing a record size.'		
	BAS	R9,PUTMSG		
	B	DOOBJMSG		
ERRFETCH	DS	0H		
	MVC	OUTLINE,=CL80'Error: FETCHing an object.'		
	BAS	R9,PUTMSG		
	B	DOOBJMSG		
ERRCREAT	DS	0H		
	MVC	OUTLINE,=CL80'Error: CREATEing an object.'		

	BAS	R9,PUTMSG	
	B	DOOBJMSG	
ERRDISCA	DS	0H	
	MVC	OUTLINE,=CL80'Error: DISCARDing object.'	
	BAS	R9,PUTMSG	
DOOBJMSG	DS	0H	
	MVC	OUTLINE,=CL80' '	
	MVC	OUTTXT1,=CL10'OBJECT:'	
	MVC	OUTDAT1,OBJECT	
	BAS	R9,PUTMSG	
	BAS	R10,DORR	
EXITERR	DS	0H	
	CLI	PLEXBLT,C'Y'	Did we CREATE the NEWPLEX?
	BNE	EXITDISC	No - just DISCONNECT and exit

*			
*		We had already CREATED the NEWPLEX when an error occurred	
*		so we want to delete the NEWPLEX before ending our program.	
*			
EXEC	CPSM	REMOVE	X
		OBJECT(=CL8'CPLEXDEF')	X
		FROM(NEWPLXD)	X
		LENGTH(NEWPLXDL)	X
		CONTEXT(MPCMAS)	X
		THREAD(THREAD)	X
		RESPONSE(RESPONSE)	X
		REASON(REASON)	
CLC	RESPONSE,EYUVALUE(OK)	RESPONSE OK?	
BE	EXITDISC	Yes - DISCONNECT and exit	
MVC	OUTLINE,=CL80'Error: REMOVEing NEWPLEX.'		
BAS	R9,PUTMSG		
BAS	R10,DORR		
B	EXITDISC	DISCONNECT and exit	
-----			*
*		End of error routines.	*
-----			*
*		Subroutines.	*
-----			*
PUTMSG	DS	0H	
	PUT	SYSPRINT,OUTLINE	
	BR	R9	
DORR	DS	0H	
-----			*
*		Subroutine: DORR	*
*		Entry: Via BAS R10,DORR	*
*		Function: Put out error messages indicating what function	*
*		failed and the RESPONSE and REASON from that	*
*		function.	*
*			*
*		Processing: - Format the EXEC CPSM RESPONSE and move to the	*
*		OUTLINE.	*
*		- Format the EXEC CPSM REASON and move to the	*
*		OUTLINE.	*
*		- Call the PUTMSG subroutine to send the	*
*		RESPONSE/REASON data to SYSPRINT.	*
*		- Return to caller.	*
-----			*
MVC	OUTLINE,=CL80' '	clear format area	
MVC	OUTTXT1,=CL10'RESPONSE:'	move in ....	
L	R3,RESPONSE	load up the RESPONSE	
CVD	R3,DOUBLE	convert to decimal	
MVC	OUTDAT1(6),=XL6'402020202120'	move in EDIT pattern	
ED	OUTDAT1(6),DOUBLE+5	EDIT RESPONSE to format area	
MVC	OUTTXT2,=CL10'REASON:'	.... constant data	
L	R3,REASON	load up the REASON	
CVD	R3,DOUBLE	convert to decimal	
MVC	OUTDAT2(6),=XL6'402020202120'	move in EDIT pattern	
ED	OUTDAT2(6),DOUBLE+5	EDIT REASON to format area	
BAS	R9,PUTMSG	SEND it	
MVC	OUTLINE,=CL80' '	clear out OUTLINE again	
BAS	R9,PUTMSG	put out blank line	
BR	R10	return to caller	

GETOBJ	DS	0H	
-----			*
*		Subroutine: GETOBJ	*
*		Entry: Via BAS R10,GETOBJ	*
*		Function: Issue the EXEC CPSM GET command to create a	*
*		result set for a specific object. Note that	*
*		all operands for GET must be preset in	*
*		mainline code - except for RESULT.	*

```

*      Processing: - Clear out the result set id - RESULT - so      *
*                  that a new result set is always built. It      *
*                  is the responsibility of mainline to DISCARD    *
*                  any previous result set for GET.               *
*                  - Determine if the GET request has a CRITERIA   *
*                  and use the proper EXEC CPSM GET call.         *
*                  - Note that GETOBJ does not check the RESPONSE *
*                  from CPSM - this is done in mainline.          *
*                  - Return to caller.                             *
*-----*
*      MVC      RESULT,=F'0'          Always get new result set
*      CLC      CRITLEN,=F'0'
*      BE       GETNOCRT
*      EXEC     CPSM GET
*                  OBJECT(OBJECT)
*                  CRITERIA(CRITERIA)
*                  LENGTH(CRITLEN)
*                  COUNT(COUNT)
*                  RESULT(RESULT)
*                  THREAD(THREAD)
*                  CONTEXT(CONTEXT)
*                  RESPONSE(RESPONSE)
*                  REASON(REASON)
*
*      GETNOCRT BR      R10
*               DS      0H
*               EXEC     CPSM GET
*                  OBJECT(OBJECT)
*                  COUNT(COUNT)
*                  RESULT(RESULT)
*                  THREAD(THREAD)
*                  CONTEXT(CONTEXT)
*                  RESPONSE(RESPONSE)
*                  REASON(REASON)
*
*      BR       R10

```

```

GETBUF   DS      0H
*-----*
*      Subroutine: GETBUF
*      Entry:      Via BAS R10,GETBUF
*      Function:   Get a buffer to hold all the records contained
*                  in the last result set we build though GET.
*      Processing: - Issue EXEC CPSM QUERY to get the length of
*                  the Resource Table record. We use the same
*                  OBJECT and RESULT from the GET. Variable
*                  RECLEN gets the record length.
*                  - Check the RESPONSE from QUERY and issue msgs
*                  and EXIT if not OK.
*                  - Multiple the RECLEN times the COUNT (returned
*                  from last GET) to determine the buffer size
*                  required and GETMAIN it.
*                  - Save the buffer length (BUFLEN) and buffer
*                  address (BUFFER) for the FREEMAIN call in
*                  the FREEBUF subroutine.
*                  - Return to caller.
*-----*
*      EXEC     CPSM QUERY
*                  OBJECT(OBJECT)
*                  DATALENGTH(RECLEN)
*                  RESULT(RESULT)
*                  THREAD(THREAD)
*                  RESPONSE(RESPONSE)
*                  REASON(REASON)
*
*      CLC      RESPONSE,EYUVALUE(OK)    RESPONSE OK?
*      BNE      ERRQUERY                No - msgs and out
*      L        R0,RECLEN
*      L        R1,COUNT
*      MR       R0,R0
*      GETMAIN  R,LV=(R1)
*      ST       R0,BUFLEN
*      ST       R1,BUFFER
*      BR       R10
*
*      FREEBUF  DS      0H
*-----*
*      Subroutine: FREEBUF
*      Entry:      Via BAS R10,FREEBUF
*      Function:   To FREEMAIN the buffer created to hold the
*                  records from the last result set we built .
*                  through GET.
*      Processing: - Use BUFLEN and BUFFER from GETBUF, FREEMAIN
*                  the buffer area.
*                  - Return to caller.

