Tivoli IBM Tivoli NetView for z/OS

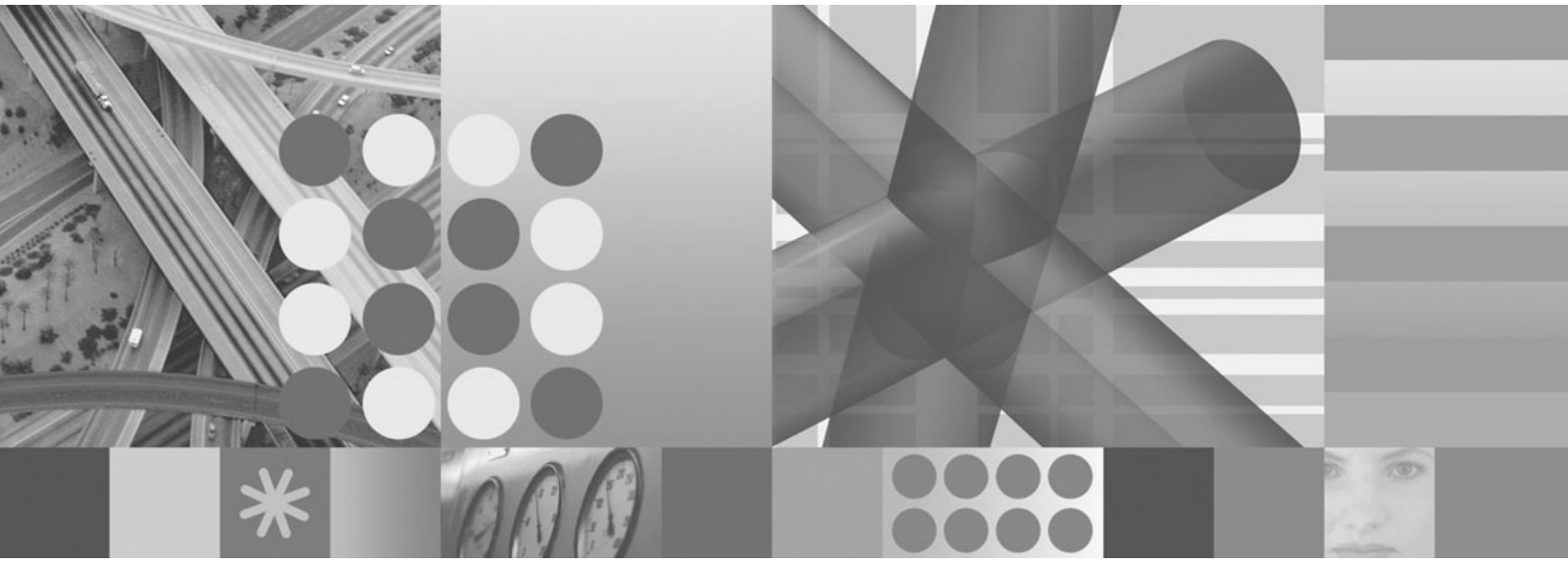
バージョン 5 リリース 3

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

SC88-9313-02
(英文原典：SC31-8865-02)

Tivoli IBM Tivoli NetView for z/OS

バージョン 5 リリース 3



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

SC88-9313-02
(英文原典：SC31-8865-02)

お願い

本書および本書で紹介する製品をご使用になる前に、 771 ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM Tivoli NetView for z/OS (製品番号 5697-ENV) バージョン 5 リリース 3、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。製品レベルに適切なエディションを使用してください。

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。

IBM 発行のマニュアルに関する情報のページ

`http://www.ibm.com/jp/manuals/`

こちらから、日本語版および英語版のオンライン・ライブラリーをご利用いただけます。また、マニュアルに関するご意見やご感想を、上記ページよりお送りください。今後の参考にさせていただきます。

(URL は、変更になる場合があります)

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： SC31-8865-02
Tivoli® IBM Tivoli NetView for z/OS
Version 5 Release 3
Resource Object Data Manager and GMFHS Programmer's Guide

発　行： 日本アイ・ビー・エム株式会社

担　当： ナショナル・ランゲージ・サポート

第1刷 2007.10

この文書では、平成明朝体™W3、平成明朝体™W7、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、および平成角ゴシック体™W7を使用しています。この(書体*)は、(財) 日本規格協会と使用契約を締結し使用しているものです。フォントとして無断複製することは禁止されています。

注* 平成明朝体™W3、平成明朝体™W7、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、平成角ゴシック体™W7

目次

図

本書について

IBM® Tivoli® NetView® for z/OS® プロダクトは、複雑な、マルチプラットフォーム、マルチベンダーのネットワークおよびシステムの可用性を、単一管理ポイントから最高度に維持するために使用できる拡張機能を提供します。 本書「*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*」には、NetView リソース・オブジェクト・データ・マネージャー (RODM) について記述されています。 NetView 管理コンソールを使用して、非 SNA ネットワークを RODM に定義し、お客様のネットワーク (非 SNA、SNA リソース、またはその両方) を管理する方法も記述されています。また、本書には、RODM を使用してネットワーク自動化機能をインプリメントする方法の説明があります。さらに、アプリケーション・プログラミングのための RODM の使用法が記述されています。

対象読者

本書は、非 SNA ネットワークを RODM に定義する必要があるネットワーク・マネージャーおよびシステム・プログラマーを対象としています。また、RODM アプリケーション、メソッド、およびデータ・モデルを作成または変更する必要があるアプリケーション・プログラマーおよびシステム・プログラマーを対象としています。

本書は、RODM を使用してネットワークを自動化する方法の計画を作成する必要があるネットワーク・プランナーにも役立ちます。

資料

このセクションでは、IBM Tivoli NetView for z/OS ライブラリーに収められている資料、およびその他の関連資料を取り上げます。 Tivoli マニュアルへのオンライン・アクセスの方法、および Tivoli マニュアルのご注文方法についても記述されています。

IBM Tivoli NetView for z/OS ライブラリー

Tivoli NetView for z/OS ライブラリーには、以下のような資料が用意されています。

- 「*アドミニストレーション・リファレンス*」(SC88-9305) には、システム管理に必要な NetView プログラムの定義ステートメントについて記述されています。
- 「*アプリケーション・プログラマーズ・ガイド*」(SC88-9306) には、 NetView プログラム間インターフェース (PPI) および NetView アプリケーション・プログラミング・インターフェース (API) の使用方法について記述されています。
- 「*Automated Operations Network カスタマイズ・ガイド*」(SC88-9318) には、イベント・ドリブン・ネットワーク自動化機能を提供する NetView Automated Operations Network (AON) コンポーネントの自動化操作機能を調整し、拡張する方法について記述されています。

- 「*Automated Operations Network ユーザーズ・ガイド*」(GC88-9302) には、Automated Operations Network コンポーネントを使用して、システムおよびネットワークの効率を向上させる方法について記述されています。
- 「*自動操作ガイド*」(SC88-9304) には、自動化操作機能を使用して、システムとネットワークの効率およびオペレーターの生産性を向上させる方法について記述されています。
- 「*コマンド解説書 第 I 巻*」(SC88-9307) および「*コマンド解説書 第 II 巻*」(SC88-9308) には、ネットワークとシステム操作およびコマンド・リストとコマンド・プロシージャーで使用することができる NetView コマンドについて記述されています。
- 「*カスタマイズ・ガイド*」(SC88-9309) には、NetView プロダクトをカスタマイズする方法が記述されており、関連情報のソースが示されています。
- 「*データ・モデル・リファレンス*」(SC88-9312) では、Graphic Monitor Facility host subsystem (GMFHS) 、SNA トポロジー・マネージャー、およびマルチシステム・マネージャーのデータ・モデルについて説明しています。
- 「*インストール: 追加コンポーネントの構成*」(SC88-9321) には、NetView の基本機能以外の追加機能の構成方法について記述されています。
- 「*インストール: グラフィカル・コンポーネントの構成*」(SC88-9322) では、NetView グラフィックス・コンポーネントをインストールおよび構成する方法について説明しています。
- 「*インストール: 概説*」(SC88-9319) には、NetView 基本機能をインストールおよび構成する方法について記述されています。
- 「*インストール: マイグレーション・ガイド*」(SC88-9320) には、NetView プロダクトの現行リリースによって提供される新規機能および前のリリースからの基本機能のマイグレーションについて記述されています。
- 「*インストール： Tivoli NetView for z/OS Enterprise Agent の構成*」(SC31-6969) では、Tivoli NetView for z/OS エンタープライズ・エージェントをインストールおよび構成する方法について説明しています。
- 「*Messages and Codes Volume 1 (AAU-DSI)*」(SC31-6965) および「*Messages and Codes Volume 2 (DUI-IHS)*」(SC31-6966) では、NetView プロダクトのメッセージ、NetView 異常終了コード、NetView メッセージ内のセンス・コード、および総称アラート・コード・ポイントについて説明しています。
- 「*マルチシステム・マネージャー ユーザーズ・ガイド*」(GC88-9301) には、NetView マルチシステム・マネージャー・コンポーネントをネットワーク管理でどのように使用できるかについて記述されています。
- 「*NetView 管理コンソール ユーザーズ・ガイド*」(GC88-9303) では、NetView プロダクトの NetView 管理コンソール・インターフェースについて説明しています。
- 「*Programming: Assembler*」(SC31-8860) には、アセンブラー言語を使用して NetView プロダクトの出口ルーチン、コマンド・プロセッサー、およびサブタスクの作成方法について記述されています。
- 「*プログラミング: パイプ*」(SC88-9311) には、NetView パイプラインを使用して NetView インストール済み環境をカスタマイズする方法について記述されています。

- 「*Programming: PL/I and C*」(SC31-8861) には、PL/I または C を使用して NetView プロダクトのコマンド・プロセッサーおよびインストール・システム出口ルーチンを作成する方法が記述されています。
- 「*プログラミング: REXX および NetView コマンド・リスト言語*」(SC88-9310) には、再構造化拡張実行プログラム言語 (REXX™) または NetView コマンド・リスト言語を使用して、NetView プロダクトのコマンド・リストを作成する方法について記述されています。
- 「*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*」(SC88-9313) では、NetView リソース・オブジェクト・データ・マネージャー (RODM) (非 SNA ネットワークの RODM への定義方法、およびネットワーク自動化とアプリケーション・プログラミングでの RODM の使用法を含む) について説明しています。
- 「セキュリティー・リファレンス」(SC88-9317) には、NetView 環境の許可検査をインプリメントする方法について記述されています。
- 「*SNA トポロジー・マネージャー インプリメンテーション・ガイド*」(SC88-9315) では、サブエリアを管理するために使用できる NetView SNA トポロジー・マネージャー、拡張対等通信ネットワーク機能 (Advanced Peer-to-Peer Networking®)、および TN3270 リソースの計画およびインプリメントについて説明しています。
- 「*Troubleshooting Guide*」(LY43-0093) では、NetView プロダクトの使用中に起こる可能性のある問題の文書化、診断、および解決について説明しています。
- 「*Tuning Guide*」(SC31-8869) には、NetView プロダクトおよびネットワーク環境の一定のパフォーマンス・ゴールを達成するために役立つチューニング情報があります。
- 「ユーザーズ・ガイド」(GC88-9300) では、NetView プロダクトを使用して、複雑なマルチベンダーのネットワークとシステムを単一ポイントから管理する方法について説明しています。
- 「*Web アプリケーション ユーザーズ・ガイド*」(SD88-6818) には、NetView Web アプリケーションを使用して、複雑なマルチベンダーのネットワークとシステムを単一ポイントから管理する方法について記述されています。
- 「*Licensed Program Specifications*」(GC31-8848) には、NetView プロダクトのライセンス情報があります。

前提資料

このリリースで提供される新機能については、「*IBM Tivoli NetView for z/OS インストール: マイグレーション・ガイド*」を参照してください。

NetView for z/OS プロダクトがどのように IBM Tivoli Monitoring プロダクトと相互に作用するかについては、以下の IBM Tivoli Monitoring の資料を参照してください。

- 「*IBM Tivoli Monitoring 6.1 紹介*」(GI88-6718) では、IBM Tivoli Monitoring のコンポーネント、概念、および機能を示しています。
- 「*IBM Tivoli Monitoring V6.1.0 Tivoli Distributed Monitoring の更新*」(GD88-6712) では、IBM Tivoli Distributed Monitoring の更新方法について説明しています。

- 「*IBM Tivoli Monitoring V6.1.0 インストールおよび設定ガイド*」(GD88-6698) では、IBM Tivoli Monitoring のインストールおよび設定について説明しています。
- 「*IBM Tivoli Monitoring V6.1.0 ユーザーズ・ガイド*」(SD88-6700) は IBM Tivoli Enterprise™ Portal オンライン・ヘルプを補完する資料であり、すべての Tivoli Enterprise Portal 機能に関する実地研修および詳細な手順が示されています。
- 「*IBM Tivoli Monitoring 6.1.0 管理者ガイド*」(SD88-6699) では、IBM Tivoli Enterprise Portal Server およびクライアントに必要なサポート・タスクおよび機能について説明しています。
- 「*IBM Tivoli Monitoring V6.1.0 Tivoli Enterprise Monitoring サーバー z/OS 版の構成*」(SD88-6713) では、z/OS システム上で稼働する IBM Tivoli Enterprise Monitoring Server を構成およびカスタマイズする方法について説明しています。
- 「*IBM Tivoli Monitoring V6.1.0 問題判別ガイド*」(GD88-6710) には、ソフトウェアに関する問題のトラブルシューティングを行う際に使用する情報およびメッセージが記載されています。
- 「*IBM Tivoli Monitoring V6.1.0 IBM Tivoli Monitoring の解説*」(SD88-6817) には、IBM Tivoli Monitoring について検討するための一連の実践内容が記載されています。
- 「*IBM Tivoli Universal Agent V6.1.0 ユーザーズ・ガイド*」(SD88-6711) では、IBM Tivoli Universal Agent について紹介しています。
- 「*IBM Tivoli Universal Agent API and Command Programming Reference Guide*」(SC32-9461) では、IBM Tivoli Universal Agent API をインプリメントする方法について説明し、API 呼び出しおよびコマンド行インターフェース・コマンドについて解説しています。

関連資料

NetView ブリッジ機能については、「*Tivoli NetView for OS/390 Bridge Implementation*」(SC31-8238-03、V1R4 ライブラリーからのみ入手可能) を参照してください。

追加の製品情報は、次の NetView for z/OS Web サイト上で検索できます。

http://www.ibm.com/software/tivoli/products/netview-zos/

オンライン用語集へのアクセス

「*Tivoli ソフトウェア用語集*」には、Tivoli ソフトウェアに関する多数の技術用語の定義が収められています。「*Tivoli ソフトウェア用語集*」は、次の Tivoli ソフトウェア・ライブラリー Web サイトでご利用いただけます。

http://publib.boulder.ibm.com/tividd/glossary/tivoliglossarymst.htm

IBM Terminology Web サイトには、多数の IBM プロダクト・ライブラリーからの用語が 1 つの便利なロケーションに統合されています。 Terminology Web サイトには、次の Web アドレスでアクセスできます。

http://www.ibm.com/software/globalization/terminology/

NetView for z/OS の用語と定義のリストについては、IBM Terminology Web サイトを参照してください。 以下の用語は、このライブラリーで使用されます。

NetView
以下のプロダクト:
- Tivoli NetView for z/OS バージョン 5 リリース 3
- Tivoli NetView for z/OS バージョン 5 リリース 2
- Tivoli NetView for z/OS バージョン 5 リリース 1
- Tivoli NetView for OS/390® バージョン 1 リリース 4

MVS™ z/OS オペレーティング・システムに関する用語

MVS エレメント
z/OS オペレーティング・システムの BCP エレメントに関する用語

CNMCMD
CNMCMD および組み込みメンバー

CNMSTYLE
CNMSTYLE および組み込みメンバー

PARMLIB
連結シーケンスでの SYS1.PARMLIB およびその他のデータ・セットに関する用語

以下の IBM の名前は、指定された Candle® の名前と置き換わります。

IBM Tivoli Monitoring Services
OMEGAMON® プラットフォームに関する用語

IBM Tivoli Enterprise Monitoring Agent
Intelligent Remote Agent に関する用語

IBM Tivoli Enterprise Monitoring Server
Candle Management Server に関する用語

IBM Tivoli Enterprise Portal
CandleNet Portal に関する用語

IBM Tivoli Enterprise Portal Server
CandleNet Portal Server に関する用語

特に断りのない限り、プログラムを参照する場合は、そのプログラムの最新のバージョンおよびリリースを指します。バージョンのみが示されている場合は、そのバージョンのすべてのリリースを参照しています。

パーソナル・コンピューターまたはワークステーションの使用に関する参照の場合は、すべてのプログラマブル・ワークステーションを使用できます。

LookAt を使用してメッセージの説明を検索する

LookAt というオンライン機能により、大部分の IBM メッセージと、いくつかのシステム異常終了 (タスクの異常終了) およびコードに関する説明を検索できます。LookAt では、通常、該当メッセージの説明がただちに表示されるため、従来の方法よりも短時間で、必要な情報を検索することができます。

LookAt を以下のロケーションから使用して、z/OS のエレメントおよびフィーチャー、z/VM®、VSE/ESA™、および Clusters for AIX® and Linux® に関する IBM メッセージの説明を検索できます。

- インターネット。LookAt Web サイト (http://www.ibm.com/eserver/zseries/zos/bkserv/lookat/) から IBM メッセージの説明に直接アクセスできます。
- z/OS TSO/E ホスト・システム。z/OS または z/OS.e システムにコードをインストールして、TSO/E コマンド行 (例えば、TSO/E プロンプト、ISPF、または OMVS が稼働中の z/OS UNIX® システム・サービス) から LookAt を使用し、IBM メッセージの説明にアクセスできます。
- Microsoft® Windows® ワークステーション。コードをインストールして、Microsoft Windows DOS コマンド行から LookAt を使用し、「*z/OS Collection*」(SK3T-4269) にある IBM メッセージの説明にアクセスできます。
- ワイヤレス・ハンドヘルド・デバイス。ハンドヘルド・デバイスで LookAt Mobile Edition を使用することによって、ワイヤレス・アクセスおよびインターネット・ブラウザー (例えば、Internet Explorer for Pocket PCs、Blazer、または Eudora for Palm OS、あるいは Opera for Linux handheld devices) を利用できるようになります。 LookAt Web サイトから LookAt Mobile Edition にリンクしてください。

LookAt をホスト・システムまたは Microsoft Windows ワークステーションにインストールするコードは、「*z/OS Collection*」(SK3T-4269) のディスクまたは LookAt Web サイト (「**Download**」をクリックし、必要なプラットフォーム、リリース、コレクション、およびロケーションを選択します) から入手できます。詳細な情報は、ダウンロード処理中に使用可能な LOOKAT.ME ファイル内にあります。

マニュアルへのオンライン・アクセス

以下は英語のみの対応となります。資料 CD には、製品ライブラリーにある資料が含まれています。資料は、PDF、HTML、および BookManager® フォーマットで入手可能です。資料へのアクセス方法の説明については、CD 上の README ファイルを参照してください。

Tivoli NetView for z/OS ライブラリーを検索するための索引が、ドキュメンテーション CD 上に収められています。ご使用のシステムに Adobe Acrobat があれば、Search コマンドを使用して、ライブラリー内で特定のテキストの場所を探索することができます。ライブラリーを検索する索引の使用方法についての詳細な情報は、Acrobat のオンライン・ヘルプを参照してください。

IBM では、この製品およびその他のすべての Tivoli 製品に関する資料を、使用可能になった時点および更新された時点で、Tivoli Information Center の Web サイト (http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp) に載せています。

「Tivoli Information Center」ウィンドウで、「**Tivoli product manuals (Tivoli 製品マニュアル)**」をクリックします。製品名の最初の文字と一致する文字をクリックし、製品ライブラリーにアクセスします。例えば、Tivoli NetView for z/OS ライブラリーにアクセスするには、**N** をクリックします。

注: PDF 文書をレターサイズ以外の用紙に印刷する場合は、Adobe Reader のメニューから「**ファイル**」→「**印刷**」を選択して表示されたウィンドウでオプションを設定し、レターサイズのページをご使用の用紙に印刷できるようにしてください。

マニュアルのご注文

以下は英語のみの対応となります。なお、日本 IBM 発行のマニュアルはインターネット経由でもご購入いただけます。詳しくは、http://www.ibm.com/jp/manuals/ の「ご注文について」をご覧ください。(URL は、変更になる場合があります)

http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss

次の電話番号からもご注文いただけます。

- 米国: 800-879-2755
- カナダ: 800-426-4968

その他の国では、Tivoli 製品資料のご注文については、ソフトウェアのお客様担当者にご連絡ください。お客様担当者の電話番号を調べるには、以下の手順を実行してください。

1. 次の Web アドレスにアクセスします。

   http://www.elink.ibmlink.ibm.com/public/applications/publications/cgibin/pbi.cgi
2. リストからお客様の国を選択し、「**Go**」をクリックします。「Welcome to the IBM Publications Center」ウィンドウが表示されます。
3. ウィンドウの左側の「**About this site**」をクリックし、お客様担当者の電話番号が記されている情報ページを表示します。

アクセシビリティー

アクセシビリティー機能は、運動障害または視覚障害など身体に障害を持つユーザーがソフトウェア・プロダクトを快適に使用できるようにサポートします。製品では、標準的なショートカット・キーおよびアクセラレーター・キーが使用され、オペレーティング・システムによって文書化されます。詳しくは、ご使用のオペレーティング・システムが提供する資料を参照してください。

追加情報については、「*ユーザーズ・ガイド*」の付録『アクセシビリティー』を参照してください。

Tivoli 技術研修

以下は英語のみの対応となります。 Tivoli 技術研修の情報については、以下の IBM Tivoli Education Web サイト (http://www.ibm.com/software/tivoli/education) を参照してください。

サポート情報

以下は英語のみの対応となります。 IBM ソフトウェアに問題がある場合は、早く解決する必要があります。お客様が必要なサポートを得られるように、IBM は以下の方法を提供しています。

オンライン

IBM Software Support サイト (http://www.ibm.com/software/support/probsub.html) にアクセスして、指示に従います。

IBM Support Assistant

IBM Support Assistant (ISA) は、IBM ソフトウェア製品に関する疑問および問題の解決に役立つ無償のローカル・ソフトウェア保守サービス・ワークベンチです。ISA により、問題判別のためのサポート関連の情報および保守サービス・ツールに迅速にアクセスできます。ISA ソフトウェアをインストールするには、http://www.ibm.com/software/support/isa にアクセスします。

問題判別ガイド

問題を解決する方法について詳しくは、「*IBM Tivoli NetView for z/OS Troubleshooting Guide*」を参照してください。

ダウンロード

ダウンロード可能なクライアントとエージェント、NetView 製品のデモンストレーション、およびいくつかの無償の NetView アプリケーションを、以下の NetView for z/OS Web サイトから入手できます。

http://www.ibm.com/software/tivoli/products/netview-zos/

これらのアプリケーションは、以下の作業に役立ちます。

- 以前のリリースから現行スタイルシートへの、カスタマイズ・パラメーターのマイグレーション
- 自動化テーブルの統計情報の入手、および自動化テーブルのリストとの統計情報のマージ
- JES (Job Entry Subsystem) ジョブの状況の表示、または指定された JES ジョブの取り消し
- プログラム間インターフェース (PPI) を使用した、NetView プログラムへのアラートの送信
- PPI を使用した、MVS コマンドの送信および受信
- TSO (Time Sharing Option) コマンドの送信および応答の受信

本書の表記規則

本書では、特殊な用語やアクション、オペレーティング・システムに依存するコマンドとパス、およびコマンド構文を表す場合に、いくつかの表記規則を使用しています。

書体の規則

本書では、書体について以下の規則を使用しています。

太字

- 周囲のテキストと見分けがつきにくい小文字のコマンドおよび大/小文字混合のコマンド
- インターフェース・コントロール (チェック・ボックス、プッシュボタン、ラジオ・ボタン、スピン・ボタン、フィールド、フォルダー、アイコン、リスト・ボックス、リスト・ボックス内の項目、複数列リスト、コン

テナー、メニューの選択項目、メニュー名、タブ、プロパティー・シート)、ラベル (**ヒント:** および**オペレーティング・システムの考慮事項**: など)
- テキスト内のキーワードおよびパラメーター

イタリック
- 引用 (例: 資料、ディスケット、および CD のタイトル)
- テキスト内で定義されている語 (例: 非交換回線は *Point-to-Point 回線* とも呼ばれる)
- 語および文字の強調 (言葉として扱われる語の例: "制限節を挿入するには、単語 *that* を使用します"。文字として扱われる場合の例: "LUN アドレスは文字 *L* で始める必要があります"。)
- テキスト中の新規用語 (定義リスト内を除く): *view* は、データが入っているワークスペース内のフレームです。
- 指定する必要のある変数および値: ... ここで *myname* が表すものは ...

モノスペース
- 例およびコード例
- 周囲のテキストと見分けがつきにくいファイル名、プログラミングのキーワード、およびその他のエレメント
- ユーザー宛てのメッセージ・テキストおよびプロンプト
- ユーザーが入力する必要のあるテキスト
- 引数またはコマンド・オプションの値

オペレーティング・システム依存の変数とパス

ワークステーション・コンポーネントの場合、本書では、環境変数およびディレクトリー表記に UNIX の規則を使用しています。

Windows コマンド行を使用する場合、環境変数では **$** *変数* を **%***変数* **%** で置き換え、ディレクトリー・パスではスラッシュ (*/*) を円記号 (**¥**) で置き換えてください。環境変数の名前は、Windows 環境と UNIX 環境とでは常に同じとは限りません。例えば、Windows 環境の %TEMP% は、UNIX 環境の $TMPDIR と同等です。

注: Windows システムで bash シェルを使用している場合は、UNIX の表記規則を使用できます。

構文図

構文図は左側の 2 つの矢印 (▸▸) で始まり、主線に沿って、相互に向かい合った 2 つの矢印 (▸◂) まで進み、ここで終了します。構文図に 2 行以上必要な場合、継続行は単一の矢印 (▸) で終了します。

構文エレメントの位置および形状

構文図では、強調表示、大括弧、または中括弧を頼りにすることはできません。構文図では、次の表に示すように、主線に対する相対的な位置によって、キーワード、変数、およびオペランドの、必須値、オプション (任意指定の) 値、およびデフォルト値を示します。

表 1. 構文エレメントの位置

エレメントの位置	意味
構文の主線上	必須
構文の主線より上	デフォルト
構文の主線より下	オプション (任意指定)

キーワードおよびオペランドは、大文字で示されます。変数は小文字で示され、イタリック体または、(NetView ヘルプおよび BookManager オンライン・ブックの場合) 色を区別して示されます。構文エレメントの形状には、以下のテーブルに示されているようなエレメントのタイプがあります。

表 2. 構文エレメントの形状

エレメント	形状
キーワード	CCPLOADF
変数	*resname*
オペランド	MEMBER=*membername*
デフォルト	*today* または INCL

必須構文エレメント

コマンド名と必須キーワード、変数、およびオペランドは、構文の主線上に示されます。 図 1 は、*resname* 変数を CCPLOADF コマンドで使用しなければならないことを示します。

CCPLOADF

図 1. 必須構文エレメント

オプションの構文エレメント

オプションのキーワード、変数、およびオペランドは、構文の主線より下に示されます。 図 2 は、ID オペランドを DISPREG コマンドで使用できるが、必須ではないことを示します。

DISPREG

図 2. オプションの構文エレメント

デフォルトのキーワードおよび値

デフォルトのキーワードおよび値は、構文の主線より上に示されます。

デフォルトがキーワードである場合、そのデフォルトは主線より上にのみ示されます。このキーワードを指定することも、または、指定せずにデフォルトにすることもできます。 図 3 は、デフォルトのキーワード STEP が主線より上にあり、その他のオプションのキーワードが主線より下にあることを示します。

オペランドにデフォルト値がある場合、そのオペランドは主線より上と下の両方に示されます。主線より下に値がある場合は、オペランドを指定するときに、デフォルト値または表示されている値のいずれかを指定する必要があることを示します。オペランドを指定しない場合は、主線より上にあるデフォルト値が使用されます。図 3 は、オペランド MODNAME=* と OPTION=* のデフォルト値が主線より上と下にあることを示します。

RID

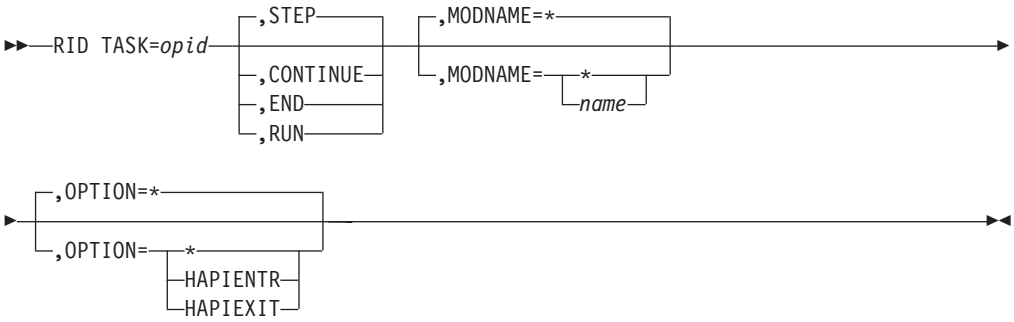
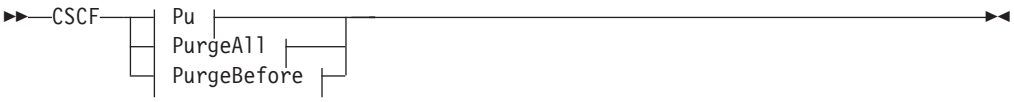


図 3. デフォルトのキーワードおよび値

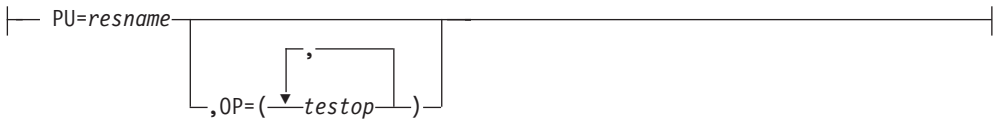
構文のフラグメント

コマンドに構文の長いセクションが含まれている場合、またはコマンド内で 1 つのセクションが 2 回以上使用されている場合は、別個のフラグメントとして主構文図の後に示します。フラグメント名は、大/小文字混合で示します。 xxviii ページの図 4 は、フラグメント Pu、PurgeAll、および PurgeBefore を持つ構文図を示します。

CSCF



Pu



PurgeAll

PurgeBefore

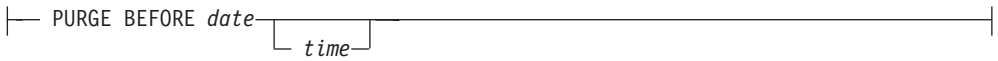


図 4. 構文のフラグメント

コンマおよび小括弧

必須のコンマおよび小括弧は、構文図内に示されます。

オペランドに複数の値を指定できる場合は、一般に値を小括弧で囲み、コンマで区切ります。例えば、図 4 では、OP オペランドの中に、*testop* 変数に複数の値を指定できることを示すコンマが入っています。

コマンドにキーワードと変数を区切る定位置コンマを入れる必要がある場合は、xxvii ページの図 3 に示すように、キーワードまたは変数の前にコンマを置きます。

コンマは、定位置オペランドが無いことを示すためにも使用されます。以下の BOSESS コマンドの例で、2 番目のコンマは、オプション (任意指定) のオペランドが使用されていないことを示します。

```
NCCF BOSESS applid,,sessid
```

末尾の定位置コンマを指定する必要はありません。定位置、非定位置にかかわらず、末尾のコンマは無視されるか、コマンドがリジェクトされる原因となります。末尾のコンマによってコマンドがリジェクトされるかどうかについては、各コマンドの制約事項を参照してください。

省略形

コマンドおよびキーワードの省略形は、各コマンドの説明の後の同義語表を参照してください。

第 1 部 RODM について理解する

第 1 章 概要

本書では、z/OS オペレーティング・システムで稼働する Tivoli NetView for z/OS V5R3 リソース・オブジェクト・データ・マネージャー (RODM) について説明します。 本書では以下を行う方法を説明します。

- ネットワーク・リソースを RODM に手動で定義し、NetView 管理コンソール (NMC) を用いてこれらのリソースを管理できるようにする。
- ネットワークの運用を、RODM に保管されたリソースの状況に応じて自動化する。
- RODM のサービスを使用したプログラムを作成する。

RODM は、オブジェクト指向のデータ・キャッシュです。 RODM のオブジェクトは、ネットワーク内のリソースを表します。データ・キャッシュは、完全にホスト・プロセッサーのメモリー内にあるため、データへのアクセスやトランザクションの処理が速くなります。多くのアプリケーションが単一の RODM と対話することができ、かつ複数の RODM を 1 つのホスト・プロセッサーで実行することができます。 RODM はさまざまなタスクに使用できます。 RODM は、ホスト・プロセッサーで実行するアプリケーションが使用できるアプリケーション・プログラミング・インターフェース (API) を備えています。

Graphic Monitor Facility ホスト・サブシステム (GMFHS) は、ホスト・プロセッサー上で実行される RODM と NetView プログラム、および NetView 管理コンソール と連動してリソースを管理するホスト・プログラムです。

GMFHS は、SNA トポロジー・マネージャーおよび NetView 管理コンソール と共に稼働して、SNA リソースを管理します。詳細については、「*IBM Tivoli NetView for z/OS SNA トポロジー・マネージャー インプリメンテーション・ガイド*」(SC88-9315) を参照してください。

GMFHS は、マルチシステム・マネージャーおよび NetView 管理コンソール と共に稼働して、非 SNA リソースを管理します。詳細については、「*IBM Tivoli NetView for z/OS マルチシステム・マネージャー ユーザーズ・ガイド*」を参照してください。

NetView により SNA より詳細な物理ビュー

NetView プログラムは、SNA トポロジー・マネージャーを使用して、NetView 管理コンソール からのサブエリアと拡張対等通信ネットワーキング機能 (APPN) ネットワーク管理を提供します。ネットワーク内のリソースのグラフィック・ビューを表示し、ビューで選択したリソースにコマンドを出すことができます。ビューには、ネットワークに関する状況情報と構成情報の両方が入っています。詳細については、「*IBM Tivoli NetView for z/OS SNA トポロジー・マネージャー インプリメンテーション・ガイド*」を参照してください。

非 SNA リソースを NetView に定義する

マルチシステム・マネージャーを使用すると、 NetView プログラムによって、NetView 管理コンソール から非 SNA ネットワークを動的に見つけ出して管理することができます。ネットワーク内のリソースのグラフィック・ビューを表示し、ビューで選択したリソースにコマンドを出すことができます。ビューには、ネットワークに関する状況情報と構成情報の両方が入っています。

非 SNA リソースを手動で定義することもできます。 NetView プログラムにネットワークに関する情報を提供して、ビューを作成し、コマンドを処理できるようにする必要があります。 SNA ネットワークの場合、 NetView は、作成した VTAM® および NCP 定義からその情報を得ます。非 SNA ネットワークの場合、 NetView は、作成した RODM 定義からその情報を得ます。本書では、作成する必要のある RODM 定義とその作成方法について説明します。

NetView 管理コンソール は GMFHS と通信します。 GMFHS がホスト内の固有のアドレス・スペースで実行され、ホスト内の固有のアドレス・スペースで実行される RODM と通信する様子を図 5 に示します。

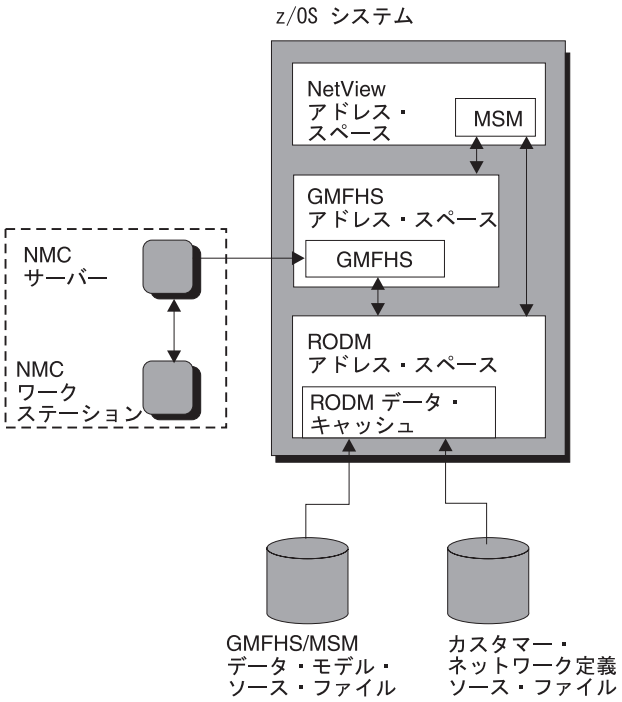


図 5. RODM を使用して NetView 管理コンソール をサポートする

リソース定義タスク

非 SNA ネットワークのリソースは、 RODM データ・キャッシュのオブジェクトによって表されます。作成できるオブジェクトには、次の 3 つのタイプがあります。

- 管理オブジェクト
- 管理されるオブジェクト

- ビュー・オブジェクト

管理オブジェクト とは、ネットワークの一部を制御し、 NetView プログラムに接続するプログラムを表します。 LAN マネージャーおよび NetView/PC は、管理オブジェクトの一例です。管理オブジェクトによって表されるプログラムは、NetView にアラートを送ってネットワーク内のリソースの状況を更新します。これらのプログラムは、自らが制御するネットワーク・リソースについて NetView プログラムからコマンドを受け取ります。

管理されるオブジェクト は、ユーザーが管理しているネットワーク・リソースを表します。管理されるオブジェクトには、状況情報と構成情報の両方が入っています。トークンリング・ローカル・エリア・ネットワーク (LAN) に接続したパーソナル・コンピューターおよびイーサネット LAN に接続したプリンターは、管理されるオブジェクトによって表されるリソースの一例です。管理されるオブジェクトには、 NetView に状況を送り、リソースについてのコマンドを受け取る、対応する管理オブジェクトがなければなりません。