```

```

*-----*
      L      R0,BUFLEN
      L      R1,BUFFER
      FREEMAIN R,A=(R1),LV=(R0)
      BR     R10

FETCH    DS      0H
*-----*
* Subroutine: FETCH *
* Entry:      Via BAS R10,FETCH *
* Function:   Issue the EXEC CPSM FETCH command to retrieve *
*             the result set created by the last GET. *
*             mainline code - except for RESULT. *
* Processing: - For FETCH we must provide a receiving area *
*             and length. We put in the area length into *
*             R2 and the area length in variable LENGTH. *
*             Note that we got both the area and length *
*             in the GETBUF routine. *
*             - Issue the FETCH request using the result set *
*             id - RESULT - from the last GET. *
*             - Check the RESPONSE - if not OK, issue msgs *
*             and exit. *
*             - Return to caller. *
*-----*
      L      R2,BUFFER
      MVC    LENGTH,BUFLEN
      EXEC   CPSM FETCH
*             ALL *
*             INTO(0(,R2)) *
*             LENGTH(LENGTH) *
*             COUNT(COUNT) *
*             RESULT(RESULT) *
*             THREAD(THREAD) *
*             RESPONSE(RESPONSE) *
*             REASON(REASON) *
      CLC    RESPONSE,EYUVALUE(OK)
      BNE    ERRFETCH
      BR     R10

CREATE   DS      0H
*-----*
* Subroutine: CREATE *
* Entry:      Via BAS R10,CREATE *
* Function:   Issue the EXEC CPSM CREATE to build a Resource *
*             Table record. *
* Processing: - Place the length of the record to be build *
*             (RECLen from GETBUF) into variable LENGTH. *
*             R2 should have been set by mainline to point *
*             to the record itself. *
*             - When CREATEing a LNKxxCG record (spec to *
*             group link) we need to specify a parm - *
*             NONE. - to indicate that we only want the *
*             CREATE to associate the spec to the group. *
*             Any systems in the group that need to be *
*             added to the spec have already been done *
*             by CREATE of LNKxxCS records (spec to *
*             system link). If this is a LNKxxCG record, *
*             set the PARM and PARMLenGth. *
*             - Issue the proper format of EXEC CPSM CREATE *
*             (either with PARM/PARMLen or without). *
*             - Check the RESPONSE - if not OK, issue msgs *
*             and exit. *
*             - Return to caller. *
*-----*
      MVC    LENGTH,RECLen
      CLC    OBJECT(4),=CL4'LNKS'
      BNE    CRENOPRM
      CLC    OBJECT+6(2),=CL2'CG'
      BNE    CRENOPRM
      MVC    PARM,=CL5'NONE.'
      MVC    PARMLen,=F'5'
      EXEC   CPSM CREATE
*             OBJECT(OBJECT) *
*             FROM(0(,R2)) *
*             LENGTH(LENGTH) *
*             PARM(PARM) *
*             PARMLen(PARMLen) *
*             THREAD(THREAD) *
*             CONTEXT(CONTEXT) *
*             RESPONSE(RESPONSE) *

```



		REASON(REASON)		
CRENOPRM	B	DS	CRECHKRR	
	DS		0H	
	EXEC	CPSM	CREATE	X
			OBJECT(OBJECT)	X
			FROM(0(,R2))	X
			LENGTH(LENGTH)	X
			THREAD(THREAD)	X
			CONTEXT(CONTEXT)	X
			RESPONSE(RESPONSE)	X
			REASON(REASON)	
CRECHKRR	DS	0H		
	CLC		RESPONSE,EYUVALUE(OK)	
	BNE		ERRCREAT	
	BR		R10	
DISCARD	DS	0H		
*-----*				
*	Subroutine: DISCARD			*
*	Entry: Via BAS R10,DISCARD			*
*	Function: Issue the EXEC CPSM DISCARD to discard a result			*
*	set built by CPSM. In our program, both GET			*
*	and PERFORM build result sets.			*
*	Processing: - Issue EXEC CPSM DISCARD for the result set.			*
*	The result set id must be placed into			*
*	RESULTD by mainline.			*
*	- Check the RESPONSE - if not OK, issue msgs			*
*	and exit.			*
*	- Return to caller.			*
*-----*				
	EXEC	CPSM	DISCARD	X
			RESULT(RESULTD)	X
			THREAD(THREAD)	X
			RESPONSE(RESPONSE)	X
			REASON(REASON)	
	CLC		RESPONSE,EYUVALUE(OK)	
	BNE		ERRDISCA	
	BR		R10	
*-----*				
*	End of subroutines.			*
*-----*				
*-----*				
*	* Copy the CPSM definitions from OrigPlex to NewPlex			*
*				*
*	* Following is a list of all CPSM Resource Tables that will			*
*	be copied into NewPlex. The order that they are in (which			*
*	is the order they will be built in our program) is			*
*	important, since some Resource Tables will reference other			*
*	Resource Tables previously built. The order of the following			*
*	list is OK for the current release of CPSM.			*
*				*
*-----*				
DEFLIST	DS	0C		
	DC	CL8'PERIODEF'	Time period definitions	
	DC	CL8'ACTION '	RTA action definitions	
	DC	CL8'CSYSDEF '	CICS system definitions	
	DC	CL8'CSYSGRP '	CICS system group definitions	
	DC	CL8'CSGLCGCS'	CICS systems in groups links	
	DC	CL8'CSGLCGCG'	CICS groups in groups links	
	DC	CL8'MONDEF '	Monitor definitions	
	DC	CL8'MONGROUP'	MON group definitions	
	DC	CL8'MONSPEC '	MON specification definitions	
	DC	CL8'MONINGRP'	MON def in MON group links	
	DC	CL8'MONINSPC'	MON spec to MON group links	
	DC	CL8'LNKSMSCS'	MON spec to CICS system links	
	DC	CL8'LNKSMSCG'	MON spec to CICS group links	
	DC	CL8'EVALDEF '	RTA evaluation definitions	
	DC	CL8'RTADEF '	Real time analysis definitions	
	DC	CL8'STATDEF '	User status probe definitions	
	DC	CL8'RTAGROUP'	RTA group definitions	
	DC	CL8'RTASPEC '	RTA specification definitions	
	DC	CL8'RTAINGRP'	RTADEF in RTA group links	
	DC	CL8'STAINGRP'	STATDEF in RTA group links	
	DC	CL8'RTAINSPC'	RTA spec to RTA group links	
	DC	CL8'LNKSRSCS'	RTA spec to CICS group links	
	DC	CL8'LNKSRSCG'	RTA spec to CICS system links	
	DC	CL8'APSPEC '	RTA/APM specification defs	
	DC	CL8'RTAINAPS'	RTA/APM spec to RTA group links	
	DC	CL8'CMDMPAPS'	RTA spec to primary CMAS links	



R13	EQU	13
R14	EQU	14
R15	EQU	15
	END	EYUAAPI3

## EYULAPI4

プログラム EYULAPI4 は、CICS 環境用に COBOL で作成されます。

### EYULAPI4 について

このプログラムは、以下の処理を行います。

- API への接続を確立します。
- バージョン 1 を指定する TS モデル (TSMDEF) 用の BAS 定義を作成します。
- 前に定義された TSMDEF を含む結果セットを作成します。
- PERFORM OBJECT コマンドを発行して、TSMDEF をターゲット・スコープに INSTALL します。
- API 接続を終了させます。
- BAS エラーは、BINCONRS、BINCONSC、および BINSTERR リソース・テーブル・レコードを使用して処理されます。