ビュー・オブジェクト は、 NetView 管理コンソール に表示できるグラフィック・ビューを表します。グラフィック・ビューのほとんどは、RODM 内に入っている構成情報をもとに自動的に作られます。固有のビューを定義することもできます。表示するリソースとその表示方法に関する情報は、ビュー・オブジェクトに入っています。

ネットワーク構成 情報は、管理されるオブジェクト間のリンクによって表されます。例えば、トークンリング・セグメントのリソースを表す管理されるオブジェクトは、それぞれセグメント上で隣接する各リソースにリンクしています。ネットワークの論理構成と物理構成の両方を定義することができます。

GMFHS がサポートするリソース

GMFHS は、標準フォーマットの NetView プログラムに状況の更新を送ることができるリソースをサポートします。*サービス・ポイント* は、非 SNA ネットワークと、NetView プログラムを含む SNA ネットワークの間のインターフェースをとるプログラムです。サービス・ポイントは、GMFHS によって RODM のオブジェクトの状況に変換されるアラートを生成します。

NetView プログラムに送られるアラートは、状況を変更したリソースを識別します。アラートが提供する名前と一致する RODM オブジェクトに、名前を割り当てる必要があります。 GMFHS がアラートからのリソース名を使用する方法については、193 ページの『第 6 章 アラートおよび解決を処理および受信するための GMFHS のカスタマイズ』を参照してください。ここでは、GMFHS アラート処理をカスタマイズして、追加のアラート・タイプを認識する方法についても説明しています。

RODM データを保管する

RODM データ・キャッシュ内のデータは、すべてメモリーに記憶されます。RODM を停止する場合に、プロセッサーを遮断するか、システムに障害が起きると、データ・キャッシュ内のデータはすべて失われます。 チェックポイント 機能を用いると、データ・キャッシュのコピーを DASD に保管することができます。RODM を再始動する際は、DASD から記憶されたデータを読み込むことができま

す。オペレーターが使用する NetView プログラムがコマンドを z/OS に送信するように設定されている場合、チェックポイント機能の要求は、プログラム、z/OS コンソール・オペレーター、または NetView オペレーターから行うことができます。RODM に記憶された状況情報は揮発性なので、 DASD から復元されたデータが適切でない場合もあります。

RODM のウォーム・スタート とは、RODM を開始して、チェックポイント・データを読み込む時点です。データ・キャッシュには、チェックポイント時点の絶対データが入ります。ウォーム・スタート後、データ・キャッシュ内のオブジェクトによっては更新しなければならない場合があります。アプリケーションは、リソース状況を保守し、RODM に送られた更新を記録していれば、チェックポイント以降の変更をすべて再送信することができます。

コールド・スタート とは、チェックポイント・データなしに RODM を開始することを意味します。 データ・キャッシュに含まれるのは、システム定義のクラスのみです。したがって、データ・モデルおよびデータをロードする必要があります。

ネットワーク自動化における RODM

SNA トポロジー・マネージャーを使用すると、サブエリア・ネットワークの管理を自動化することができます。詳細については、「*IBM Tivoli NetView for z/OS SNA トポロジー・マネージャー インプリメンテーション・ガイド*」を参照してください。

RODM を使用すると、非 SNA ネットワーク・リソースの管理を自動化することもできます。 GMFHS は、RODM データ・キャッシュ内の非 SNA ネットワーク・リソースの状況を保守しているので、 RODM 内のデータを使用して自動化ルーチンを書くことができます。以下に示す RODM の概念は、自動化を行う上で重要です。

自動化の概念

RODM の処理は、ユーザー・アプリケーションとメソッドの 2 つのタイプのプログラムが行います。 RODM *ユーザー・アプリケーション* は、 RODM とは異なるアドレス・スペースで実行し、かつ API を用いて RODM と通信するプログラムです。ユーザー・アプリケーションは、RODM と同じ z/OS ホストで実行する必要があります。 ユーザー・アプリケーションは、どのプログラム言語ででも作成できます。 API 用に提供されているサンプル制御ブロックは、PL/I および C で使用するようになっています。したがって、この 2 つのいずれかの言語を使用すると便利です。

メソッド は、RODM アドレス・スペースで実行し、別の API を使用して RODM と通信するプログラムです。メソッドは、データ・キャッシュ内のデータで特定のタスクを実行する、通常は小さなプログラムです。メソッドを実行することを、メソッドをトリガーする と言います。メソッドは、PL/I または C/370 で作成する必要があります。したがって、これらの言語で実行できるタイプの機能に限定されます。メソッドには、以下の 6 タイプがあります。

- RODM は、フィールドの値が照会されると、フィールドの*照会メソッド* を起動します。例えば RODM は、ネットワーク・リソースにコマンドを出して、その現在の状況を要求することができます。 *query* サブフィールド は、フィールドの照会メソッドを指定します。
- RODM は、別のメソッドもしくはユーザー・アプリケーションがフィールド値の変更を要求すると、フィールドの*変更メソッド* を起動します。例えば、変更メソッドは、コマンドを出してネットワーク・リソースの実際の状況を RODM でのリソースを表すオブジェクトの新しい状況と一致するように変更することができます。 *change* サブフィールド は、フィールドの変更メソッドを指定します。
- RODM は、フィールドの値が変更になると、*通知メソッド* を起動します。フィールドには、通知メソッドをいくつでも定義することができます。 RODM は、ユーザー・アプリケーションに変更を通知します。この通知メソッドは、特に自動化タスクには重要です。 *notify* サブフィールド は、フィールドの通知メソッドを指定します。
- RODM は、別のメソッドまたはユーザー・アプリケーションから要求があると、*名前付きメソッド* を起動します。名前付きメソッドは、オブジェクトもしくはクラスのフィールドにより指定されます。名前付きメソッドは、特定のオブジェクトまたはクラスになんらかのアクションを実行するときに使用することができます。例えば、メソッドが関連するオブジェクトを活動化するコマンドが含まれる名前付きメソッドを作ることができます。
- *オブジェクト独立メソッド* は、特定のオブジェクトまたはクラスに関連しない任意のメソッドです。オブジェクト独立メソッドは、多数のオブジェクトおよびクラスで実行できます。例えば、オブジェクト独立メソッドは、指定された LAN 上のワークステーションを表す全オブジェクトの状況を照会することができます。
- *初期化メソッド* は、特殊なタイプのオブジェクト独立メソッドです。初期化メソッドが指定されている場合は、RODM が開始すると自動的に起動されます。

照会メソッド、変更メソッド、通知メソッド、および名前付きメソッドは、特定のクラスもしくはオブジェクトに関連するため、*オブジェクト特有* メソッドとも呼ばれます。 NetView プログラムには、自動化タスクに使用できるサンプルのメソッドがあります。

RODM 自動化プラットフォーム という名前の一連の NetView サービスによって、自動化が容易になります。 NetView 自動化テーブル、コマンド・リスト、およびアプリケーションは、RODM に要求を出して、フィールドの値を変更し、メソッドを起動することができます。 NetView 提供のメソッドは、 NetView タスクが出すコマンドを送ります。また、RODM 自動化プラットフォームには拡張 API が備えられており、 NetView アドレス・スペース内のアプリケーションは、これを用いると少ないプログラミング作業で RODM 機能を出すことができるようになります。

自動化の例

代表的な自動化の実施では、メソッド、ユーザー・アプリケーション、および RODM 自動化プラットフォームを使用する傾向があります。例えば、リソースに障害があるときは、通知メソッドを使用して自動化アプリケーションに知らせることができます。自動化アプリケーションは、RODM を照会して、障害を起こしたリソースに関連するネットワーク内のリソースを探すことができます。自動化アプリケ

ーションは、関連リソースの状況を照会して、最も可能性の高い問題の個所を判断し、問題修正のコマンドを出すことができます。

RODM 自動化プラットフォームを用いると、 NetView コマンドを出す RODM の特定のオブジェクトに関連するメソッドを作成することができます。オブジェクト特有のメソッドには、そのメソッドに関連するリソースを活動化するコマンドを入れることができます。オブジェクト特有のメソッドは、自動化アプリケーションにより起動されると、 NetView 提供のメソッド EKGSPPI にコマンドを送ります。このコマンドは NetView プログラムに渡されて、自動タスクによって出されます。これにより、同じアプリケーションが、各リソースに特有のコマンドを知らなくても異なるタイプのリソースを活動化できるようになります。

詳細情報

本書では、特に自動化に関しては 2 つの章を設けています。 GMFHS データ・モデルによる自動化の詳細については、 209 ページの『第 7 章 自動化コードを作成する』をお読みください。 RODM 自動化プラットフォーム・サービスの詳細については、 217 ページの『第 8 章 RODM 自動化プラットフォームを使用する』をお読みください。

RODM プログラミング・タスク

この概要では、RODM を用いた NetView 管理コンソール およびネットワークの自動化のサポートに焦点をあてていますが、 RODM は、他のタイプのネットワークおよびシステム管理プログラムをサポートすることができます。このセクションでは、RODM プログラミング・タスクを全般的に説明します。

RODM は、高速データ・キャッシュ・マネージャーを必要とするどのタスクにも使用することができます。 RODM には、ユーザー・アプリケーション・プログラム用のアプリケーション・プログラミング・インターフェースがあり、またメソッド用には別のアプリケーション・プログラミング・インターフェースがあります。 さらに、データのデータ・キャッシュへのロードとデータの保守を単純化するロード機能も備えています。

ユーザー・アプリケーションおよびメソッドには、 RODM にきわめて類似したインターフェースがあります。 RODM が備えている機能の多くは、両タイプのプログラムで使用することができます。ユーザー・アプリケーションおよびメソッドは、RODM に機能要求を送ります。 RODM は、戻りコードと理由コードで応答し、要求が正常であったかどうかを示します。機能要求によっては、RODM がデータを戻す場合もあります。 RODM に出される単一の機能要求と RODM からの応答で、*トランザクション* が形成されます。

RODM トランザクション

トランザクションの多くは、RODM に対して、データ・キャッシュの特定クラス、オブジェクト、フィールド、もしくはサブフィールドになんらかのアクションをとることを要求します。例えば、ユーザー・アプリケーションは、ネットワーク・リソースの状況を表すフィールドの値を変更するよう RODM に要求します。トラン

ザクションが指定する特定のクラス、オブジェクト、フィールド、もしくはサブフィールドは、トランザクションのターゲット です。一般に、トランザクションがもつターゲットは 1 つです。

各トランザクションは、そのトランザクションの必須パラメーターを渡す RODM の呼び出しを用いて作られます。パラメーターは、以下の 6 つの制御ブロックにグループ分けされます。

- アクセス・ブロック
- トランザクション情報ブロック
- 機能ブロック
- 応答ブロック
- エンティティー・アクセス情報ブロック
- フィールド・アクセス情報ブロック

個々のトランザクションは、必要に応じて異なるブロックを使用します。

アクセス・ブロック は、ユーザー・アプリケーションを RODM に識別します。メソッドは、RODM 内で実行します。したがって、アクセス・ブロックを使用することはありません。 RODM 自動化プラットフォーム・サービスの CNMQAPI および DSINOR は、 NetView アドレス・スペースで実行するアプリケーションのアクセス・ブロックを処理します。

トランザクション情報ブロック は、RODM で各トランザクションをトレースするのに使用します。 RODM は、トランザクションの戻りコードと理由コードをこの制御ブロック内に入れます。トランザクションは、すべてこのブロックを使用します。

機能ブロック は、実行する RODM 機能を指定します。このブロックには、RODM が機能を実行する際に必要とする特定のパラメーターが入ります。トランザクションは、すべてこのブロックを使用します。

応答ブロック には、RODM から要求されたデータはすべて入ります。照会機能など、データを要求する機能は応答ブロックを使用します。

エンティティー・アクセス情報ブロック は、トランザクションのターゲットである特定のクラスおよびオブジェクトを識別します。このブロックは、クラス、オブジェクト、フィールド、もしくはサブフィールドがトランザクションのターゲットであるときに使用されます。

フィールド・アクセス情報ブロック は、トランザクションのターゲットである特定のフィールドを識別します。このブロックは、フィールドまたはサブフィールドがトランザクションのターゲットであるときに使用されます。

RODM 機能

RODM には、ユーザー・アプリケーション用とメソッド用の機能があります。ユーザー・アプリケーションでのみ使用できる機能もあれば、メソッドでのみ使用できる機能もあります。多くの機能は、両方に使用できます。機能ごとに特定の許可レベルが必要なので、特定のアプリケーションに使用できる機能を限定することができます。

RODM には、RODM と接続したり、切断したりする機能があります。また、RODM のチェックポイントを取り、停止する機能を備えています。

RODM には、データ・キャッシュ内のエレメントの構造を変更する一連の機能があります。クラス、オブジェクト、フィールド、およびサブフィールドを作成し削除する機能があります。リンクおよびリンク解除機能を用いると、オブジェクト間の関係を定義することができます。

RODM には、クラスおよびオブジェクトのフィールドとサブフィールドの値を変更する一連の機能があります。フィールドの値を変更すると、その変更メソッドが定義されていれば、起動されます。サブフィールドの値を変更しても、変更メソッドは起動されません。

RODM は、データ・キャッシュ内のクラスおよびオブジェクトに関する情報を入手する照会機能を備えています。プログラムは、任意のフィールドもしくはサブフィールドの値を照会することができます。フィールドの値を照会すると、その照会メソッドが定義されていれば、起動されます。サブフィールドの値を照会しても、照会メソッドは起動されません。プログラムは、データ・キャッシュ内のエレメントの構造を照会することもできます。 RODM は、文字フィールドの値に応じて、RODM のオブジェクトを探し出す機能も備えています。

RODM には、通知処理をサポートする機能があります。プログラムは、通知加入を追加および削除することができます。ユーザー・アプリケーションは、通知キューから情報を得ることができます。通知メソッドは、RODM 通知処理をサポートしています。

その他の機能によって、RODM ログに診断情報を書き込んだり、メソッドを起動することができます。機能のリストは、RODM の一回の呼び出しで発行することができます。 RODM への要求は非同期に出すこともできます。

各機能の詳細は、 423 ページの『第 14 章 アプリケーション・プログラミングの解説』で説明します。 RODM が提供している機能ごとに、サンプルの機能ブロックとプログラミング例があります。

プログラム言語

ユーザー・アプリケーションは、RODM ユーザー・アプリケーション・プログラミング・インターフェースを使用して RODM にアクセスします。ユーザー・アプリケーションは、z/OS 環境によってサポートされていれば、どのプログラム言語でも作成することができます。ただし、RODM サンプルと例が提供されるのは、PL/I と C の場合のみです。

メソッドは、RODM メソッド・アプリケーション・プログラミング・インターフェースを使用して RODM にアクセスします。 RODM メソッドは、PL/I または C でのみ作成できます。 NetView 提供のメソッドの多くは、ソース形式で提供されます。これらのメソッドをモデルとして使用し、それぞれの RODM メソッドを作成することができます。

RODM 通知プロセス

RODM *通知処理* により、ユーザー・アプリケーションはイベントの非同期通知を受け取ることができます。ユーザー・アプリケーションは、データ・キャッシュのフィールドに*申請* します。 フィールドの値が変更になると、そのフィールドに関連する通知メソッドが起動されます。通知メソッドは、変更に関する情報を*通知キュー* に書き込み、 RODM はユーザー・アプリケーションの*イベント制御ブロック* (ECB) を通知します。

ユーザー・アプリケーションは、その ECB が RODM により通知されるまで待機します。ユーザー・アプリケーションは、*EKGWAIT* モジュールを呼び出して、ECB が通知されるまで待機します。ユーザー・アプリケーションは、通知キューから情報を得て、妥当なアクションをとります。ユーザー・アプリケーションは、イベントの処理を終了すると、次のイベントの通知を待ちます。

RODM ロード機能

RODM ロード機能には、クラス構造およびオブジェクトを RODM データ・キャッシュにロードする簡単な方法があります。データ・モデル、クラス構造、フィールド、およびオブジェクトに関する詳細については、「*IBM Tivoli NetView for z/OS データ・モデル・リファレンス*」を参照してください。

ロード機能が処理するクラスおよびオブジェクトごとに、入力ステートメントを作成します。ロード機能を使用すると、初期構造およびオブジェクトをデータ・キャッシュにロードすることができ、かつデータ・キャッシュを随時更新し、保守することもできます。

RODM ロード機能は、次の 2 つのタイプの入力ステートメントを受け入れます。

- *高水準 RODM ロード機能*ステートメント。これを用いると、クラスおよびオブジェクトの作成、削除を行うことができます。作成ステートメントごとに、1 つのクラスまたはオブジェクト、およびそのフィールドのすべてを定義します。RODM ロード機能ステートメント 1 つで、多数の RODM トランザクションの作業を行うことができます。
- *RODM ロード機能プリミティブ・ステートメント*。これを用いると、高水準 RODM ロード機能ステートメントでは不可能な RODM データ・キャッシュへの変更を行うことができます。例えば、RODM ロード機能プリミティブ・ステートメントを使用すると、オブジェクト独立メソッドを起動することも、データ・キャッシュのサブフィールドの値を変更することもできます。

RODM に関するその他の資料

本書では、ネットワークの GMFHS データ・モデルへの定義、データ・モデルの RODM データ・キャッシュへのロード、および RODM を使用するアプリケーション・プログラムおよびメソッドの作成に関して説明します。 NetView ライブラリー内の他の資料では、本書で概括したタスクを実行する際に使用できる RODM に関する情報を記載しています。

IBM Tivoli NetView for z/OS インストール: グラフィカル・コンポーネントの構成
NetView プログラムをインストールするための手順、およびシステムをカス

タマイズし、ネットワークを個々の要件に合わせて調整するための手順を説明しています。以下のトピックが入っています。
- RODM を MVS サブシステムとして定義する
- セキュリティーを設定する
- RODM ログを定義する
- RODM 開始プロシージャーを更新する
- RODM のグローバル変数を定義する
- EKGCUST メンバーを使用して RODM を定義する
- RODM DSIQTSK タスクの初期設定値を定義する

IBM Tivoli NetView for z/OS アドミニストレーション・リファレンス
以下の情報が記載されています。
- RODM および RODM 自動化タスクの定義に使用するステートメント
- EKGCUST メンバーを使用して RODM を定義する

IBM Tivoli NetView for z/OS セキュリティー・リファレンス
本書には、RODM セキュリティーの定義に関する情報が含まれています。

IBM Tivoli NetView for z/OS 自動操作ガイド
RODM を NetView 自動化の一部として使用する方法を説明しています。

IBM Tivoli NetView for z/OS Troubleshooting Guide
本書では、以下を含む診断および障害追及に関する情報を記載しています。
- デバッグのメソッド
- RODM ログ
- RODM ダンプ・ユーティリティー
- RODM ロード・ユーティリティー・エラー・リスト
- RODM API 統計を使用して RODM のパフォーマンスを改善する

IBM Tivoli NetView for z/OS Messages and Codes Volume 2 (DUI-IHS)
RODM が戻すメッセージを説明しています。 RODM メッセージは、接頭部に EKG がつきます。

IBM Tivoli NetView for z/OS SNA トポロジー・マネージャー インプリメンテーション・ガイド
SNA トポロジー・マネージャーの使用方法を説明しています。

IBM Tivoli NetView for z/OS データ・モデル・リファレンス
GMFHS、SNA トポロジー・マネージャー、およびマルチシステム・マネージャーのデータ・モデルについて説明しています。

IBM Tivoli NetView for z/OS Tuning Guide
本書では、RODM および GMFHS のチューニングに関する情報を記載しています。

IBM Tivoli NetView for z/OS ユーザーズ・ガイド
本書では、RODM および GMFHS を含む、 NetView の使用方法に関するオペレーターならびにシステム・プログラマー用の情報を記載しています。

RODM 用のツール

NetView は、RODM とともに使用するために、以下のツールを提供します。

- RODMView
- RODM アンロード機能
- FLCARODM (RODM アクセス機能)
- BLDVIEWS
- ビジュアル BLDVIEWS (VBV)

これらのツールの詳細については、 575 ページの『付録 A. RODM ツール』を参照してください。

RODM のサンプルおよびマクロ

NetView プログラムには、 RODM でそれぞれのネットワークをセットアップし、アプリケーション・プログラムおよびメソッドの作成方法を学習するときに使用できるサンプル・コードが備えられています。また、作成するアプリケーション・プログラムおよびメソッドに組み込むマクロも提供しています。サンプル・コードとマクロは、 NetView プロダクトとともに出荷され、次のライブラリーに入っています。

NETVIEW.V5R3M0.CNMSAMP
このライブラリーには、ネットワークを定義し、 RODM にロードするときに使用できるサンプル・コードが入っています。このライブラリーには、RODM への接続方法、ならびに GMFHS 自動化を使用するアプリケーション・プログラムとメソッドの作成方法を学ぶときに使用できるサンプル・コードが入っています。機能サンプルの名前の接頭部には、EKG5 および EKG6 が付きます。

NETVIEW.V5R3M0.SCNMMAC1
このライブラリーには、アプリケーション・プログラムおよびメソッドに組み込むマクロが入っています。これらのマクロの名前の接頭部には、EKG1、 EKG2、EKG3、EKG4 が付きます。これらのマクロの詳細については、 423 ページの『第 14 章 アプリケーション・プログラミングの解説』を参照してください。

これらのマクロのいくつかと、サンプル・コードの一部については、本書で説明しています。個々のマクロもしくは機能の名前は、それぞれの説明がある個所に列挙してあります。

第 2 部 リソースを NetView に定義する

第 2 章 ネットワークを GMFHS に定義する

この章では、GMFHS データ・モデルに基づいてネットワーク構成を手動で NetView に定義する方法について説明します。 この章では、まずサンプル・ネットワークを説明してから、手動でネットワークを定義するステップを紹介します。

注:

1. SNA ネットワークを RODM に定義するために、SNA トポロジー・マネージャーを使用することができます。詳細については、「*IBM Tivoli NetView for zOS SNA トポロジー・マネージャー インプリメンテーション・ガイド*」を参照してください。
2. 非 SNA ネットワークを RODM に定義する場合には、マルチシステム・マネージャー・アクセス機能を使用することができます。

手動による RODM へのネットワークの定義を支援するために、 NetView とともにサンプル・オブジェクト・ロード・ファイル DUIFSNET が提供されています。サンプル・ファイルには、サンプルのネットワークを RODM に定義する RODM ロード機能ステートメントが入っています。

手動によるネットワークの定義は、RODM ロード機能ステートメントを使用して行います。このステートメントは、次の任意の方法で生成することができます。

- 構成情報をリポジトリーに保管している場合は、変換プログラムを作成して、275 ページの『第 10 章 RODM ロード機能を使用する』に示す RODM ロード・ファイル形式に情報を変換する。
- テキスト・エディターを使って構成定義を作成する。

ネットワークは、RODM ロード機能を使用せずに定義することもできます。ネットワーク構成情報をデータベースに保管している場合は、構成情報を直接 RODM に入れる RODM ユーザー・アプリケーションを作成することができます。 ユーザー・アプリケーションは、RODM ユーザー API を呼び出して、データを RODM に入れます。 RODM ユーザー・アプリケーションの作成については、343 ページの『第 11 章 RODM を使用するアプリケーションを作成する』を参照してください。

手動によるネットワーク定義の概要

ネットワーク構成を手動で RODM に定義するときは、次のタスクをリストされた順序で実行します。

1. 構成を分析し、RODM に定義する必要のあるネットワーク・エレメントを識別する。
2. ネットワーク内に管理オブジェクトを定義する。管理オブジェクトは、以下のものです。
   - SNA ドメイン
   - ネットワーク管理ゲートウェイ
   - 非 SNA ドメイン

3. ネットワーク内の管理されるオブジェクトを定義する。管理されるオブジェクトは、以下のものです。
   - サービス・ポイントを介した状況、アラート、または両方を受信するための実在の非 SNA オブジェクト
   - 非 SNA オブジェクトを定義したビューに現れる SNA オブジェクト
   - 集合オブジェクト
4. ネットワーク内のリソースの接続関係を定義する。接続関係の例としては、論理接続と物理接続、親子、論理構成、物理構成、部分構成があります。
5. オペレーターが使用しやすいような構成のビューのタイプを定義する。

サンプル・ネットワーク

この章では、サンプル・ネットワーク (21 ページの図 6) を使用してネットワークを RODM に定義する方法について説明します。このネットワークには、SNA コンポーネントと非 SNA コンポーネントの両方が入っています。

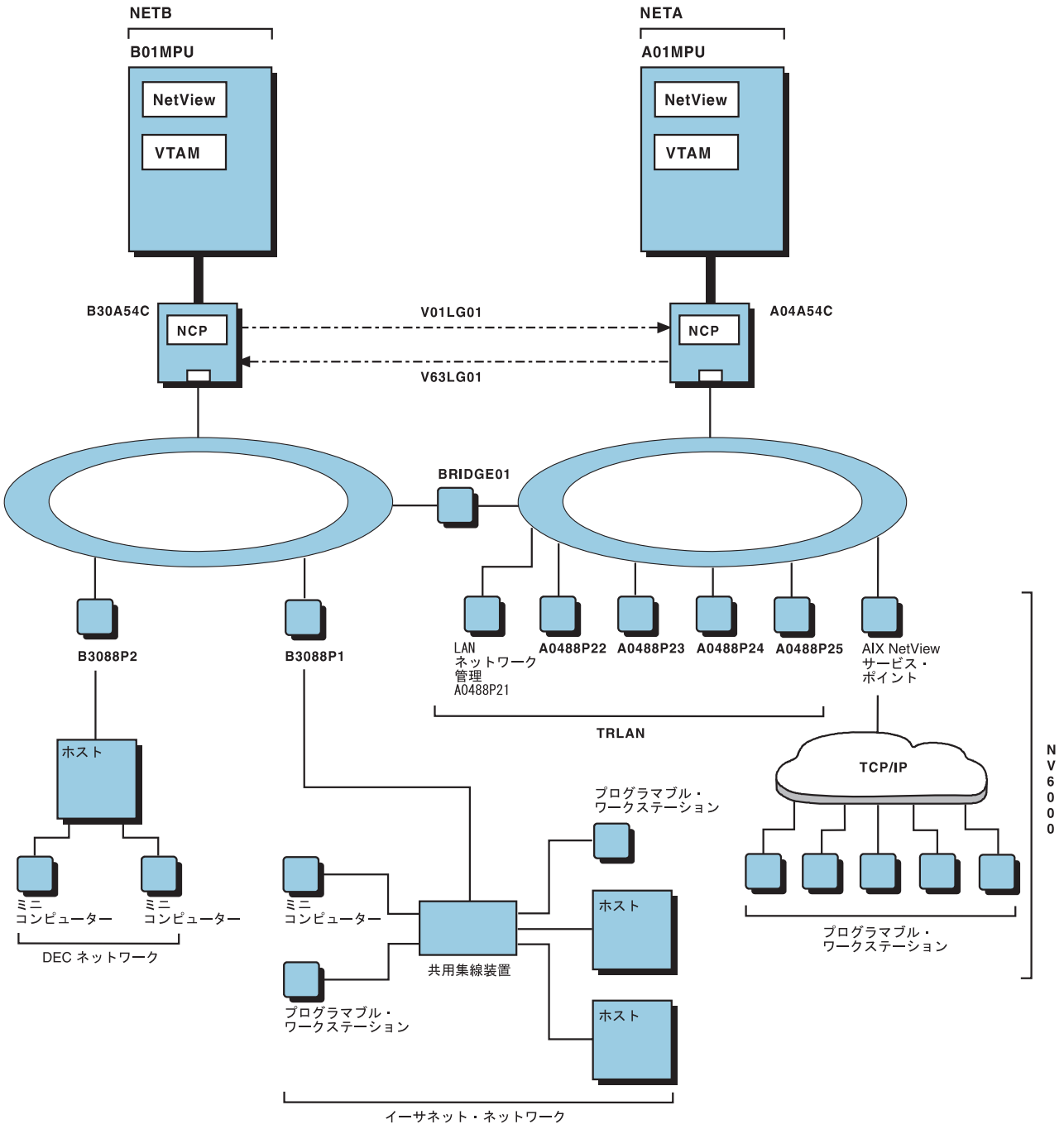


図 6. サンプル・ネットワーク

サンプル・ネットワークの SNA コンポーネント

このサンプル・ネットワークは、ネットワーク NETA およびネットワーク NETB の 2 つのネットワーク・ドメインから構成されています。

ネットワーク NETA は次の要素から構成されています。

- NetView プログラムおよび VTAM を実行中のホスト・プロセッサー A01MPU
- ホスト・プロセッサーをトークンリングに接続する NCP A04A54C

- TRLAN ネットワークを管理する NMG A0488P21
- NV6000 を管理する NMG A0488P31
- TRLAN ネットワーク
- NV6000 ネットワーク

ネットワーク NETB は次の要素から構成されています。
- NetView プログラムおよび VTAM を実行中のホスト B01MPU
- ホストをトークンリングに接続する NCP B30A54C
- イーサネット・ネットワークを管理する NMG B3088P1
- DEC ネットワークを管理する NMG B3088P2
- イーサネット・ネットワーク
- DEC ネットワーク

2 つのホスト・システムが、NCP/ トークンリング相互接続 (NTRI) を介して、 2 つの論理ゲートウェイ・コネクターの V01LG01 および V63LG01 によって接続されます。 2 つの NCP 間のこれらの論理ゲートウェイ・コネクターは、その間のブリッジによって 2 つのトークンリング LAN に結び付けられます。サービス・ポイントとその NCP を接続する SNA リンクも、その基盤となる物理接続にトークンリングを使用します。

サンプル・ネットワーク内のホスト、NCP、サービス・ポイント、ゲートウェイ・コネクター、およびリンク・コネクターは、 NetView および VTAM プログラムによって管理される SNA リソースです。フォーカル・ポイント NetView、GMFHS、および RODM は、ホスト A01MPU で実行されます。 NetView 管理コンソールは、これらの SNA リソースをモニターして、リソースのビューを作成します。

サンプル・ネットワークの非 SNA コンポーネント

NetView 管理コンソール は、サンプル・ネットワークの非 SNA コンポーネントを認識しません。 NetView 管理コンソール がこれらの非 SNA コンポーネントを管理するには、 GMFHS データ・モデルを使用して、非 SNA コンポーネントを RODM に定義する必要があります。

サービス・ポイント

サンプル・ネットワークには、ネットワーク管理ゲートウェイとして定義されている 4 つのサービス・ポイントがあります。

- イーサネット・ネットワークを管理するトランザクション・プログラム SYNOPTAP を実行する NMG B3088P1
- DEC ネットワークを管理するトランザクション・プログラム NAP を実行する NMG B3088P2
- トークンリング LAN 内で稼働し、 TRLAN ネットワークを管理するトランザクション・プログラム LANMGR を実行する NMG A0488P21
- NV6000 ネットワークを管理するトランザクション・プログラム NMG A0488P31 を実行する NMG A0488P31

DEC ネットワーク

23 ページの図 7 は、サンプル・ネットワークに示されている DEC ネットワークの詳細です。 DEC ネットワークは、以下から構成されます。

- サービス・ポイント B3088P2 に接続されている DEC ホスト RALV4
- ミニコンピューター RALXT1 に RALV4 を接続するリンク TX-0-2
- ミニコンピューター RALXT2 に RALV4 を接続するリンク TX-1-2

NAP トランザクション・プログラムは、サービス・ポイント B3088P2 内で稼働し、これらのリソースに関連したイベントをアラートに変換してから、このアラートを NetView 管理コンソール・フォーカル・ポイント・ホスト A01MPU に送信します。このトランザクション・プログラムは、これらのリソース用のコマンドも受け入れます。

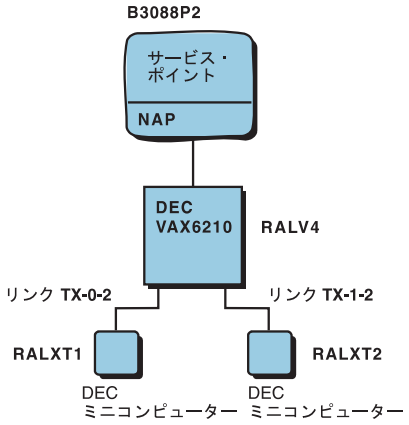


図 7. DEC ネットワーク

イーサネット・ネットワーク

24 ページの図 8 は、サンプル・ネットワークに示されているイーサネット・ネットワークの詳細です。サービス・ポイント B3088P1 上のアダプターは、サービス・ポイントを共用集線装置 CNTR3000 に接続します。集線装置は、次の 3 つのコネクターによってホストおよびワークステーションに接続されます。

- コネクター OEMLAB。非 SNA ホストの VAX6210 および 9370 を関連付けています。
- コネクター NSL_ENET。 DOS ワークステーション DOSTCPIP および RISC System/6000® ワークステーション RS6000 に関連付けられています。
- コネクター NSL_B202。ホスト AS400 に関連付けられています。

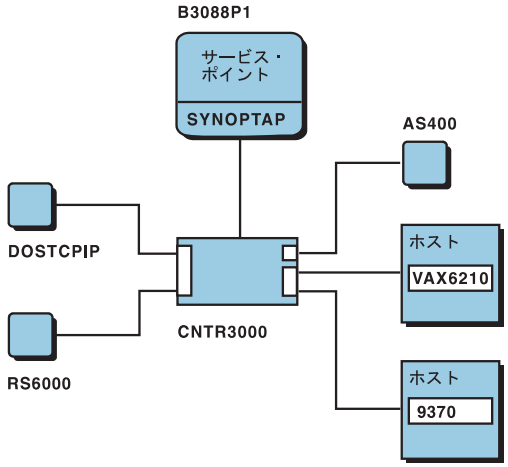


図 8. イーサネット・ネットワーク

トークンリング・ローカル・エリア・ネットワーク

25 ページの図 9 は、トークンリング・ネットワーク TRLAN です。以下から構成されます。

- トークンリングに NCP A04A54C を接続するためのアダプター TRADPTR
- トークンリング・インターフェース・カプラー (TIC) に相当する SNA 回線のリソース A04N1088
- TIC 用の SNA 物理装置 (PU) として定義されているリソース A04P1088
- プログラマブル・ワークステーション用のトークンリング・アダプターで、 LAN マネージャー内の該当するアダプター・アドレスに関連付けられるリソース A0488P21 から A0488P25 まで
- NETB 内の別のトークンリングに接続する LAN のブリッジである BRIDGE01

サンプル・ネットワークは、プログラマブル・ワークステーションを表す SNA PU 2 リソースを SNA に定義し、 SNA PU を A0488P21 から A0488P25 に指定して、 PU をサポートする各ワークステーションに常駐するアダプターに SNA PU を関連付けます。サンプル・ネットワークは、DisplayResourceName フィールドを使用して、トークンリング・ネットワークのリソースごとに表示する名前を指定します。例えば、オブジェクト LANMGR.10005AC35CA0 にはその DisplayResourceName フィールドがあって、 A0488P21 に設定されています。これで、オペレーターが区別できるリソースの名前を表示することができます。

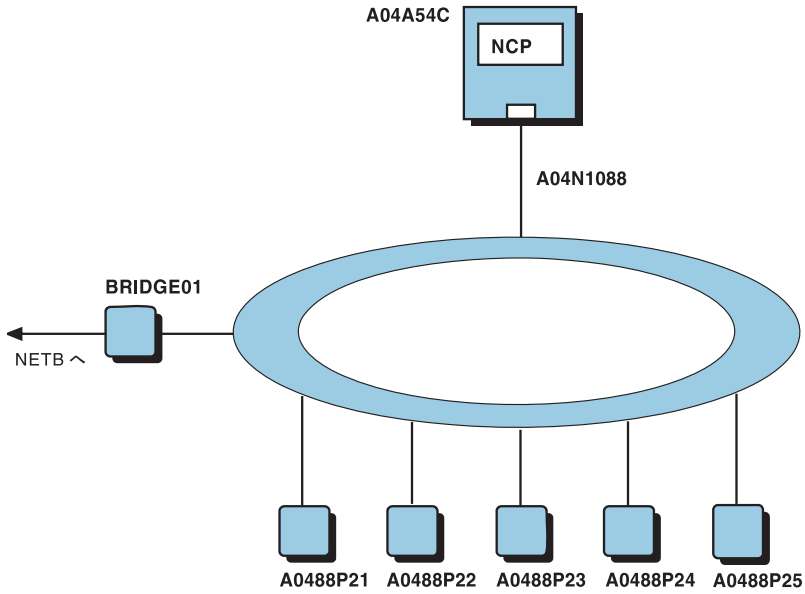


図 9. トークンリング LAN

NV6000 ネットワーク

26 ページの図 10 は、サンプル・ネットワークに示された NV6000 ネットワークの詳細です。 NV6000 ネットワークは、以下から構成されます。

- Tivoli NetView for AIX、 AIX NetView サービス・ポイント、および AIX SNA Server/6000 が稼働する RS/6000® ホスト
- プログラマブル・ワークステーション T46A、T47A、T47B、T48A、および T48B

AIX SNA サーバー /6000 は、PU 名 A0488P31 で構成され、 TivoliNetView for AIX の SPAPPLD アプリケーションは、A94306F8 で構成されます。ワークステーション T46A、T47A、T47B、T48A、および T48B は、 Tivoli NetView for AIX が導入されている TCP/IP ネットワークに接続されます。 Tivoli NetView for AIX は、これらのリソースに関連する選択されたトラップをアラートに変換してから、このアラートをフォーカル・ポイント・ホスト A01MPU に送信します。 A94306F8 トランザクション・プログラムは、これらのリソース用のコマンドも受け入れます。