**使用されているコマンド:** CONNECT、CREATE、GET、PERFORM OBJECT、FEEDBACK、FETCH、TERMINATE、TRANSLATE

```

IDENTIFICATION DIVISION.
PROGRAM-ID. EYULAPI4
*****
*
* MODULE NAME = EYULAPI4
*
* DESCRIPTIVE NAME = CPSM SAMPLE API PROGRAM 4
*                   (SAMPLE COBOL VERSION)
*
* COPYRIGHT = Licensed Materials - Property of IBM
*             5695-081
*             (C) Copyright IBM Corp. 1995, 1997
*             All Rights Reserved
*
*             US Government Users Restricted Rights - Use,
*             duplication or disclosure restricted by GSA ADP
*             Schedule Contract with IBM Corp.
*
* STATUS = %CP00
*
* FUNCTION =
*
* TO PROVIDE AN EXAMPLE OF THE USE OF THE FOLLOWING EXEC CPSM
* COMMANDS: CONNECT, CREATE, FEEDBACK, FETCH, GET,
*           PERFORM OBJECT, TERMINATE.
*
* WHEN INVOKED, THE PROGRAM DEPENDS UPON THE VALUES HELD IN THE
* W-CONTEXT AND W-SCOPE DECLARATIONS WHEN ESTABLISHING A
* CONNECTION WITH CICSplex SM. THEY MUST TAKE THE FOLLOWING
* VALUES:
*
* W-CONTEXT  = THE NAME OF A CMAS OR CICSplex. REFER TO THE
*              DESCRIPTION OF THE EXEC CPSM CONNECT COMMAND
*              FOR FURTHER INFORMATION REGARDING THE CONTEXT
*              OPTION.
*
* W-SCOPE    = THE NAME OF A CICSplex, CICS SYSTEM, OR CICS
*              SYSTEM GROUP WITHIN THE CICSplex. REFER TO THE
*              DESCRIPTION OF THE EXEC CPSM CONNECT COMMAND
*              FOR FURTHER INFORMATION REGARDING THE SCOPE
*              OPTION.
*
* THIS SAMPLE REQUIRES NO PARAMETERS AT INVOCATION TIME.
*
* WHEN CREATING THE BAS DEFINITION THE PROGRAM DEPENDS UPON THE
* VALUES HELD IN THE W-DEFNAME AND W-DEFPREFIX DECLARATIONS.
* THEY MUST TAKE THE FOLLOWING VALUES:
*

```

```

* W-DEFNAME = THE NAME OF THE CREATED BAS DEFINITION. A      *
*             1 TO 8 CHARACTER VALUE.                        *
*                                                     *
* W-DEFPFIX = THE MODEL PREFIX OF THE CREATED BAS DEFINITION.*
*             A 1 TO 16 CHARACTER VALUE.                    *
*                                                     *
*                                                     *

```

```

* WHEN INSTALLING THE BAS DEFINITION THE PROGRAM USES THE    *
* VALUE HELD IN THE W-TSCOPE DECLARATION AS THE TARGET FOR   *
* THE INSTALL OPERATION. IT MUST TAKE THE FOLLOWING VALUE :  *
*                                                     *
* W-TSCOPE = THE NAME OF A CICS SYSTEM, OR CICS              *
*            SYSTEM GROUP WITHIN THE CICSplex. REFER TO THE  *
*            DESCRIPTION OF THE TARGET PARAMETER OF AN        *
*            INSTALL ACTION IN THE RESOURCE TABLE REFERENCE *
*            FOR FURTHER INFORMATION REGARDING THE TARGET     *
*            SCOPE VALUE.                                     *
*                                                     *
* THE SAMPLE ESTABLISHES AN API CONNECTION AND ISSUES A CREATE *
* COMMAND TO CREATE A BAS DEFINITION. A GET COMMAND IS ISSUED *
* TO OBTAIN A RESULT SET CONTAINING THE CREATED BAS DEFINITION.*
*                                                     *
* USING THE PERFORM OBJECT ACTION(INSTALL) COMMAND EACH RECORD *
* IN THE RESULT SET IS INSTALLED INTO THE TARGET SCOPE        *
* IDENTIFIED BY THE W-SCOPE DECLARATION.                      *
*                                                     *
* FINALLY, THE API CONNECTION IS TERMINATED.                  *
*                                                     *
* ANY BAS ERRORS ARE REPORTED USING THE BINCONRS, BINCONSC, AND *
* BINSTERR RESOURCE TABLES.                                  *
*                                                     *

```

```

* NOTES :                                                     *
* DEPENDENCIES = S/390, CICS                                  *
* RESTRICTIONS = NONE                                         *
* REGISTER CONVENTIONS =                                       *
* MODULE TYPE = EXECUTABLE                                     *
* PROCESSOR = COBOL                                           *
* ATTRIBUTES = READ ONLY, SERIALY REUSABLE                    *
*                                                     *
* ----- *
* ENTRY POINT = EYULAPI4                                       *
*                                                     *
* PURPOSE = ALL FUNCTIONS.                                     *
*                                                     *
* LINKAGE = FROM CICS EITHER WITH EXEC CICS LINK OR AS A CICS *
* TRANSACTION.                                                 *
*                                                     *
* INPUT = NONE.                                                *
* ----- *
* ENVIRONMENT DIVISION.                                       *
* DATA DIVISION.                                             *
* WORKING-STORAGE SECTION.                                     *
* ----- *
* CHANGE W-CONTEXT AND W-SCOPE TO MATCH YOUR INSTALLATION    *
* CHANGE W-DEFNAME AND W-DEFPFIX FOR THE CREATE COMMAND.      *
* CHANGE W-TSCOPE FOR THE PERFORM OBJECT COMMAND.             *
* ----- *
01 W-CONTEXT          PIC X(8) VALUE 'RTGA  '.
01 W-SCOPE             PIC X(8) VALUE 'RTGA  '.
01 W-DEFNAME          PIC X(8) VALUE 'EYULAPI4'.
01 W-DEFPFIX          PIC X(16) VALUE 'EYUL*           '.
01 W-TSCOPE           PIC X(8) VALUE 'RTGF  '.
* ----- *

```

```

01 W-RESPONSE         PIC S9(8) USAGE BINARY.
01 W-REASON           PIC S9(8) USAGE BINARY.
01 W-BUFFER           PIC X(32767).
01 W-BUFFERLEN        PIC S9(8) COMP.
01 W-FBBUFF           PIC X(248).
01 W-FBTTKN           PIC S9(8) COMP.
01 W-THREAD           PIC S9(8) USAGE BINARY.
01 W-RESULT           PIC S9(8) USAGE BINARY.
01 W-RECCNT           PIC S9(8) USAGE BINARY.
01 W-CRITERIA         PIC X(80) VALUE SPACES.
01 W-CRITERIALEN      PIC S9(8) USAGE BINARY.

```

```

01 W-PARM          PIC X(80) VALUE SPACES.
01 W-PARMLN        PIC S9(8) USAGE BINARY.
01 W-MSG-TEXT.
  02 W-TEXT        PIC X(80) VALUE SPACES.
  02 W-LINECTL      PIC X(1) VALUE X'13'.
01 ARRAYS.
  02 CH8ARR        OCCURS 20 TIMES PIC X(8).
  02 FULLARR       OCCURS 60 TIMES PIC S9(8) COMP.
01 III            PIC S9(8) VALUE ZERO.
01 CODEV          PIC S9(8) COMP.
01 CHARV          PIC X(12).
01 LASTCMD        PIC X(20).
01 LASTTHR        PIC S9(8) COMP.
01 LASTRES        PIC S9(8) COMP VALUE 0.
01 BINZERO        PIC X(1) VALUE X'00'.
01 BLNKPAD        PIC X(40)
  VALUE '          '.
01 FBCHAR2        PIC X(2).
01 FBHALF4        REDEFINES FBCHAR2.
  02 FBHALF        PIC S9(4) COMP.
01 PICZZZ9A       PIC ZZZ9.
01 PICZZZ9B       PIC ZZZ9.
01 PICZZZ9        PIC ZZZ9.
01 PYCZZZ9        PIC ZZZ9.
01 PIKZZZ9        PIC ZZZ9.
01 PYKZZZ9        PIC ZZZ9.
01 PICZZZZZZZ9    PIC ZZZZZZZ9.
01 CHR8           PIC X(8).
01 CHR12          PIC X(12).
01 CHAR6          PIC X(6).
01 CHAR12         PIC X(12).
* Include the resource table copybooks...
COPY TSMDEF.
COPY FEEDBACK.
COPY BINCONRS.
COPY BINCONSC.
COPY BINSTERR.