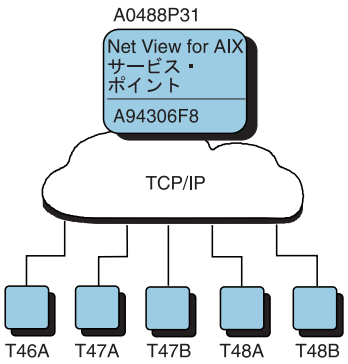


図 10. NV6000 ネットワーク

定義するネットワーク・エレメントを識別する

ネットワークを RODM に正しく定義するには、ネットワーク・コンポーネントとその構成を見定めてから、ネットワーク・エレメントを識別します。識別するエレメントは、次のとおりです。

- 管理オブジェクト
- 管理されるオブジェクト
- 接続関係
- 望ましいビュー

管理オブジェクトを識別する

管理オブジェクトとは、ネットワークのコンポーネントを制御し、コンポーネントを NetView プログラムに接続するプログラムを表します。これらのプログラムは、NetView プログラムにアラートを送り、ネットワーク内のリソースの状況を更新し、制御するリソースの NetView プログラムからコマンドを受け取ります。
RODM が識別する必要がある管理オブジェクトには、次の 3 タイプがあります。

- SNA ドメイン
- ネットワーク管理ゲートウェイ
- 非 SNA ドメイン

SNA ドメイン

SNA ドメインは、1 つの NetView プログラムを表します。これらの SNA リソースがシャドー・オブジェクトとして RODM に定義される場合は、 SNA リソースのアラートを出すことができる NetView プログラムごとに 1 つの SNA ドメインを RODM に定義する必要があります。

SNA シャドー・オブジェクトを定義していなくても、 NetView プログラムに報告する非 SNA ドメインがある場合は、その NetView プログラムごとに SNA ドメインを定義する必要があります。これで、非 SNA オブジェクトへのコマンドのサポートが確保され、 GMFHS は非 SNA ドメインのリソースの状況が認識されているかどうかを判断できるようになります。シャドー・オブジェクトについては、 28 ページの『管理されるオブジェクトを識別する』を参照してください。

サンプル・ネットワークでは、ホスト B01MPU および A01MPU に常駐する NetView プログラムごとに SNA ドメインが 1 つ定義されます。

ネットワーク管理ゲートウェイ

ネットワーク管理ゲートウェイ (NMG) は、SNA ネットワーク管理システムである NetView プログラムと、 1 つまたは複数の非 SNA ネットワークのネットワーク管理機能の間のゲートウェイです。 1 つまたは複数のトランザクション・プログラムを実行する、 AIX および NetView/PC サービス・ポイントは、NMG の例です。 NMG は、サービス・ポイント・コマンド・サービス (SPCS) サポートを使用するか、他のなんらかの手段によってアラートを送るユーザー作成のサービス・ポイントの場合もあります。

ネットワーク管理ゲートウェイをサポートする他の 2 つの NetView 機能は、プログラム間インターフェース (PPI) とオペレーター・ステーション・タスク (OST) です。プログラム間インターフェースは、ネットワーク管理情報を交換する際のパス、ならびに非 SNA リソースを管理し、 NetView アドレス・スペース以外のアドレス・スペースのフォーカル・ポイント・ホストで実行するアプリケーション用のコマンドを提供します。 OST は、非 SNA リソースのネットワーク管理コマンドを受け入れ、その状況を提供する、コマンド・プロシージャーおよびコマンド処理プログラムを実行します。

サンプル・ネットワークには、ネットワーク管理ゲートウェイとして定義されている 4 つのサービス・ポイントがあります。

- B3088P2
- B3088P1
- A0488P31
- A0488P21

非 SNA ドメイン

非 SNA ドメインは、モニターする非 SNA ネットワークごとに定義する必要があります。非 SNA ドメインは、サービス・ポイント、トランザクション・プログラム、およびエレメント管理システムの任意の組み合わせごとに、個別に識別されます。

トランザクション・プログラム (TP) は、SNA ネットワーク内から非 SNA ネットワークを管理します。エレメント管理システム (EMS) は、ネットワークの他の側面から、または固有の側面から非 SNA ネットワークを管理します。トランザクション・プログラムは、ネットワークを管理する際にエレメント管理システムと対話します。

トランザクション・プログラムおよびエレメント管理システムは、使用するトランザクション・プログラムによって、非 SNA リソースの NetView に到着するアラートに関係する場合と、関係しない場合があります。 Non_SNA_Domain_Class オブジェクトは、 NetView プログラムに流れるアラートで識別される、サービス・ポイント、トランザクション・プログラム、およびエレメント管理システムの組み合わせごとに定義する必要があります。

サンプル・ネットワークでは、以下に挙げたそれぞれのネットワークごとに非 SNA ドメインが定義されています。

- B3088P1 という名前のサービス・ポイント、SYNOPTAP という名前のトランザクション・プログラムがあり、エレメント管理システムがないイーサネット・ネットワーク。

- B3088P2 という名前のサービス・ポイントおよび NAP という名前のトランザクション・プログラムがある DEC ネットワーク。
- A0488P21 という名前のサービス・ポイントおよび LANMGR という名前のトランザクション・プログラムがある TRLAN ネットワーク。
- A0488P31 という名前のサービス・ポイントおよび A94306F8 という名前のトランザクション・プログラムがある NV6000 ネットワーク。

管理されるオブジェクトを識別する

管理されるオブジェクトは、ユーザーが管理するネットワーク・リソースを表します。これらのオブジェクトには、オブジェクトが表すネットワーク・リソースに関する状況情報と構成情報が入っています。管理されるオブジェクトには、 NetView プログラムに状況を送り、リソースのコマンドを受け取る管理オブジェクトが必要です。 RODM を用いて管理したいネットワーク・リソースごとに、1 つの管理オブジェクトを識別します。 RODM には、次の 4 つのタイプの管理されるオブジェクトを定義することができます。

- SNA トポロジー・マネージャー・クラス・オブジェクト。 SNA トポロジー・マネージャー・オブジェクトは、サンプル・ネットワーク DUIFSNET には組み込まれていません。詳細については、「*IBM Tivoli NetView for z/OS SNA トポロジー・マネージャー インプリメンテーション・ガイド*」を参照してください。
- GMFHS_Shadow_Objects_Class オブジェクト
- GMFHS_Managed_Real_Objects_Class オブジェクト
- GMFHS_Aggregate_Objects_Class オブジェクト

GMFHS_Shadow_Objects_Class オブジェクト

SNA トポロジー・マネージャーは、管理するリソースの SNA オブジェクトを作成します。 SNA トポロジー・マネージャーに管理されない他の SNA リソースがある場合は、それを表す GMFHS_Shadow_Objects_Class オブジェクトを作成することができます。 GMFHS_Shadow_Objects_Class オブジェクトは、非 SNA リソースに関係付けたい SNA リソースを表します。シャドー・オブジェクトの状況は、 RODM 内には保存されませんが、 NetView 管理コンソール SNA サポートによって維持されます。 NetView 管理コンソール・ワークステーションでシャドー・オブジェクトが入っているビューが表示されると、 NetView 管理コンソールは各オブジェクトの状況を記入して維持します。

注: NetView 管理コンソールは、シャドー・オブジェクトの状況を維持しません。シャドー・オブジェクトは NetView 管理コンソールに表示されますが、状況は常に不明です。

SNA リソースを、サンプル・ネットワークの 4 つの非 SNA ネットワークのような、非 SNA リソースに関係付けたい場合は、 GMFHS_Shadow_Objects_Class で SNA リソースをオブジェクトとして定義する必要があります。これらの GMFHS_Shadow_Objects_Class オブジェクトは、物理装置 (PU)、論理装置 (LU)、およびリンク接続などの SNA リソースであり、RODM で定義されるため、関連する非 SNA リソースに関係付けることができます。

サンプル・ネットワークでは、論理リンク・コネクター V01LG01 と V63LG01 が定義されており、 2 つの NCP と 2 つのトークンリング LAN を接続する物理パ

スに関係付けられます。論理リンク・コネクターのいずれかが正常でない状況で表示されると、オペレーターは、そのコネクターを選んで、そのリソースに関する詳細情報を要求することができます。次に、GMFHS は、RODM のコネクターに関する GMFHS_Shadow_Objects_Class オブジェクトを探し、構成の関係に従ってコネクターを構成するリソースを判別し、詳細情報を構成するビューを動的に構成して表示します。

GMFHS_Managed_Real_Objects_Class オブジェクト

GMFHS_Managed_Real_Objects_Class オブジェクトは、 NetView 管理コンソールにより管理される非 SNA リソースを表します。これらのリソースのそれぞれの状況は、ネットワークを通じて送られたアラートおよびコマンド応答によって判別され、 RODM に保管されます。このようなリソースの例としては、マルチプレクサー、モデム、ソフトウェア・アプリケーション、および T1 エレメント・マネージャーなどがあります。管理するリソースごとに、
GMFHS_Managed_Real_Objects_Class オブジェクトを GMFHS に定義する必要があります。 GMFHS_Managed_Real_Objects_Class に子クラスを追加した場合は、代わりに子クラスのオブジェクトを作成してください。詳細については、「*IBM Tivoli NetView for z/OS データ・モデル・リファレンス*」を参照してください。

サンプル・ネットワークでは、4 つの非 SNA ネットワークの当該リソースごとに、 GMFHS_Managed_Real_Objects_Class オブジェクトが定義されます。例えば、図 7 に示されている DEC ネットワークでは、次のリソース用に
GMFHS_Managed_Real_Objects_Class オブジェクトが定義されています。

- DEC ホスト RALV4
- ミニコンピューター RALXT1 および RALXT2
- リンク TX-0-2 および TX-1-2

GMFHS_Aggregate_Objects_Class オブジェクト

GMFHS_Aggregate_Objects_Class オブジェクトは、オブジェクトのグループを表します。このオブジェクト・グループは、任意の数および組み合わせの実オブジェクトおよび集合オブジェクトから構成されます。集合オブジェクトの例には、データ・センター、複数コンポーネントから成る複合回線、および散在したリソースのグループがあります。

集合オブジェクトは、GMFHS に定義し、基礎となる
GMFHS_Managed_Real_Objects_Class オブジェクトに関係付けることができます。集合オブジェクトの状況は、集合オブジェクトが表す実オブジェクトの状況によって判別されます。 GMFHS_Aggregate_Objects_Class に子クラスを追加した場合は、代わりに子クラスのオブジェクトを作成してください。

他の集合オブジェクトから構成する集合オブジェクトを定義することもできます。高水準集合オブジェクトの状況は、低水準集合オブジェクトの状況に影響を与える実オブジェクトの状況によって判別されます。低水準集合オブジェクトの状況は、高水準集合オブジェクトの状況に影響を与えません。集合オブジェクトの状況に影響を与えるのは実オブジェクトのみです。

GMFHS_Shadow_Objects_Class オブジェクトには状況フィールドがないので、それらが表す実リソースは、集合オブジェクトの状況に影響を与えません。

GMFHS は、9 レベルまでの集約をサポートします。 1 つのレベルの集約は、1 つまたは複数の実もしくは集合オブジェクトから構成する 1 つの集合オブジェクトです。実オブジェクトが、集合体の親オブジェクトの子として定義され、この集合体の親オブジェクトが別の親の集合オブジェクトの子として定義される場合には、 2 つのレベルの集約が定義されています。

集合オブジェクトの定義は、完全な階層内で行う必要があります。集合オブジェクトは、集約階層内でそれより下にある集合オブジェクトの子集合オブジェクトとして定義することはできません。

集約の使用に関する詳細については、 153 ページの『集約の概念』を参照してください。

サンプル・ネットワークでは、集合オブジェクトが非 SNA ネットワーク (イーサネット、DEC、NV6000、および TRLAN) ごとに定義されています。これらの集合オブジェクトのそれぞれが、それぞれのネットワークの実リソースのすべてを表します。これらの集合オブジェクトのそれぞれの状況は、基礎となる実リソースの全体の状況を反映します。

このほかに、次の 2 つの集合オブジェクトも定義されています。

- 集合オブジェクト ETHERNET および DEC から構成される集合オブジェクト WESTCTR。 WESTCTR の状況は、イーサネットおよび DEC ネットワーク内の実リソースの状況によって判別されます。
- 集合オブジェクト NV6000 および TRLAN から構成される集合オブジェクト EASTCTR。 EASTCTR の状況は、NV6000 および TRLAN ネットワーク内の実リソースの状況によって判別されます。

これらの集合オブジェクトは、 33 ページの『ビューを識別する』で説明されている高水準ビュー内に表示されます。

接続関係を識別する

接続関係は、RODM で定義されたリソースを相互に接続可能にする方法です。 これらの関係は、物理、論理、または対等のいずれであってもかまいません。
GMFHS データ・モデルは、次の関係をサポートします。

- ComposedOfLogical および IsPartOf
- ComposedOfPhysical および IsPartOf
- AggregationParent および AggregationChild
- ParentAccess および ChildAccess
- PhysicalConnPP
- LogicalConnPP
- PhysicalConnUpstream および PhysicalConnDownstream
- LogicalConnUpstream および LogicalConnDownstream
- BackboneConnPP

ComposedOfLogical および IsPartOf

ComposedOfLogical および IsPartOf は、 1 つのオブジェクトが他の複数のオブジェクトから論理的に構成される場合の論理関係を作成します。他のオブジェクトは、

順番に最初のオブジェクトの一部となります。この論理関係は、任意の数の実オブジェクト、集合オブジェクト、もしくはシャドー・オブジェクト間であってもかまいません。

サンプル・ネットワークでは、シャドー・オブジェクト NETV.WECONN は、 NCP A04A54C と NCP B30A54C 間のゲートウェイ・コネクターを表します。これには、シャドー・オブジェクト V01LG01 および V63LG01 との ComposedOfLogical 関係があります。これらの GMFHS_Shadow_Objects_Class オブジェクトは、順番に、 GMFHS_Shadow_Objects_Class オブジェクト NETV.WECONN との IsPartOf 関係をもちます。

SNA トポロジー・マネージャーがインストールされている場合、ComposedOfLogical 関係は、シャドー・オブジェクトの代わりに SNA トポロジー・マネージャー・オブジェクトを使用して設定することができます。

オペレーターがビューで NETV.WECONN オブジェクトを選択し、詳細を要求すると、 GMFHS は NETV.WECONN オブジェクトの ComposedOfLogical 関係に従って、この関係を満足させるすべてのオブジェクトを検索します。 GMFHS はこれらのオブジェクトから構成するビューを作成し、それをワークステーションに送って表示します。 NETV.WECONN オブジェクトに ComposedOfPhysical 関係も定義された場合、 GMFHS はその関係のビューも作成し、それをワークステーションに送って表示します。

ComposedOfPhysical および IsPartOf

ComposedOfPhysical および IsPartOf は、 1 つのオブジェクトが他の複数のオブジェクトから物理的に構成される場合の物理関係を作成します。他のオブジェクトは、順番に最初のオブジェクトの一部となります。

サンプル・ネットワークの場合は、 23 ページの図 7 に見られるように、非 SNA ネットワーク全体を表す DEC という名前の GMFHS_Aggregate_Objects_Class オブジェクトには、ホストと 2 つのミニコンピューターを表す RODM のオブジェクトとの ComposedOfPhysical 関係があります。これらのリソースを表す RODM の GMFHS_Managed_Real_Objects_Class オブジェクトは、順番に、集合オブジェクト DEC との IsPartOf 関係をもちます。

オペレーターがビューで DEC オブジェクトを選択し、詳細を要求すると、GMFHS は DEC オブジェクトの ComposedOfPhysical 関係に従って、この RODM からの関係を満足させるすべてのオブジェクトを検索し、これらのオブジェクトから構成するビューを作成して、それをワークステーションに送り、要求側のオペレーターに表示します。 DEC オブジェクトに ComposedOfLogical 関係も定義された場合、GMFHS はその関係ビューも作成して、 ComposedOfPhysical 関係ビューとともに、それをワークステーションに送って表示します。

ComposedOfPhysical および IsPartOf は、一般に集合オブジェクトと基礎となる実オブジェクト間の関係を定義するときに使用されますが、これがこの関係の唯一の使い方ではありません。例えば、GMFHS_Managed_Real_Objects_Class のオブジェクトは、他の GMFHS_Managed_Real_Objects_Class オブジェクトの構成の際に定義することができます。この場合、集約は行われませんが、オペレーターが最初のオブジェクトを選択し、詳細を求めると、最初のオブジェクトが構成されるオブジェクトのビューが表示されます。

AggregationParent および AggregationChild

AggregationParent および AggregationChild は、 1 つのオブジェクトが 1 つまたは複数の集合体子に対する集合体親であるという関係を作成します。集合体親の状況は、集合体子の状況によって判別されます。

実オブジェクトの AggregationParent フィールドは、実オブジェクトが状況に影響を与えている集合オブジェクトのすべてにリンクします。実オブジェクトは、任意の数の集合オブジェクトの状況に影響を与えることができます。集合オブジェクトの AggregationChild フィールドは、その集合オブジェクトの状況に影響を与える実オブジェクトのすべてにリンクします。

GMFHS データ・モデルでは、 AggregationParent フィールドと AggregationChild フィールド間には直接リンクを作成しません。代わりに、GMFHS がこれらのフィールドをリンクするメソッド、DUIFCUAP を提供します。例えば、次の RODM ロード機能プリミティブ・ステートメントにより、実オブジェクト DECNET.RALV4.RALXT2 の AggregationParent フィールドは集合オブジェクト DEC の AggregationChild フィールドにリンクします。

```
OP DUIFCUAP INVOKED_WITH (SELFDEFINING)
   ((CHARVAR)'LINK'
   (CHARVAR)'GMFHS_Managed_Real_Objects_Class.DECNET.RALV4.RALXT2'
   (CHARVAR)'GMFHS_Aggregate_Objects_Class.DEC');
```

DUIFCUAP メソッドは、これらのリンクを外すときにも使用されます。

ParentAccess および ChildAccess

ParentAccess および ChildAccess フィールドは、 GMFHS が構成親ビューと構成子ビューを作成するときに使用します。 ParentAccess および ChildAccess は、 1 つのオブジェクトが 1 つまたは複数の子オブジェクトの親であるという関係を作成します。

オペレーターがリソースを選び、構成親ビューを求めると、 GMFHS は RODM からリソースを検索し、リソースの上位階層全体を判別します。 GMFHS は、次にこの関係を満足させるオブジェクトのビューを作成し、そのビューをワークステーションで表示します。

この関係は、階層状に配置されたネットワークで、リソースの所有者へのパスを判別する場合にしばしば役立ちます。構成親ビューか構成子ビューのどちらかを使用する場合は、 ParentAccess および ChildAccess 関係の両方を定義する必要があります。

PhysicalConnPP

PhysicalConnPP は、1 つのリソースが対等関係にある他のリソースに物理的に接続する際の関係を作成します。この接続は、リンク接続へのノードであっても、ノード接続へのノードであってもかまいません。接続がノードへのノードである場合は、GMFHS は 2 つのオブジェクトを入れるビューを表示する際に、 2 つのノードの間にヌル・コネクターを挿入します。

サンプル・ネットワークでは、DEC ネットワークのホストが PhysicalConnPP 関係によって 2 つのリンクに接続され、 2 つのリンクは、順番に PhysicalConnPP 関係によってミニコンピューターに接続されます。オペレーターがリソースを選び、物

理的に接続されたこれらのリソースから構成されるビューの表示を求めると、GMFHS は、この関係を用いてビューを作成し、表示します。

LogicalConnPP

LogicalConnPP 関係の働きは、この関係が物理関係ではなく論理関係である点を除き、 PhysicalConnPP 関係と同じです。

サンプル・ネットワークでは、NCP B30A54C は、LogicalConnPP 関係によってゲートウェイ・コネクター V01LG01 に接続されます。ゲートウェイ・コネクター V01LG01 は、この同じ関係によって順番に NCP A04A54C に接続されます。

PhysicalConnUpstream および PhysicalConnDownstream

PhysicalConnUpstream および PhysicalConnDownstream は、方向が重要なオブジェクトを物理的に接続するときに使用します。これらの関係は、リソースを接続の一方または他方の終端でグループにまとめることが重要な場合に使用します。

例えば、マルチポイント・リンクとそれに接続するリソースを定義する場合は、PhysicalConnUpstream を使用してコントローラーをリンクにリンクし、PhysicalConnDownstream を使用していくつかの端末をリンクにリンクすることができます。この場合、オペレーターが物理接続性を表示するビューを要求したときに、リンクの一方の終端でコントローラーがリンクされ、端末はすべて他方の終端でリンクされます。

LogicalConnUpstream および LogicalConnDownstream

LogicalConnUpstream および LogicalConnDownstream は、方向が重要なオブジェクトを論理的に接続するときに使用します。これらの関係は、PhysicalConnUpstream および PhysicalConnDownstream 関係の論理的な対をなします。

BackboneConnPP

BackboneConnPP は、サブエリア・バックボーンの一部であるオブジェクトを示すときに使用します。

ビューを識別する

GMFHS は、大部分のビューを、ワークステーションで表示されるオブジェクト間で定義される関係に基づいて作成します。しかし、表示するオブジェクトの指定を行うビューには、例外、ネットワーク、構成、またはより詳細なビューの 4 つのタイプを定義することができます。定義するビューは、ネットワークによって異なります。

例外ビュー

例外ビューは、例外として定義された実オブジェクト、シャドー・オブジェクト、および集合オブジェクトの集合体です。これらのオブジェクト間で示される接続関係はありません。例外ビューは、オブジェクトの単なる図形リストです。このリストは、リソース・オブジェクトの DisplayStatus 値もしくは UserStatus 値でフィルターにかけることができます。

以下のリストに、多様なビジネスのニーズに合わせて例外ビューを定義する方法の例をいくつか示します。

- 非アクティブなすべての NCP を表示する場合。
- 自動化ルーチンによって再活動化される NCP を除く、非アクティブなすべての NCP を表示する場合。
- オペレーターの責任範囲に限定した障害リソースを入れるビューを定義する場合。
- 障害のあるすべての回線を表示する場合。
- リソースを例外ビューに組み込める日時を定義する場合。例えば、トークンリング LAN 上に、PU として表されるワークステーションがあるとします。 1 日の作業中に、このワークステーションをモニターして、それが適合の状態であることを確認します。その日の終わりにワークステーションの電源を切ると、PU が不良の状況に変わります。例外ビューの定義によっては、PU が例外ビューに組み込まれる場合があります。これを防ぐために、通常の稼働時間と非稼働時間のそれぞれに対して 2 つの定義を作成することができます。営業日の終了時には、タイマーで自動化ルーチンを開始して、通常の稼働時間の定義から非稼働時間の定義に変更し、 PU を例外ビューから除外します。詳細については、 118 ページの『例外ビューのオブジェクトおよび基準を定義する』を参照してください。

図 11 は、例外ビューの例です。

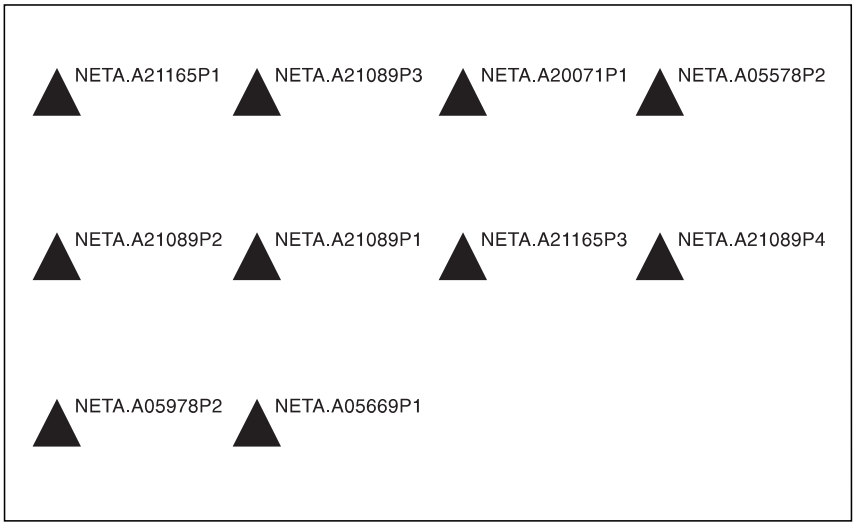


図 11. 例外ビューの例

ネットワーク・ビュー

ネットワーク・ビューは、オペレーターが一緒に表示する実オブジェクト、集合オブジェクト、およびシャドー・オブジェクトの集合体です。オペレーターがネットワーク・ビューを選ぶと、 GMFHS は RODM から該当するビュー・オブジェクトを検索し、このビューの一部として指定されるオブジェクトを判別します。次に GMFHS はこれらのオブジェクトを検索し、それらが入ったビューを作成して、そのビューをワークステーションで表示します。オブジェクトにオブジェクト間で定義された論理接続関係または物理接続関係があれば、これらの関係はビューに表示されます。

以下は、サンプル・ネットワーク用に定義されたネットワーク・ビューのうちの 2 つです。

- 高水準ネットワークの非 SNA コンポーネントの状況を示す BIGPIC という高水準ビュー
- 非 SNA ネットワークの管理に関与する SNA および非 SNA の主なコンポーネントを示す SAMPNET という管理ビュー

図 12 は、BIGPIC という高水準ビューです。このビューでは、WESTCTR は集合オブジェクト ETHERNET および DEC から構成する集合オブジェクトです。EASTCTR は集合オブジェクト TRLAN および NV6000 から構成する集合オブジェクトです。集合オブジェクト ETHERNET、DEC、TRLAN、および NV6000 は、管理される非 SNA ネットワークのそれぞれの実オブジェクトを表します。

実オブジェクトが状況を変えると、その状況は集合オブジェクト ETHERNET、DEC、TRLAN、および NV6000 にまで反映し、集合オブジェクト WESTCTR および EASTCTR にも反映します。したがって、高水準ビュー BIGPIC は、管理される非 SNA 実オブジェクトを表すビューをオペレーターに表示します。

WESTCTR の状況が適合から劣化に変わると、オペレーターは WESTCTR オブジェクトを選んで、詳細を要求することができます。集合オブジェクト ETHERNET および DEC から構成されるビューが、表示されます。あるいは、オペレーターは、そのオブジェクトを選んで、障害をもつリソース・ビューへの高速パスを要求することができます。このビューは、例外状態にある集合オブジェクト ETHERNET および DEC の実オブジェクトから構成されます。このタイプのビューは、多数の実オブジェクトと集合オブジェクトが入っているネットワークでは貴重な場合があります。

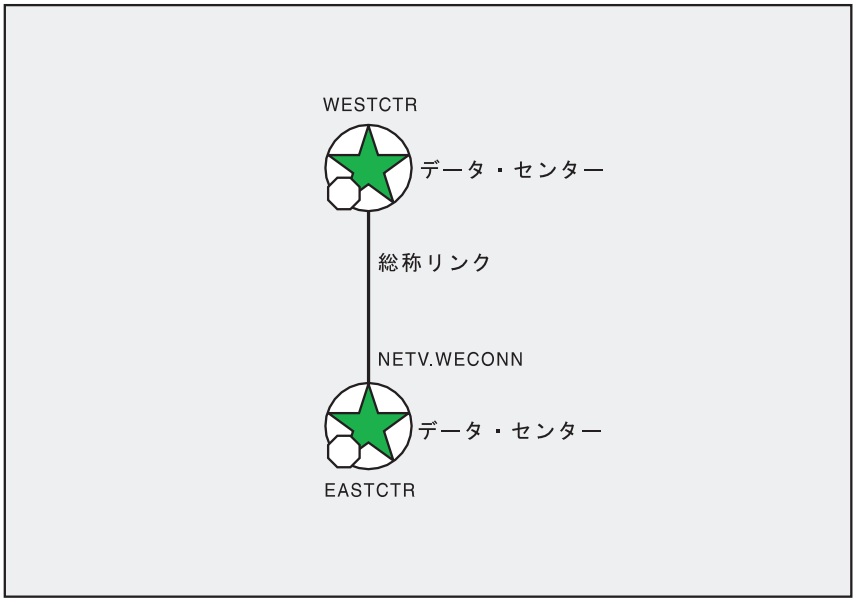


図 12. 高水準ビュー BIGPIC

36 ページの図 13 は、SAMPNET という名前の管理ビューです。このビューは、ネットワークの主な SNA および非 SNA コンポーネントを表示します。このビューには、SNA ホスト、NCP、およびサービス・ポイントに加えて、 2 つの NCP に

リンクする論理ゲートウェイ・コネクターが入ります。ネットワーク管理ゲートウェイであるサービス・ポイントに接続されるのは、集合オブジェクト ETHERNET、DEC、TRLAN、および NV6000 です。表示されている SNA リソースは、GMFHS_Shadow_Objects_Class オブジェクトとして GMFHS に定義されます。

このビューは、サンプルの非 SNA ネットワークの管理に関係する主な SNA および非 SNA コンポーネントと、その間の関係を示します。オペレーターは、SNA と非 SNA の両方の状況を調べることができます。非 SNA 集合体が状況を変えると、オペレーターはそれを選び、より詳細なビューを要求して、状況変更の原因を探すことができます。

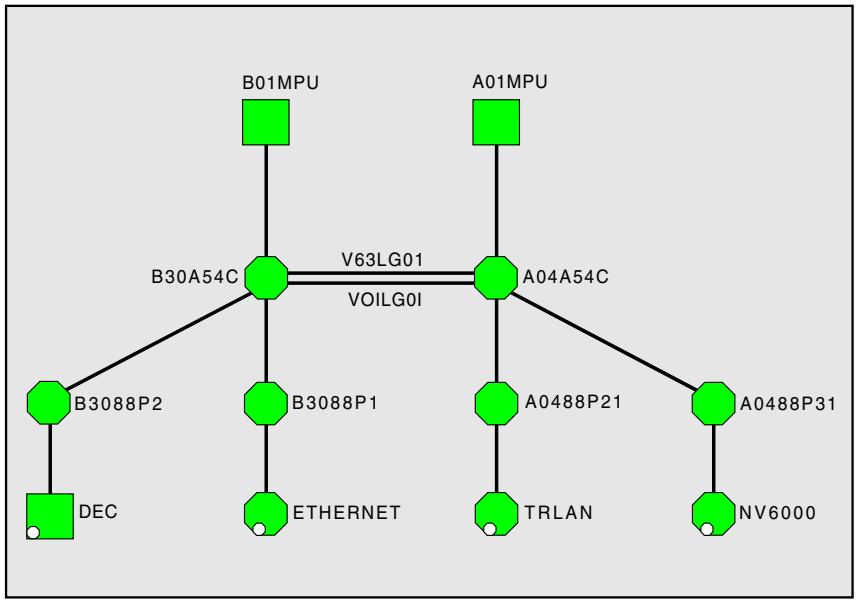


図 13. 管理ビュー SAMPNET

構成ビュー

次の構成ビューは、事前定義ビューです。これらは、オブジェクトを他のオブジェクトとの関係で示すのに使用します。

ビュー・タイプ
説明

対等 (Peer) 対等関係をもつオブジェクトを表示します。

物理 (Physical)
オブジェクト間の物理関係に基づくネットワーク内のオブジェクトを表示します。

論理 (Logical) オブジェクト間の論理関係に基づくネットワーク内のオブジェクトを表示します。

バックボーン (Backbone)
サブエリア・バックボーンをなすオブジェクトを表示します。

次の構成ビューは、動的に作成されたビューである可能性もあります。

- バックボーン (Backbone)
- 論理 (Logical)
- 物理 (Physical)

構成ビューの詳細については、111 ページの『特定ビューの場合のオブジェクトの展開処理の説明』を参照してください。サンプル・ネットワークには、次に説明する構成対等機能ビューが入っています。

構成対等機能ビューは、ビューで表示されるネットワークの対等関係を共用するオブジェクトの集合です。ビューを定義する際に、構成対等機能ビューに表示するオブジェクトを指定します。対等ビューには表示可能なオブジェクトならばどのタイプも指定できますが、選択するのは、対等関係を共用するオブジェクトのみを選択してください。そのような関係をもつオブジェクトは、ユーザーの責任で決めます。

オペレーターがビューでリソースを選択して、そのオブジェクトが定義される対等ビューの参照を求めると、 GMFHS は、適切なビューの作成のためにユーザーが定義した対等ビュー・オブジェクトを使用して、表示のため要求元オペレーターのワークステーションへそれらを転送します。ネットワーク・ビューを指定しているため、オブジェクトにオブジェクト間で定義された論理接続関係または物理接続関係があれば、これらの関係はビューに表示されます。

38 ページの図 14 は、サンプル・ネットワークの ETHERNET ネットワークのオブジェクトが 3 つ入っている対等ビューです。以下は、ビューの内容です。

- コネクター OEMLAB
- コネクター NSL_ENET
- コネクター NSL_B202

この対等ビューで使用される名前は、オブジェクトの DisplayResourceName フィールドで判別されます。例えば、OEMLAB と表示されるオブジェクトの MyName 値は、 LATTVIEW.656_MAIN.CNTR3000.SL02P0 です。

この対等ビュー内の 3 つの各オブジェクトは、 DisplayResourceType オブジェクト DUIXC_RTN_LAN_ADAPTER にリンクしています。アイコン DUIU5N00 および台形状の端末シンボルは、DUIXC_RTN_LAN_ADAPTER へのリンクによって指定されます。サンプル・ネットワーク定義のこれらのオブジェクト間では、関係の定義は行われず、したがってビューでの関係の表示はありません。

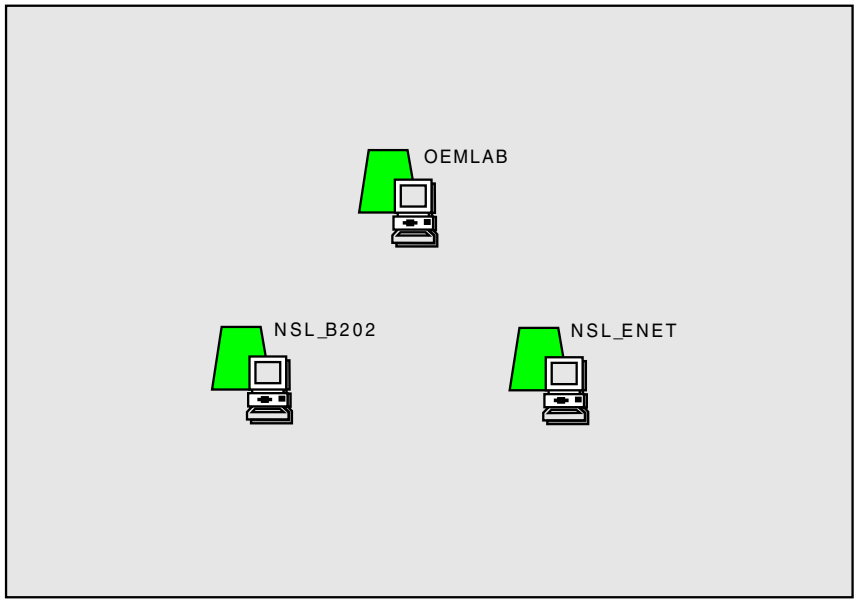


図 14. ETHERNET ネットワークの対等ビュー

より詳細なビュー

次のより詳細なビューは事前定義ビューです。これらは、オブジェクトを他のオブジェクトとの関係で示すのに使用します。

ビュー・タイプ
説明

論理 (Logical) オブジェクト間の論理関係に基づくネットワーク内にあるオブジェクトの次の低い層を表示します。

物理 (Physical)
オブジェクト間の物理関係に基づくネットワーク内にあるオブジェクトの次の低い層を表示します。

より詳細なビューは、動的に作成することもできます。詳細については、115 ページの『より詳細なビュー』を参照してください。

ネットワーク構成を RODM に定義する

SNA トポロジー・マネージャーは APPN およびサブエリア・ネットワークの定義に使用し、マルチシステム・マネージャーは RODM の非 SNA リソースの定義に使用することができます。次の説明のように、RODM の非 SNA リソースは手動で定義することもできます。

GMFHS でモニターしたいネットワークのリソースを識別したら、それらのリソースを RODM に定義します。リソースの定義は、すべて RODM ロード機能ステートメントと GMFHS データ・モデルの用語で行われます。定義のソースは、定義ステートメントが入っている 1 つまたは複数の RODM ロード・ファイルです。

このセクションでは、前のセクションで識別したオブジェクトのそれぞれを RODM に対して定義する方法を説明します。前述したオブジェクトのそれぞれのタイプごとに、そのタイプのオブジェクトの定義方法を説明し、そのオブジェクト・タイプの場合に定義する必要のあるフィールドを示し、さらに RODM ロード機能ステートメントを使用するサンプル・オブジェクトの定義を示します。 RODM ロード機能ステートメントの詳細については、
275 ページの『第 10 章 RODM ロード機能を使用する』を参照してください。

エディターを用いてネットワークを GMFHS に定義する際に必要な RODM ロード機能ステートメントを作成することも、あるいは独自の構成データベース形式から RODM ロード機能が必要とする形式に変換するプログラムを作成することもできます。

管理オブジェクトを定義する

管理オブジェクトには、ネットワーク管理ゲートウェイ、SNA ドメイン、および非 SNA ドメインが組み込まれています。ネットワーク管理ゲートウェイごとに、NMG_Class オブジェクトを 1 つ作成します。 SNA ドメインごとに、SNA_Domain_Class オブジェクトを 1 つ作成します。ドメインから送られるアラートに入っている特定の情報によって、非 SNA ドメインごとに 1 つまたは複数の Non_SNA_Domain_Class オブジェクトを作成します。

SNA ドメインを定義する

SNA リソースに含まれるサービス・ポイントにアクセスする構成で認識される SNA_Domain_Class オブジェクトをそれぞれの SNA ドメインに 1 つずつ定義します。このオブジェクトはビューに表示することができますが、 SNA_Domain_Class オブジェクトは GMFHS によって維持されません。

サンプル・ネットワークでは、SNA ドメイン B01NV が以下の RODM ロード機能ステートメントで定義されています。

```
-- Create SNA Domain Object for B01NV --
CREATE INVOKER  ::= 0000003;
       OBJCLASS ::= SNA_Domain_Class;
       OBJINST  ::= MyName = (CHARVAR) 'B01NV';

       ATTRLIST
       SNANet   ::= (CHARVAR) 'NETB';
END;
```

RODM の SNA_Domain_Class オブジェクトの名前は、 5 文字の NetView ドメイン ID です。

この例の場合、B01NV という SNA_Domain_Class オブジェクトは、 NETB という名前の SNA ネットワーク内にあります。オブジェクト名は OBJINST パラメーターで指定され、ネットワーク名はこのオブジェクトの CREATE ステートメントに関係する ATTRLIST パラメーターのフィールド SNANet で指定されます。定義する SNA ドメインが複数の場合でも、定義内の基本情報は各ドメインとも変わらず、ドメインが関連するオブジェクトと SNA ネットワークの名前を指定するだけで済みます。

ネットワーク管理ゲートウェイを定義する

ネットワークのネットワーク管理ゲートウェイごとに、ネットワーク管理ゲートウェイ・オブジェクトを作成します。

サンプル・ネットワークでは、ネットワーク管理ゲートウェイ B3088P2 が、以下の RODM ロード機能ステートメントで定義されています。

```
-- Create NMG Object for B3088P2 --
CREATE INVOKER  ::= 0000004;
       OBJCLASS ::= NMG_Class;
       OBJINST  ::= MyName = (CHARVAR) 'B3088P2';

       ATTRLIST
       Domain   ::= (OBJECTLINK)
       ('SNA_Domain_Class'.'B01NV'.'ContainsResource'),
       CommandRouteLUName ::= (CHARVAR) 'B01NV',
       NMGCharacteristics ::= (ANONYMOUSVAR) x'80',
       AgentStatusEffect      ::= (ANONYMOUSVAR) x'80',
       TransportProtocolName  ::= (CHARVAR) 'COS',
       WindowSize             ::= (INTEGER) 1;
END;
```

RODM ではネットワーク管理ゲートウェイの名前は、次のように判別されます。

- ゲートウェイで NetView プログラムの共通操作サービス (COS) 機能を使用してコマンドを受信する場合には、ネットワーク管理ゲートウェイ・オブジェクトの名前は、サービス・ポイントを含む SNA リソースに関連した PU 名または LU 名になります。
- ゲートウェイで PPI インターフェースを使用してコマンドの送達、およびコマンド応答とアラートの受信を行う場合には、ネットワーク管理ゲートウェイ・オブジェクト名は、コマンドの送信先のネットワーク管理アプリケーションに関連したプログラム間インターフェース受信側名になります。
- ゲートウェイが OST で実行中のコマンド処理プログラムおよびコマンド・プロシージャーを使用する場合には、ネットワーク管理ゲートウェイ・オブジェクト名は、このタイプのオブジェクトに固有の任意の名前にすることができます。

この例の場合、TransportProtocolName フィールドの値は COS です。これは、サービス・ポイント B3088P2 と NetView プログラム間でコマンドとアラートを転送する際に、共通操作サービス (COS) アーキテクチャーを用いた SSCP-PU もしくは LU-LU セッションを使用することを指定します。ウィンドウ・サイズは 1 で、NMG に対抗できるのは 1 コマンドのみであることを指定します。

CommandRouteLUName フィールドは B01NV に設定されて、ホスト A01MPU のコマンドが RMTCMD コマンドによってサービス・ポイント B3088P2 に経路指定されることを指定します。このことは、コマンドが、NetView-NetView セッション上をホスト B01MPU に常駐する NetView プログラムに最初に送られることを指定します。この NetView プログラムは、 RUNCMD コマンドをサービス・ポイント B3088P2 に送り、返しの応答をホスト A01MPU の NetView プログラムに経路指定します。

TransportProtocolName フィールドでは、GMFHS が、コマンドを送り、コマンドへの応答を受ける際にネットワーク管理ゲートウェイと通信する方法を指定します。以下は、このフィールドの有効値です。

- COS

- PPI
- OST
- NONE

非 SNA ドメインを定義する

Non_SNA_Domain_Class オブジェクトは、ネットワークのサービス・ポイント (SP)、トランザクション・プログラム (TP)、およびエレメント管理サブシステム (EMS) の個々独自の組み合わせごとに、1 つ定義します。次の組み合わせは、Non_SNA_Domain_Class のオブジェクトを個別に指定します。

- SP
- SP.TP
- SP.TP.EMS
- TP
- TP.EMS

前掲リストで、DOMS010 セッション・プロトコルに有効なのは先頭の 3 項目のみであることに注意してください。

Non_SNA_Domain_Class のオブジェクトの DisplayStatus フィールドの値は、GMFHS とドメインに関連するトランザクション・プログラム間のコマンドと応答の通信セッションの状況を表します。この値は、トランザクション・プログラムがドメインに関するアラート情報を GMFHS に送信できるかどうかを示すものではありません。アラート処理の詳細については、193 ページの『第 6 章 アラートおよび解決を処理および受信するための GMFHS のカスタマイズ』を参照してください。

サンプル・ネットワークでは、Non_SNA_Domain_Class オブジェクト DECNET は、次の RODM ロード機能ステートメントで定義されています。

```
-- Create Non_SNA Domain Object for DECNET --
CREATE INVOKER  ::= 0000003;
       OBJCLASS ::= Non_SNA_Domain_Class;
       OBJINST  ::= MyName = (CHARVAR) 'B3088P2.NAP.DECNET';

       ATTRLIST
       EMDomain ::= (CHARVAR) 'DECNET',
       DomainCharacteristics ::= (ANONYMOUSVAR) x'3672',
       InitialResourceStatus ::= (INTEGER) 129,
       PresentationProtocolName ::= (CHARVAR) 'DOMP020',
       SessionProtocolName ::= (CHARVAR) 'PASSTHRU',
       TransactionProgram ::= (CHARVAR) 'NAP',
       ReportsToAgent ::= (OBJECTLINK)
       ('NMG_Class'.'B3088P2'.'ReportsOnDomain'); END;
```

この例では、次のフィールド値が Non_SNA_Domain_Class のオブジェクトに指定されています。

- MyName フィールドは、ピリオドで分割された 3 つの名前から構成されています。
  - サービス・ポイントの名前 (B3088P2)
  - トランザクション・プログラムの名前 (NAP)
  - エレメント管理サブシステムの名前 (DECNET)

  この例では、エレメント管理サブシステムには、エレメント管理ドメイン名 DECNET が 1 つだけ指定されています。

- DomainCharacteristics フィールドは、次を指定します。
  - トランザクション・プログラム NAP は、ネイティブ・コマンド、状況表示コマンド、活動化コマンド、および非活動化コマンドをサポートする。
  - リソース名エレメントが、報告済みリソースのフルネームを作るときにピリオドで連結される。
  - トランザクション・プログラムは、コマンドの応答を戻す。
  - ドメインにある実オブジェクトのリソース状況を送信請求する処理が抑制される。
- InitialResourceStatus フィールドでは、実際のリソース状況が、アラートまたは応答でコマンドに報告されるまで、トランザクション・プログラム NAP で管理されるリソースに適合状況が報告されるように指定されます。
- PresentationProtocolName フィールドでは、DOMP020 を指定します。 DOMP020 プロトコルは、GMFHS が各総称コマンドをコマンド・ストリングに置き換えることを指定します。 GMFHS は、総称コマンドの宛先である GMFHS_Managed_Real_Objects_Class のオブジェクト、もしくは総称コマンドの宛先のドメインである Non_SNA_Domain_Class のオブジェクトからのコマンド・ストリングを使用します。例えば、GMFHS は、活動化総称コマンドが選ばれたとき、 ActivateCommandText フィールドの値を置き換えます。
- SessionProtocolName フィールドには、PASSTHRU という値があります。この値は、このドメインと関連付けられているトランザクション・プログラムとともにセッションが存在していると GMFHS が想定することを示します。
- ReportsToAgent フィールドでは、サービス・ポイント、およびこのサービス・ポイント (B3088P2) 用に定義された NMG_Class オブジェクトにドメインが関連するようになります。

このサンプルではドメインがどのビューでも表示されないため、ドメインに接続性は定義されません。

管理されるオブジェクトを定義する

管理されるオブジェクトには、SNA リソース、非 SNA 実リソース、および集合リソースが組み込まれています。 SNA トポロジー・マネージャーを使用して、 SNA オブジェクトを RODM にロードしたり、次に説明する処理を用いて、GMFHS_Shadow_Objects_Class オブジェクトを手動で定義することもできます。このセクションでは、これらのリソースを RODM に定義する方法を説明します。

注: NetView プログラムに送られたアラートによって、状況を変更したリソースが識別されるので、管理されるオブジェクトにはアラートによって与えられた名前と一致する名前を割り当てます。 GMFHS がアラートからのリソース名を使用する方法については、 193 ページの『第 6 章 アラートおよび解決を処理および受信するための GMFHS のカスタマイズ』を参照してください。

SNA リソースを定義する

RODM に定義したい SNA リソースごとに、 GMFHS_Shadow_Objects_Class のオブジェクトを 1 つ定義します。 SNA リソースの状況は RODM に保管されていませんが、以下の 1 つまたは複数の理由で、SNA リソースを RODM に定義することができます。

- SNA と非 SNA リソースとの間の関係を表示するため

- SNA リソースのアラート・ヒストリーを入手するため
- ユーザー状況を保留中の SNA アラートを入手するため

サンプル・ネットワークでは、SNA ホスト B01MPU のシャドー・オブジェクトが、以下の RODM ロード機能ステートメントで定義されています。

```
-- Create GMFHS Shadow Object for SNA Host B01MPU --
CREATE INVOKER  ::= 0000003;
       OBJCLASS ::= GMFHS_Shadow_Objects_Class;
       OBJINST  ::= MyName = (CHARVAR) 'NETB.B01MPU';
       ATTRLIST
       LocateName ::= (INDEXLIST)((CHARVAR) 'NETB.B01MPU'),
       DisplayResourceName ::= (CHARVAR) 'B01MPU';
END;
OP DUIFCLRT INVOKED_WITH (SELFDEFINING)
   ((CHARVAR)'LINK'
   (CHARVAR)'GMFHS_Shadow_Objects_Class.NETB.B01MPU'
   (CHARVAR)'Display_Resource_Type_Class.DUIXC_RTS_HOST');
```

シャドー・オブジェクトの名前は、SNA ネットワークの名前であり、 SNA オブジェクト、ピリオド (.)、リソースの SNA 名で構成されます。この例での名前は、NETB.B01MPU です。

この例で、ホスト B01MPU は B01MPU の DisplayResourceName をもちます。この名前は、リソースが入っているすべてのビューでリソースの隣に表示されています。シャドー・オブジェクトには DUIXC_RTS_HOST の DisplayResourceType が割り当てられ、これが SNA ホストであることを示します。

GMFHS_Shadow_Objects_Class オブジェクトの関係は、オブジェクトそのものを定義するときは定義しませんが、すべてのオブジェクトが定義された後に限り定義します。したがって、他のオブジェクトへのリンクについては、このセクションの後半で明らかにします。

非 SNA 実リソースを定義する

RODM に定義したい非 SNA 実リソースごとに、GMFHS_Managed_Real_Objects_Class のオブジェクトを定義します。このオブジェクトの名前は、リソースについて受け取ったアラートをリソースを表すオブジェクトに相関付けるのに使用します。

GMFHS_Managed_Real_Objects_Class に子クラスを追加した場合は、代わりにその子クラスのフィールドおよびオブジェクトを作成する必要があります。詳細については、「*IBM Tivoli NetView for z/OS データ・モデル・リファレンス*」を参照してください。

所定のレイアウト・アルゴリズムを使用して、定義中のオブジェクトを定義済みのネットワーク・ビュー、構成ビュー、またはより詳細なビューに表示するためには、このオブジェクトに対する Layout_Parameters_For_Object_Class のオブジェクトの定義が必要になる場合があります。 Layout_Parameters_For_Object_Class オブジェクトの定義については、 54 ページの『ネットワーク・ビュー、構成ビュー、およびより詳細なビューのレイアウト・パラメーターを定義する』で説明します。

サンプル・ネットワークでは、ミニコンピューター RALXT1 が DEC ネットワークに常駐する非 SNA 実リソースです。 RALXT1 は、次の RODM ロード機能ステートメントによって、 RODM に GMFHS_Managed_Real_Objects_Class オブジェクトとして定義されます。

```
-- Create a GMFHS Managed Real Object for RALXT1 --
CREATE INVOKER  ::= 0000003;
       OBJCLASS ::= GMFHS_Managed_Real_Objects_Class;
       OBJINST  ::= MyName = (CHARVAR) 'DECNET.RALV4.RALXT1';

       ATTRLIST
       DisplayResourceName ::= (CHARVAR) 'RALXT1',
       Domain   ::= (OBJECTLINK)
       ('Non_SNA_Domain_Class'.'B3088P2.NAP.DECNET'.'ContainsResource'),
       DisplayStatusCommandText ::= (CHARVAR)
'DECCMD/00,SHOW NODE RALXT1 SUMMARY';
END;
OP DUIFCLRT INVOKED_WITH (SELFDEFINING)
   ((CHARVAR)'LINK'
   (CHARVAR)'GMFHS_Managed_Real_Objects_Class.DECNET.RALV4.RALXT1'
   (CHARVAR)'Display_Resource_Type_Class.DUIXC_RTN_MINI');
```

GMFHS_Managed_Real_Objects_Class オブジェクトの名前は、実リソースについて着信するアラートを解決するのに使用します。名前は、実リソースが常駐する非 SNA ドメインを表す Non_SNA_Domain_Class オブジェクトの EMDomain フィールドで指定される文字ストリング、およびそのトランザクション・プログラムとエレメント管理システムに認識されるリソースの名前 (ピリオドで区切られる) から構成されます。

この例では、ミニコンピューター RALXT1 に、 Non_SNA_Domain_Class オブジェクト B3088P2.NAP.DECNET が関連付けられ、 DUIXC_RTN_MINI の DisplayResourceType が指定されています。 DisplayResourceName フィールドが指定されているので、このリソースと一緒にオペレーターに対してビューで表示されるときの名前は、RALXT1 です。

GMFHS_Managed_Real_Objects_Class のオブジェクトと Display_Resource_Type_Class のオブジェクトの間のリンクは、 DUIFCLRT メソッドを起動する RODM ロード機能プリミティブ・ステートメントによって作成されます。 RODM ロード機能プリミティブ・ステートメントについては、279 ページの『ロード機能プリミティブ・ステートメント』で説明します。 DUIFCLRT メソッドについては、557 ページの『DUIFCLRT: リソース・タイプ・リンク・メソッド』で説明します。

GMFHS 集合オブジェクトを定義する

集合オブジェクトは、リソースをモニターする目的で高水準リソースにグループ化するときに使用することができます。例外ビューを使用してリソースを直接モニターすることもできます。詳細については、118 ページの『例外ビューのオブジェクトおよび基準を定義する』を参照してください。

GMFHS_Aggregate_Objects_Class オブジェクトは、ビューに表示したい集合オブジェクトごとに 1 つずつ定義してください。 GMFHS_Aggregate_Objects_Class に子クラスを追加した場合は、代わりに子クラスのオブジェクトを作成してください。 GMFHS 集合オブジェクトを定義するときは、以下のようにします。

- 集合オブジェクトのエレメントの複合関係を指定する。
- 集合オブジェクトに属するリソースを指定する。

- 集約親と集約子との間の階層をセットアップする。

サンプル・ネットワークの場合、DEC ネットワークはサービス・ポイント B3088P2 により管理されます。 DEC ネットワークは、 23 ページの図 7 での図解のように、ホスト、 2 つのミニコンピューター、およびリンクから構成します。 DEC という名前の集合オブジェクトが定義され、DEC ネットワークを表します。 DEC 集合オブジェクトは高水準ビューに組み込まれ、その状況はそれが表すリソースの集合の状況を表します。ネットワーク DEC の GMFHS_Aggregate_Objects_Class オブジェクトは、以下の RODM ロード機能ステートメントによって定義されます。

```
-- Create a GMFHS Aggregate Object for DEC --
CREATE INVOKER  ::= 0000004;
       OBJCLASS ::= GMFHS_Aggregate_Objects_Class;
       OBJINST  ::= MyName = (CHARVAR) 'DEC';
       ATTRLIST
       ThresholdDegraded ::= (INTEGER) 1,
       ThresholdSeverelyDegraded ::= (INTEGER) 2,
       ThresholdUnsatisfactory ::= (INTEGER) 3,
       ComposedOfPhysical ::= (OBJECTLINKLIST)
('GMFHS_Managed_Real_Objects_Class'.'DECNET.RALV4'.'IsPartOf')
('GMFHS_Managed_Real_Objects_Class'.'DECNET.RALV4.RALXT1'.'IsPartOf')
('GMFHS_Managed_Real_Objects_Class'.'DECNET.RALV4.RALXT2'.'IsPartOf')
('GMFHS_Managed_Real_Objects_Class'.'DECNET.RALV4.TX02'.'IsPartOf')
('GMFHS_Managed_Real_Objects_Class'.'DECNET.RALV4.TX12'.'IsPartOf');
END;
OP DUIFCLRT INVOKED_WITH (SELFDEFINING)
   ((CHARVAR)'LINK'
   (CHARVAR)'GMFHS_Aggregate_Objects_Class.DEC'
   (CHARVAR)'Display_Resource_Type_Class.DUIXC_RTN_HOST_AGG');

OP DUIFCUAP INVOKED_WITH (SELFDEFINING)
   ((CHARVAR)'LINK'
   (CHARVAR)'GMFHS_Managed_Real_Objects_Class.DECNET.RALV4'
   (CHARVAR)'GMFHS_Aggregate_Objects_Class.DEC');

OP DUIFCUAP INVOKED_WITH (SELFDEFINING)
   ((CHARVAR)'LINK'
   (CHARVAR)'GMFHS_Managed_Real_Objects_Class.DECNET.RALV4.RALXT1'
   (CHARVAR)'GMFHS_Aggregate_Objects_Class.DEC');

OP DUIFCUAP INVOKED_WITH (SELFDEFINING)
   ((CHARVAR)'LINK'
   (CHARVAR)'GMFHS_Managed_Real_Objects_Class.DECNET.RALV4.RALXT2'
   (CHARVAR)'GMFHS_Aggregate_Objects_Class.DEC');

OP DUIFCUAP INVOKED_WITH (SELFDEFINING)
   ((CHARVAR)'LINK'
   (CHARVAR)'GMFHS_Managed_Real_Objects_Class.DECNET.RALV4.TX02'
   (CHARVAR)'GMFHS_Aggregate_Objects_Class.DEC');

OP DUIFCUAP INVOKED_WITH (SELFDEFINING)
   ((CHARVAR)'LINK'
   (CHARVAR)'GMFHS_Managed_Real_Objects_Class.DECNET.RALV4.TX12'
   (CHARVAR)'GMFHS_Aggregate_Objects_Class.DEC');
```

集合オブジェクトの定義は、ComposedOfPhysical と IsPartOf 関係、および AggregationParent と AggregationChild 関係の 2 組の関係をともないます。 ComposedOfPhysical と IsPartOf 関係によって、オペレーターが別のビューのオブジェクトを選び、より詳細を要求したときにビューで表示するオブジェクトを判別します。 AggregationParent と AggregationChild 関係によって、集合リソースの状況を計算するときに使用する実リソースを判別します。

この例では、DEC 集合オブジェクトの ComposedOfPhysical フィールドが、次の GMFHS_Managed_Real_Objects_Class オブジェクトの IsPartOf フィールドにリンクされています。

- DECNET.RALV4
- DECNET.RALV4.RALXT1
- DECNET.RALV4.RALXT2
- DECNET.RALV4.TX02
- DECNET.RALV4.TX12

この ComposedOfPhysical および IsPartOf の関係は、オペレーターがビュー内の DEC オブジェクトを選択してより詳細を要求する場合には、指定された GMFHS_Managed_Real_Objects_Class オブジェクトから構成されるビューを GMFHS が作成してワークステーションでこのビューを表示するように指定します。

DEC 集合オブジェクトには、 DUIXC_RTN_HOST_AGG の DisplayResourceType が割り当てられ、オブジェクトが非 SNA 集合体ホストを表すことを示します。 GMFHS_Aggregate_Objects_Class のオブジェクトと Display_Resource_Type_Class のオブジェクトの間のリンクは、 DUIFCLRT メソッドを起動する RODM ロード機能プリミティブ・ステートメントによって作成されます。 DUIFCLRT メソッドについては、 557 ページの『DUIFCLRT: リソース・タイプ・リンク・メソッド』で説明します。

DEC オブジェクトは、DEC ネットワークの基礎となる実リソースを表す集合体ホストです。 DUIFCUAP メソッドを使用し、RODM ロード機能ステートメントによって、この集合体親とその集合体子の間に AggregationParent および AggregationChild リンクが作成されます。 DUIFCUAP メソッドについては、 560 ページの『DUIFCUAP: 集約パス更新メソッド』で説明します。

一般に、ComposedOfPhysical と IsPartOf 関係、および AggregationParent と AggregationChild 関係は一緒に使用されますが、別々に使用することもできます。 例えば、実リソースを集合リソースのより詳細なビューで表示するが、集合リソースの状況に影響を与えないようにする場合は、集合オブジェクトの ComposedOfPhysical と IsPartOf 関係と実オブジェクトを対で定義して、AggregationParent と AggregationChild 関係を定義しないこともできます。

別の例では、GMFHS_Managed_Real_Objects_Class オブジェクトを他の GMFHS_Managed_Real_Objects_Class オブジェクトで構成するように定義することができます。この場合、ユーザーが最初のオブジェクトを選んで、より詳細を要求すると、最初のオブジェクトの一部として定義されたオブジェクトが表示されます。 最初のオブジェクトは集合オブジェクトではないため、この場合、AggregationParent と AggregationChild 関係は定義されません。

オブジェクト間で接続関係を定義する

オブジェクト間の接続関係で、ビューで表示するオブジェクトと、集合オブジェクトの状況に影響を与えるリソースを判別することができます。 シャドー・オブジェクトを含む関係を除き、 30 ページの『接続関係を識別する』で説明されたこれらの接続関係は、オブジェクトの定義時、もしくはオブジェクトの定義後の随時に定義することができます。シャドー・オブジェクトを含む接続関係を定義できるのは、

シャドー・オブジェクトが定義された後に限られます。このセクションでは、このような関係のいくつかについて、その定義方法をサンプル・ネットワークでの例を使用して説明します。

論理接続性を定義する

オブジェクトの論理リンクへの接続は、接続されるオブジェクトの LogicalConnPP フィールド、もしくは LogicalConnUpstream および LogicalConnDownstream フィールドを使用して行うことができます。サンプル・ネットワークでは、SNA ホスト B01MPU を表すシャドー・オブジェクトが、 SNA NCP B30A54C を表すシャドー・オブジェクトに論理的に接続され、以下の RODM ロード機能ステートメントを用いて、 36 ページの図 13 で図解された関係を作成しています。

```
-- Link Host B01MPU to NCP B30A54C --
OP 'GMFHS_Shadow_Objects_Class'.'NETB.B01MPU'.'LogicalConnPP'
IS_LINKED_TO
'GMFHS_Shadow_Objects_Class'.'NETB.B30A54C'.'LogicalConnPP';
```

リンクされるオブジェクトごとに、オブジェクトのクラス情報、オブジェクト名、および定義されるリンクのタイプを判別するフィールドを指定する必要があります。

物理接続性を定義する

物理リンクへのオブジェクトの接続は、接続されるオブジェクトの PhysicalConnPP フィールド、もしくは PhysicalConnUpstream および PhysicalConnDownstream フィールドを使用して行うことができます。サンプル・ネットワークでは、非 SNA ホスト RALV4 は、以下の RODM ロード機能ステートメントを用いてリンク TX-0-2 に物理的にリンクされています。

```
-- Link RALV4 to TX-0-2 --
OP 'GMFHS_Managed_Real_Objects_Class'.'DECNET.RALV4'.'PhysicalConnPP'
IS_LINKED_TO
'GMFHS_Managed_Real_Objects_Class'.'DECNET.RALV4.TX02'.'PhysicalConnPP';
```

リンクされるオブジェクトごとに、オブジェクトのクラス情報、オブジェクト名、および定義されるリンクのタイプを判別するフィールドを指定する必要があります。

親子関係を定義する

親と子のリンクは、リンクされるオブジェクトの ChildAccess および ParentAccess フィールドを用いて定義されます。サンプル・ネットワークでは、ミニコンピューター RALXT1 は、以下の RODM ロード機能ステートメントを用いて、 23 ページの図 7 で図解されている構成の DEC ホスト RALV4 にリンクされています。

```
-- Link RALV4 to RALXT1 --
OP 'GMFHS_Managed_Real_Objects_Class'.'DECNET.RALV4'.'ChildAccess'
IS_LINKED_TO
'GMFHS_Managed_Real_Objects_Class'.'DECNET.RALV4.RALXT1'.'ParentAccess';
```

リンクされるオブジェクトごとに、オブジェクトのクラス情報、オブジェクト名、およびオブジェクトが親か子かを判別するフィールドを指定する必要があります。

ビューを定義する

RODM では、以下の種類のビューを定義することができます。
- 例外
- ネットワーク
- 構成
- より詳細

ビュー・オブジェクトを定義する場合は、必ず RODM 高水準ロード機能ステートメントを使用してください。 RODM 高水準ロード機能ステートメントによって、オブジェクトの使用前にオブジェクトの全フィールドを定義することができます。RODM プリミティブ・ステートメントを使用すると、 GMFHS が、全情報が定義される前にビュー・オブジェクトに関する情報にアクセスしようとするため、予期しないエラーが発生する場合があります。高水準ロード機能ステートメントおよびプリミティブ・ステートメントの詳細については、 275 ページの『第 10 章 RODM ロード機能を使用する』を参照してください。

RODM で作成されるビューは、NetView 管理コンソールによって表示されます。以下のセクションでは、グラフィック機能によって使用されるパラメーターおよびレイアウト・アルゴリズムについて説明します。ビューの詳細については、 755 ページの『付録 B. ビュー・レイアウト機能』を参照してください。

例外ビューを定義する

例外ビューは、Exception_View_Class のオブジェクトによって表されます。表示したい例外ビューごとに、このクラスのオブジェクトを 1 つ作成します。 NetView 管理コンソールを使用して、事前定義ビューのすべてのリストを表示します。

サンプル・ネットワークには、例外ビューは含まれていません。しかし、サンプル DUIFDEXV には例外ビュー・オブジェクトの定義例が備えられており、またこのセクションの RODM ロード機能ステートメントを使用すれば例外ビューを定義することができます。 49 ページの図 15 は、DisplayStatus が極度に低下もしくは不良状況にある GMFHS_Displayable_Objects_Parent_Class のすべてのオブジェクトの例外ビューを示しています。

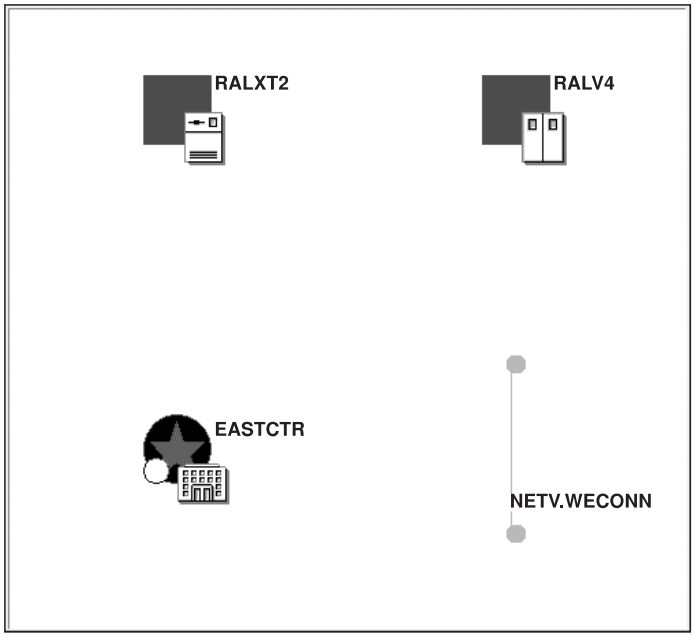


図 15. ネットワークの例外ビュー

例外ビュー EXCEPTIONVIEW1 は、次の RODM ロード機能ステートメントによって定義します。

```
CREATE INVOKER  ::= 0000001;
       OBJCLASS ::= Exception_View_Class;
       OBJINST  ::= MyName = (CHARVAR) 'EXCEPTIONVIEW1';
       ATTRLIST
       Annotation ::= (CHARVAR) 'Monitored by Operator A',
       ExceptionViewName ::= (CHARVAR) 'EXVIEW1',
END;
```

次のステートメントは、クラス GMFHS_Displayable_Objects_Parent_Class のすべてのオブジェクトが EXCEPTIONVIEW1 内にあると定義するときに使用します。ExceptionViewList フィールドは、クラス・レベルで定義する必要はありません。ExceptionViewList フィールドは、オブジェクト・レベルで定義することもできます。

```
OP 'GMFHS_Displayable_Objects_Parent_Class'..
   'ExceptionViewList'
HAS_VALUE (INDEXLIST)((CHARVAR) 'EXVIEW1');
```

オブジェクトを例外ビューに定義する際の詳細については、 118 ページの『例外ビューのオブジェクトおよび基準を定義する』を参照してください。

ネットワーク・ビューを定義する

ネットワーク・ビューは、 Network_View_Class のオブジェクトによって表されます。表示したいネットワーク・ビューごとに、このクラスにオブジェクトを 1 つ作成します。 NetView 管理コンソールは、すべての事前定義ビューのリストを表示します。

50 ページの図 16 は、サンプル・ネットワークの DEC ネットワーク・コンポーネントのネットワーク・ビューです。オブジェクトごとに表示されるアイコンおよびシンボルは、リンク先の DisplayResourceType オブジェクトによって判別されま

す。例えば、リソース DECNET.RALV4.RALXT1 は DUIXC_RTN_MINI にリンクしています。アイコン DUIU2N00 および四角形のホスト・シンボルは、DUIXC_RTN_MINI によって指定されます。ビューで表示される名前 RALXT1 は、オブジェクト DECNET.RALV4.RALXT1 の DisplayResourceName フィールドによって指定されます。

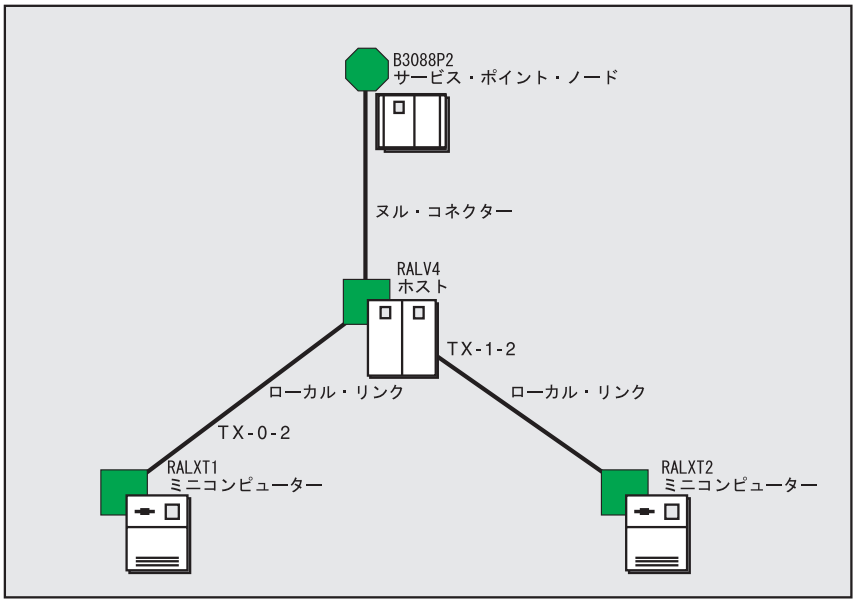


図 16. DEC ネットワークのネットワーク・ビュー

DEC ネットワークのネットワーク・ビューは、以下の RODM ロード機能ステートメントで定義されます。

```
-- Create Network View for DECNET --
CREATE INVOKER   ::= 0000004;
       OBJCLASS ::= Network_View_Class;
       OBJINST  ::= MyName = (CHARVAR) 'DECNET';
       ATTRLIST
       Annotation ::= (CHARVAR) 'DEC NETWORK',
       LayoutType ::= (INTEGER) 8,
       ConnType ::= (ANONYMOUSVAR) x'80',
       ContainsObjects ::= (OBJECTLINKLIST)
('GMFHS_Managed_Real_Objects_Class'.'DECNET.RALV4.RALXT1'.'ContainedInView')
('GMFHS_Managed_Real_Objects_Class'.'DECNET.RALV4.RALXT2'.'ContainedInView')
('GMFHS_Managed_Real_Objects_Class'.'DECNET.RALV4.TX02'.'ContainedInView')
('GMFHS_Managed_Real_Objects_Class'.'DECNET.RALV4.TX12'.'ContainedInView')
('GMFHS_Managed_Real_Objects_Class'.'DECNET.RALV4'.'ContainedInView');
('GMFHS_Shadow_Objects_Class'.'NETB.B3088P2'.'ContainedInView')
END;
```

この例では、DECNET という Network_View_Class オブジェクトが定義されて、DEC ネットワークのネットワーク・ビューを表しています。オブジェクトの Annotation フィールドには、値 DEC NETWORK が割り当てられ、ビューによりワークステーションで表示されます。 LayoutType フィールドには値 8 が割り当てられ、ビューが接続ツリー・レイアウトで表示されることを指定しています。 ConnType フィールドには値 80 が割り当てられ、このタイプのビューには、ノードからリンクの接続に加えて、ノードからノードへの接続が有効であることを指定していま

す。 DECNET オブジェクトの ContainsObjects フィールドは、管理される実オブジェクト (DEC ネットワークを構成する実リソースを表す) の ContainedInView フィールドにリンクしています。

構成ビューを定義する

構成ビューは、次のいずれかのクラスにビューを表すオブジェクトを定義して、作成します。

ビュー・タイプ	定義されるクラス
対等 (Peer)	Configuration_Peer_View_Class
物理 (Physical)	Configuration_Physical_Connectivity_View_Class
論理 (Logical)	Configuration_Logical_Connectivity_View_Class
バックボーン (Backbone)	Configuration_Backbone_View_Class

表示したい構成ビューごとに、そのそれぞれのクラスにオブジェクトを 1 つ作成します。サンプル・ネットワークには構成対等機能ビューが入っているので、Configuration_Peer_View_Class オブジェクトの定義例が続きます。他の構成ビュー・タイプ・クラスのどれにオブジェクトを定義するときも、類似のプロシージャーを使用します。次の構成ビューは、動的に作成されたビューである可能性もあります。

- バックボーン (Backbone)
- 論理 (Logical)
- 物理 (Physical)

構成ビューの詳細については、111 ページの『特定ビューの場合のオブジェクトの展開処理の説明』を参照してください。

対等ビューを定義する: 52 ページの図 17 は、トークンリング LAN コンポーネントの対等ビューです。対等ビューは、Configuration_Peer_View_Class のオブジェクトにより表示されます。表示したい対等ビューごとに、このクラスのオブジェクトを 1 つ作成します。

52 ページの図 17 は、サンプル・ネットワークのトークンリング LAN コンポーネントの対等ビューです。オブジェクトごとに表示されるアイコンおよびシンボルは、リンク先の DisplayResourceType オブジェクトによって判別されます。例えば、集合リソース BRIDGE01 は DUIXC_RTN_BRIDGE_AGG にリンクしています。アイコン DUIU4N02 および六角形のノード・シンボルは、DUIXC_RTN_BRIDGE_AGG によって指定されます。 BRIDGE01 は集合リソースであるため、ノード・シンボルにはより小さい集合体シンボルも入っています。ビューで表示される名前 BRIDGE01 は、オブジェクト BRIDGE01 の DisplayResourceName フィールドによって指定されます。

サンプル・ネットワークでは、 BRIDGE01 の DisplayResourceName 値ももつ LANMGR.BRIDGE01 という名前の実オブジェクトを定義していることに留意してください。このビューの BRIDGE01 は、GMFHS_Aggregate_Objects_Class のオブジ

エクトです。

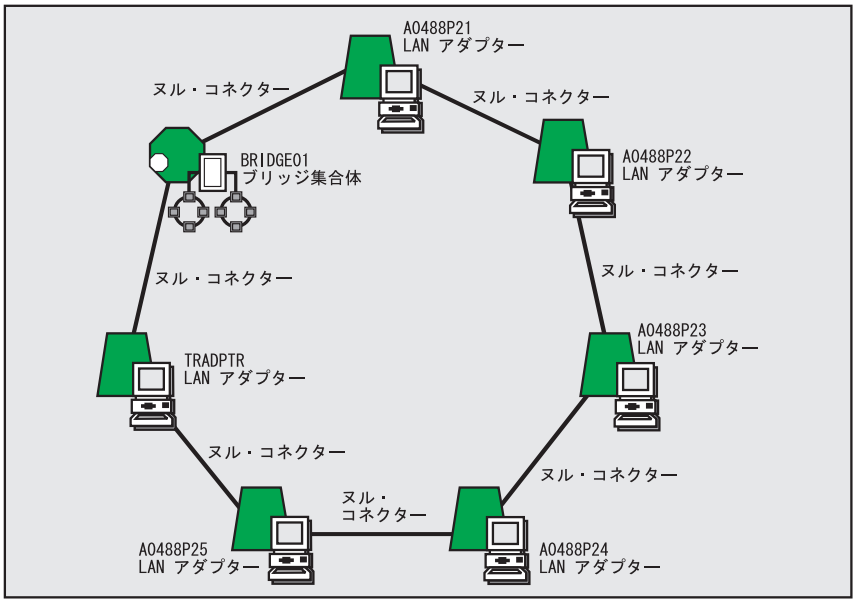


図 17. トークンリング・ネットワーク TRLANNET の対等ビュー

トークンリング LAN ネットワークの構成対等機能ビューは、以下の RODM ロード機能ステートメントで定義されます。

```
-- Create Configuration Peer View TRLANNET --
CREATE INVOKER   ::= 0000004;
       OBJCLASS ::= Configuration_Peer_View_Class;
       OBJINST  ::= MyName = (CHARVAR) 'TRLANNET_Peer';
       ATTRLIST
       Annotation ::= (CHARVAR) 'Token Ring Network',
LayoutType ::= (INTEGER) 4,
       ConnType ::= (ANONYMOUSVAR) x'80',
       FirstNode  ::= (OBJECTLINK)
('GMFHS_Managed_Real_Objects_Class'.'LANMGR.10005AC35CA0'.'IsFirstNode'),
       SecondNode ::= (OBJECTLINK)
('GMFHS_Managed_Real_Objects_Class'.'LANMGR.10005A95E7CC'.'IsSecondNode'),
       ContainsObjects ::= (OBJECTLINKLIST)
('GMFHS_Managed_Real_Objects_Class'.'LANMGR.10005AC35CA0'.'ContainedInView')
('GMFHS_Managed_Real_Objects_Class'.'LANMGR.10005A95E7CC'.'ContainedInView')
('GMFHS_Managed_Real_Objects_Class'.'LANMGR.10005A89A267'.'ContainedInView')
('GMFHS_Managed_Real_Objects_Class'.'LANMGR.10005A966BAB'.'ContainedInView')
('GMFHS_Managed_Real_Objects_Class'.'LANMGR.10005A95A08C'.'ContainedInView')
('GMFHS_Managed_Real_Objects_Class'.'LANMGR.400076041088'.'ContainedInView')
('GMFHS_Aggregate_Objects_Class'.'BRIDGE01'.'ContainedInView');
END;
```