```

```

*****
* Start of LINKAGE section *
*****
LINKAGE SECTION.

PROCEDURE DIVISION.
EYULAPI4-START SECTION.
EYULAPI4-00.

*-----*
*   OBTAIN A CPSM API CONNECTION.                               *
*   *                                                             *
*   THE API WILL RETURN A TOKEN IDENTIFYING THE THREAD IN      *
*   VARIABLE W-THREAD.                                         *
*-----*
*   MOVE 'Establishing Connection...' TO W-TEXT.
*   DISPLAY W-TEXT.
*   EXEC CICS SEND FROM(W-TEXT) LENGTH(81) ERASE END-EXEC.
*   EXEC CPSM CONNECT
*       CONTEXT(W-CONTEXT)
*       SCOPE(W-SCOPE)
*       VERSION('0140')
*       THREAD(W-THREAD)
*       RESPONSE(W-RESPONSE)
*       REASON(W-REASON)
*   END-EXEC.
*   IF W-RESPONSE NOT = EYUVALUE(OK) GO TO NO-CONNECT.

```

```

*-----*
*   CREATE A TS MODEL DEFINITION (TSMDEF)                       *
*   *                                                             *
*   A TSMDEF is created with a version of 1.                   *
*-----*
*   INITIALIZE TSMDEF.
*   MOVE X'01' TO DEFVER OF TSMDEF.
*   MOVE W-DEFNAME TO NAME-R OF TSMDEF.
*   MOVE W-DEFPFIX TO PREFIX OF TSMDEF.
*   MOVE DFHVALUE(AUXILIARY) TO LOCATION OF TSMDEF.
*   MOVE EYUVALUE(NO) TO RECOVERY OF TSMDEF.
*   MOVE EYUVALUE(NO) TO SECURITY-R OF TSMDEF.

```

```

        MOVE 'Sample TSMDEF definition' TO DESCRIPTION OF TSMDEF.
* Copy the definition into our buffer...
        MOVE TSMDEF TO W-BUFFER.
        MOVE TSMDEF-TBL-LEN TO W-BUFFERLEN.
        MOVE 'Creating TSMDEF...' TO W-TEXT.
*
        EXEC CICS SEND FROM(W-TEXT) LENGTH(81) WAIT END-EXEC.
        EXEC CPSM CREATE
              OBJECT('TSMDEF')
              FROM(W-BUFFER)
              LENGTH(W-BUFFERLEN)
              THREAD(W-THREAD)
              RESPONSE(W-RESPONSE)
              REASON(W-REASON)
        END-EXEC.
        MOVE 'CREATE' TO LASTCMD.
        MOVE W-THREAD TO LASTTHR.
        MOVE 0 TO LASTRES.
        IF W-RESPONSE NOT = EYUVALUE(OK) GO TO UNEXPECTED.

```

```

*-----*
*   GET THE TSMDEF RESOURCE TABLE.                                     *
*-----*
*   CREATE A RESULT SET CONTAINING ENTRIES FOR ALL TSMDEFS           *
*   WITH NAMES EQUAL TO THE VALUE OF W-DEFNAME. .                   *
*   THE NUMBER OF ENTRIES MEETING THE CRITERIA IS RETURNED          *
*   IN VARIABLE W-RECCNT.                                           *
*-----*
*   MOVE 'Get the created TSMDEF Resource Table...' TO W-TEXT.
*   DISPLAY W-TEXT.
        EXEC CICS SEND FROM(W-TEXT) LENGTH(81) WAIT END-EXEC.
        STRING 'NAME=' DELIMITED BY SIZE
              W-DEFNAME DELIMITED BY SIZE
              '.' DELIMITED BY SIZE
              INTO W-CRITERIA.
        MOVE LENGTH OF W-CRITERIA TO W-CRITERIALEN.
        MOVE BINZERO TO W-RESULT.
        EXEC CPSM GET OBJECT('TSMDEF')
              CRITERIA(W-CRITERIA)
              LENGTH(W-CRITERIALEN)
              COUNT(W-RECCNT)
              RESULT(W-RESULT)
              THREAD(W-THREAD)
              RESPONSE(W-RESPONSE)
              REASON(W-REASON)
        END-EXEC.
        IF W-RESPONSE NOT = EYUVALUE(OK) GO TO NO-GET.

```

```

*-----*
*   INSTALL EACH RECORD INTO THE SCOPE IDENTIFIED BY THE           *
*   VALUE OF W-TSCOPE.                                             *
*-----*
        MOVE W-RECCNT TO PICZZZZZZ9.
        STRING 'Installing ' DELIMITED BY SIZE
              PICZZZZZZ9 DELIMITED BY SIZE
              ' TSMDEF Entries...' DELIMITED BY SIZE
              INTO W-TEXT.
*   DISPLAY W-TEXT
        EXEC CICS SEND FROM(W-TEXT) LENGTH(81) WAIT END-EXEC.
        STRING '(USAGE(LOCAL) TARGET(' DELIMITED BY SIZE
              W-TSCOPE DELIMITED BY SIZE
              ')).' DELIMITED BY SIZE
              INTO W-PARM.
        MOVE LENGTH OF W-PARM TO W-PARMLEN.

        EXEC CPSM PERFORM OBJECT('TSMDEF')
              ACTION('INSTALL')
              PARM(W-PARM)
              PARMLEN(W-PARMLEN)
              RESULT(W-RESULT)
              THREAD(W-THREAD)
              RESPONSE(W-RESPONSE)
              REASON(W-REASON)
        END-EXEC.
        MOVE 'PERFORM OBJECT' TO LASTCMD.
        MOVE W-THREAD TO LASTTHR.
        MOVE W-RESULT TO LASTRES.
        IF W-RESPONSE NOT = EYUVALUE(OK) GO TO UNEXPECTED.

```

```

MOVE 'Completed. Remove TSMDEF to re-run.' TO W-TEXT.
GO TO SCRNLG2.

```

```

*****
* Branch here if an unexpected CPSM error occurs *
*****
UNEXPECTED.
    MOVE W-RESPONSE TO PICZZZ9.
    STRING '*** RESPONSE=' DELIMITED BY SIZE PICZZZ9
    DELIMITED BY SIZE BLNKPAD DELIMITED BY SIZE INTO W-TEXT.
    PERFORM SCRNLG2.
    MOVE W-REASON TO PICZZZ9.
    STRING '*** REASON=' DELIMITED BY SIZE PICZZZ9
    DELIMITED BY SIZE BLNKPAD DELIMITED BY SIZE INTO W-TEXT.
    PERFORM SCRNLG2.
    MOVE '*** Unexpected error condition arose' TO W-TEXT.
    PERFORM SCRNLG2.
* Obtain FEEDBACK information
    IF LASTCMD = 'DISCONNECT' GO TO NOFEED.
    IF LASTCMD = 'FEEDBACK' GO TO NOFEED.
    IF LASTCMD = 'TERMINATE' GO TO NOFEED.
    STRING
    '*** Getting FEEDBACK data for ' DELIMITED BY SIZE
    LASTCMD DELIMITED BY SIZE
    INTO W-TEXT.
    PERFORM SCRNLG2.
    STRING
    BLNKPAD DELIMITED BY SIZE
    BLNKPAD DELIMITED BY SIZE
    INTO W-TEXT.
* Get the FEEDBACK data
    GETFEED.
* Clear error result set count
    MOVE 0 TO FULLARR(1).
    PERFORM GETFB THROUGH EGETFB
* Display FEEDBACK information
* Display information
    IF W-RESPONSE = EYUVALUE(OK)
        PERFORM DISPFEED
        IF FULLARR(1) NOT = 0 PERFORM GETFERT THROUGH EGETFER END-I
-F
    IF LASTRES NOT = 0 GO TO GETFEED END-IF
    MOVE '*** End of FEEDBACK data' TO W-TEXT
    PERFORM SCRNLG2
    GO TO NOFEED
END-IF.
    MOVE W-RESPONSE TO PICZZZ9.
    MOVE W-REASON TO PYCZZZ9.
    STRING '*** FEEDBACK not available (' DELIMITED BY SIZE
    PICZZZ9 DELIMITED BY SIZE ',' DELIMITED BY SIZE
    PYCZZZ9 DELIMITED BY SIZE ')' DELIMITED BY SIZE
    BLNKPAD DELIMITED BY SIZE INTO W-TEXT END-STRING.
    PERFORM SCRNLG2.
NOFEED.
    EXEC CICS DELAY FOR SECONDS(10) END-EXEC.
* Exit from test case
    EXEC CICS RETURN END-EXEC.
    GOBACK.
    EXIT.

```

```

*****
* This subroutine obtains the FEEDBACK data *
*****
GETFB.
* Use exact buffer size
    MOVE FEEDBACK-TBL-LEN TO W-BUFFERLEN.
    IF LASTRES = 0 GO TO NORESULT.
RESULT.
    EXEC CPSM FEEDBACK
        INTO(W-FBUFF) LENGTH(W-BUFFERLEN)
        RESULT(LASTRES)
        THREAD(LASTTHR)
        RESPONSE(W-RESPONSE)
        REASON(W-REASON)
    END-EXEC.

* If command didn't execute, get FEEDBACK no result set
* Command didn't execute?

```

```

        IF W-RESPONSE = EYUVALUE(NODATA)
            MOVE 0 TO LASTRES
            GO TO NORESULT
        END-IF.
        GO TO ENDFBACK.
    NORESULT.
* Use exact buffer size
    MOVE FEEDBACK-TBL-LEN TO W-BUFFERLEN.
    EXEC CPSM FEEDBACK
        INTO(W-FBBUFF) LENGTH(W-BUFFERLEN)
        THREAD(LASTTHR)
        RESPONSE(W-RESPONSE)
        REASON(W-REASON)
    END-EXEC.

    ENDFBACK.
    EGETFB.
    EXIT.

*****
* Branch here if FEEDBACK Error Result Token available *
*****
    GETFERT.
        MOVE ERR-OBJECT OF FEEDBACK TO CH8ARR(1).
        STRING
            '*** Getting ' DELIMITED BY SIZE
            CH8ARR(1) DELIMITED BY SIZE
            ' error result set data for FEEDBACK' DELIMITED BY SIZE
        INTO W-TEXT.
        PERFORM SCRNLG2.
    FERTRES.
* Use largest buffer size
    MOVE FEEDBACK-TBL-LEN TO W-BUFFERLEN.
    EXEC CPSM FETCH
        INTO(W-BUFFER) LENGTH(W-BUFFERLEN)
        RESULT(ERR-RESULT OF FEEDBACK)
        THREAD(LASTTHR)
        RESPONSE(W-RESPONSE)
        REASON(W-REASON)
    END-EXEC.

* Display FEEDBACK Error Result Token information
* Display information
    IF W-RESPONSE = EYUVALUE(OK)
        IF CH8ARR(1)= 'FEEDBACK'
            MOVE W-BUFFER TO W-FBBUFF
            PERFORM DISPFEEED
        END-IF
        IF CH8ARR(1)= 'BINSTERR'
            PERFORM DISPBIER
        END-IF
        IF CH8ARR(1)= 'BINCONRS'
            PERFORM DISPBIRS
        END-IF
        IF CH8ARR(1)= 'BINCONSC'
            PERFORM DISPBISC
        END-IF
        GO TO FERTRES
    END-IF.
    MOVE W-RESPONSE TO PICZZZ9.
    MOVE W-REASON TO PYCZZZ9.
    STRING '*** FEEDBACK not available (' DELIMITED BY SIZE
    PICZZZ9 DELIMITED BY SIZE ',' DELIMITED BY SIZE
    PYCZZZ9 DELIMITED BY SIZE ')' DELIMITED BY SIZE
    BLNKPAD DELIMITED BY SIZE INTO W-TEXT END-STRING.
    PERFORM SCRNLG2.
    EGETFER.
    EXIT.

*****
* This subroutine displays FEEDBACK information *
*****
    DISPFEEED.
        MOVE W-FBBUFF TO FEEDBACK.
        STRING BINZERO COMMAND OF FEEDBACK DELIMITED BY SIZE
        INTO FBCHAR2.
        MOVE FBHALF TO PICZZZ9.
        MOVE RESPONSE OF FEEDBACK TO PYCZZZ9.

```



```

MOVE REASON OF FEEDBACK TO PIKZZZ9.
MOVE RSLTRECID OF FEEDBACK TO PYKZZZ9.
MOVE SPACES TO W-TEXT.
STRING 'Cmd=' PICZZZ9 ' Attr=' ATTRDATAVAL OF
FEEDBACK ' Eib=' CEIBDATAVAL OF FEEDBACK ' Err='
ERRCODEVAL OF FEEDBACK ' Rspn=' PYCZZZ9 ' Reas='
PIKZZZ9 ' ResId=' PYKZZZ9
DELIMITED BY SIZE INTO W-TEXT END-STRING.
PERFORM SCRNLG2.
MOVE ERROR-CODE OF FEEDBACK TO PICZZZ9.
MOVE CEIBRESP OF FEEDBACK TO PYCZZZ9.
MOVE CEIBRESP1 OF FEEDBACK TO PIKZZZ9.
MOVE CEIBFN OF FEEDBACK TO PYKZZZ9.
MOVE SPACES TO W-TEXT.
STRING ' ECode=' PICZZZ9 ' RESP=' PYCZZZ9
' RESP1=' PIKZZZ9 ' EibFn=' PYKZZZ9 ' Obj='
OBJECT-A OF FEEDBACK ' OAct=' OBJECT-ACT OF FEEDBACK
DELIMITED BY SIZE INTO W-TEXT END-STRING.
PERFORM SCRNLG2.
MOVE SPACES TO W-TEXT.
STRING ' Att1=' ATTR-NM1 OF FEEDBACK ' 2='
ATTR-NM2 OF FEEDBACK ' 3=' ATTR-NM3 OF FEEDBACK
' 4=' ATTR-NM4 OF FEEDBACK ' 5=' ATTR-NM5 OF
FEEDBACK DELIMITED BY SIZE INTO W-TEXT END-STRING.
PERFORM SCRNLG2.
MOVE ERR-COUNT OF FEEDBACK TO PICZZZ9.
MOVE SPACES TO W-TEXT.
STRING ' FObj=' ERR-OBJECT OF FEEDBACK
' FCnt=' PICZZZ9
DELIMITED BY SIZE INTO W-TEXT END-STRING.
PERFORM SCRNLG2.
MOVE ERR-COUNT OF FEEDBACK TO FULLARR(1).
EXIT.

```

```

*****
* This subroutine displays BINSTERR information *
*****
DISPBIER.
MOVE W-BUFFER TO BINSTERR.
MOVE SPACES TO W-TEXT.
STRING 'CMAS=' CMASNAME OF BINSTERR ' Plex='
PLEXNAME OF BINSTERR ' CSys=' CICSNAME OF BINSTERR
' ResName=' RESNAME OF BINSTERR
DELIMITED BY SIZE INTO W-TEXT END-STRING.
PERFORM SCRNLG2.
MOVE RESVER OF BINSTERR TO PICZZZ9.
MOVE ERRCODE OF BINSTERR TO PYCZZZ9.
MOVE CRESP1 OF BINSTERR TO PIKZZZ9.
MOVE CRESP2 OF BINSTERR TO PYKZZZ9.
MOVE SPACES TO W-TEXT.
STRING ' ResVer=' PICZZZ9 ' ECode=' PYCZZZ9
' RESP=' PIKZZZ9 ' RESP1=' PYKZZZ9
DELIMITED BY SIZE INTO W-TEXT END-STRING.
PERFORM SCRNLG2.
MOVE CEIBFN OF BINSTERR TO PICZZZ9.
MOVE SPACES TO W-TEXT.
STRING ' EibFn=' PICZZZ9
DELIMITED BY SIZE INTO W-TEXT END-STRING.
PERFORM SCRNLG2.
EXIT.