この例では、TRLANNET_Peer という名前の Configuration_Peer_View_Class オブジェクトが定義されて、トークンリング LAN ネットワークの構成対等機能ビューを表しています。オブジェクトの Annotation フィールドには、値 `Token Ring Network` が割り当てられます。LayoutType フィールドには値 4 が割り当てられ、トークンリング・ネットワークの放射状レイアウトを指定します。 ConnType フィールドには、前のネットワーク・ビューの例のように、値 80 が割り当てられます。

ビューを作成する際に、ビューに表示されるオブジェクトのオブジェクト名を指定します。サンプル・ネットワークは、トークンリング・ネットワークのリソースごとに表示する名前を指定するときは DisplayResourceName フィールドを使用するので、この例の RODM ロード機能ステートメントのオブジェクト名は、 52 ページの図 17 に示されている名前とは異なります。例えば、オブジェクト LANMGR.10005AC35CA0 にはその DisplayResourceName フィールドがあって、A0488P21 に設定されています。

TRLANNET_Peer オブジェクトの FirstNode フィールドは、構成対等機能ビューのリングの最上部に表示されるオブジェクトの IsFirstNode フィールドにリンクされます。 SecondNode フィールドは、ビューの最初のノードの右に表示されるオブジェクトにリンクします。 ContainsObjects フィールドは、ビューに表示される残りのオブジェクトにリンクします。これらのオブジェクトは、定義される順序でビューに表示されます。

より詳細なビューの定義

より詳細なビューを作成するには、以下のクラスのいずれかでビューを表すオブジェクトを定義します。

ビュー・タイプ
定義されるクラス

物理 (Physical)
More Detail_Physical_View_Class

論理 (Logical) More Detail_Logical_View_Class

表示したいより詳細なビューごとに、それぞれのクラスに 1 つのオブジェクトを作成します。これらのビューも動的に作成することができます。

サンプル・ネットワークには、定義済みより詳細なビューは含まれていません。より詳細なビューに関する詳細については、 115 ページの『より詳細なビュー』を参照してください。

レイアウト・パラメーターの定義

レイアウト・パラメーターは、以下のタイプのビューに指定することができます。

- ネットワーク
- 構成
- より詳細
- 例外

例外ビューのレイアウト・パラメーターを定義する

グリッド・レイアウトは、例外ビューと併用できる唯一のレイアウト・アルゴリズムであり、グリッド・レイアウト・アルゴリズムに定義できる唯一のビュー・パラメーターはレイアウト幅です。グリッド・レイアウト・アルゴリズムについては、755 ページの『付録 B. ビュー・レイアウト機能』を参照してください。

ネットワーク・ビュー、構成ビュー、およびより詳細なビューのレイアウト・パラメーターを定義する

ネットワーク・ビュー、構成ビュー、またはより詳細なビューを定義する場合は、レイアウト・アルゴリズムを指定することができます。これを行うには、ビューを表すために定義するビュー・オブジェクトの LayoutType フィールドに値を指定します。ビュー・オブジェクトは以下のクラスに定義することができます。

- Network_View_Class
- Configuration_Peer_View_Class
- Configuration_Backbone_View_Class
- Configuration_Logical_Connectivity_View_Class
- Configuration_Physical_Connectivity_View_Class
- More_Detail_Logical_View_Class
- More_Detail_Physical_View_Class

レイアウト・アルゴリズムを指定しない場合は、リンク・タイプのレイアウト・アルゴリズムによるデフォルトの放射状表示が使用されます。

使用するレイアウト・アルゴリズムの種類の選び方、および各レイアウト・アルゴリズムの利点、欠点については、 755 ページの『付録 B. ビュー・レイアウト機能』を参照してください。

ある種のレイアウト・アルゴリズムでは、追加情報を提供して、ビューの正しいレイアウトに役立てることが必要です。この情報は、ビュー・オブジェクト自体のフィールドで指定することもあります。例えば、LinkCrossOptionValue フィールドは、放射状レイアウト・アルゴリズムが交差したリンクを解くのにかかる作業量を指定します。別の例の場合、FirstNode フィールド および SecondNode フィールドは、トークンリングの放射状レイアウト・アルゴリズムで、リングの最上部に置かれるノードを指定し、最上部ノードの右に置かれるノードを指定します。

Layout_Parameters_For_Object_Class オブジェクトのフィールドには、追加情報も指定することができます。これらのオブジェクトは、ビューとビューで表示するオブジェクトをリンクします。これらのオブジェクトは、オブジェクトを特定のレイアウト・アルゴリズムによって特定のビューにレイアウトするときに用いられるパラメーターを指定します。 1 つの Layout_Parameters_For_Object_Class オブジェクトを同じレイアウト・パラメーターをもつすべてのオブジェクトにリンクすることができます。

例としては、この Layout_Parameters_For_Object_Class オブジェクトにリンクするリソースが、接続ツリー・レイアウトが使用されるときの接続ツリーのルート・ノードであることを指定する RootNode フィールドや、この Layout_Parameters_For_Object_Class オブジェクトにリンクするオブジェクトが、一連のオブジェクトで表示される場合の所定のレイアウト・アルゴリズムについて指定する LayoutSequence フィールドがあります。

55 ページの表 3 に、以下のクラスのオブジェクトに指定することができるフィールドをリストします。

- Network_View_Class
- Configuration_Peer_View_Class
- Configuration_Backbone_View_Class
- Configuration_Logical_Connectivity_View_Class

- Configuration_Physical_Connectivity_View_Class
- More_Detail_Logical_View_Class
- More_Detail_Physical_View_Class

これらのフィールドは、使用するレイアウト・アルゴリズムによって、オプションになることも、必須になることも、適用不能になることもあります。 表 3 は、オプション (***O***) フィールドと必須 (***R***) フィールドを示しています。 ***N/A*** は、そのタイプのレイアウト・アルゴリズムにはパラメーターが適用不能であることを示します。

表 3. レイアウト・アルゴリズムおよびビュー・パラメーター

レイアウト・アルゴリズム	リンク・クロス・オプション値	ビン・パッキング・フラグ	バス・ノード	最初のノード	2 番目のノード	レイアウトの方向付け	デフォルトの行スペース	楕円の縦横比 (幅 / 高さ)	レイアウトの幅
クラスター ID ごとの放射状	O	O	N/A	N/A	N/A	N/A	N/A	N/A	N/A
リンク・タイプごとの放射状	O	O	N/A	N/A	N/A	N/A	N/A	N/A	N/A
ローカル・エリア・ネットワーク	O	O	N/A	N/A	N/A	N/A	N/A	N/A	N/A
トークンリング・ネットワーク	N/A	N/A	N/A	R	R	N/A	N/A	N/A	N/A
中央パスを使った LAN	N/A	N/A	R	N/A	N/A	N/A	N/A	N/A	N/A
近接した階層	N/A	N/A	N/A	N/A	N/A	O	O	N/A	N/A
単一の楕円	N/A	N/A	N/A	N/A	N/A	N/A	N/A	O	N/A
接続ツリー	N/A	N/A	N/A	N/A	N/A	O	O	N/A	N/A
グリッド	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	O

レイアウト・パラメーターおよびレイアウト・アルゴリズムの詳細については、755 ページの『付録 B. ビュー・レイアウト機能』を参照してください。

レイアウト・パラメーター: 表 4 は、Layout_Parameters_For_Object_Class オブジェクトで指定できるレイアウト・パラメーターのリストで、レイアウト・パラメーターがオプション (***O***) か必須 (***R***) であるレイアウト・アルゴリズムのタイプを示します。 ***N/A*** は、そのタイプのレイアウト・アルゴリズムにはパラメーターが適用不能であることを示します。これらのレイアウト・パラメーターおよびレイアウト・アルゴリズムの詳細については、 755 ページの『付録 B. ビュー・レイアウト機能』を参照してください。

表 4. レイアウト・アルゴリズムおよびレイアウト・パラメーター

レイアウト・アルゴリズム	リソースをレイアウトするときの文字	レイアウトの順序	階層優先順位	ルート・ノード	クラスター ID 値
クラスター ID ごとの放射状	N/A	N/A	N/A	N/A	R
リンク・タイプごとの放射状	O	N/A	N/A	N/A	N/A

表 4. レイアウト・アルゴリズムおよびレイアウト・パラメーター (続き)

レイアウト・アルゴリズム	リソースをレイアウトするときの文字	レイアウトの順序	階層優先順位	ルート・ノード	クラスター ID 値
ローカル・エリア・ネットワーク	N/A	O	N/A	N/A	N/A
トークンリング・ネットワーク	N/A	O	N/A	N/A	N/A
中央パスを使った LAN	N/A	O	N/A	N/A	N/A
近接した階層	N/A	N/A	R	N/A	N/A
単一の楕円	N/A	O	N/A	N/A	N/A
接続ツリー	N/A	O	N/A	R	N/A
グリッド	N/A	O	O	N/A	N/A

サンプル・ネットワークでは、 36 ページの図 13 で図解されているように、Layout_Parameters_For_Object_Class オブジェクト LPTRLAN に、集合オブジェクト TRLAN をネットワーク・ビュー SAMPNET で表示する方法を指定するパラメーターが入っています。以下は、LPTRLAN オブジェクトを定義する RODM ロード機能ステートメントです。

```
-- Create Layout Parameters for Object TRLAN --
CREATE INVOKER   ::= 0000004;
       OBJCLASS  ::= Layout_Parameters_For_Object_Class;
       OBJINST  ::= MyName = (CHARVAR) 'LPTRLAN';
       ATTRLIST
       Object ::= (OBJECTLINK)
('GMFHS_Aggregate_Objects_Class'.'TRLAN'.'LayoutParmList'),
       View ::= (OBJECTLINKLIST)
('Network_View_Class'.'SAMPNET'.'LayoutParmList'),
       HierarchicalPriority ::= (INTEGER) 4;
END;
```

Object フィールドは、レイアウト・パラメーターを適用するオブジェクトを指定し、 View フィールドは、レイアウト・パラメーターを適用するビューを指定します。 HierarchicalPriority フィールドは、 TRLAN オブジェクトがネットワーク・ビューの階層レイアウトの 4 行目に表示されることを指定します。

50 ページの図 16 で図解されているように、 Layout_Parameters_For_Object_Class オブジェクト LPB3088P2P には、シャドー・オブジェクト NETB.B3088P2 をネットワーク・ビュー DECNET で表示する方法を指定するパラメーターが入っています。以下は、LPB3088P2P レイアウト・パラメーター・オブジェクトを定義する RODM ロード機能ステートメントです。

```
-- Create Layout Parameters for Object B3088P2 --
CREATE INVOKER   ::= 0000004;
       OBJCLASS  ::= Layout_Parameters_For_Object_Class;
       OBJINST  ::= MyName = (CHARVAR) 'LPB3088P2P';
       ATTRLIST
       Object ::= (OBJECTLINK)
('GMFHS_Shadow_Objects_Class'.'NETB.B3088P2'.'LayoutParmList'),
       View ::= (OBJECTLINKLIST)
```

```
('Network_View_Class'.'DECNET'.'LayoutParmList'),
       LayoutSequence ::= (INTEGER) 0,
       RootNode ::= (ANONYMOUSVAR) X'80';
END;
```

前の例のように、Object および View フィールドは、これらのパラメーターが関連するオブジェクトおよびビューを指定します。 LayoutSequence フィールドには値 0 が割り当てられ、ノードがビューで順不同でレイアウトされることを指定しています。 RootNode フィールドは、シャドー・オブジェクト NET.B3088P2 が接続性ツリーのルート・ノードのように表示されることを指定します。

動的に作成されたより詳細なビューにレイアウト・パラメーターを定義する

あらゆるタイプのより詳細なビューを動的に作成することができます。より詳細なビューを明示的に定義しなくても、より詳細なビューのレイアウトを指定することができます。より詳細なビューは、 NetView 管理コンソール オペレーターがコンテキスト・メニューから「**More Detail**」を選ぶと作成されます。 GMFHS は、RODM で定義されたオブジェクトに以下のより詳細なビューを作成しようとします。

- より詳細な論理ビューには、選択されたオブジェクトの ComposedOfLogical フィールドで指定されたすべてのオブジェクトが含まれます。
- より詳細な物理ビューには、選択されたオブジェクトの ComposedOfPhysical フィールドで指定されたすべてのオブジェクトが含まれます。
- 構成子 II ビューには、構成子 II ビューに関して、View_Information_Object_Class オブジェクトの RelFieldNamesA フィールドによって指定されたオブジェクトのすべてが含まれます。
- 構成子 III ビューには、構成子 III ビューに関して、View_Information_Object_Class オブジェクトの RelFieldNamesA フィールドによって指定されたオブジェクトのすべてが含まれます。

ComposedOfLogical フィールドもしくは ComposedOfPhysical フィールドの値がヌルならば、対応するビューは作成されません。より詳細なビューの表示については、「*IBM Tivoli NetView for z/OS NetView 管理コンソール ユーザーズ・ガイド*」の『ビューの概説』を参照してください。

レイアウト・パラメーターは、選択したオブジェクトから作成されるより詳細なビューごとに指定することができます。より詳細なビューのレイアウト・パラメーターを指定するには、以下のステップを完了します。 59 ページの図 18 は、作成するオブジェクト（**A**、**B**、および **C**）ならびにリンク（**1** および **2**）を示しています。

1. より詳細なビュー・レイアウト・パラメーターの定義を行いたいオブジェクトを選ぶ。定義するのは、このオブジェクトが他のビューで選ばれたときに作成されるより詳細なビューのレイアウト・パラメーターです。

   この例では、サンプル・ネットワークの集合オブジェクト TRLAN（**A**）を選びます。
2. 定義中のレイアウト・パラメーターに対応するより詳細なビュー (より詳細な論理ビューまたはより詳細な物理ビュー) を選択する。

TRLAN オブジェクトがもつ ComposedOfLogical と ComposedOfPhysical の値は両方とも有効であるため、より詳細なビューは 2 つ作成されます。この例では、より詳細な物理ビューについてのレイアウト・パラメーターの定義を選びます。

3. ビューを表すための Layout_Parameters_For_View_Class のオブジェクトを作成する。

   ヒント: Layout_Parameters_For_View_Class オブジェクトは、Network_View_Class オブジェクトに類似しています。

   以下は、この例のオブジェクト (**B**) を作成する RODM ロード機能ステートメントの一部です。サンプル・メンバー DUIFSNET には、全ステートメントが入っています。

```
CREATE INVOKER   ::= 0000004;
       OBJCLASS ::= Layout_Parameters_For_View_Class;
       OBJINST  ::= MyName = (CHARVAR)
                    'View_Layout_Parms_For_TRLAN_More_Detail_Physical';
```

4. ステップ 3 で作成したオブジェクトの SelectedResource フィールドを、ステップ 1 (57 ページ) で選んだオブジェクトの DetailViewLayoutForSelectedResource フィールドにリンクする。

   以下は、このリンク (59 ページの図 18 で **1** と示されている) を作成する RODM ロード機能ステートメントの一部です。

```
SelectedResource ::= (OBJECTLINKLIST) ('GMFHS_Aggregate_Objects_Class'.
'TRLAN'.'DetailViewLayoutForSelectedResource'),
```

5. この Layout_Parameters_For_View_Class オブジェクト (**B**) が表す詳細なビューのタイプを指定する。ビュー・タイプの指定は、このオブジェクトの ViewClass フィールドを、ビュー・タイプを表す View_Information_Reference_Class のオブジェクト (**C**) の DetailViewLayout フィールドにリンクして行います。
   - More_Detail_Logical_View_Reference
   - More_Detail_Physical_View_Reference
   - Configuration_Children_II_View_Reference
   - Configuration_Children_III_View_Reference

   以下は、リンク (より詳細な物理ビューを指定し、 59 ページの図 18 で **2** と示されている) を作成する RODM ロード機能ステートメントの一部です。

```
ViewClass ::= (OBJECTLINKLIST) ('View_Information_Reference_Class'.
'More_Detail_Physical_View_Reference'.'DetailViewLayout'),
```

6. 定義するビューのレイアウト・パラメーターを指定する。
   Layout_Parameters_For_View_Class オブジェクト (**B**) の他のフィールドでは、レイアウト・アルゴリズムおよび他のビュー・パラメーターが指定されます。 55 ページの表 3 は、レイアウト・アルゴリズムごとの必須レイアウト・パラメーターのリストです。

   この例では、レイアウト・アルゴリズムとしてトークンリング・アルゴリズムの放射状レイアウトが選ばれています。 55 ページの表 3 は、このレイアウトに必須の FirstNode フィールドと SecondNode フィールドを示しています。以下は、レイアウト・アルゴリズム、および FirstNode と SecondNode フィールドを指定する RODM ロード機能ステートメントの一部です。

```
LayoutType ::= (INTEGER) 4,
FirstNode  ::= (OBJECTLINK) ('GMFHS_Managed_Real_Objects_Class'.
     'LANMGR.10005AC35CA0'.'IsFirstNode'),
SecondNode ::= (OBJECTLINK) ('GMFHS_Managed_Real_Objects_Class'.
     'LANMGR.10005A95E7CC'.'IsSecondNode');
```

7. 追加のオブジェクトもしくはビューに、この同じ Layout_Parameters_For_View_Class オブジェクトを使用したい場合は、追加のリンクを作成する。 リンク・フィールドは、すべて複数の値を受け入れます。
8. より詳細なビューの個々のオブジェクトのレイアウトを制御する必要がある場合は、オブジェクトのレイアウト・パラメーターを定義する。レイアウト・アルゴリズムによっては、オブジェクトのレイアウト・パラメーターを必要とするものもあります。 55 ページの表 4 に必須パラメーターのリストがあります。

   レイアウト・パラメーターを定義する際の手順は、『より詳細なビューにオブジェクト用のレイアウト・パラメーターを定義する』を参照してください。

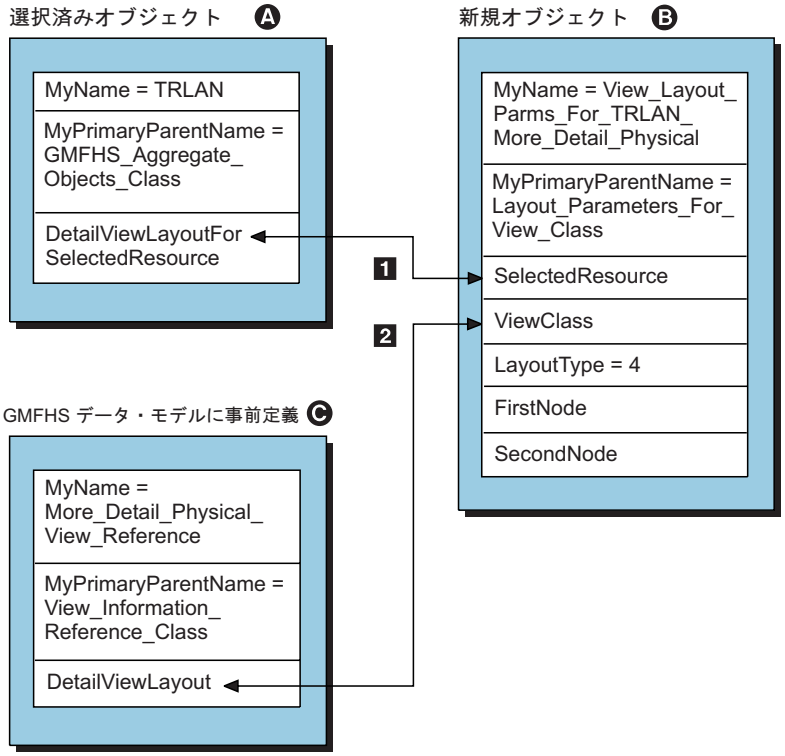


図 18. より詳細なビューのためのレイアウト・パラメーターを定義する

より詳細なビューにオブジェクト用のレイアウト・パラメーターを定義する:

注: より詳細なビューに表示される個々のオブジェクトのレイアウト・パラメーターを定義することもできます。これらのレイアウト・パラメーターは、Layout_Parameters_For_Object_Class オブジェクトで定義します。リンクは、レイアウト・パラメーターを適用するオブジェクトとビューを指定します。より詳細なビューのレイアウト・パラメーターを指定するには、以下のステップを完了します。 62 ページの図 19 は、作成するオブジェクトとリンクを示しています。

1. レイアウト・パラメーターの定義を行いたい、より詳細なビューのオブジェクトを識別する。オブジェクトの指定は、より詳細なビューで表示するように、ステップ 1 (57 ページ) で指定した元のオブジェクトの、ComposedOfLogical、ComposedOfPhysical、もしくは RelFieldNamesA フィールドによって行わなければなりません。

   この例では、GMFHS_Managed_Real_Objects_Class のオブジェクト (**E**) LANMGR.10005A89A267 のレイアウト・パラメーターを定義します。

2. Layout_Parameters_For_Object_Class のオブジェクトを作成して、特定のビュー内にあるときにオブジェクトのレイアウト・パラメーターを表す。

   以下は、 62 ページの図 19 に示されているこのオブジェクト (**D**) を作成するときの RODM ロード機能ステートメント (DUIFSNET 内ではない) の一部です。

   ```
   CREATE INVOKER   ::= 0000004;
          OBJCLASS  ::= Layout_Parameters_For_Object_Class;
          OBJINST   ::= Detail_Layout_LANMGR.10005A89A267;
   ```

3. ステップ 2 で作成した Layout_Parameters_For_Object_Class オブジェクトの Object フィールドを、表示されたオブジェクトの DetailLayoutParmList フィールドにリンクする。

   この例では、Detail_Layout_LANMGR.10005A89A267 オブジェクト (**D**) の Object フィールドをオブジェクト (**E**) LANMGR.10005A89A267 の DetailLayoutParmList フィールドにリンクします。以下は、このリンク (62 ページの図 19 で **3** と示されている) を作成する RODM ロード機能ステートメントの一部です。

   ```
   Object ::= (OBJECTLINKLIST) ('GMFHS_Managed_Real_Objects_Class'.
   'Detail_Layout_LANMGR.10005A89A267'.'DetailLayoutParmList'),
   ```

4. これらのレイアウト・パラメーターが適用されるビューを指定する。

   a. Layout_Parameters_For_Object_Class オブジェクトの SelectedResource フィールドを、このより詳細なビュー (ステップ 1 (57 ページ) で選択されたオブジェクト) を生成する目的で選択されるオブジェクトの DetailLayoutParmListForSelectedResource フィールドにリンクする。

      この例では、オブジェクト (**D**) Detail_Layout_LANMGR.10005A89A267 の SelectedResource フィールドをオブジェクト (**A**) TRLAN の DetailLayoutParmListForSelectedResource フィールドにリンクします。以下は、このリンク (62 ページの図 19 で **4** と示されている) を作成する RODM ロード機能ステートメントの一部です。

      ```
      SelectedResource ::= (OBJECTLINKLIST)
      ('GMFHS_Aggregate_Objects_Class'.
      'TRLAN'.'DetailLayoutParmListForSelectedResource'),
      ```

   b. これらのレイアウト・パラメーターが適合する、より詳細なビューのタイプを指定する。ビュー・タイプの指定は、このオブジェクト (**D**) の ViewClass フィールドを、ビュー・タイプを表す View_Information_Reference_Class のオブジェクト (**C**) の DetailLayoutParmList フィールドにリンクして行います。

      - More_Detail_Logical_View_Reference
      - More_Detail_Physical_View_Reference

- Configuration_Children_II_View_Reference
- Configuration_Children_III_View_Reference

以下は、リンク (より詳細な物理ビューを指定し、 62 ページの図 19 で 5 と示されている) を作成する RODM ロード機能ステートメントの一部です。

```
ViewClass ::= (OBJECTLINKLIST)
('View_Information_Reference_Class'.
'More_Detail_Physical_View_Reference'.
'DetailLayoutParmList'),
```

5. オブジェクトのレイアウト・パラメーターを指定する。 55 ページの表 4 は、レイアウト・アルゴリズムごとのオプションおよび必須レイアウト・パラメーターのリストです。

   この例では、トークンリング・アルゴリズムの放射状レイアウトが使用されています。 55 ページの表 4 は、LayoutSequence フィールドが指定できる、唯一の任意指定パラメーターであることを示しています。このオブジェクト (D) の LayoutSequence フィールドには値 3 を指定します。以下は、LayoutSequence フィールドの値を設定する RODM ロード機能ステートメントの一部です。

   ```
   LayoutSequence ::= (INTEGER) 3;
   ```

6. 追加のオブジェクトもしくはビューに、この同じ Layout_Parameters_For_Object_Class オブジェクトを使用したい場合は、追加のリンクを作成する。リンク・フィールドは、すべて複数の値を受け入れます。

   例えば、この同じオブジェクトが OTHER_AGG という名前の GMFHS_Aggregate_Objects_Class のオブジェクトが選ばれたとき (OTHER_AGG はサンプル・ネットワークの一部ではない) に生成されたより詳細な物理ビュー内にあるときは、それを使用してオブジェクト LANMGR.10005A89A267 のレイアウト・パラメーターを定義します。オブジェクト Detail_Layout_LANMGR.10005A89A267 の SelectedResource フィールドからオブジェクト OTHER_AGG の DetailLayoutParmListForSelectedResource フィールドへのリンクを作成します。以下は、このリンクを作成する RODM ロード機能プリミティブ・ステートメントです。

   ```
   OP 'Layout_Parameters_For_Object_Class'.
      'Detail_Layout_LANMGR.10005A89A267'.'SelectedResource'
   IS_LINKED_TO 'GMFHS_Aggregate_Objects_Class'.'TRLAN'.
      'DetailLayoutParmListForSelectedResource';
   ```

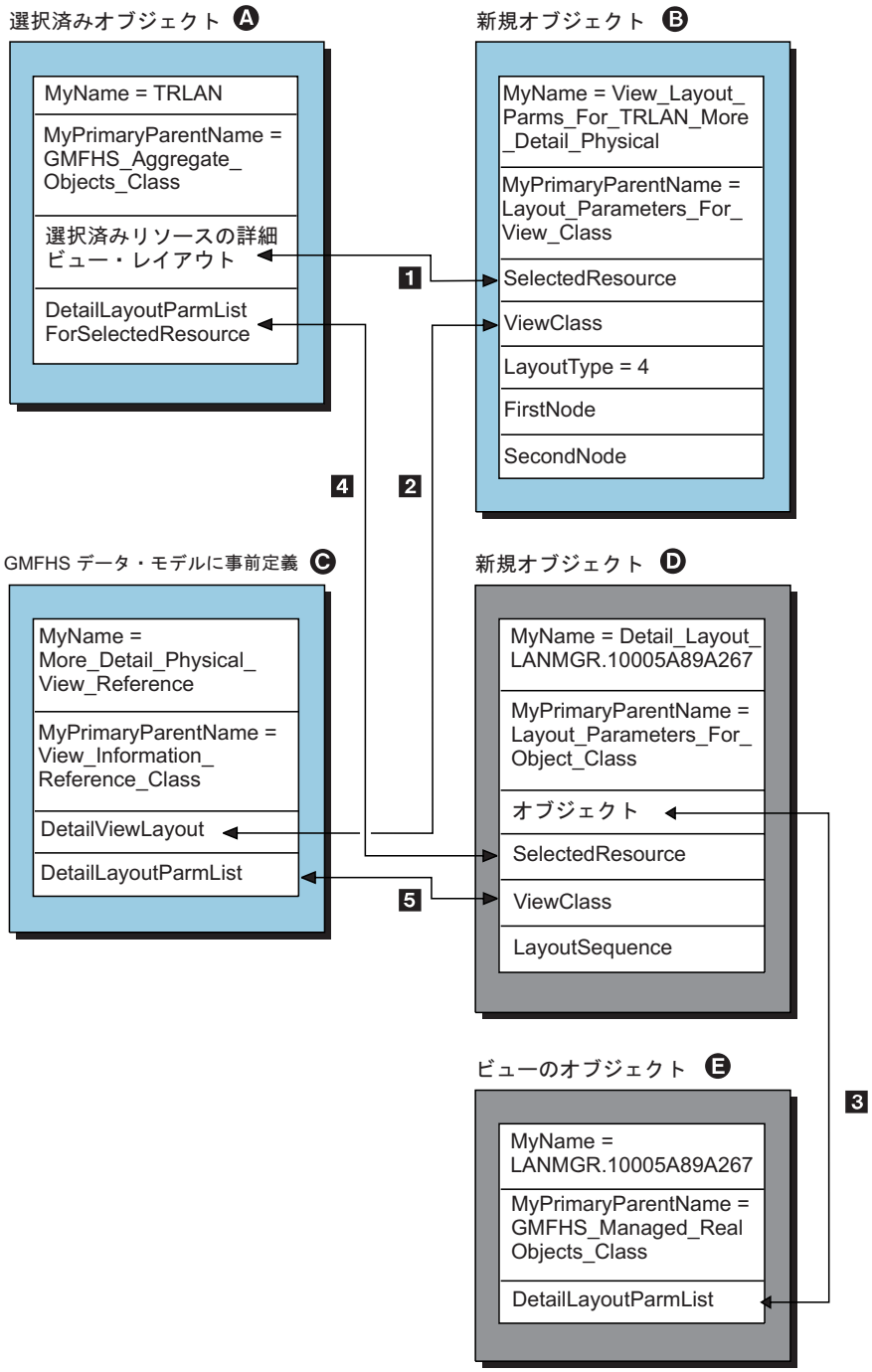


図 19. より詳細なビューのオブジェクトにレイアウト・パラメーターを定義する

まとめ

構成とネットワークを示すオブジェクトを定義したら、 RODM ロード機能を使って RODM にロードします。この方法については、 65 ページの『第 3 章 GMFHS データ・モデルをロードする』を参照してください。

クラスのオブジェクトの定義をロードする前に、そのクラスの定義をロードしておく必要があります。オブジェクトを実際にリンクする前に、リンクするオブジェクトを同じトークンによって定義する必要があります。従う順序の例として、サンプル・メンバー DUIFSNET のロード機能ステートメントを使用します。サンプル・ネットワークのオブジェクトとリンクは、以下の順序でロードするように配置されています。

1. SNA_Domain_Class オブジェクト
2. GMFHS_Shadow_Objects_Class オブジェクト
3. NMG_Class オブジェクト
4. Non_SNA_Domain_Class オブジェクト
5. GMFHS_Managed_Real_Objects_Class オブジェクト
6. GMFHS_Aggregate_Objects_Class オブジェクト
7. オブジェクト間のリンク
   - 論理リンク
   - 物理リンク
   - 親/子リンク
8. Exception_View_Class オブジェクト
9. Network_View_Class オブジェクト
10. Configuration_Peer_View_Class オブジェクト
11. Layout_Parameters_For_View_Class オブジェクト
12. Layout_Parameters_For_Object_Class オブジェクト

注: サンプル・ロード・ファイル DUIFSNET で定義されたサンプル・ネットワークには、例外ビューは含まれていませんが、ロードすべき位置の前のリスト内に含まれています。

個々のネットワークを定義するときは、事前にサンプル・ロード・ファイル DUIFSNET のネットワークを十分に調べてください。 RODM ロード機能の構文については、275 ページの『第 10 章 RODM ロード機能を使用する』を参照してください。

第 3 章 GMFHS データ・モデルをロードする

この章では、GMFHS および SNA トポロジー・マネージャーのデータ・モデル、ネットワーク定義、およびメソッドを RODM にロードする方法について説明します。また GMFHS がアクティブのときに、オブジェクトを追加、変更、または削除する方法についても説明します。

GMFHS クラス構造は、RODM ロード機能入力ファイルの DUIFSTRC に含まれており、 NetView プログラムと一緒に出荷されます。

SNA トポロジー・マネージャーのクラス構造は、 NetView プログラムとともに出荷される RODM ロード機能入力ファイル FLBTRDM*x* で提供されています。FLBTRDM*x* ロード機能入力ファイルに関する詳細については、「*IBM Tivoli NetView for z/OS インストール: グラフィカル・コンポーネントの構成*」を参照してください。

DUIFSTRC およびすべての FLBTRDMx 入力ファイルは、サンプル CNMSJH12 を用いてロードされます。 DUIFSTRC およびすべての FLBTRDMx 入力ファイルは、 GMFHS 操作用にロードされなければなりません。入力ファイル DUIFSTRC をロードしてから FLBTRDM*x* 入力ファイルをロードしなければならないことに注意してください。これはサンプル CNMSJH12 で指定された順番であり、変更してはなりません。

データ・モデルおよびネットワーク定義をロードする

RODM が実行中のときは、サンプル CNMSJH12 を用いて、 GMFHS データ・モデルおよびネットワーク定義をロードします。

1. RODM ステートメントを作成して、非 SNA ネットワークを定義する。ネットワークを RODM に定義する方法については、 19 ページの『第 2 章 ネットワークを GMFHS に定義する』を参照してください。
2. サンプル・ジョブ CNMSJH12 を以下の手順で更新する。
   a. JOB ステートメントを変更して、インストールの会計情報を指定する。
   b. ステップ 1 で作成された RODM ロード・ファイル名を、サンプルの最後の行の EKGIN1 DD ステートメントに入力する。例えば、オブジェクト定義がデータ・セット NETVIEW.V5R3M0.MYDEFS(OBJECTS) にある場合、CNMSJH12 の最後の行は次のようになります。

      ```
      //          DD DSN=NETVIEW.V5R3M0.MYDEFS(OBJECTS),DISP=SHR
      ```

   c. RODMNAME を、EXEC ステートメントの RODM の名前に置き換える。
3. RODM が実行されていることを確認する。
4. CNMSJH12 を開始する。
5. GMFHS を開始する。

GMFHS が実行中にネットワーク定義を変更する

非 SNA オブジェクトを RODM データ・キャッシュで追加、変更、削除する際に GMFHS が実行されている場合は、 GMFHS CONFIG コマンドが必要になることがあります。 GMFHS CONFIG コマンドは、それらに応答する際に必要な変更の有効範囲と処理のタイプを、GMFHS に対して識別します。

SNA トポロジー・マネージャーによって管理されるサブエリア・リソースは、GMFHS CONFIG コマンドを使用せずに随時変更することができます。

注:

1. NMG およびドメインは、GMFHS CONFIG コマンドを使用せずに動的に追加することができます。詳細については、 69 ページの『GMFHS がアクティブのときに NMG およびドメインを追加する』を参照してください。
2. GMFHS がアクティブのときに RODM に保管された GMFHS を変更すると、適切な GMFHS CONFIG コマンドが出されて完了するまでは、予期しない結果を招く場合があります。

GMFHS CONFIG コマンドの 3 つのタイプとは、 DOMAIN、NETWORK および VIEW です。以降のセクションで、変更するフィールドとクラスに基づいて発行する GMFHS CONFIG コマンドを列挙します。

DOMAIN
変更に GMFHS_Managed_Real_Objects_Class オブジェクトと Non_SNA_Domain_Class オブジェクトの関連付けの変更が含まれているが、GMFHS CONFIG NETWORK コマンドの使用を必要とする変更が含まれていないときに使用します。 CONFIG DOMAIN コマンドの動作の詳細については、 NetView のオンライン・ヘルプを参照してください。

NETWORK
行われる変更に、NMG とドメインの特性および構造を記述する情報に対する変更が含まれるときに限り使用します。

VIEW マイグレーションの目的だけに残されているものであり、必要ありません。

GMFHS CONFIG コマンドにも LOAD パラメーターがあります。 CONFIG VIEW にデフォルトの LOAD=NO が指定されると、操作は実行されません。 CONFIG DOMAIN および NETWORK の場合に、デフォルトの LOAD=NO が指定されると、 RODM ロード機能の呼び出しを除くすべてのコマンド処理が完了します。例えば、ジョブ通知、または GMFHS 以外のなんらかの RODM アプリケーションによる RODM ロード機能の実行によって、キャッシュの内容が変更される場合は、LOAD=NO を指定した GMFHS CONFIG コマンドを使用します。こうすると、変更に必要な GMFHS 内の処理が完了します。

LOAD=YES を指定すると、RODM ロード機能がコマンド処理の一貫で実行されます。 INDD=*ddname* ならば、データ・セットもしくは *ddname* で識別されるセットが入力として RODM ロード機能に渡されます。 INDD パラメーターが指定されない場合のデフォルトは、EKGIN3 です。

注: GMFHS CONFIG コマンドは、注意して使用してください。このコマンドを使用すると、1 つまたは複数の非 SNA ドメインのもとにある RODM オブジェクトによっては再初期設定される場合があります。このため、定義される実オ