```

```

*****
* This subroutine displays BINCONRS information *
*****
DISPBIRS.
MOVE W-BUFFER TO BINCONRS.
MOVE ERROP OF BINCONRS TO PICZZZ9.
MOVE SPACES TO W-TEXT.
STRING 'CMAS=' CMASNAME OF BINCONRS ' Plex='
PLEXNAME OF BINCONRS ' CSys=' CICSNAME OF BINCONRS
' ResType=' RESTYPE OF BINCONRS ' EOp=' PICZZZ9
DELIMITED BY SIZE INTO W-TEXT END-STRING.
PERFORM SCRNLG2.
MOVE CANDVER OF BINCONRS TO PICZZZ9.
MOVE SPACES TO W-TEXT.
STRING ' CandName=' CANDNAME OF BINCONRS
' CandVer=' PICZZZ9 ' CResGrp=' CANDRGRP OF BINCONRS
' CResAss=' CANDRASG OF BINCONRS ' CResDes='

```

```

CANDRDSC OF BINCONRS
DELIMITED BY SIZE INTO W-TEXT END-STRING.
PERFORM SCRNLG2.
MOVE CANDUSAGE OF BINCONRS TO CODEV.
MOVE 'BINCONRS' TO CHR8.
MOVE 'CANDUSAGE' TO CHR12.
PERFORM XCV2CH
MOVE CHARV TO CHAR6.
MOVE CANDTYPE OF BINCONRS TO CODEV.
MOVE 'BINCONRS' TO CHR8.
MOVE 'CANDTYPE' TO CHR12.
PERFORM XCV2CH
MOVE CHARV TO CHAR12.
MOVE CANDASGOVR OF BINCONRS TO CODEV.
MOVE 'BINCONRS' TO CHR8.
MOVE 'CANDASGOVR' TO CHR12.
PERFORM XCV2CH
MOVE SPACES TO W-TEXT.
STRING ' CandUsa=' CHAR6
' CandSGrp=' CANDSGRP OF BINCONRS
' CandSTyp=' CHAR12 ' CandAss0=' CHARV
DELIMITED BY SIZE INTO W-TEXT END-STRING.
PERFORM SCRNLG2.
MOVE EXISTVER OF BINCONRS TO PICZZZ9.
MOVE EXISTUSAGE OF BINCONRS TO CODEV.
MOVE 'BINCONRS' TO CHR8.
MOVE 'EXISTUSAGE' TO CHR12.
PERFORM XCV2CH
MOVE SPACES TO W-TEXT.
STRING ' ExistName=' EXISTNAME OF BINCONRS
' ExistVer=' PICZZZ9 ' EResGrp=' EXISTRGRP OF
BINCONRS ' EResAss=' EXISTRASG OF BINCONRS
' EResDes=' EXISTRDSC OF BINCONRS ' ExistUsa=' CHARV
DELIMITED BY SIZE INTO W-TEXT END-STRING.
PERFORM SCRNLG2.
MOVE EXISTTYPE OF BINCONRS TO CODEV.
MOVE 'BINCONRS' TO CHR8.
MOVE 'EXISTTYPE' TO CHR12.
PERFORM XCV2CH
MOVE CHARV TO CHAR12.
MOVE EXISTASGOVR OF BINCONRS TO CODEV.
MOVE 'BINCONRS' TO CHR8.
MOVE 'EXISTASGOVR' TO CHR12.
PERFORM XCV2CH
MOVE SPACES TO W-TEXT.
STRING ' ExistSGrp=' EXISTSGRP OF BINCONRS
' ExistSTyp=' CHAR12 ' ExistAss0=' CHARV
DELIMITED BY SIZE INTO W-TEXT END-STRING.
PERFORM SCRNLG2.
EXIT.

```

```

*****
* This subroutine displays BINCONSC information *
*****
DISPBISC.
MOVE W-BUFFER TO BINSTERR.
MOVE ERRORP OF BINCONSC TO PICZZZ9.
MOVE ERRCODE OF BINCONSC TO PYCZZZ9.
MOVE SPACES TO W-TEXT.
STRING 'CMAS=' CMASNAME OF BINCONSC ' Plex='
PLEXNAME OF BINCONSC ' EOp=' PICZZZ9 ' ECode='
PYCZZZ9 ' TScope=' TARGSCOPE OF BINCONSC
' TAssgn=' TARGRASG OF BINCONSC
DELIMITED BY SIZE INTO W-TEXT END-STRING.
PERFORM SCRNLG2.
MOVE SPACES TO W-TEXT.
STRING ' TDesc=' TARGRDSC OF BINCONSC ' RScope='
RELScope OF BINCONSC ' RAssgn=' RELRASG OF BINCONSC
' RDesc=' RELRDSC OF BINCONSC ' CSys=' CICSNAME OF
BINCONSC
DELIMITED BY SIZE INTO W-TEXT END-STRING.
PERFORM SCRNLG2.
EXIT.

```

```

*****
* This subroutine converts coded value to character string *
*****
XCV2CH.

```

```

* Use new thread for TRANSLATE
  EXEC CPSM CONNECT
        VERSION('0140')
        THREAD(W-FBTTKN)
        RESPONSE(W-RESPONSE)
        REASON(W-REASON)
  END-EXEC.

* Translate internal coded value to character value
  EXEC CPSM TRANSLATE
        OBJECT(CHR8)
        ATTRIBUTE(CHR12)
        FROMCV(CODEV) TOCHAR(CHARV)
        THREAD(W-FBTTKN)
        RESPONSE(W-RESPONSE)
        REASON(W-REASON)
  END-EXEC.
  EXIT.

*-----*

*      PROCESSING FOR API FAILURES.                                *
*-----*
NO-CONNECT.
  MOVE 'ERROR CONNECTING TO API.' TO W-MSG-TEXT.
  GO TO SCRNLG.
NO-CREATE.
  MOVE 'ERROR CREATING DEFINITION.' TO W-MSG-TEXT.
  GO TO SCRNLG.
NO-GET.
  MOVE 'ERROR GETTING RESOURCE TABLE.' TO W-MSG-TEXT.
  GO TO SCRNLG.
NO-INSTALL.
  MOVE 'ERROR INSTALLING RESULT SET.' TO W-MSG-TEXT.
  GO TO SCRNLG.
NO-TRANSLATE.
  MOVE 'ERROR TRANSLATING ATTRIBUTE.' TO W-MSG-TEXT.
  GO TO SCRNLG.
SCRNLG.
*  DISPLAY W-MSG-TEXT.
  EXEC CICS SEND FROM(W-MSG-TEXT) LENGTH(81) WAIT END-EXEC.
  MOVE W-RESPONSE TO PICZZZ9A.
  MOVE W-REASON TO PICZZZ9B.
  STRING 'RESPONSE=' DELIMITED BY SIZE
        PICZZZ9A DELIMITED BY SIZE
        ' REASON= ' DELIMITED BY SIZE
        PICZZZ9B DELIMITED BY SIZE
        INTO W-MSG-TEXT.

SCRNLG2.
*  DISPLAY W-MSG-TEXT.
  EXEC CICS SEND FROM(W-MSG-TEXT) LENGTH(81) WAIT END-EXEC.

  ENDIT.
*-----*
*      TERMINATE API CONNECTION.                                *
*-----*
  EXEC CPSM TERMINATE RESPONSE(W-RESPONSE) REASON(W-REASON)
  END-EXEC.
  EXEC CICS RETURN END-EXEC.
*  GOBACK
  EXIT.
EYULAPI4-END.