ブジェクトの数によっては、かなり CPU を使用することになります。その CPU 使用率は、GMFHS を最初に開始したときに似た使用率になる場合もあります。

GMFHS CONFIG コマンドに関する詳細については、 NetView のオンライン・ヘルプを参照してください。

必須の GMFHS CONFIG コマンドを選択する

下記の表は、RODM キャッシュのオブジェクトがそのフィールド値を変更するときに必要な GMFHS CONFIG コマンドを表しています。使用する必要がある CONFIG コマンドを判別するには、以下の該当する最初の規則を使用します。

- オブジェクト・フィールドの変更に CONFIG NETWORK コマンドが必要な場合には、 CONFIG NETWORK コマンドを使用する。
- オブジェクト・フィールドの変更に CONFIG DOMAIN コマンドが必要な場合には、 CONFIG DOMAIN コマンドを使用する。
- 最後に、フィールドがリストされていなければ、追加または削除を行うオブジェクトにも、あるいは変更を加えるオブジェクト・フィールド値にも、 CONFIG コマンドは必要ありません。しかし、RODM ロード機能ジョブの完了後に RODM CHKPT コマンドを発行します。これによって、RODM キャッシュの新しいチェックポイント・イメージが書き込まれ、必要に応じてキャッシュのリカバリーに使用できます。

オブジェクト自体の追加または削除用に、別の表は用意されていません。なぜならば、SNA ドメイン・オブジェクトの場合を除き、新しいオブジェクトは、別のオブジェクトにリンクするまでは効力をもたず、またオブジェクトは、その他のオブジェクトへのリンクがすべて削除されるまでは削除できないためです。オブジェクト・リンクの確立と削除は、データ・タイプが OBJECTLINK または OBJECTLINKLIST のフィールドのフィールド値を変更して行われます。これらのタイプのフィールドの変更については、表に記載しています。

Non_SNA_Domain_Class の変更

表 5 は、Non_SNA_Domain_Class のオブジェクトのフィールドを変更する際に使用する GMFHS CONFIG コマンドを示しています。

表 5. Non_SNA_Domain_Class オブジェクトの GMFHS CONFIG コマンド

フィールド	GMFHS CONFIG コマンド
AlertProc	NETWORK
CommandTimeoutInterval	NETWORK
ContainsResource	NETWORK, DOMAIN (注を参照)
DomainCharacteristics	NETWORK
DomainCharacteristics2	NETWORK
EMDomain	NETWORK
InitialResourceStatus	NETWORK
PresentationProtocolName	NETWORK
ReportsToAgent	NETWORK
SessionProtocolName	NETWORK

表 5. Non_SNA_Domain_Class オブジェクトの GMFHS CONFIG コマンド (続き)

フィールド	GMFHS CONFIG コマンド
TransactionProgram	NETWORK
WindowSize	NETWORK

注: Non_SNA_Domain_Class オブジェクトの ContainsResource フィールドは、ドメインに属する GMFHS 管理の実リソースか GMFHS-NMG オブジェクトのどちらかを指定することができます。非 SNA ドメイン・オブジェクトのリソース (Resources) フィールドが、GMFHS-NMG オブジェクトのドメイン (Domain) フィールドにリンクしている場合は、CONFIG NETWORK コマンドを使用します。非 SNA ドメイン・オブジェクトとの間でリンクもしくはリンク解除が行われるのが GMFHS 管理の実リソースのみならば、 CONFIG DOMAIN コマンドを使用することができます。 CONFIG DOMAIN コマンドを使用する前に、 NetView オンライン・ヘルプでこのコマンドの詳しい説明を参照してください。

SNA_Domain_Class の変更

表 6 は、 SNA_Domain_Class のオブジェクトのフィールドを変更する際に使用する GMFHS CONFIG コマンドを示しています。 SNA_Domain_Class のオブジェクトを作成もしくは削除するときは、 GMFHS CONFIG NETWORK コマンドを出します。

表 6. SNA_Domain_Class オブジェクトの GMFHS CONFIG コマンド

フィールド	GMFHS CONFIG コマンド
ContainsResource	NETWORK
SNANet	NETWORK

NMG_Class の変更

表 7 は、 NMG_Class のオブジェクトのフィールドを変更するときに使用する GMFHS CONFIG コマンドを示しています。

表 7. NMG_Class オブジェクトの GMFHS CONFIG コマンド

フィールド	GMFHS CONFIG コマンド
AgentStatusEffect	NETWORK
CommandRouteLUName	NETWORK
Domain	NETWORK
NMGCharacteristics	NETWORK
ReportsOnDomain	NETWORK
TransportProtocolName	NETWORK
WindowSize	NETWORK

GMFHS_Managed_Real_Objects_Class の変更

表 8 は、 GMFHS_Managed_Real_Objects_Class のオブジェクトのフィールドを変更するときに使用する GMFHS CONFIG コマンドを示しています。

表 8. GMFHS_Managed_Real_Objects_Class オブジェクトの GMFHS CONFIG コマンド

フィールド	GMFHS CONFIG コマンド
Domain	DOMAIN (注を参照)

表 8. GMFHS_Managed_Real_Objects_Class オブジェクトの GMFHS CONFIG コマンド (続き)

フィールド	GMFHS CONFIG コマンド

注: 非 SNA ドメイン・オブジェクトとの間でリンクもしくはリンク解除が行われるのが GMFHS 管理の実リソースのみならば、 CONFIG DOMAIN コマンドを使用することができます。 CONFIG DOMAIN コマンドを使用する前に、 NetView オンライン・ヘルプでこのコマンドの詳しい説明を参照してください。

GMFHS がアクティブのときに NMG およびドメインを追加する

NMG および非 SNA ドメインは、GMFHS CONFIG コマンドを使用せずに、GMFHS の実行中に RODM に追加することができます。 RODM にオブジェクトを定義するときは、次のガイドラインに従ってください。

- NMG または非 SNA ドメインを動的に追加したいことを示す、適切なビットを設定する。
- GMFHS に、非 SNA ドメインのリソースに初期状況もしくは未確認状況を適用してほしくないことを示すときは、 DomainCharacteristics フィールドに適切なビットを設定する。

  注: これを適用するのは、GMFHS が最初に NMG または非 SNA ドメインを処理する場合のみです。 GMFHS は、通常後続のすべての処理に初期状況および未確認状況を適用します。
- GMFHS に非 SNA ドメインのリソース状況を請求させたくないときは、DomainCharacteristics フィールドに該当するビットを設定する。
- NMG およびドメインが RODM に定義された後、NMG を非 SNA ドメインにリンクする。 GMFHS は、このリンクをシグナルとして使用し、新しい NMG またはドメインの処理を開始します。

第 4 章 ネットワーク管理ゲートウェイと通信する

この章では、GMFHS がネットワーク管理ゲートウェイ (NMG) と通信する方法について説明します。 NMG は、非 SNA ネットワークに関する状況情報を GMFHS に送ります。 GMFHS は、非 SNA ネットワークのコマンドを NMG に送ります。

非 SNA リソースは、GMFHS の非 SNA ドメインに関連しています。非 SNA ドメインを GMFHS に定義するときは、各非 SNA ドメインとその関連リソースを所有する NMG を指定します。また、GMFHS が NMG と通信する方法も指定します。

NMG が実行するワークステーションのクロックは、 GMFHS が実行するホストのクロックと同期している必要があります。 DOMP010 表示プロトコルは、これらのクロックと同期化します。他の表示プロトコルの場合は、それぞれのルーチンを作成してクロックを同期化してください。 NMG が遠隔操作サービスをインストールした OS/2® オペレーティング・システムで実行されている場合は、 NetView から RUNCMD を出して、 ROP サービスを用いてワークステーション・クロックを設定してください。

ROP サービスの使用については、「*サービス・ポイント・アプリケーション・ルーターとリモート・オペレーション・サービスの手引き*」を参照してください。クロックが同期化されていない場合には、GMFHS がアラートを正しく処理しないことがあります。

この章は、以下の GMFHS フィールドの正しい値を選ぶ際に役立ちます。

- PresentationProtocolName
- SessionProtocolName
- TransportProtocolName

この章は、DomainCharacteristics フィールドの一部のビットに正しい値を選ぶ際にも役立ちます。

この章を使用して、GMFHS が NMG に期待する内容を理解することもできます。独自のサービス・ポイントもしくは NMG を作成する場合は、この情報が必要です。

最後に、この章では、NETCENTER プロトコルと GMFHS プロトコル間の相違について説明します。 NETCENTER からのマイグレーションを行う場合は、この章を用いて既存の NMG を GMFHS と併用する方法を理解してください。

72 ページの表 9 は、代表的な NMG の 3 つの GMFHS プロトコル・フィールドの値を示しています。

表 9. 代表的な NMG の GMFHS プロトコルの値

NMG 名	Presentation ProtocolName	Session ProtocolName	Transport ProtocolName
LAN ネットワーク・マネージャー	DOMP020	PASSTHRU	COS
NAP	DOMP010	DOMS010	COS
NetView OST[1]	DOMP020	PASSTHRU	OST
NetView OST	PASSTHRU	PASSTHRU	OST
NetView/PC	DOMP010	DOMS010	COS
NetView PPI	NONE	NONE	PPI
NetView/6000 V1	DOMP020	PASSTHRU	COS
NetView/6000 V2	DOMP010	DOMS010[2]	COS
NetView for AIX V3	DOMP010	DOMS010[2]	COS
NetView for AIX V4	DOMP010	DOMS010[2]	COS
オープン・トポロジー・インターフェース・エージェント[3]	DOMP010	NONE	COS
PPI	DOMP020	PASSTHRU	PPI

注:

[1] パラメーター置換を使用したい場合は、 DOMP020 表示プロトコルを使用してください。

[2] 詳細については、 90 ページの『NetView/6000 V2、NetView for AIX V3、NetView for AIX V4、および DOMS010 のセッションの確立』を参照してください。

[3] IBM Tivoli NetView for z/OS オープン・トポロジー・インターフェース・エージェント。

この表に掲載されたプロトコル・パラメーターの値は代表的なものであることを忘れないでください。パラメーター値はほかの組み合わせも可能で、使用する値は NMG がサポートする内容で決まります。

非 SNA 表示プロトコルを定義する

表示プロトコルは、エレメント管理システムが使用する構文との間でコマンドの変換を行います。 変換は、コマンドのターゲットであるリソースに関連するドメインの規則に従って行われます。

Non_SNA_Domain_Class オブジェクトの PresentationProtocolName フィールドは、非 SNA ドメインに使用されるプロトコルを指定します。以下は、有効なプロトコル名です。

- DOMP010
- DOMP020
- PASSTHRU
- NONE

DOMP010 表示プロトコル

DOMP010 プロトコルを使用すると、総称コマンドはドメインに関連するゲートウェイへの送達用に変換できるようになり、 DOMP010 プロトコルを用いて形式化されたコマンドへの応答を DisplayStatus に変換できるようになります。 DisplayStatus は、ビューのオブジェクトの形に反映します。ネイティブ・コマンドおよびリソース特有のコマンドも、ドメインに関連するネイティブ・エレメント・マネージャーまたはトランザクション・プログラムによってサポートされる DOMP010 プロトコルを用いて送ることができます。

DOMP010 表示プロトコルは、コマンド・メッセージおよび NMG からのコマンド応答メッセージが、 76 ページの『DOMP010 の形式化の規則』に説明されている規則に従って形式化されることを指定します。

DOMP010 プロトコルは、次のタイプのコマンドの変換を行います。

- 総称コマンド:
  - Activate (活動化)
  - Display Abnormal Status (異常状況表示)
  - Display Status (状況表示)
  - Inactivate (非活動化)
  - Reconfigure (再構成)
  - Recycle (再生)
- セッション・プロトコル・コマンド
- ネイティブ・コマンド・テキストおよびリソース特有のコマンド・テキスト

DOMP010 プロトコルも、任意のコマンドのネイティブ・エレメント・マネージャーからのコマンド応答の変換を行います。

ネイティブ・コマンドの場合、DOMP010 は、オペレーターが入力したコマンドでパラメーター置換を行います。 GMHFS は、コマンドのトークンを以下のように置き換えます。

トークン
: **GMFHS がとるアクション**

%APPL%
: Non_SNA_Domain_Class オブジェクトの TransactionProgram フィールドの値に置き換える。

%DOMAIN%
: Non_SNA_Domain_Class オブジェクトの EMDomain フィールドの値に置き換える。

%RESOURCE%
: リソースの MyName フィールドの値に置き換える。

%SPNAME%
: NMG_Class オブジェクトの MyName フィールドの値に置き換える。

%TYPE%
: リソースに関連する Display_Resource_Type_Class オブジェクトの TypeName フィールドの値に置き換える。

GMFHS は、ネイティブ OST テキストの以下のパラメーターを受け入れます。
- %RESPONSE%
- %NORESPONSE%

%RESPONSE% パラメーターによって、すべての有効なコマンド応答が強制的にワークステーションに戻されます。 %RESPONSE% パラメーターは、Non_SNA_Domain_Class DomainCharacteristics フィールドの Response Expected ビットをオーバーライドします。 %NORESPONSE% パラメーターによって、OST コンソールでネイティブ・コマンドが強制的に出され、ワークステーションに応答は戻されません。

DOMP010 プロトコルは、NETCENTER NSI1 表示プロトコルに似ていますが、DOMP010 プロトコルは若干の拡張機能を備えています。これらの拡張機能を使いたくなければ、 Non_SNA_Domain_Class オブジェクトの DomainCharacteristics フィールドのビット 13 をオンに設定します。 GMFHS は、NETCENTER 総称 Enable (使用可能) コマンドおよび Disable (使用不能) コマンドをサポートしていません。GMFHS および NETCENTER 間の相違の詳細説明については、 100 ページの『NETCENTER プロトコルから GMFHS プロトコルにマイグレーションする』を参照してください。

DOMP010 表示プロトコルは、COS およびプログラム間インターフェース NMG にのみ適用できます。

DOMP020 表示プロトコル

DOMP020 プロトコルを使用すると、総称コマンドは、ドメインに関連する NMG への送達用に変換できるようになります。 DOMP020 プロトコルは、ネイティブ・コマンド・テキストおよびリソース特有のコマンド・テキストをサポートします。これらのコマンドへの応答は、元のワークステーションのコマンド応答ウィンドウに未変更のまま戻されます。 GMFHS は、これらの応答からは状況情報を引き出しません。

総称コマンドのテキストは、RODM から取り出されます。 GMFHS は、コマンドのターゲットを表す GMFHS_Managed_Real_Objects_Class オブジェクトからのコマンド・テキストを要求します。このオブジェクトがコマンド・テキストを定義しなければ、GMFHS が、コマンドのターゲットのドメインを表す Non_SNA_Domain_Class オブジェクトからのコマンド・テキストを要求します。Display Abnormal Status (異常状況表示) 総称コマンドおよび Reconfigure (再構成) 総称コマンドが有効なのは、コマンドのターゲットが Non_SNA_Domain_Class のオブジェクトである場合に限られます。総称コマンドに使用されるフィールドは以下のとおりです。

総称コマンド	**GMFHS フィールド**
Activate (活動化)	ActivateCommandText
Deactivate (非活動化)	DeactivateCommandText
Display Abnormal Status (異常状況表示)	DisplayAbnormalStatusCommandText
Display Status (状況表示)	DisplayStatusCommandText
Reconfigure (再構成)	ReconfigureCommandText
Recycle (再生)	RecycleCommandText

GMFHS は、コマンドを探す際にパラメーター置換を行います。 GMFHS は、コマンドで次のいずれかのトークンを見つけ、それらを以下のように置き換えます。

トークン **GMFHS がとるアクション**

%APPL% Non_SNA_Domain_Class オブジェクトの TransactionProgram フィールドの値に置き換える。

%DOMAIN%
Non_SNA_Domain_Class オブジェクトの EMDomain フィールドの値に置き換える。

%RESOURCE%
リソースの MyName フィールドの値に置き換える。

%SPNAME%
NMG_Class オブジェクトの MyName フィールドの値に置き換える。

%TYPE%
リソースに関連する Display_Resource_Type_Class オブジェクトの TypeName フィールドの値に置き換える。

注: 異常状況表示コマンドおよび再構成コマンドが関係するのはドメインのみで、したがって、コマンド・テキストで探索されるのはドメイン・オブジェクトのみです。

DOMP020 プロトコルは、すべての NMG タイプで使用されます。コマンドが GMFHS からである場合は、ゲートウェイによって、ワークステーション・オペレーターに関連する OST もしくは中央側の NetView 基本プログラム・オペレーター・インターフェース・タスク (PPT) にコマンドを送ることができます。コマンドについて実行するコマンド・プロシージャーもしくはコマンド処理プログラムは、直接または間接にアラートを生成する場合があります。アラートは、結果としてのリソース状況を報告します。

PASSTHRU 表示プロトコル

PASSTHRU プロトコルは、ワークステーション・オペレーターによって入力されたネイティブ・ネットワークのコマンド・テキストがネイティブ・エレメント管理システムに直接未変更で渡り、かつネイティブ・ネットワークのコマンド応答テキストが GMFHS の介入なしにワークステーション・オペレーターに戻ることを指定します。

PASSTHRU 表示プロトコルは、コマンドの実際のテキストが RODM から取り出されることを指定します。 PASSTHRU と DOMP020 の相違点は、PASSTHRU が総称コマンドをサポートせず、パラメーター置換を行わないことです。

NONE 表示プロトコル

ドメインに関連する NMG にコマンドが送られない場合は、ドメインの PresentationProtocolName 値に NONE を指定します。例えば、ドメイン内のリソースについてのアラートのみを受け取るようにドメインが定義される場合、 NONE を指定します。

すべての表示プロトコルの出力の形式化

このセクションでは、DOMP020 および PASSTHRU プロトコル、ならびに DOMP010 プロトコルの出力の形式化について説明します。

DOMP020 および PASSTRU 出力の形式化

NMG が COS トランスポート・プロトコルを使用する場合は、サブベクトル 31 に RUNCMD への応答が入ります。サブベクトル 31 の応答は、以下のように形式化されます。ネイティブ・エレメント・マネージャーが複数行の応答テキストを GMFHS に送る場合は、応答テキストの各行を別々のサブベクトル 31 に入れなければなりません。この結果、応答テキストの個々の行は、必ずテキストの独立した行としてワークステーションの「Command Responses」ウィンドウに表示されます。

DOMP010 の出力の形式化

複数行応答のテキストの各独立行の前には、個別のテキスト・キーワード (TX) が付きます。 DOMP010 プロトコル用 TX キーワードの使用の詳細については、 85 ページの『テキスト − TX』を参照してください。

DOMP010 の形式化の規則

このセクションでは、COS NMG のコマンドか、プログラム間インターフェース NMG に送られたデータに含まれるテキスト・データの形式について説明します。このセクションで、パケット の用語はこれらのサブベクトル内の情報を意味します。

一般のパケット形式

パケットは、コンマで区切られた 1 つまたは複数のキーワード・パラメーターから構成されます。これらのパラメーターは、コマンドまたは応答を識別するなどの機能を実行します。テキスト・パケット内の値は、すべて表示可能文字です。

- NetView/PC API/CS 環境での表示可能文字は、ASCII でコーディングされます。
- SNA ネットワークでの文字は、表示可能な EBCDIC でコーディングされます。NetView/PC API/CS は、必要なコード・セットの変換を行います。

各パラメーターには、次のような一般的な形式があります。

```
keyword=value
```

各キーワードは長さが 2 文字で、常に等記号がつきます。値は、可変長の値です。例えば、CP が値 MINIA をもつキーワードならば、キーワード・パラメーターは以下のようになります。

```
CP=MINIA
```

キーワード値は、複数のデータ項目から構成することができ、コンマで区切られて、1 組の括弧で囲まれます。例えば、次のようになります。

```
CP=(MINIA,MINIB)
```

代表的なパケットでは、いくつかのキーワード・パラメーターが指定されます。キーワード・パラメーターもコンマで区切られます。例えば、次のようになります。

```
CM=AE,SQ=10,DM=DOMAIN,CP=(MINIA,MINIB)
RP=AE,SQ=10,DM=DOMAIN,CP=MINIA,ST=U,TM=930601120000,CP=MINIB,ST=U,TM=930601120000,
```

ほとんどの場合、個々のパラメーターの順序は重要ではありません。この規則に対する例外は、キーワードの説明のところで注記します。

キーワードおよび値の定義

パケット・キーワードとその説明を以下に示します。

キーワード	説明
CE	コマンドの実行
CM	コマンド ID (コマンドに必須)
CP	コンポーネント ID
DM	ドメイン ID
PT	プロトコルのテキスト
RN	理由
RP	応答 ID (応答に必須)
SN	コマンド送信側の ID
SQ	メッセージ順序番号 (コマンドおよび応答に必須)
ST	状況 ID
TM	タイム・スタンプ
TX	ネイティブ・コマンドまたは応答テキスト

以下のセクションでは、各キーワードとその値を説明します。

コマンドの実行 − CE

コマンド実行状況キーワード CE は、コマンドの実行が正常でなかったことを示します。これは、否定応答がコマンド全体に適用されるという点で、否定応答 (RP=X) とは異なります。コマンド実行の失敗は、コマンドのサブセットに適用されます。

CE のキーワード値は、テキスト・ストリングに含まれる値リストです。この値は、理由 (RN) キーワードの値と同じです。これらの値については、81 ページの『理由 − RN』を参照してください。

コマンドが状況表示 (CM=D) もしくは異常状況表示 (CM=A) であり、応答が複数のコンポーネントの状況を伴う場合、コマンドの実行はコンポーネントのいずれかで失敗しました。これは、以下のように示されます。

```
CP=component_name,ST=X,CE=(reason text)
```

同じコマンド応答は、同じくコマンドが正常であったコンポーネントの状況をともないます。コマンドの実行がコンポーネントごとに個々に失敗した場合は、 CE キーワードと ST=X がコンポーネントごとに戻されます。

注: ST=X の使用 (必須) は、このコンポーネントについて報告済みの状況があれば依然有効であることを指定します。

CE キーワードは位置依存です。 CE の位置は、その対象コンポーネントの CP キーワードの後、かつ他のすべてのコンポーネントの前になければなりません。すなわち、特定のコンポーネントの CP と CE のペアの間を他の CE キーワードで分割してはなりません。

CE キーワードは、状況表示および異常状況表示 (CP=A および CP=D) についてサポートされています。

コマンド − CM

コマンド・キーワード CM は、エレメント・マネージャーに出されるコマンドです。このキーワードは、ホストからエレメント・マネージャーに送られるどのパケットにも必須です。

CM 値には、2 つの部分からなる定義があります。

- この値の先頭バイトは、コマンド・タイプです。コマンド・タイプにより、非 SNA 装置に出すコマンドのタイプを分類します。以下のリストは、コマンド・タイプの説明です。

値	説明
A	異常状況表示
C	再構成ドメイン
D	名前付きリソースまたはリソースの状況表示
I	非活動化リソース
N	ネイティブ・コマンド
P	プロトコル・メッセージ
R	再生リソース
V	活動化リソース
X	否定応答

- 2 番目のバイトは継続です。

  継続バイトは、複数の応答を要求することができるコマンド・タイプと一緒に使用されます。

値	説明
E	これは、初期要求もしくは初期要求に対する最後の応答です。
M	これは、継続要求であるか、最後の応答でないか、のいずれかです (複数の応答が初期要求にサービスする必要があるとき)。

  継続バイトの重要性の詳細については、 86 ページの『複数応答プロトコル』を参照してください。

コンポーネント ID − CP

CP キーワードによって指定されるコンポーネント ID は、 RODM データ・キャッシュ内 GMFHS_Managed_Real_Resource オブジェクトの MyName 値のリソース部分と一致している必要があります。例えば、リソースの MyName が OTTAWA.MINIA ならば、`CP=MINIA` と指定します。

値リストを用いると、1 つの CP キーワードで複数のリソースを指定することができます。例えば、1 つのコマンドに 3 つのリソースが組み込まれる場合の CP キーワードは次のようになります。

```
CP=(MINIA,MINIB,MINIC)
```

注: コマンド応答は、応答が複数のリソース用であれば、コンポーネント ID リストではなく複数の CP 値を使用します。

CP キーワード値のサイズは、以下によって異なります。

- エレメント・マネージャーを入れる NMG のタイプ

- コマンドの必須キーワードのサイズ
- コマンドのオプション・キーワードのサイズ

最大のコマンド・サイズは、NMG タイプによって異なります。最大サイズは、以下のいずれかにすることができます。
- COS ゲートウェイには、240 文字
- OST ゲートウェイには、256 文字
- プログラム間インターフェース・ゲートウェイには、253 文字

CP キーワードで有効なリソース名の最大サイズを判別するには、以下のように行います。

1. ベース・コマンドの文字数と CP キーワードの構文の文字数を加算する。
2. その合計を、NMG がサポートする最大の長さから減算する。

例えば、以下のコマンドには 24 文字が含まれています。

```
CM=DE,SQ=5,DM=DOMAIN,CP=aaa
```

したがって、リソース名 *aaa* の最大サイズは、 COS ゲートウェイの場合は 216 文字、OST ゲートウェイの場合は 232 文字、プログラム間インターフェース・ゲートウェイの場合は 229 文字です。

次のコマンドには 28 文字が含まれています。

```
CM=DE,SQ=5,DM=DOMAIN,CP=(aaa,bbb,ccc)
```

したがって、リソース名 *aaa*、*bbb*、および *ccc* の最大サイズは、 COS ゲートウェイの場合は 212 文字、OST ゲートウェイの場合は 228 文字、プログラム間インターフェース・ゲートウェイの場合は 225 文字です。

コマンドに複数のコンポーネントを指定し、コマンドのサイズが最大を超えると、GMFHS は自動的にコマンド内のリソース数を減らし、コマンド・サイズを縮小します。

ドメイン − DM

ドメイン・キーワード DM は、複数の非 SNA ドメインがサポートされるときのリソースの非 SNA ドメインを指定します。ドメイン・キーワードはオプションです。

DM は、GMFHS が CP キーワードで指定されたリソースに結びつくドメインを示します。 DM は、Non_SNA_Domain_Class オブジェクトの EMDomain フィールドと一致する必要があります。例えば、リソースの MyName が OTTAWA.MINIA ならば、キーワード・パラメーターの形式は以下のようになります。

```
DM=OTTAWA
```

DM 値の長さは、8 文字まで使用できます。

プロトコル − PT

プロトコル・キーワード PT は、コマンド ID (CM) または応答 ID (RP) コマンド・タイプが、プロトコル・コマンド (P) と等しいときに使用します。例えば、CM=PE (E は継続バイト)。

PT 値は、ホスト上およびコマンドのターゲット (ネイティブ・エレメント・マネージャー) 上の 2 つの連携処理間の通信セッションを制御するプロトコル・コマンドです。コマンドにはすべて応答が必要なので、どのプロトコル・コマンド要求もプロトコル・タイプの応答をもつ必要があります。

表 10 は、定義済み PT 値のリストで、 DOMS010 プロトコルに使用するセッション・プロトコル・コマンドを表示しています。

表 10. プロトコル・コマンドの値

プロトコル・コマンド	意味
SESSION_REQUEST	GMFHS によってエレメント・マネージャーに送られ、セッションの確立を要求する。
SESSION_REQUEST_ACCEPT	SESSION_REQUEST プロトコル・コマンドを確認する応答。このコマンドは、セッションの確立は示しません。
INIT_ACCEPT	GMFHS により戻され、INIT アラートの受信を確認する。
INIT_ACCEPT_ACCEPT	INIT_ACCEPT プロトコル・コマンドを確認する応答。
SET_CLOCK	GMFHS が、INIT_ACCEPT_ACCEPT プロトコル・コマンドを受け取った後で、かつ SET_CLOCK プロトコル・コマンドがドメインのネイティブ・エレメント・マネージャーによってサポートされている場合に、 GMFHS により送られる。このメッセージが送られるのは、サポート・セット・クロック・ビットが DomainCharacteristics フィールドで「オン」で設定される場合に限ります。 SET_CLOCK は、その TM パラメーター値に現在のローカル時刻を指定します。このメッセージは、セッションがアクティブである限り 24 時間ごとに出されます。
SET_CLOCK_ACCEPT	ネイティブ・エレメント・マネージャーにより戻され、SET_CLOCK プロトコル・コマンドを確認する。

注: GMFHS から着信するコマンドの PT キーワードの値は、小文字です。 GMFHS は、応答値では大文字小文字を区別しません。

例えば、GMFHS が NMG から INIT アラートに応答する場合のパケットの形式は、以下のようになります。

```
CM=PE,DM=DURHAM,SQ=7,PT=(INIT_ACCEPT)
```

INIT_ACCEPT への応答は次のようになります。

```
RP=PE,DM=DURHAM,SQ=7,PT=(INIT_ACCEPT_ACCEPT)
```

SET_CLOCK プロトコル・コマンドがサポートされている場合、 GMFHS はそれを 24 時間ごとに NMG に送り、 NMG がそのクロックを正しい時刻に設定できるようにします。現在時刻は、TM キーワードにより設定され、 INIT アラートで指定された NMG のオフセットが考慮されます。例えば、次のようになります。

```
CM=PE,SQ=8,DM=DURHAM,PT=(SET_CLOCK),TM=930101120000
RP=PE,SQ=8,DM=DURHAM,PT=(SET_CLOCK_ACCEPT)
```

これらのプロトコルの詳細については、 89 ページの『DOMS010 のセッションの確立』を参照してください。

理由 − RN

理由キーワード (RN) は、要求が受け入れられなかった理由を示します。 RN キーワードには、RP=XE が常に使用されます。

理由値は、値リスト形式のテキスト・ストリングです。例えば、次のようになります。

```
RN=(execution node inaccessible)
```

表 11 は、サポートされているテキスト値のリストです。

表 11. 理由値

値	説明
Aborted	要求の完了を禁止するエラーが発生した (メモリー、CPU、ディスクなどの障害)。
Canceled	要求が取り消され、完了できなかった。
Component unknown	ターゲット・コンポーネントが不明。
Currently not allowed	コマンド・タイプはサポートされているが、現時点では、ターゲット・コンポーネントが実行できない。
Execution node inaccessible	要求されたコマンドを実行するターゲット・ノードがアクセス不能。
Failed	コマンド処理は完了したが、予定の結果が得られなかった (ACTIVATE によってコンポーネントが活動化しなかった)。
Invalid command ID	コマンド・タイプが無効。
Invalid parameter	キーワード・パラメーターが誤りで、コマンドの実行が禁止された。
No resources	要求の実行に使用できるリソースが十分でなかった (メモリー、CPU、ディスクなど)。
Not allowed	コマンド・タイプはサポートされているが、ターゲット・コンポーネントには使用できない。
Not supported	コマンド・タイプが、コマンドを処理するエンティティーによってサポートされていない。
Preempted	別の処理により優先使用され、要求が完了できなかった。
Timed out	要求がタイムアウトになり、有効な応答が処理できなかった。

注: GMFHS は、応答値では大文字小文字を区別しません。

応答 − RP

応答キーワード RP は、コマンド応答パケットを識別します。応答キーワード値は、78 ページの『コマンド − CM』でのコマンド・キーワード、CM の説明内容と同じです。 RP 値も、CM 値で説明した継続バイトを使用します。

例えば、単一のコンポーネントに状況表示コマンドを出す場合、応答は肯定で、継続メッセージは必要とされません。以下は、キーワード・パラメーターの形式です。

```
RP=DE,SQ=5,DM=DOMAIN,CP=MINIA
```

要求への応答が否定ならば (要求は正常に完了できない)、コマンド・タイプの先頭バイトに X が入れられます。例えば、次のようになります。

```
RP=XE,SQ=5,DM=DOMAIN,RN=(no resources)
```

コマンドの送信側 ID － SN

コマンドの送信側 ID キーワード SN は、コマンドの送信側を識別します。 SN キーワードは、すべてのコマンドに組み込まれています。キーワード値は、常に GMFHS です。

```
SN=GMFHS
```

メッセージ順序番号 － SQ

メッセージ順序番号キーワード SQ には、要求もしくは応答を識別する固有のメッセージ順序番号が含まれています。応答のメッセージ順序番号は、元の要求で使用された順序番号と同じです。例えば、あるコンポーネントに順序番号 6 で状況表示コマンドを出すと、その要求に対する応答の順序番号も 6 になります。

SQ には、継続応答についての相互関係があります。単一の要求で複数の応答が必要な場合は、応答のすべてを元の要求に相関させるのにメッセージ順序番号が使用されます。例えば、メッセージ順序番号 35 で異常状況表示 COMPONENTS コマンドを出すと、一連の応答中の最初の応答のメッセージ順序番号は 35 で、継続バイトはそれより多く (M) 設定されます。例えば、次のようになります。

```
CM=AM,SQ=35
```

発信側は、継続バイトを M に設定し、メッセージ順序番号 35 で別の要求を送ることができます。応答側は、この要求を受け取ると、要求が、前の応答パケットに収まりきらないデータを続けて送ってくることがわかります。この交換の繰り返しは、応答の継続バイトが終了 (E) に設定されて、元の要求が満たされるまで継続します。

メッセージ順序番号は、999 に達すると最初に戻ります。

状況 － ST

状況キーワード ST は、以下のいずれかを記述するのに使用します。

- 状況表示 (CM=A または CM=D) コマンドへの応答でのコンポーネントの状況
- 活動化 (CM=V)、非活動化 (CM=V)、もしくは再生 (CM=R) コマンドへの応答での、結果としてのコンポーネント状況

状況キーワードの値は、リソースの GMFHS 外部状況であっても NETCENTER 内部状況であってもかまいません。

- リソースの GMFHS 外部状況を記述するときは、1 バイト値が使用されます。
- リソースの NETCENTER 内部状況を記述するときは、値リストが使用されます。

応答メッセージの特定のリソースに使用できる状況値タイプは、1 つだけです。

複数のリソースに関する状況が報告されるときは、 ST キーワード・パラメーターおよび値は、関連する各コンポーネント ID キーワード (CP) の直後に続く必要があります。 ST と TM キーワードが一緒に送られる場合、それらがともに関連する CP キーワードの後に続いている限り、その固有の順序は重要ではありません。

表 12 は、単一バイトの外部状況と NETCENTER の外部状況です。

注: 非 SNA ドメインを定義して、いずれかのタイプの状況を認識することができます。 Non_SNA_Domain_Class のオブジェクトで、DomainCharacteristics フィールドのビット 13 がオンになると、 GMFHS は、NETCENTER 状況キーワードを GMFHS 状況キーワードに変換します。

表 12. NETCENTER から NetView キーワードへの変換および説明

GMFHS 状況	NETCENTER 状況	NETCENTER の説明
U (不良)	A	異常 − 装置は稼働しているが、異常状態がある。
U (不良)	B	使用不能 − オペレーターもしくはシステムによる意図的な非活動化
? (不明)	C	構成なし − 装置がネットワーク定義の一部になっていない。
U (不良)	D	ダウン
S (適合)	N	正常
I (中間)	P	パフォーマンス − パフォーマンス上の問題
I (中間)	T	途中 − 装置が現在状況を変更中。
? (不明)	U	使用不能 − 状況が使用不能。
変更なし	X	状況の要求が実行不可 − それまでに報告された状況が依然有効と考えられる。

リソースの GMFHS 外部状況が不良ならば、 ST キーワード・パラメーターの形式は以下のとおりです。

```
ST=U
```

コンポーネント NODE1 および NODE2 の GMFHS 外部状況が報告され、その状況がそれぞれ適合と不良ならば、ST キーワード・パラメーターの形式は以下のとおりです。

```
CP=NODE1,ST=S,TM=890315120801,CP=NODE2,ST=U,TM=890315120814
```

GMFHS は、マイグレーションを行うためにすべての NETCENTER 状況値をサポートします。 GMFHS は、NETCENTER 内部リソース状況値を自動的に GMFHS 状況値に変換します。

内部状況の 3 つの NETCENTER カテゴリー (構成、操作、使用状況) が、値リストに入れられます。 例えば、次のようになります。

```
ST=(configuration,operation,utilization)
```

リスト内の各位置で、コンポーネントのそのカテゴリーの状況を定義します。長さは 1 バイトです。 各リスト・エレメントのさまざまなリソースの状況の説明に使用する値については、 84 ページの表 13 で説明します。

表 13. リソース状況値

カテゴリー	値	意味
構成	C	非構成
	U	使用不能
	V	アクティブ
	X	非アクティブ
操作	A	異常
	L	作動不能
	O	作動可能
	U	使用不能
使用状況	N	正常
	R	過負荷
	U	使用不能

GMFHS は、ビューもしくは総称コマンド応答でリソース状況を表示する際に、 3 つの内部状況値から成ります。

例えば、リソースの 3 つのカテゴリーが、構成=使用不能、操作=操作可能、および使用状況=正常ならば、ST キーワード・パラメーターは以下のとおりです。

```
ST=(U,O,N)
```

タイム・スタンプ − TM

タイム・スタンプ・キーワード TM は、ローカルの日付および時刻を記述します。TM 値とキーワードは、コマンド応答でコンポーネントが指定されるつど必要であり、かつ応答で指定されたコンポーネント状況ごとに必要です。これには、D、A、I、V、および R コマンドが含まれます。タイム・スタンプ・キーワードは他の応答内にあってもかまいませんが、無視されます。 TM キーワードは、SET_CLOCK セッション・プロトコル・コマンドにも組み込まれて、エレメント・マネージャーのクロックの設定値を指定します。

複数のリソースに関する時刻が報告される場合は、 TM キーワード・パラメーターおよび値は、関連する各コンポーネント ID キーワード (CP) の直後に続く必要があります。 TM と ST キーワードが一緒に送られる場合、それらがともに関連する CP キーワードの後に続いている限り、その固有の順序は重要ではありません。以下は、タイム・スタンプの形式です。

```
TM=yymmddhhmmss
```

タイム・スタンプの変数は次のように定義されます。

yy 年
mm 月 (01 から 12)
dd 日 (01 から 31)
hh 時 (00 から 23)
mm 分 (00 から 59)
ss 秒 (00 から 59)

例えば、状況の報告が 1993 年 5 月 28 日のローカル時刻で午後 3 時 58 分 21 秒現在であるならば、 TM キーワード・パラメーターは次のようになります。

```
TM=930528155821
```

テキスト − TX

テキスト・キーワード TX は、ネイティブ・コマンドとその応答をサポートします。 TX の値はテキストのストリングです。

コマンドの場合の TX 値は、ネイティブ・エレメント・マネージャーのコンソールから入力されたコマンドのような、ネイティブ・ネットワーク・コマンドのテキストです。以下は、SHOW CIRCUIT A ネイティブ・コマンドのデータ項目形式です。

```
TX=(SHOW CIRCUIT A)
```

応答の場合の TX は、ネイティブ・エレメント・マネージャーのコンソールで受信された応答テキストです。コマンドがオペレーターから出された場合、コマンド応答は Command Response (コマンド応答) ウィンドウに表示されます。 TX キーワードが出現するごとに、 NetView ワークステーションでテキストが 1 行表示されます。コマンドに対する応答が CIRCUIT A CONFIGURED AND OPERATIONAL である場合は、以下が応答キーワード・パラメーターの形式です。

```
TX=(CIRCUIT A CONFIGURED AND OPERATIONAL)
```

コマンドに対する応答が複数行応答ならば、応答キーワード・パラメーターの形式は以下のようになります。

```
TX=(    COMMAND FAILURE STATISTICS),
TX=(ROUTES    ERRORS    HITS    MISSES),
TX=(  40       250      2000       4)
```

個々のパラメーター行はコンマで区切られます。テキスト応答の場合は、パラメーターの順序は重要です。個々の TX キーワードごとに Command Response ウィンドウでのテキストは別々の行になります。

) 文字 (右括弧) で TX テキスト・ストリングは終了します。テキストに組み込みの) 文字が含まれる場合は、その) を 2 番目の) 文字の前に置きます。コマンドへの応答が CIRCUIT (A) CONFIGURED AND OPERATIONAL である場合は、以下が応答キーワード・パラメーターの形式です。

```
TX=(CIRCUIT (A)) CONFIGURED AND OPERATIONAL)
```

コマンドの形式化およびプロトコルの例

このセクションでは、必須の表示処理プロトコルの例を記載します。プロトコルは、機能的に以下の 2 つがあります。

- 単一応答プロトコル
- 複数応答プロトコル

コマンドやコマンドの応答パケットを構成する各種キーワードおよび値については、 77 ページの『キーワードおよび値の定義』を参照してください。コマンド・タイプおよび継続バイトのリストについては、 78 ページの『コマンド − CM』を参照してください。

単一応答プロトコル

単一応答プロトコルは、初期コマンドとして指定されたコマンドと最後の応答として指定された応答から構成されています。 86 ページの図 20 は、状況表示コマンド

と応答の場合に交換されるパケットを示しています。

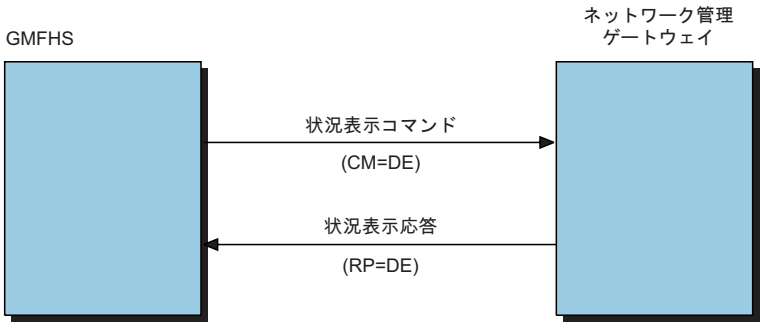


図 20. 単一応答プロトコル

GMFHS から送られるコマンドには、CM キーワードが含まれています。プロトコルを維持するために、CM 値の先頭文字 D は交換可能になっています。この文字は、状況表示のコマンド・タイプを意味します。この値は、コマンドに有効ならばどのコマンド・タイプでもかまいません。

しかし、継続文字の E の値は、初期コマンドを指定します。この文字は、追加のコマンド・パケット継続 (継続値 = M) が必要かどうかに関係なく、コマンド・パケットの最初の出現に必ず組み込まれている必要があります。

ネイティブ・エレメント・マネージャーからの応答では、RP キーワードは値 DE をもっています。コマンド・タイプ文字は交換可能です。継続文字の E の値は、生成された最後の応答を指定します。

プロトコルは、SQ キーワードでさらにチェックを行います。応答の SQ 値はコマンドの SQ 値と等しくなければなりません。

以下の例に見られるように、単一応答プロトコルは、応答に単一リソースの場合を超えるデータが含まれることを見越しています。

コマンドは、単一の CP キーワード・パラメーターの 3 つのリソース、RALV4.RALXT1、RALV4.RALXT2、および RALV4.TX02 の状況を要求します。

```
CM=DE,DM=EASTSIDE,CP=(RALV4.RALXT1,RALV4.RALXT2,RALV4.TX02),SQ=1
```

応答には、要求されたリソースごとに別々の CP キーワードが入ります。

```
RP=DE,DM=EASTSIDE,CP=RALV4.RALXT1,ST=N,TM=901201135901,
CP=RALV4.RALXT2,ST=N,TM=901201135912,
CP=RALV4.TX02,ST=D,TM=901201135914,SQ=1
```

注: CM および SQ キーワード・パラメーターはコマンド内に、 RP および SQ パラメーターは応答内にあります。

複数応答プロトコル

応答データが大きすぎて単一の応答に収まらないときは、 GMFHS および NMG は複数応答プロトコルを使用します。

複数応答プロトコルは、以下から構成されます。

- 初期コマンドとして指定されたコマンド

- 数が無限の継続応答およびコマンド
- 最後の応答

図 21 は、最も単純な複数応答の例で、状況表示コマンドと応答の場合に交換されるパケットを示しています。

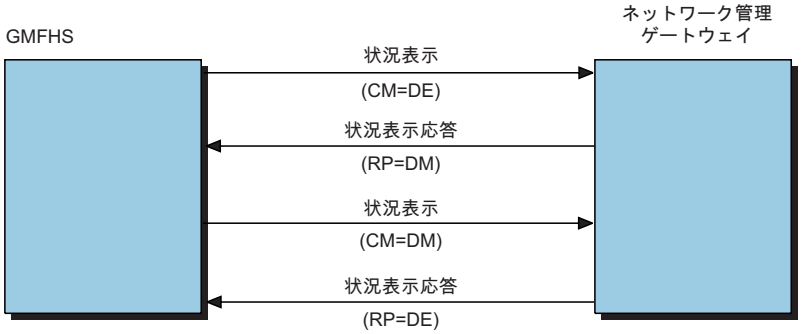


図 21. 複数応答プロトコル

NetView プログラムから送られる初期コマンドには、継続文字が E に設定された CM キーワード (CM=AE) が入っています。 NMG 応答は、RP キーワード継続パラメーターとして値 M を組み込むことで (RP=AM)、応答にはデータのすべてが含まれないことを示します。

それ以上の応答データを得るには、GMFHS は再度要求を出します。要求パラメーターは、M に設定される継続パラメーター (CM=DM) の場合を除き、すべて初期要求と同じです。 NMG は残りのデータを送り、継続パラメーターを E に設定して (RP=AE)、送るデータがこれ以上ないことを示します。

次の初期コマンドは、非 SNA ドメイン B3088P2 (状況が異常の) のすべてのリソースの表示を要求しています。

```
CM=AE,DM=B3088P2,SQ=44
```

以下はその結果の応答です。

```
RP=AM,DM=B3088P2,SQ=44,CP=TIM,ST=A,TM=911231235959,
CP=A0488P23,ST=C,TM=920101000000,
CP=A0488P24,ST=U,TM=920101000001
```

この応答は、応答が継続すること (RP = AM) を示し、3 つのリソース、A0488P22、A0488P23、および A0488P24 の状況を示しています。

コマンドは再度送られます。

```
CM=AM,DM=B3088P2,SQ=44
```

継続文字は M に設定され (CM = AM)、コマンドが順序番号 44 (SQ=44) の前のコマンドの継続であることを示します。

最後に、もう一つの応答が交換を終了します。

```
RP=AE,DM=B3088P2,SQ=44,CP=RALV4.TX02,ST=A,TM=920101000002
```

継続文字が E に設定され (RP=AE)、これが最後の応答であることを示します。

タイミングに関する考慮事項

状況情報は、総称アラートとコマンド応答の両方に入っているので、 GMFHS はアラートまたは応答を処理する時点にはタイム・スタンプを備えます。アラートの日付および時刻は、ネイティブ・エレメント・マネージャーもしくは NMG のそのエージェントによって用意されます。

アラート

NetView プログラムは、アラートの有効時刻は、 NetView プログラムがアラートを受け取った時刻と見なします。

しかし、この標準では、NMG によって報告される非 SNA アラートの場合に問題が発生します。アラートは、 VTAM プログラムに送られてから、 GMFHS に送られるまでに、非 SNA ネットワークおよび NMG 内でかなり遅れることがあります。遅延の結果、アラートのタイム・スタンプが不正確になり、ネットワーク問題解決を複雑にすることも、無駄にすることもあり得ます。 GMFHS は次の規則を用いて、これらの欠点を解決しています。

- アラートの発信側は、日付/時刻サブベクトルをアラートに組み込むことができます。これは、NetView プログラムがアラートを受け取った時刻をオーバーライドします。 サブベクトルのグリニッジ標準時 (GMT) オフセットが、オプションの GMT オフセット・サブフィールドで指定されていれば、使用されます。
- アラートの日付/時刻サブベクトルに GMT オフセットが組み込まれずに、ネイティブ・エレメント・マネージャーが、セッションの確立でその GMT オフセットを報告した場合は、ネイティブ・エレメント・マネージャーのオフセットが使用されます。
- アラートの日付/時刻サブベクトルに GMT オフセットが組み込まれずに、セッションの確立でオフセットが指定されない場合は、日付/時刻サブベクトルの時刻が使用され、 NetView プログラムのローカルの GMT オフセットによって正規化されます。

コマンド応答

GMFHS では、タイム・スタンプ・キーワード・パラメーター (TM) はコンポーネント状況を含むどのコマンド応答にも組み込まれている必要があります。しかし、状況応答は、同じコンポーネントのより最近のアラートの後に、GMFHS に到着することができます。これが起こるのは、ネイティブ・エレメント・マネージャーが複数のコンポーネントからの状況で応答を組み立て、かつコンポーネントの状況の変更が、その応答の後であっても、応答の送信前に行われる場合です。ネイティブ・エレメント・マネージャーがこのコンポーネントのアラートをコマンド応答の送信前に送ると、 GMFHS は状況の標識を誤った順序で受け取ります。

GMFHS は、タイム・スタンプを比較することでこの状態から回復します。状況更新 (アラートもしくはコマンド応答) が、報告された最新の状況より前にタイム・スタンプが取られている場合、 GMFHS はその新しい状況を適用しません。 GMFHS は、監査メッセージおよびコンソール・メッセージをログに記録します。

タイム・スタンプ・キーワードには、GMT オフセットは組み込まれていません。GMFHS は、タイム・スタンプを正常にしてそれらを比較します。 GMFHS とネイティブ・エレメント・マネージャーの間のセッションを確立するために使用されている INIT アラートにネイティブ・エレメント・マネージャーの GMT オフセット

が含まれている場合、このオフセットが使用されます。それ以外の場合は、GMFHS のローカル GMT オフセットが使用されます。

非 SNA セッション・プロトコルを定義する

非 SNA ドメインに指定するセッション・プロトコルは、GMFHS がそのドメインのコマンドおよび応答通信セッションの確立、保守、および終了を行う方法を示します。ドメインに使用するプレゼンテーション・プロトコルは、 RODM の非 SNA ドメイン・オブジェクトの SessionProtocolName フィールドで指定します。以下は、有効なセッション・プロトコル名です。

- DOMS010
- PASSTHRU
- NONE

GMFHS も、エレメント・マネージャーとの通信セッションの確立、保守、および終了を行います。 GMFHS は、Non_SNA_Domain_Class オブジェクトの SessionProtocolName フィールドの値を用いて、エレメント・マネージャーとのセッションの確立方法を判別します。

DOMS010

DOMS010 プロトコルは、 GMFHS と非 SNA ドメイン間のコマンド・セッションの確立を調整する一連の規則およびコマンド構文を指定します。

DOMS010 セッション・プロトコルは、GMFHS がセッションの存在を判別する前に、 GMFHS およびエレメント・マネージャーが相互の身元確認をする必要があることを指定します。 GMFHS がエレメント・マネージャーに送るコマンド、および予期する応答については、79 ページの『プロトコル － PT』で説明しています。また、『DOMS010 のセッションの確立』に、この識別順序の例が示されています。

ドメインが DOMS010 を指定すると、コマンドの形式化は、PresentationProtocolName フィールドの値に関係なく、 DOMP010 の形式化の規則に従って行われます。

PASSTHRU

PASSTHRU プロトコルは、セッション確立情報の交換なしに、 GMFHS と非 SNA ドメイン間にコマンド・セッションが存在することを指定します。 GMFHS は、GMFHS の初期化と同時に、コマンド・セッションがアクティブであると見なします。

NONE

NONE プロトコルは、ドメインにはコマンド・サポートがないことを示します。

DOMS010 のセッションの確立

DOMS010 セッション・プロトコルは、他のコマンドが使用可能になる前に、GMFHS がドメインとのセッションを獲得する必要があることを指定します。セッションは、GMFHS によって、もしくはエレメント・マネージャーから開始します。 90 ページの図 22 に、エレメント・マネージャーから開始されるセッションの確立を示します。

ドメインの状況として GMFHS が報告する内容を表示するときは、GMFHS SHOW DOMAIN コマンドを使用します。 SHOW コマンドについては、NetView オンライン・ヘルプを参照してください。

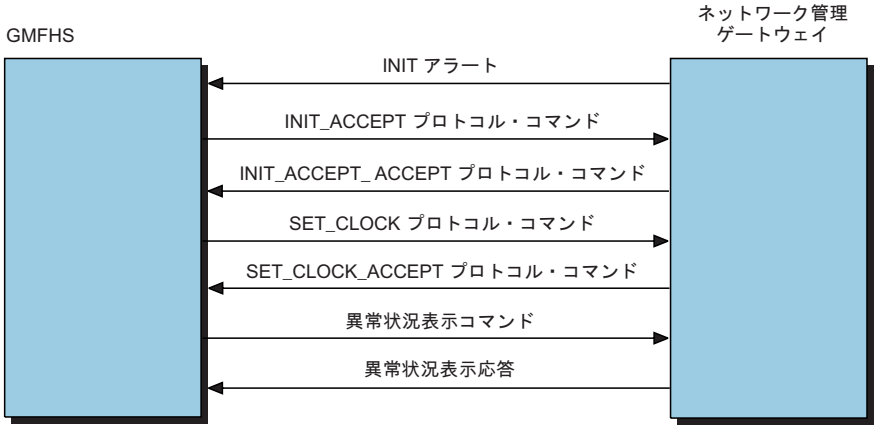


図 22. NMG の要求によるセッションの確立: この図で示されているコマンドは、 79 ページの『プロトコル － PT』で説明しています。

エレメント・マネージャーは、INIT 総称アラートを送って、 GMFHS とのセッションを開始することができます。 GMFHS がこのアラートを受け取ると、以下のことを行います。

- INIT_ACCEPT プロトコル・コマンドで NMG に応答する。 INIT アラートについては、 92 ページの『セッションの確立の INIT 総称アラート』で説明します。
- SET_CLOCK プロトコル・コマンドを送る (サポートされている場合)。
- 1 つまたは複数の異常状況表示総称コマンドもしくは状況表示総称コマンドを送り、すべてのリソースの現在の状況を検索する。異常状況表示がサポートされていない場合は、 GMFHS がそれぞれのリソースに状況表示総称コマンド (サポートされている場合) を出します。これらのコマンドのサポートの有無は、ドメインを GMFHS に定義する Non_SNA_Domain_Class オブジェクトの DomainCharacteristics フィールドによって指定されます。

NetView/6000 V2、NetView for AIX V3、NetView for AIX V4、および DOMS010 のセッションの確立

NetView/6000 バージョン 2 (V2)、 NetView for AIX バージョン 3 (V3)、NetView for AIX バージョン 4 (V4) は、 NETCENTER のみを直接サポートします。 GMFHS のドメイン命名規則は NETCENTER の場合とは異なるので、NetView では、サンプルの CNMS4406 を提供して、GMFHS と NetView/6000 V2、 NetView for AIX V3、および NetView for AIX V4 間のセッションの確立を容易にしています。

このサンプルは、DOMS010 のセッションの確立の INIT および DOWN アラート部分を備えています。このサンプルによって、ユーザーは以下の指定を行うことができます。

- 非 SNA ドメインの 3 つの名前つきエレメント (41 ページの『非 SNA ドメインを定義する』を参照)。 サンプル CNMS4406 では、サービス・ポイント (SP)

は sp_name、トランザクション・プログラム (TP) は tp_name、エレメント管理サブシステム (EMS) は domain_name です。

- INIT または DOWN アラートを送信するかどうか。この場合、このアラートは、RODM 内の類似した名前をもつドメイン・オブジェクトと NetView/6000 V2 サービス・ポイントとの突き合わせを行います。

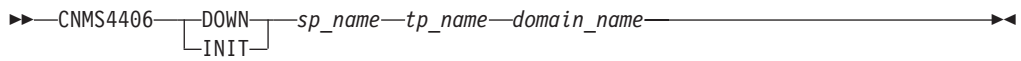
サンプル CNMS4406 は、 C 言語でコーディングされた NetView コマンド処理プログラムです。これを使用するには、まず LONGNAME コンパイル・オプションを指定して C 言語でコンパイルし、実行可能な NetView ライブラリーに入れる必要があります。

注: サンプルのコンパイル方法に関する詳細については、「*IBM Tivoli NetView for z/OS Programming: PL/I and C*」を参照してください。 LONGNAME コンパイル・オプションについては、「*OS/390 C/C++ プログラミング・ガイド*」(SC88-7720) を参照してください。

以下の CMDDEF ステートメントを DSIPARM メンバー CNMCMD に入れておくことも必要です (マイグレーションのために組み込みファイル CNMCMDU を使用します)。

```
CMDDEF.CNMS4406.MOD=CNMS4406
CMDDEF.CNMS4406.RES=N
```

以下は、このサンプルの構文図です。



例えば、 A0488P31.A94306F8.NETVIEW という名の NetView/6000 V2 ドメイン・オブジェクトのサンプル CNMS4406 を実行するときは、 NetView コマンド機能もしくは NetView 自動化テーブルから以下のコマンドを用いて、INIT アラートを送ることができます。

```
CNMS4406 INIT A0488P31 A94306F8 NETVIEW
```

GMFHS と、NetView/6000 または NetView for AIX 間にセッションを確立するには (ともにアクティブのとき)、このサンプルを自動化テーブルに入れて、常に該当する INIT および DOWN アラートに送ります。

GMFHS － 開始済みセッションの確立

GMFHS は、受動のセッション相手側ですが、エレメント・マネージャーにセッションを開始するようにプロンプトを出すことができます。 Non_SNA_Domain_Class オブジェクトの DomainCharacteristics フィールドは、 GMFHS セッションが確立されたことを確認し、ドメインの NMG からの状況を送信請求します。このプロンプトは、以下のときに起こることがあります。

- GMFHS の始動時、およびセッションが得られるまでのユーザー定義の時間間隔ごと
- NMG の状況が適合へと変化したのを GMFHS が検出したとき、および GMFHS に NMG 下のエレメント・マネージャーとのセッションがないとき

DOMS010 プロトコルは、 DOMP010 プロトコルと同じプロトコル・コマンド (80 ページの表 10 に記載されている) を使用します。交換は、図 23 の図解のように行われます。

図 23. GMFHS の要求によるセッションの確立

GMFHS は、SESSION_REQUEST プロトコル・コマンドを送信して、エレメント・マネージャーとのセッションを始めます。エレメント・マネージャーは、このコマンドを受け取ると、SESSION_REQUEST_ACCEPT プロトコル・コマンドで応答し、総称 INIT アラートを生成します。この処理の残りの部分については、 89 ページの『DOMS010 のセッションの確立』で説明しています。

セッションの確立の INIT 総称アラート

プロトコル・コマンド以外に、DOMS010 プロトコルには INIT アラートが組み込まれています。エレメント・マネージャーは、INIT アラートを生成して、GMFHS とのセッションを確立します。

表 14 は、 INIT 総称アラートで表示する必要があるサブベクトルおよびデータのリストです。

注: 特にオプションの注記がない限り、サブベクトルとデータはすべて必須です。

表 14. 総称アラート・サブベクトル

サブベクトル	説明
総称アラート・データ・サブベクトル	アラート・タイプ: X'12' (不明) アラート記述コード: X'FE00' (判別不能エラー)
推定原因サブベクトル	推定原因コード・ポイント: X'1001' (アプリケーション・プログラム)
原因判別不能サブベクトル	推奨アクション・コード・ポイント: X'0700' (アクション不要)

表 14. 総称アラート・サブベクトル (続き)

サブベクトル	説明
第 1 プロダクト・セット ID サブベクトル	プロダクト種別: X'xC' (非 IBM ソフトウェア) ソフトウェア・プロダクトの共通名: NMG API 全体で通信する NMG アプリケーション (非 SNA ネットワーク内) の ID。 ソフトウェア・プロダクトの共通レベル: 000000 ソフトウェア・プロダクトのプログラム番号: USER0 **注:** 第 1 プロダクト・セット ID サブベクトルは、 SNA には準拠しても、有効な情報をともなわないときに組み込まれます。
第 2 プロダクト・セット ID サブベクトル	プロダクト種別: X'xC' (非 IBM ソフトウェア) ソフトウェア・プロダクトの共通名: コマンドを受け取るネイティブ・エレメント・マネージャーの名前 ソフトウェア・プロダクトの共通レベル: 000000 ソフトウェア・プロダクトのプログラム番号: USER0 **注:** 第 2 プロダクト・セット ID サブベクトルは、 SNA には準拠しても有効な情報を伴わないときに組み込まれます。
日付/時刻サブベクトル (オプション)	日付および時刻情報が入っている X'01' サブベクトル
階層リソース・リスト・サブベクトル	第 1 リソース名 (強制): サービス・ポイントの名前 第 1 リソース・タイプ ID (強制): X'81' (サービス・ポイント) トランザクション・プログラム・リソース (オプション): トランザクション・プログラム ID (オプション): X'18' (トランザクション・プログラム) 追加のリソース名 (オプション): ドメインを個別に識別するときに、必要に応じて 追加のリソース・タイプ ID (オプション): 任意 **注:** サービス・ポイントから始まり、名前と名前の間の区切り文字としてピリオド (.) を用いるリソース名の連結は、RODM Non_SNA_Domain_Class オブジェクトにあるオブジェクトの MyName フィールドと同一である必要があります。

表 14. 総称アラート・サブベクトル (続き)

サブベクトル	説明
自己定義テキスト・メッセージ・サブベクトル	テキスト・メッセージ: INIT[,GMT=*chhmm*] オプションの GMT キーワード・パラメーターは、状況情報が入ったすべてのアラートおよびコマンド応答のグリニッジ標準時 (GMT) に対するオフセットを記述します。キーワード値は、以下のように形式化されます。 *c* は GMT 時刻修飾コード、+、-、もしくは Z です。 • GMT 修飾子をローカル時刻に加算するときは + を指定します。 • GMT 修飾子をローカル時刻から減算するときは - を指定します。 • ローカル時刻がすでに GMT である場合は Z を指定します。この場合、*hhmm* は 0000 です。 *hhmm* は時および分の GMT 修飾子です。 • *hh* の 24 時間形式による有効な値の範囲は、00 から 23 までです。 • *mm* の有効な分の範囲は、00 から 59 までです。

セッション終了

図 24 は、セッション終了時のアラートの交換を示しています。

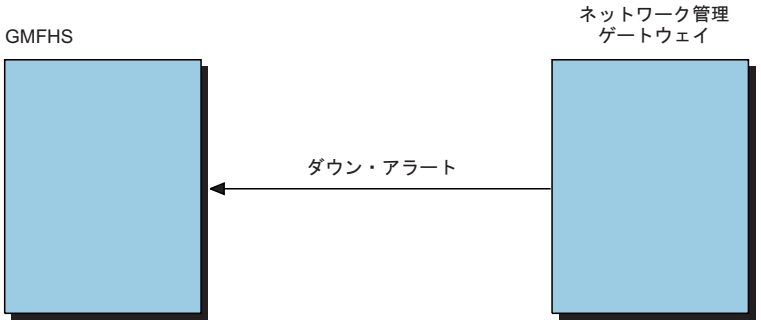


図 24. セッション終了

注: セッション終了アラートは、自己定義テキスト・メッセージ・サブベクトルにテキスト DOWN が含まれる点を除き、 92 ページの『セッションの確立の INIT 総称アラート』で説明されているアラートと同じです。

GMFHS は、このアラートを受け取ると、セッションがダウンしていると見なし、セッションが再び確立されるまで NMG にコマンドを送りません。

GMFHS は、次のいずれかの理由でダウン状態を検出したときも、セッションを終了します。
- NMG の状況が不良に変更になる。
- アラートが、エレメント・マネージャーの不良への状況の変更を報告する。
- GMFHS が、エレメント・マネージャーからの INIT アラートを受け取る。

INIT アラートを受け取ると、セッションは終了し、ただちに再確立されます。

非 SNA トランスポート・プロトコルを定義する

トランスポート・プロトコル定義は、ネットワーク制御コマンドがその非 SNA リソースの宛先に転送される方法を制御します。定義するトランスポート・プロトコルによっては、ワークステーションでコマンドを出して非 SNA リソースを制御することができます。

トランスポート・プロトコル・フィールドでは、GMFHS が、コマンドを送りコマンドへの応答を受け取る際に、ネットワーク管理ゲートウェイ (NMG) と通信する方法を指定します。以下は、有効なプロトコル名です。

- COS は、NMG がサービス・ポイントであり、かつ GMFHS はサービス・ポイントと通信する際に RUNCMD を使用する必要があることを示します。
- PPI は、NMG がプログラム間インターフェース (PPI) を使用し、かつ GMFHS は、NetView 管理コンソールと通信するフォーカル・ポイント・ホストの他のアドレス・スペースで実行するシステムまたはネットワーク管理トランザクション・プログラムと通信するときは、 PPI を使用する必要があることを示します。
- OST は、NMG が NetView プログラムであり、かつコマンドは NetView OST に送られることを指定します。
- NONE は、この NMG がコマンドを受け入れないことを指定します。

注: NMG がサービス・ポイントを表す場合、その名前はサービス・ポイントの SNA 名でなければなりません。 NMG が PPI を使用する際のその名前は、NMG が使用する PPI 受信側 ID でなければなりません。 NMG が OST ならば、その名前は 1 から 8 文字の任意の名前にすることができます。

COS ゲートウェイ・サポート

NetView 共通操作サービス (COS) ゲートウェイ・サポートは、RUNCMD コマンドを用いて、分散ホストの中央側 SSCP もしくはリモート SSCP が所有するサービス・ポイントとの間で、ネットワーク制御コマンドを送ったりコマンド応答を受け取ったりします。これらのサービス・ポイントは、 NetView プログラムのサービス・ポイント・コマンド・サービス (SPCS) からアクセスを受けるため、GMFHS は、この通信用の VTAM の通信ネットワーク管理インターフェース (CNMI) を直接使用しません。

ネットワーク制御コマンドを出すと、トランスポート層は、ネットワーク管理ゲートウェイ (NMG) オブジェクトの TransportProtocolName フィールドを検査します。フィールド値が COS ならば、GMFHS ホストは、コマンドを NetView アドレス・スペースで実行する GMFHS 有効範囲検査プログラム OPT に送ります。有効範囲検査プログラムは、コマンドを別々の自動タスクで実行する GMFHS COS コマンド処理プログラムに渡します。 COS コマンド処理プログラムは、コマンドのコンテキスト情報をいくつか保存し、そのコマンドが入った RUNCMD コマンドを作成して、コマンドを発行します。 RUNCMD コマンドへの応答は、GMFHS COS コマンド処理プログラムにより受け取られ、未解決のコマンドに相関させられて、GMFHS に戻されます。コマンド・リストは、RUNCMD コマンドを出し、それに対する応答を得ます。すべての応答が使用可能ならば、それらは COS コマンド処

理プログラムに戻されます。コマンド処理プログラムは、応答を、コマンド処理プログラムが保存しているコマンド・コンテキストに相関させ、応答を GMFHS に戻します。

サービス・ポイントが分散 NetView システムに常駐する場合、 COS コマンド処理プログラムは MS トランスポートを用いてコマンドを LU 6.2 セッションで経路指定します。 NetView プログラムはコマンドを分散 NetView システムに経路指定し、コマンドを分散ルーター自動タスクで実行して、 COS コマンド処理プログラムへの応答の送達が行われる中央設置場所の NetView プログラムに、応答を戻します。コマンド応答は、ローカル・サービス・ポイントからの応答の場合と同じように、GMFHS に戻されます。

COS トランスポート・プロトコルを使用するときは、 TransportProtocolName フィールドの値をそのゲートウェイの NMG_Class オブジェクトの COS に設定してください。

NetView プログラムが LU 6.2 を用いてサービス・ポイントと通信し、サービス・ポイント LU の NETID が、 RUNCMD を出す NetView プログラムと異なる場合は、NMGCharacteristics フィールドのビットで、 SNA ネットワーク名が RUNCMD の `NETID=` キーワード・パラメーターに組み込まれることを指定する必要があります。

NetView プログラムが SSCP-PU セッションを用いてサービス・ポイントと通信し、 RUNCMD を出す NetView プログラムがサービス・ポイント PU と通信する CNMI を所有していない場合は、 CNMI を所有する NetView プログラムのドメイン名を、サービス・ポイントの NMG_Class オブジェクトの CommandRouteLUName フィールドで指定します。

プログラム間インターフェース・ゲートウェイ

ゲートウェイ・トランスポート用のプログラム間インターフェース (PPI) を使用すると、 GMFHS もしくは NetView のアドレス・スペース以外のアドレス・スペースの処理が、 GMFHS から総称ネットワーク・コマンドおよびネイティブ・ネットワーク・コマンドを受け取り、コマンド応答を戻すことができます。 PPI トランスポート・タイプを使用するときは、 PPI の TransportProtocolName フィールド値を指定した NMG オブジェクトを定義します。この NMG オブジェクトの MyName フィールドは、 GMFHS がこのゲートウェイのコマンドを送る先の PPI の受信側名でなければなりません。

プログラム間インターフェースを用いて交換されるメッセージは、以下の場合を除き、実行主ベクトルおよび応答実行間主ベクトルを使用します。

- DomainCharacteristics フィールドで、コマンド応答がネイティブ・エレメント・マネージャーから受け取られると指定する場合は、実行主ベクトルにサポート・データ相関 MS 共通サブベクトルを組み込む必要があります。サポート・データ相関サブフィールドの PCID には、コマンド相互関係子が入っています。
- GMFHS が実行コマンドを送れない場合は、センス・データ・サブベクトルに、PPI の送信要求が失敗した理由を説明する PPI 戻りコードが入ります。 PPI 戻りコードについては、「*IBM Tivoli NetView for z/OS アプリケーション・プログラマーズ・ガイド*」を参照してください。

OST/PPT ゲートウェイ

NetView OST/PPT には、コマンドの発信元であるワークステーションに関連する NetView オペレーター・ステーション・タスク (OST) を使用するか、あるいは関連するワークステーション・オペレーターがいない場合は、基本プログラム・オペレーター・インターフェース (POI) タスク (PPT) を使用して、ネットワーク制御コマンドを出すことができる、ゲートウェイ・トランスポート機能があります。
NetView コマンド・リストおよびコマンド処理プログラムは、ワークステーションのオペレーターが入力したコマンドに応答して開始されます。このゲートウェイには、次の特性が有効です。

- DomainCharacteristics フィールドの応答予期ビットがオンであっても、コマンド応答を作成しない一部の OST/PPT コマンド。
- このゲートウェイにより開始されるコマンド・リストまたはコマンド処理プログラムは、 NetView GENALERT 機能を用いて現行のリソース状況または結果のリソース状況を報告し、それがビューに反映されるようにすることができます。この機能により開始されるコマンドが、それ以外ではターゲット・リソースにアラートが生成される変更を起こさせる場合は、 GENALERT を使用する必要はありません。

非ネットワーク装置をモニターする

NetView プログラムを用いると、ライン・プリンターなどの非ネットワーク装置をモニターすることができます。総称アラートを生成する GENALERT コマンドを出すコマンド・リストを書くことができます。非ネットワーク装置を表す RODM 実リソースの名前、およびそれらの装置について報告する RODM 非 SNA ドメイン・オブジェクトの名前を、それらが GENALERT アラート・リソース階層が使用する命名規則に従うように定義します。

NMG のタイプ

GMFHS は、3 つのタイプの NMG と通信することができます。

- 共通操作サービス NMG
- オペレーター・ステーション・タスク NMG
- プログラム間インターフェース NMG

NMG のタイプは、NMG_Class オブジェクトの TransportProtocol フィールドによって判別されます。 NMG が管理するドメインは、すべて同じタイプでなければなりません。

共通操作サービス NMG

GMFHS は、NetView RUNCMD コマンドにより共通操作サービス (COS) NMG と通信します。ネットワーク・コマンド・マネージャー・タスクは、表示プロトコルおよびセッション・プロトコルに従ってコマンド・テキストを作成してから、 COS ゲートウェイ・コマンド処理プログラム自動タスクを用いて RUNCMD コマンドを出し、応答を待ちます。 RUNCMD についての詳細は、 NetView のオンライン・ヘルプを参照してください。

COS NMG には、以下の利点があります。

- GMFHS は、コマンド応答を受け取ることができる。

- 表示プロトコルによっては、コマンド応答にネットワーク・コマンド・マネージャー・タスクが解釈できる状況情報を含めることができる。
- 現行サービス・ポイント・アプリケーションのいくつかが、このアーキテクチャーに従っている。
- オペレーター開始コマンドへの応答が、 Non-SNA Command Response (非 SNA コマンド応答) ウィンドウで表示される。

COS NMG に対するコマンドの最大サイズは、240 バイトです。表示プロトコル・コマンドまたはセッション・プロトコル・コマンドのコマンド・テキストの長さが、コマンド変数の置換後に 240 バイトを超えると、GMFHS はコマンドを拒否します。

オペレーター・ステーション・タスク NMG

GMFHS は、要求オペレーターの OST、または GMFHS 開始コマンドの PPT にコマンドを送って、オペレーター・ステーション・タスク (OST) NMG と通信します。ネットワーク・コマンド・マネージャー・タスクは、表示プロトコルおよびセッション・プロトコルに従ってコマンド・テキストを作成してから、ホスト・タスク・マネージャー OPT のメッセージ・キューイング・サービスを使用してコマンドをオペレーターの OST または PPT に送信します。 GMFHS は、OST コマンド応答を解釈できないので、状況変更はすべてアラートとして GMFHS に報告されなければなりません。

OST NMG に対するコマンドの最大サイズは、256 バイトです。表示プロトコル・コマンドまたはセッション・プロトコル・コマンドのコマンド・テキストの長さが、コマンド変数の置換後に 256 バイトを超えると、GMFHS はコマンドを拒否します。

プログラム間インターフェース NMG

GMFHS は、プログラム間インターフェースに登録済みの他のアプリケーションと情報を交換して、プログラム間インターフェース NMG と通信します。コマンドは、実行コマンド主ベクトル (X'8061') 内で形式化されます。コマンド応答は、2 つの応答主ベクトル (X'0061' および X'1300') に戻されます。ネットワーク・コマンド・マネージャー・タスクは、表示プロトコルおよびセッション・プロトコルに従ってコマンド・テキストを作成し、それをプログラム間インターフェースを経てエレメント・マネージャーに送ります。エレメント・マネージャーは、プログラム間インターフェースを経て GMFHS に応答します。

プログラム間インターフェース NMG には、以下の利点があります。

- GMFHS は、コマンド応答を受け取ることができる。
- 表示プロトコルによっては、コマンド応答にネットワーク・コマンド・マネージャー・タスクが解釈できる状況情報を含めることができる。
- オペレーター開始コマンドへの応答が、 Non-SNA Command Response (非 SNA コマンド応答) ウィンドウで表示される。

プログラム間インターフェース NMG に対するコマンドの最大サイズは、253 バイトです。表示プロトコル・コマンドまたはセッション・プロトコル・コマンドのコマンド・テキストの長さが、コマンド変数の置換後に 253 バイトを超えると、GMFHS はコマンドを拒否します。

PPI コマンド・トランスポート・エンベロープ

GMFHS コマンドのテキストは、実行コマンド主ベクトル (X'8061') のプログラム間インターフェース NMG にトランスポートされます。 この主ベクトルについては、「*System Network Architecture Formats*」で説明しています。しかし、GMFHS にはコマンド応答に相互関係子がなければならず、かつ実行コマンド主ベクトルのアーキテクチャーに相互関係子サブベクトルが組み込まれていないため、 GMFHS は相互関係子を含むサブベクトルを組み込むことで、このアーキテクチャーから離れます。この追加の相互関係子は、サポート・データの相関サブベクトル (X'48') です。

表 15 は、実行コマンド主ベクトルに組み込まれたサブベクトルおよびサブフィールドを示しています。

表 15. 実行コマンド主ベクトルのサブベクトルおよびサブフィールド

サブベクトル	サブフィールド	説明
名前リスト	宛先アプリケーション名	Non_SNA_Domain_Class オブジェクトの TransactionProgram フィールドの値
自己定義テキスト・メッセージ	コード化文字セット ID	X'00000037'
自己定義テキスト・メッセージ	テキスト・メッセージ	表示層によって作成されるコマンド・テキスト
サポート・データ相関	完全修飾セッション PCID	PCID: GMFHS 内部相互関係子 ネットワーク修飾 CP 名: GMFHS.NETCMD

コマンド応答は、以下の 2 つの主ベクトルから構成されます。
- 実行コマンドへの応答
- テキスト・データ・パラメーター

GMFHS は、応答実行コマンド間主ベクトルのすべてのサブベクトルを無視します。サブベクトルは不要です。表 16 に、テキスト・データ・パラメーター主ベクトルのサブベクトルおよびサブフィールドを示します。