```

EYUxAPI4 の COBOL バージョンは、CICS 環境用に作成されているため、**EXEC CICS SEND** コマンドをコメント化し、上記の言語固有の出力ステートメントをアンコメントして、MVS/ESA バッチ環境で実行するよう変換することができます。



## 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せん。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涉外

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を提供し、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。

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をお願いします。

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を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 6 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](#)
- [システム・プログラムの開発](#)
- [CICS TS セキュリティー](#)
- [外部インターフェースに向けた開発](#)
- [アプリケーション開発のリファレンス](#)
- [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 6 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。

- [トラブルシューティングおよびサポート](#)
- [CICS TS 診断参照](#)

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 6 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
- CICS における Java™ アプリケーション

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 6 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]

- アクション、実行 [36](#)
- アクションの実行 [36](#)
- アセンブラ言語プログラム
  - サポート対象環境 [2](#)
- アセンブラ言語プログラムプログラム
  - 言語に関する考慮事項 [78](#)
  - コンパイル [81](#)
  - 変換 [80](#)
  - ランタイムの考慮事項 [84](#)
  - リソース・テーブルのコピーブックの使用 [71](#)
  - リンク・エディット [83](#)
- イベント、listen [46](#)
- イベント制御ブロック (ECB)
  - 説明 [47](#)
- イベントの listen [46](#)
- エラー結果セット
  - 説明 [92](#)
  - BAS 定義の [94](#), [95](#)
  - CICS 定義の更新のための [93](#)
  - CICS リソースのインストールでの [93](#)
  - FEEDBACK レコードのフィールド [91](#)
- エラー・コード [109](#)
- エラー処理
  - エラー結果セットの使用 [92](#)
  - FEEDBACK データの使用 [88](#)
  - MASQRYER データの使用 [97](#)
  - RESPONSE と REASON の使用 [85](#)
  - REXX プログラムでの [105](#)
- 応答、コマンド
  - テスト
    - コマンド・レベル・インターフェースの使用 [87](#)
- 応答、コマンド
  - タイプ [85](#)
  - テスト
    - ランタイム・インターフェースの使用 [88](#)
- オブジェクト、CICSplex SM によって管理
  - 選択 [16](#)
  - タイプ [11](#)
  - 変更 [35](#)

## [カ行]

- 可用性、CICS リリース [1](#)
- 環境
  - 考慮事項 [79](#)
  - 互換性 [6](#)
  - サポート [1](#)
- 関数パッケージ、REXX [99](#)
- 管理対象オブジェクト
  - 選択 [16](#)
  - タイプ [11](#)
  - 変更 [35](#)

- 管理対象オブジェクトの選択
  - コンテキストおよびスコープの使用 [16](#)
  - フィルター式の使用 [17](#)
- 管理対象の CICS リソース
  - 説明 [11](#)
  - リソース・テーブル [13](#)
- 結果セット
  - コマンド
    - 概要 [21](#)
  - 作成 [20](#)
  - 説明 [20](#)
  - レコード
    - 位置指定 [27](#)
    - カスタマイズ [14](#)
    - 集計 [30](#)
    - 取得 [24](#)
    - ソート [34](#)
    - フィルタリング [17](#)
  - レコード・ポインタの配置 [27](#)
- 結果セット、エラー
  - 説明 [92](#)
  - BAS 定義の [94](#), [95](#)
  - CICS 定義の更新のための [93](#)
  - CICS リソースのインストールでの [93](#)
  - FEEDBACK レコードのフィールド [91](#)
- 結果セット・レコードの位置指定 [27](#)
- 結果セット・レコードの集計 [30](#)
- 結果セット・レコードの取得 [24](#)
- 結果セット・レコードの順序付け [34](#)
- 結果セット・レコードのソート [34](#)
- 結果セット・レコードのフィルタリング [17](#)
- 言語に関する考慮事項
  - アセンブラ [78](#)
  - PL/I [79](#)
- コピーブック、リソース・テーブル
  - アクセス [69](#)
  - アセンブラ [71](#)
  - 形式 [70](#)
  - 説明 [69](#)
  - データ特性 [70](#)
  - 名前と別名 [69](#)
  - BINCONRS [94](#)
  - BINCONSC [95](#)
  - BINSTERR [93](#)
  - C [77](#)
  - COBOL [74](#)
  - PL/I [72](#)
- コマンド応答
  - タイプ [85](#)
  - テスト
    - コマンド・レベル・インターフェースの使用 [87](#)
    - ランタイム・インターフェースの使用 [88](#)
- コマンド・レベル・インターフェース
  - 環境に関する考慮事項 [78](#)
  - 言語に関する考慮事項 [78](#)
  - サポート対象環境 [2](#)
  - プログラムのコンパイル [81](#)

コマンド・レベル・インターフェース (続き)  
プログラムの変換 [80](#)  
プログラムのリンク・エディット [82](#)  
ランタイムの考慮事項 [84](#)  
リソース・テーブルのコピーブックの使用 [69](#)  
コマンド・レベル・プログラムのコンパイル [81](#)  
コマンド・レベル・プログラムのリンク・エディット [82](#)  
コンテキスト  
コマンドでの指定 [17](#)  
説明 [16](#)

## [サ行]

サポート対象環境 [1](#)  
サンプル・プログラム  
記述 [10](#)  
提供対象のリスト [10](#)  
リスト [111](#)  
式  
順序 [15, 34](#)  
属性  
フィルター式内 [18](#)  
変更式内 [35](#)  
フィルター (filter) [17](#)  
変更 [35](#)  
parameter [36, 38](#)  
summary [32](#)  
集計オプション  
説明 [32](#)  
集計結果セット  
説明 [31](#)  
集計式  
説明 [32](#)  
状況プログラム  
CICSplex SM API [80](#)  
状況プログラム内の CICSplex SM API [80](#)  
スコープ  
コマンドでの指定 [17](#)  
説明 [16](#)  
制限されたリソース・テーブル属性 [14](#)  
セキュリティ  
考慮事項 [5](#)  
属性、リソース・テーブル  
順序付け [14](#)  
変換  
REXX プログラムでの [102](#)  
変更 [35](#)  
属性式  
フィルター式内 [18](#)  
変更式内 [35](#)

## [タ行]

タスク関連ユーザー出口 [67](#)  
ダンプ  
要求 [36](#)  
通知、CICSplex SM  
処理 [46](#)  
説明 [12](#)  
リソース・テーブル [13](#)  
定義、CICS  
作業 [37](#)  
説明 [12](#)

定義、CICSplex SM  
作業 [37](#)  
説明 [12](#)  
リソース・テーブル [13](#)  
統合 CICS 変換プログラム [80](#)  
トークン  
ユーザー定義 [46](#)  
CICSplex SM [48](#)

## [ハ行]

配列式  
説明 [15, 34](#)  
パラメーター式  
アクションを実行するタイミング [36](#)  
CICS 定義 [38](#)  
CICS 定義用 [29](#)  
CICSplex SM 定義 [38](#)  
非同期処理  
概要 [45](#)  
トークンの使用 [46](#)  
ADDRESS の使用 [47](#)  
LISTEN の使用 [46](#)  
NOWAIT の使用 [46](#)  
RECEIVE の使用 [47](#)  
ビュー  
説明 [14](#)  
標識フィールド  
説明 [47](#)  
フィルター (filter)  
説明 [17](#)  
フィルター式  
説明 [17](#)  
総称値 [19](#)  
プログラム、サンプル  
記述 [10](#)  
提供対象のリスト [10](#)  
リスト [111](#)  
変換  
コマンド・レベル・プログラム [80](#)  
リソース・テーブル属性  
REXX プログラムでの [102](#)  
RESPONSE 値および REASON 値  
コマンド・レベル・インターフェースの使用 [87](#)  
ランタイム・インターフェースの使用 [88](#)  
変換エラー、REXX [105](#)  
変更式  
説明 [35](#)

## [マ行]

メタデータ、CICSplex SM  
説明 [13](#)  
リソース・テーブル [13](#)

## [ヤ行]

ユーザー置き換え可能プログラム  
CICSplex SM API [79](#)  
ユーザー置き換え可能プログラム内の CICSplex SM API [79](#)  
ユーザー・トークン [46](#)  
要求のスケジューリング [46](#)

## [ラ行]

ランタイム・エラー、REXX [106](#)  
ランタイムの考慮事項、コマンド・レベル [84](#)  
リソース・テーブル  
「SCOPE applies」フィールド [16](#)  
カスタマイズ [14](#)  
コピーブック [69](#)  
コマンド・レベル・インターフェースでの使用 [69](#)  
制限付き属性 [14](#)  
説明 [13](#)  
属性の変換  
REXX プログラムでの [102](#)  
ビュー [14](#)  
REXX との使用 [100](#)  
リソース・テーブルのコピーブック  
アクセス [69](#)  
アセンブラー [71](#)  
形式 [70](#)  
説明 [69](#)  
データ特性 [70](#)  
名前と別名 [69](#)  
BINCONRS [94](#)  
BINCONSC [95](#)  
BINSTERR [93](#)  
C [77](#)  
COBOL [74](#)  
PL/I [72](#)  
リソース・テーブル・レコードのカスタマイズ [14](#)  
リリース属性の変更 [35](#)  
リリースの互換性 [6](#)  
レコードの展開  
集計結果セット [31](#)  
レコード・ポインター、配置 [27](#)  
ローカル・ファイル  
無効にする [36](#)

## A

API プログラムの互換性  
環境間 [6](#)  
リリース間 [6](#)  
API プログラムのマイグレーション [6](#)  
ASYNCREQ レコード  
取得 [47](#)  
説明 [46](#)  
ASYNCREQ レコードの取り出し [47](#)

## B

BINCONRS リソース・テーブル・レコード [94, 95](#)  
BINCONSC リソース・テーブル・レコード [94, 95](#)  
BINSTERR リソース・テーブルのレコード [93](#)

## C

C プログラム  
コンパイル [82](#)  
サポート対象環境 [2](#)  
での実行 [79](#)  
変換 [81](#)  
ランタイムの考慮事項 [84](#)  
リソース・テーブルのコピーブックの使用 [77](#)

C プログラム (続き)  
リンク・エディット [83](#)  
C++ プログラム  
コンパイル [82](#)  
CHANGEAGENT 属性  
説明 [38](#)  
CHANGEAGREL 属性  
説明 [38](#)  
CHANGETIME 属性  
説明 [14, 38](#)  
REXX を使用した処理 [102](#)  
CHANGEUSRID 属性  
説明 [38](#)  
CICS グローバル・ユーザー出口プログラム [79](#)  
CICS 定義  
作業 [37](#)  
説明 [12](#)  
CICS 定義の変更 [37](#)  
CICS リソース、管理対象  
説明 [11](#)  
リソース・テーブル [13](#)  
CICS リリース可用性 [1](#)  
CICSplex SM API タスク関連ユーザー出口 [67](#)  
CICSplex SM 通知  
処理 [46](#)  
説明 [12](#)  
リソース・テーブル [13](#)  
CICSplex SM 定義  
作業 [37](#)  
説明 [12](#)  
リソース・テーブル [13](#)  
CICSplex SM 定義の変更 [37](#)  
CICSplex SM トークン [48](#)  
CICSplex SM へのアクセス [2](#)  
CICSplex SM への接続 [2](#)  
CICSplex SM マネージャー・リソース  
説明 [12](#)  
リソース・テーブル [13](#)  
CICSplex SM メタデータ  
説明 [13](#)  
リソース・テーブル [13](#)  
COBOL プログラム  
コンパイル [82](#)  
サポート対象環境 [2](#)  
変換 [81](#)  
ランタイムの考慮事項 [84](#)  
リソース・テーブルのコピーブックの使用 [74](#)  
リンク・エディット [83](#)  
CONNECT コマンド  
使用 [3](#)  
CREATETIME 属性  
説明 [38](#)  
REXX を使用した処理 [102](#)  
CRESxxxx リソース・テーブル [67](#)

## E

ECB フィールド  
説明 [47](#)  
ERR\_RESULT トークン [103](#)  
EXPAND コマンド [31](#)  
EYU\_ 属性 [14](#)  
EYU\_TRACE ステム変数 [107](#)  
EYU9XESV セキュリティー・ルーチン

EYU9XESV セキュリティー・ルーチン (続き)  
考慮事項 [5](#)  
EYU9XLAP [67](#)  
EYUAPI 関数  
使用 [100](#)  
EYUREAS 関数  
使用 [88](#)  
EYURESP 関数  
使用 [88](#)  
EYUTERM 関数  
使用 [99](#)  
EYUVALUE 関数  
応答と理由に対する使用 [87](#)

## F

FEEDBACK コマンド  
使用 [88](#)  
FEEDBACK 属性  
REXX を使用した処理 [103](#)  
feedback レコード  
可用性 (availability) [91](#)  
サンプル [92](#)  
取得 [88](#)  
説明 [89](#)  
location [88](#)  
FEEDBACK レコードの取り出し [88](#)  
FETCH コマンド  
使用 [24](#)

## G

GROUP コマンド  
使用 [30](#)

## L

LISTEN コマンド  
使用 [46](#)  
LOCATE コマンド  
使用 [27](#)  
LOCFILE  
無効にする [36](#)

## M

MARK コマンド  
使用 [28](#)  
MASQRYER コマンド  
使用 [97](#)  
masqryer レコード  
可用性 (availability) [98](#)  
説明 [97](#)  
MASQRYER レコードの取り出し [97](#)

## N

NOWAIT オプション、使用 [46](#)

## O

OBJSTAT リソース・テーブルのレコード [25](#)

OBJSTAT レコード  
集計結果セット内 [31](#)  
取得 [24](#)  
説明 [24](#)  
OBJSTAT レコードの取得 [24](#)  
ORDER コマンド  
使用 [34](#)

## P

PERFORM OBJECT コマンド  
パラメーター式を使用する [36](#)  
PL/I プログラム  
言語に関する考慮事項 [79](#)  
コンパイル [82](#)  
サポート対象環境 [2](#)  
変換 [81](#)  
ランタイムの考慮事項 [84](#)  
リソース・テーブルのコピーブックの使用 [72](#)  
リンク・エディット [83](#)

## R

REASON オプション  
使用 [85](#)  
RECEIVE コマンド  
使用 [47](#)  
RESPONSE オプション  
使用 [85](#)  
REXX から API へのアクセス [99](#)  
REXX からのリソース・テーブルへのアクセス [100](#)  
REXX 関数パッケージ [99](#)  
REXX 処理  
CHANGETIME 属性 [102](#)  
CREATETIME 属性 [102](#)  
FEEDBACK 属性 [103](#)  
REXX ランタイム・インターフェース  
関数パッケージ [99](#)  
サポート対象環境 [2](#)  
使用 [99](#)  
変換エラー [105](#)  
メッセージ [106](#)  
ランタイム・エラー [106](#)  
リソース・テーブルへのアクセス [100](#)  
EYU\_TRACE データ [107](#)  
STATUS 値 [106](#)

## S

SPECIFY VIEW コマンド  
使用 [14](#)  
STATUS 値、解釈 [106](#)

## T

TBUILD コマンド  
使用 [100](#)  
処理エラー [106](#)  
TPARSE コマンド  
使用 [100](#)  
処理エラー [106](#)

## U

UNMARK コマンド  
使用 [28](#)

## W

Web ユーザー・インターフェース  
フィルター式 [19](#)

## X

XICEREQ [79](#)