表 16. テキスト・データ・パラメーター主ベクトルのサブベクトルおよびサブフィールド

サブベクトル	サブフィールド	説明
サポート・データ相関	完全修飾セッション PCID	コマンドのサブベクトルと同じでなければなりません。 PCID: GMFHS 内部相互関係子 ネットワーク修飾 CP 名: GMFHS.NETCMD
自己定義テキスト・メッセージ	テキスト・メッセージ	コマンド応答テキスト
自己定義テキスト・メッセージ	他のサブフィールド	GMFHS は、このサブベクトルの他のサブフィールドをすべて無視します。

NETCENTER プロトコルから GMFHS プロトコルにマイグレーションする

GMFHS が使用するプロトコルは、NETCENTER が使用するプロトコルに類似しています。 表 17 は、NETCENTER プロトコルに指定された値と、 GMFHS プロトコルに指定できる対応する値を示しています。

表 17. NETCENTER から GMFHS への定義名の変換

フィールド	NETCENTER	GMFHS
SessionProtocolName	NSI1	DOMS010
	PASSTHRU	PASSTHRU
	NONE	NONE
PresentationProtocolName	NSI1	DOMP010
	相当するものなし	DOMP020
	PASSTHRU	PASSTHRU
	NONE	NONE
TransportProtocolName	CNMI	COS
	相当するものなし	PPI
	相当するものなし	OST
	NONE	NONE

表 18 は、プロトコルの指定に使用する NETCENTER 属性の名前、およびプロトコルを指定する際に使用する対応 GMFHS フィールドの名前です。

表 18. NETCENTER 属性名から GMFHS フィールド名への変換

プロトコル	NETCENTER 属性	GMFHS フィールド	GMFHS オブジェクト
セッション	SESS	SessionProtocolName	非 SNA ドメイン
表示	FORMAT	PresentationProtocolName	非 SNA ドメイン
トランスポート	TRAN	TransportProtocolName	ネットワーク管理ゲートウェイ

以下は、NETCENTER NSI1 プロトコルと GMFHS DOMP010 プロトコルとの相違点です。

- GMFHS は、24 時間ごとに (DomainCharacteristics フィールドの内容により異なる) 条件付きで、 SET_CLOCK プロトコル・コマンドをエレメント・マネージャーに送ります。
- GMFHS は、すべてのコマンドの送信側 ID キーワード (SN=GMFHS) を組み込んでいます。
- GMFHS は、GMFHS DisplayStatus 値に略号によって関連する新規の状況値を認識します。 GMFHS は、これらの値が DomainCharacteristics フィールドで指定されている場合は、 NETCENTER タイプ状況をそのいずれかに変換します。
- GMFHS は、NETCENTER 総称 Enable (使用可能) コマンドおよび Disable (使用不能) コマンドをサポートしていません。
- GMFHS の場合、リソース名は (使用するゲートウェイによって) 可能な限り多くの文字を使用することができます。 NETCENTER は、リソース名を最大 8 文字に限定します。

- GMFHS の場合は、TX テキスト・ストリングで、) 文字 (右括弧) を使用することができます。詳細については、 85 ページの『テキスト − TX』を参照してください。

第 5 章 GMFHS が RODM を使用する方法

グラフィック・モニター機能ホスト・サブシステム (GMFHS) は、 RODM および NetView 管理コンソール (NetView 管理コンソール) とともに作動して、ネットワークのグラフィック・ビューを表示し、ビューから選んだリソースにコマンドを出します。ビューには、ネットワーク・リソースに関する状況情報と構成情報の両方が含まれます。この章では、GMFHS が RODM を使用する方法を説明します。この情報を用いて、RODM の内容を修正し、 GMFHS および NetView 管理コンソール の実行方法を変更することができます。

GMFHS の初期化

GMFHS は、次の 2 つのオプションのいずれかで開始することができます。
- 集約ウォーム・スタート
- リソース状況ウォーム・スタート

デフォルトでは、オプションは実行されず、GMFHS は正常に開始されます。

集約ウォーム・スタート

集約ウォーム・スタートは、 GMFHS 始動プロシージャー CNMGMFHS に AGGRST=YES パラメーターをコーディングすることにより行います。オブジェクト独立メソッド、DUIFFAWS は、 RODM データ・キャッシュの実オブジェクトおよび集合オブジェクトの状況集約に関連するフィールドを初期化するときに実行します。詳細については、 565 ページの『DUIFFAWS: 集約ウォーム・スタート・メソッド』を参照してください。

リソース状況ウォーム・スタート

リソース状況ウォーム・スタートは、 GMFHS 始動プロシージャー CNMGMFHS に RESWS=YES パラメーターをコーディングすることにより行います。

リソース状況ウォーム・スタートには、 GMFHS をすみやかに復元するメカニズムがあります。 GMFHS は異常終了しても、GMFHS に管理された RODM のリソースの状況が依然正しい場合は、リソース状況ウォーム・スタート・オプションを使用します。 GMFHS は、すべてのドメイン・リソースについての正常なリソース状況初期化処理をバイパスし、代わりに RODM 内の既存の状況情報を使用します。

GMFHS はドメイン単位にリソースの状況を設定します。リソース状況ウォーム・スタートが起こるには、ドメインは次のいずれかの条件を満たさなければなりません。
- GMFHS の最後の初期化時に、リソースの状況請求が正常に完了した。
- 状況請求がサポートされていない。
- 状況請求スキップが示されている。

リソース状況ウォーム・スタートには、RODM の現行の状況データが必要です。現行状況を RODM で確実に保守するには、現行ドメインとリソース値の保存に

RODM の定期的なチェックポイントが必要です。この場合、RODM はそれまでのチェックポイント・データが入ったデータ・セットを用いてロードすることができます。

注:

1. RODM の最後のチェックポイントと GMFHS を再初期化した時点の間の、すべての状況更新が失われます。
2. GMFHS および RODM がバックアップ・ホストでウォーム・スタートされる場合、チェックポイント・ファイルが入った DASD はバックアップ・ホストからアクセスできなければなりません。

GMFHS 初期化処理の概要

正常な GMFHS の初期化には、次の 2 つのサブプロセスがあります。

- 設定
- セッションの確立

これらのサブプロセスで、非 SNA ドメインごとのリソースの初期状況を判別します。しかし、一定の環境下では GMFHS はこれらのステップを実行しません。この判別は、以下の GMFHS 開始オプションと RODM フィールドの値によって行われます。

- GMFHS ウォーム・スタート・オプション (resws=yes|no)
- NMG_Class オブジェクトに定義される AgentStatus フィールド
- NMG_Class オブジェクトに定義される AgentStatusEffect フィールド
- Non_SNA_Class オブジェクトに定義される DomainCharacteristics フィールド
- Non_SNA_Class オブジェクトに定義される DomainCharacteristics2 フィールド

設定サブプロセス

各ドメインのもとのリソースは、次の条件下の場合を除き、初期状況、もしくは不明状況に設定されます。

- GMFHS がリソース状況ウォーム・スタート・オプション (resws=yes) で開始し、 DomainCharacteristics2 フィールドで状況完了ビットを**オン**にする。
- DomainCharacteristics フィールドの状況設定スキップ・ビットを**オン**にする。

セッションの確立サブプロセス

状況請求がサポートされていれば、各ドメイン内のリソースの状況が請求されます。状況請求の詳細については、71 ページの『第 4 章 ネットワーク管理ゲートウェイと通信する』を参照してください。

GMFHS がリソース状況ウォーム・スタート・オプション (resws=yes) で開始し、DomainCharacteristics2 フィールドの状況完了ビットを**オン**にすると、 GMFHS はドメインのセッションの確立サブプロセスを実行しません。しかし、GMFHS がリソース状況ウォーム・スタート・オプション (resws=yes) で開始し、DomainCharacteristics2 フィールドの状況完了ビットを**オフ**にすると、 GMFHS はドメインのセッションの確立サブプロセスを実行します。

状況請求がドメインにサポートされていない場合、リソース状況は次の条件によって設定されます。

- AgentStatusEffect フィールドの値が X'80' であって、 DomainCharacteristics2 フィールドで状況完了ビットを**オン**にすると、リソースの状況は変わらない。
- AgentStatusEffect フィールドの値が X'80' であって、 DomainCharacteristics2 フィールドで状況完了ビットを**オフ**にすると、次のようになる。
  - AgentStatus フィールドの値が 1 か 3 ならば、リソースの状況は、InitialResourceStatus フィールドの値で示される値に設定される。
  - AgentStatus フィールドの値が 0 か 2 ならば、リソースの状況は不明 (Unknown) に設定される。
- AgentStatusEffect フィールドの値が X'00' ならば、リソースの状況は、InitialResourceStatus フィールドの値で示される状況に設定される。

トポロジー・マネージャーをモニターする

GMFHS は、トポロジー・マネージャーの状況をモニターし、その状況をオペレーターに示します。各トポロジー・マネージャーを表すには、 Topology_Manager クラスでオブジェクトを 1 つ作成します。 SNA トポロジー・マネージャーはこのオブジェクトを自動的に作成しますので、注意します。

Topology_Manager クラス・オブジェクトのフィールドを用いて、マネージャーごとに以下の指定を行うことができます。

- その状況
- GMFHS が各マネージャーが使用不能と見なすまでの間隔 (この間にマネージャーがその状況を示しておかなければならない)
- そのコマンド・インディケーターの範囲

マネージャーごとにそのオブジェクトの StatusIndicator フィールドを定期的に更新して、それがアクティブにあることを GMFHS に知らせる必要があります。このフィールドが StatusInterval フィールドで指定された間隔内に更新されないと、GMFHS はマネージャーが使用不能であると報告します。トポロジー・マネージャーの状況は、NetView 管理コンソール の状況域に表示され、オープン・ビューの状況バーに要約されます。

ビューを作成する

GMFHS は、すべてのビューを次の 2 ステップ処理を用いて作成します。

- オブジェクトの展開
- オブジェクトの接続

オブジェクトの*展開* は、ビューに表示するオブジェクトのリストの判別に使用される処理です。この処理は、要求されるビューのタイプによって異なります。

オブジェクトの*接続* は、リストのオブジェクトがビューで相互接続される方法を決めるのに使用する処理です。この処理は、ビューの各タイプの処理と同じです。 この処理の説明については、 118 ページの『オブジェクトの接続処理』を参照してください。

オブジェクトの展開処理

GMFHS が作成するビューは、すべて次の 2 つのカテゴリーに分類することができます。

- 事前定義
- 動的作成

事前定義ビュー

事前定義ビューは、RODM のビュー・オブジェクトによって表されます。ビュー・オブジェクトには、ビューに存在する各リソースへのリンクが含まれています。唯一必要なオブジェクトの展開処理は、現在ビュー・オブジェクトにリンクしているオブジェクトのリストの照会です。例外ビューのオブジェクトはリンクされていませんので、注意します。

動的に作成されたビュー

動的に作成されたビューは RODM 内のビュー・オブジェクトによっては表されません。動的に作成されたビューは、オープン・ビュー上のオブジェクトを選択してからそのオブジェクトに対するアクションを起こすか、または特定のオブジェクトに対してリソース位置指定要求を出すことによって選択します。いずれの場合も、GMFHS は要求を受信し、ビューの作成に必要なオブジェクトのセットを検出するために、指定されたオブジェクトの照会するフィールドを判別します。照会すべきフィールドは、ビューのタイプにより異なります。

動的に作成されたビューによっては、GMFHS は、ビューに表示するオブジェクトの全リストを判別するのに再帰的処理を使用することもあります。例えば、オブジェクトについての構成親ビューが要求されると、 GMFHS はオブジェクトの親を判別します。次に、この親に親があるかどうかを判別します。親のない親が見つかるまで、この処理が繰り返されます。詳細については、134 ページの『再帰的ビューを制限する』を参照してください。この処理を使用するビューについては、111 ページの『特定ビューの場合のオブジェクトの展開処理の説明』で記載しています。

次のオブジェクトには、ビュー作成処理における重要な役割があります。

- Display_Resource_Type_Class オブジェクト
- View_Information_Object_Class オブジェクト

以降で、これらのオブジェクトの概要説明を行い、111 ページの『特定ビューの場合のオブジェクトの展開処理の説明』では、これらのオブジェクトを各タイプのビューに使用する方法を説明します。

Display_Resource_Type_Class オブジェクト: Display_Resource_Type_Class オブジェクトは、リソースを表示する際にそれにアイコンを関連付けるときに使用します。ビューに入れることができる表示可能オブジェクトは、Display_Resource_Type_Class のオブジェクトにリンクされていなければなりません。表示可能オブジェクトの Display_Resource_Type_Class オブジェクトへのリンクは、以下の図で説明と図解を行っている 2 つの方法で行うことができます。

注: 表示可能オブジェクトは、両方の方法で Display_Resource_Type_Class オブジェクトにリンクすることができます。 GMFHS がこの状態を検出すると、107 ページの図 25 に見られる技法が使用されます。

NetView バージョン 3 より前は、リンクを行うのにメソッド DUIFCLRT が実行されるのが普通でした。 図 25 に見られるように、DUIFCLRT は、表示可能オブジェクトの DisplayResourceType フィールドを、 Display_Resource_Type_Class オブジェクトの Resources (リソース) フィールドにリンクします。この場合の欠点は、オブジェクトごとにこのメソッドを実行しなければならない点です。

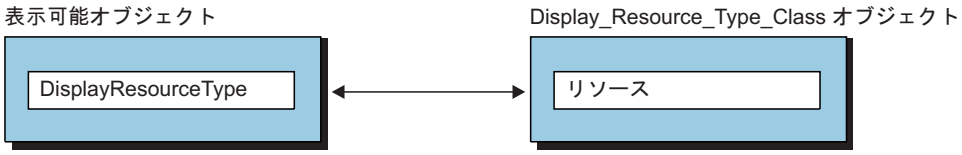


図 25. NetView バージョン 3 の前までの Display_Resource_Type_Class オブジェクトのリンクの技法

現在では、図 26 に示すように、 Display_Resource_Type_Class オブジェクトを RODM のオブジェクト・クラスに関連付けることができるようになっています。これは、View_Information_Reference_Class オブジェクトを作成し、そのオブジェクト ID をオブジェクト・クラスの ViewInfoRefObjDRT フィールドに入れることで行います。 View_Information_Reference_Class オブジェクトの DisplayResourceType フィールドは、次にメソッド DUIFCLRT を用いて、 Display_Resource_Type_Class オブジェクトの ResourceClasses フィールドにリンクさせます。リンクはクラス・レベルで定義できないため、 View_Information_Reference_Class オブジェクトが使用されます。 ViewInfoRefObjDRT フィールドは、クラスのすべてのオブジェクトから継承されます。この技法の利点は、リンクの定義をオブジェクトごとに個々に行うのではなく、クラス・レベルで一度だけ行えばよいという点です。

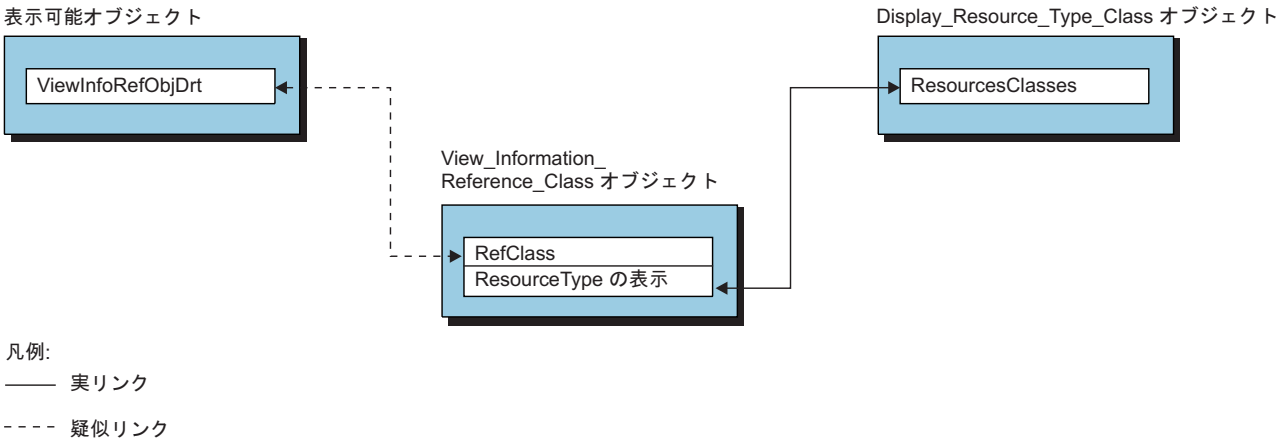


図 26. Display_Resource_Type_Class オブジェクトの新しいリンク技法

View_Information_Object_Class オブジェクト: GMFHS は、View_Information_Object_Class オブジェクトを以下の目的に使用します。

- いくつかの動的に作成されたビューの作成時に、他のすべての関連オブジェクトを探す際に照会するオブジェクトのフィールドを判別する。
- ビューのオブジェクトを接続する方法を判別する。詳細については、 118 ページの『オブジェクトの接続処理』を参照してください。

しかし、GMFHS は、両方の目的で使用する View_Information_Object_Class オブジェクトを判別するのに、共通技法を使用します。 GMFHS が定義するあらゆるリソース・タイプとビュー・タイプの対に対して、 View_Information_Object_Class オブジェクトが 1 つあります。すべてのリソース・タイプは、最終的には、それらを表示できるビューのタイプを表す View_Information_Object_Class オブジェクトを指し示します。

すべてのビュー・タイプは、最終的には、特定タイプのビューで表示できるリソース・タイプを表す View_Information_Object_Class オブジェクトを指し示します。オブジェクト・タイプおよびビュー・タイプごとに、この組み合わせを表す有効な View_Information_Object_Class オブジェクトは 1 つだけあります。以下の 2 つの技法を使用して、リソースについて View_Information_Object_Class オブジェクト **A** を判別することができます。

1. 第 1 の技法は、 NetView バージョン 3 より前のバージョンで使用できた唯一の技法です。この技法で使用されるオブジェクトおよびフィールドについては、109 ページの図 27 で図解しています。
2. NetView バージョン 3 から、第 2 の技法が使用できます。この技法で使用されるオブジェクトおよびフィールドについては、 110 ページの図 28 で図解しています。

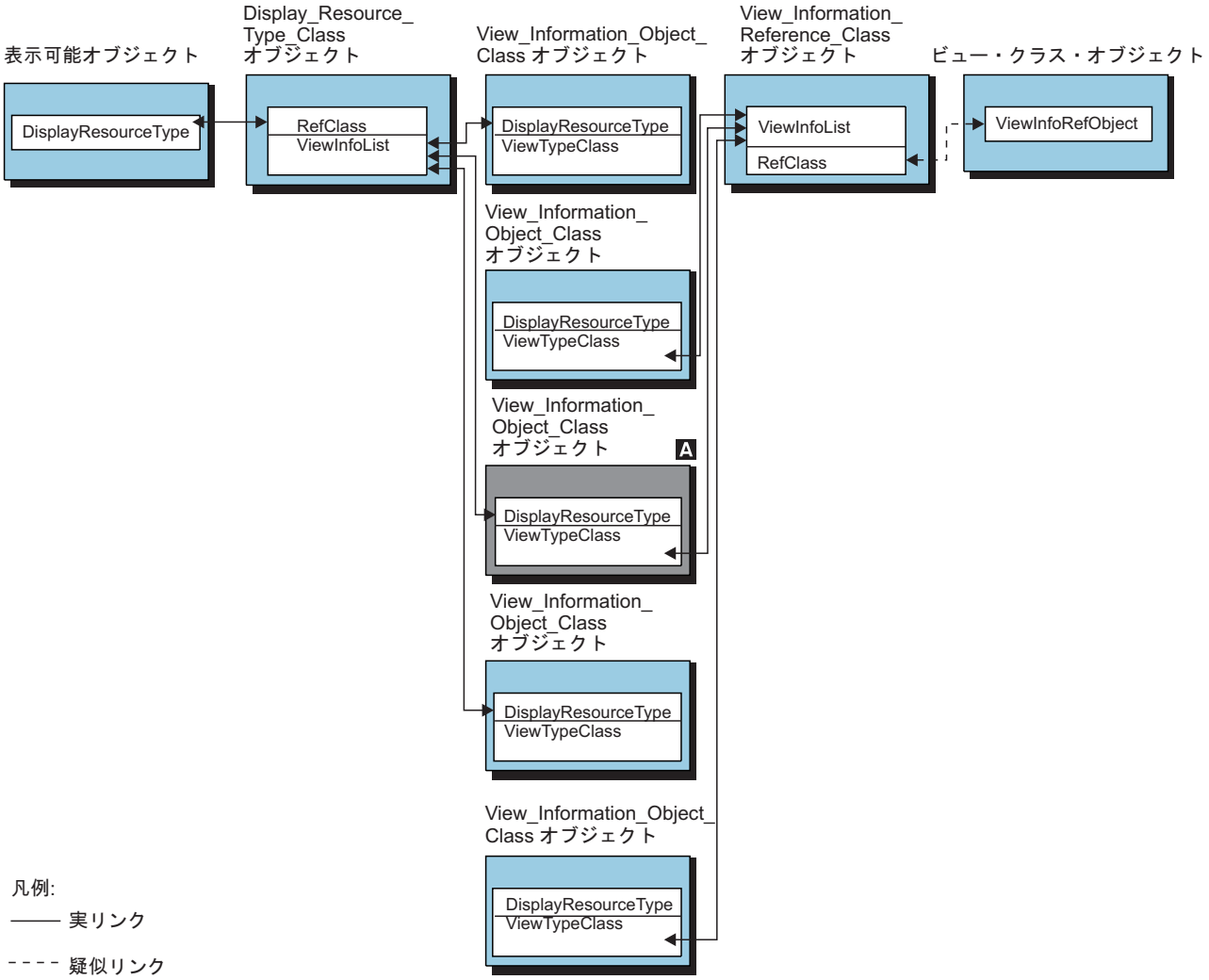


図 27. View_Information_Object_Class オブジェクトの判別技法 1

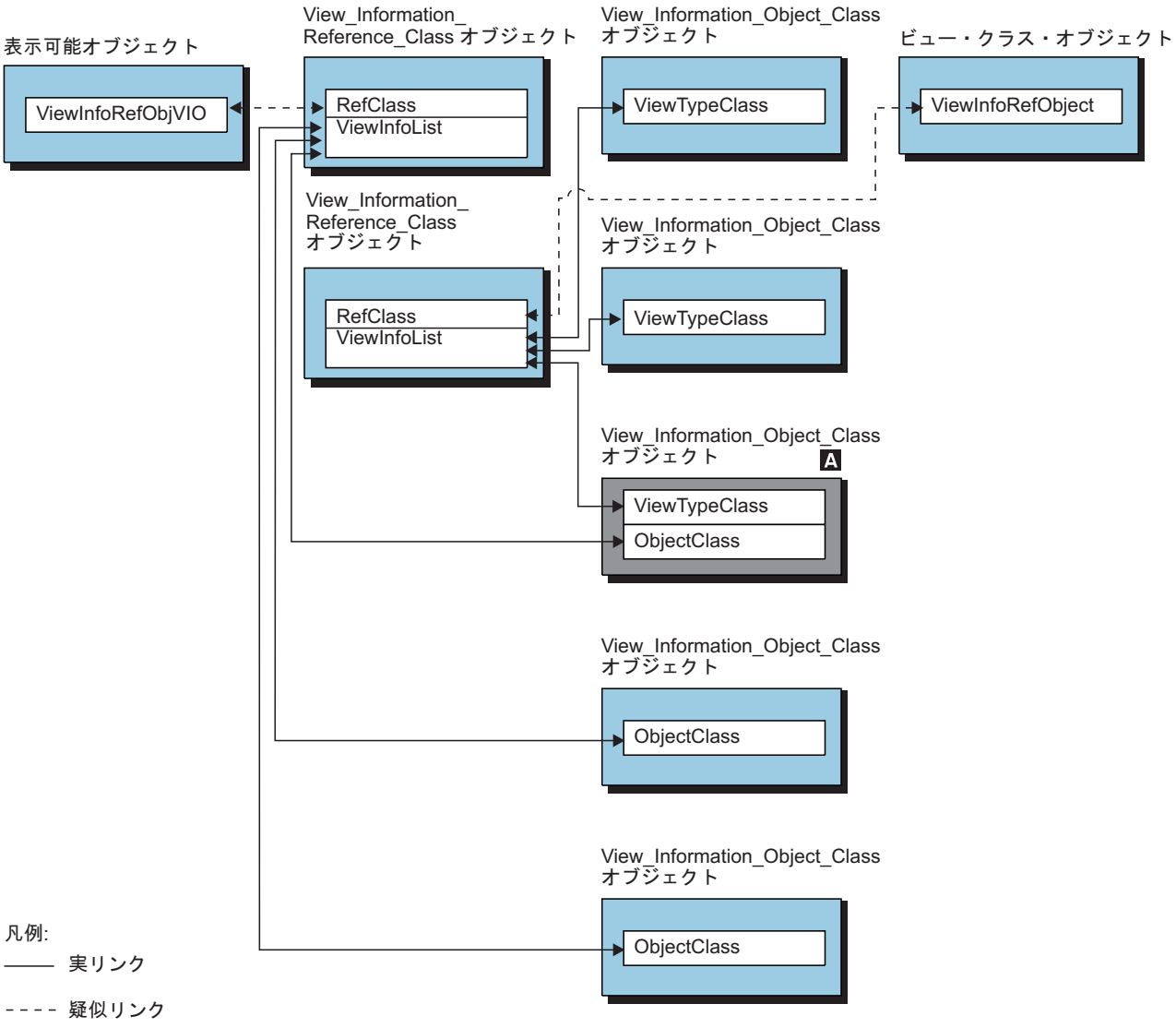


図 28. View_Information_Object_Class オブジェクトの判別技法 2

表示可能オブジェクトは、 DisplayResourceType フィールド (109 ページの図 27 参照) および ViewInfoRefObjVIO フィールド (図 28 参照) の両方を用いて、View_Information_Object_Class オブジェクトを指定することができます。 GMFHS は、この状態を検出すると、 ViewInfoRefObjVIO フィールドが指し示す View_Information_Object_Class オブジェクトを使用します。

GMFHS が、表示可能オブジェクトに有効な View_Information_Object_Class オブジェクトを見つけられない場合は、次の 2 つのシナリオのどちらかが起こっている可能性があります。

- オペレーターが、ルート・オブジェクトと呼ばれるリソース・オブジェクトに定義されていないビュー・タイプを選んだときに、 View_Information_Object_Class オブジェクトが見つからない。この場合、GMFHS は、このビュー・タイプにはこのタイプのオブジェクトが使用できない旨のメッセージを表示します。
- ビュー内にルート・オブジェクト以外のオブジェクトがあるはずであるのに、GMFHS がその View_Information_Object_Class オブジェクトを検出できない場合、 GMFHS はそのオブジェクトを省略して、ビューを作成します。 NetView

バージョン 3 より前のバージョンでは、 GMFHS がリソース・オブジェクトの View_Information_Object_Class オブジェクトを検出できない場合は、ビューを作成できませんでした。

特定ビューの場合のオブジェクトの展開処理の説明

このセクションでは、GMFHS がビューに組み込むオブジェクトの判別方法を説明します。ネットワーク・ビューおよび例外ビューは、それらを NMC ツリー・ビューから選んでオープンします。他のタイプのビューはすべて、ビュー名ではなくオブジェクトを選んでオープンします。

ビューごとに、以下の情報が表示されます。

- ビューが事前定義されたか、動的に作成されたか。ビューによっては、事前定義のものと動的に作成されたものの両方の場合がありますので、注意する必要があります。
- GMFHS がオブジェクトのすべてを展開するのに使用する論理の高水準の記述。
- オブジェクトの展開処理が使用するフィールド。

ネットワーク・ビュー: ネットワーク・ビューは、事前定義ビューです。各ビューは、RODM の Network_View_Class オブジェクトによって表されます。 NetView 管理コンソール サーバーが GMFHS とのセッションを確立するときに、このクラス下のあらゆるオブジェクトが照会され、 NMC ツリー・ビューに表示されます。ネットワーク・ビューの追加もしくは削除を行うたびに、このビューのリストは自動的に最新表示されます。リストに表示されるビューの名前は、Network_View_Class オブジェクトの MyName フィールドの値です。

ネットワーク・ビューがオープンされると、要求が GMFHS に渡されます。GMFHS は、Network_View_Class オブジェクトの ContainsObjects フィールドを照会します。戻されたオブジェクトのリストは、GMFHS の接続処理で使用されます。この処理の説明については、 118 ページの『オブジェクトの接続処理』を参照してください。

構成対等機能ビュー: 構成対等機能ビューは、事前定義ビューです。各ビューは、RODM の Configuration_Peer_View_Class オブジェクトによって表されます。構成対等機能ビューはネットワーク・ビューに似ていますが、以下のように、大きな相違点が 2 つあります。

- 構成ビューは、NMC ツリー・ビューでは使用できません。
- 構成ビューは、名前ではなく、オブジェクトで実行されます。

構成対等機能ビューがオープンされると、要求が GMFHS に渡されます。 GMFHS は、選択したリソース・オブジェクトの ContainedInView フィールドを照会します。このフィールドは、このリソースが現在定義されているあらゆる事前定義ビューを指し示します。これらのビュー・オブジェクトごとに、GMFHS はオブジェクトが作成されたクラスを検出して、そのビュー・タイプを判別します。Configuration_Peer_View_Class オブジェクトごとに、 GMFHS は、指定されたビュー・オブジェクトの ContainsObjects フィールドを照会して、ビューに入れるオブジェクトのリストを獲得します。戻されたオブジェクトのリストは、GMFHS の接続処理で使用されます。この処理の説明については、 118 ページの『オブジェクトの接続処理』を参照してください。

NMC により検出された障害のあるリソースのビュー: NMC により検出された障害のあるリソースのビューは、動的に作成されるビューです。これは、オープン・ビューで集合オブジェクトを選び、 NMC により検出された障害のあるリソースのビューを要求することにより作成されます。

NMC により検出された障害のあるリソースのビューがオープンされると、 NMC は GMFHS に要求を渡します。 GMFHS は、選択された集合オブジェクトの AggregationChild フィールドを照会して、すべての集合体子オブジェクト、および集合オブジェクトの実の子オブジェクトのリストを獲得します。集合体子オブジェクトごとに、 GMFHS はそのオブジェクトの AggregationChild フィールドを照会して、その子オブジェクトを獲得します。この処理は、GMFHS が元の集合体でのすべての実オブジェクトの全リストを得るまで繰り返されます。

GMFHS は、リスト内のすべての集合オブジェクト、および次の基準の*どれか* に合致する実オブジェクトを取り除きます。

- 例外状態にマップしない (ResourceTraits には NOXCPT が入っている)。
- オブジェクトが集合から中断状態にあることを示す UserStatus がある (UserStatus のビット 0x40 がオンになっている)。
- 集合が使用中でないことを示す AggregationPriorityValue がある (AggregationPriorityValue = -1)。

これらの基準に合致しないオブジェクトのリストは、GMFHS の接続処理に渡されます。この処理の説明については、 118 ページの『オブジェクトの接続処理』を参照してください。

障害のあるリソースに対する高速パスのビューをカスタマイズする: オブジェクトの DisplayStatus をオブジェクトの例外状態にマップする方法をカスタマイズすることで、 NMC により検出された障害のあるリソースのビューに表示するオブジェクトを決めることができます。表示状況を例外状態にマップする際の詳細については、 118 ページの『例外ビューのオブジェクトおよび基準を定義する』を参照してください。

構成子ビュー: 構成子ビューは、オープン・ビューでオブジェクトを選び、構成子ビューを選択することで要求される、動的に作成されたビューです。このビューは、選択したオブジェクトに定義されているすべての子をオペレーターに表示します。 GMFHS は、選択したオブジェクトの子のオブジェクトを探すのに次の処理を使用します。

- View_Information_Object_Class オブジェクトを探す。
- View_Information_Object_Class オブジェクトの RelFieldNamesA フィールドを照会する。基本的な GMFHS データ・モデルの場合では、このフィールドが ChildAccess フィールドを指定します。 RelFieldNamesA フィールドはユーザー修正可能であって、他の値が入っている可能性があるので、注意します。
- ChildAccess フィールドには、オブジェクトの子であるすべてのオブジェクトを指すポインターが入っている。

この処理は、子の全リストが識別されるまで、選択したオブジェクトの子オブジェクトごとに繰り返されます。オブジェクトのリストは、GMFHS の接続処理に渡されます。この処理の説明については、 118 ページの『オブジェクトの接続処理』を参照してください。

構成親ビュー: 構成親ビューは、オープン・ビューでオブジェクトを選び、構成親ビューを選択することで要求される、動的に作成されたビューです。このビューは、選択されたオブジェクト、中間の親への接続、および選択されたオブジェクトの最終の親への接続を表示します。 GMFHS は、選択したオブジェクトの親オブジェクトを探すのに次の処理を使用します。

- View_Information_Object_Class オブジェクトを探す。
- View_Information_Object_Class オブジェクトの RelFieldNamesA フィールドを照会する。基本的な GMFHS データ・モデルの場合では、このフィールドが ParentAccess フィールドを指定します。 RelFieldNamesA フィールドはユーザー修正可能であって、他の値が入っている可能性があるので、注意します。
- ParentAccess フィールドには、選択したオブジェクトの親のオブジェクトであるすべてのオブジェクトを指すポインターが入っている。

この処理は、親オブジェクトの全リストが識別されるまで、選択したオブジェクトの親オブジェクトごとに繰り返されます。オブジェクトのリストは、GMFHS の接続処理に渡されます。この処理の説明については、118 ページの『オブジェクトの接続処理』を参照してください。

構成論理ビュー: 構成論理ビューは、オープン・ビューでオブジェクトを選び、構成論理ビューを選択することで要求されます。このビューは、選択したオブジェクトと、それに論理的に接続されたすべてのリソース・オブジェクトを表示します。構成論理ビューは、動的に作成されたものであっても、事前定義されたものであってもかまいません。

動的に作成された構成論理ビューの場合、GMFHS は、次の処理を用いて、選択したオブジェクトに論理的に接続されたオブジェクトを探します。

- View_Information_Object_Class オブジェクトを探す。
- 基本的な GMFHS データ・モデルの以下のフィールドを照会する。
  - RelFieldNamesA。LogicalConnUpstream フィールドを指定する
  - RelFieldNamesB。LogicalConnDownstream フィールドを指定する
  - RelFieldNamesAB。LogicalConnPP フィールドを指定する

  RelFieldNamesA、RelFieldNamesB、および RelFieldNamesAB は、ユーザー修正可能であって、他の値が入っている可能性があるので、注意します。
- これらのフィールドには、選択したオブジェクトに論理的に接続されたオブジェクトを指すポインターが入っている。

この処理は、オブジェクトの全リストが識別されるまで、選択したオブジェクトに論理的に接続されたリソース・オブジェクトごとに繰り返されます。

定義済み構成論理ビューの場合、要求は GMFHS に渡されます。 GMFHS は、選択したリソース・オブジェクトの ContainedInView フィールドを照会します。このフィールドは、このリソースが現在定義されているあらゆる事前定義ビューを指し示します。これらのビュー・オブジェクトごとに、GMFHS はオブジェクトが作成されたクラスを検出して、そのビュー・タイプを判別します。
Configuration_Logical_View_Class オブジェクトごとに、 GMFHS は、指定されたビュー・オブジェクトの ContainsObjects フィールドを照会して、ビューに入れるオブジェクトのリストを獲得します。

動的に作成された構成論理ビューと定義済み構成論理ビューの両方の場合、オブジェクトのリストは GMFHS 接続処理に渡されます。この処理の説明については、118 ページの『オブジェクトの接続処理』を参照してください。

構成物理ビュー: 構成物理ビューは、オープン・ビューでオブジェクトを選んでから、構成論理ビューを選択することで要求されます。このビューは、選択したオブジェクトと、それに物理的に接続されたすべてのリソース・オブジェクトを表示します。構成物理ビューは、動的に作成することも事前定義することもできます。

動的に作成された構成物理ビューの場合、GMFHS は、次の処理を用いて、選択したオブジェクトに物理的に接続されたオブジェクトを探します。

- View_Information_Object_Class オブジェクトを探す。
- 基本的な GMFHS データ・モデルの以下のフィールドを照会する。
  - RelFieldNamesA。PhysicalConnUpstream フィールドを指定する
  - RelFieldNamesB。PhysicalConnDownstream フィールドを指定する
  - RelFieldNamesAB。PhysicalConnPP フィールドを指定する

  RelFieldNamesA、RelFieldNamesB、および RelFieldNamesAB は、ユーザー修正可能であって、他の値が入っている可能性があるので、注意します。
- これらのフィールドには、選択したオブジェクトに物理的に接続されたオブジェクトを指すポインターが入っている。

この処理は、オブジェクトの全リストが識別されるまで、選択したオブジェクトに物理的に接続されたリソース・オブジェクトごとに繰り返されます。

定義済み構成物理ビューの場合、要求は GMFHS に渡されます。 GMFHS は、選択したリソース・オブジェクトの ContainedInView フィールドを照会します。このフィールドは、このリソースが現在定義されているあらゆる事前定義ビューを指し示します。これらのビュー・オブジェクトごとに、GMFHS はオブジェクトが作成されたクラスを検出して、そのビュー・タイプを判別します。
Configuration_Physical_View_Class オブジェクトごとに、GMFHS は、指定されたビュー・オブジェクトの ContainsObjects フィールドを照会して、ビューに入れるオブジェクトのリストを獲得します。

動的に作成された構成物理ビューと定義済み構成物理ビューの両方の場合、オブジェクトのリストは GMFHS 接続処理に渡されます。この処理の説明については、118 ページの『オブジェクトの接続処理』を参照してください。

構成バックボーン・ビュー: 構成バックボーン・ビューは、オープン・ビューでオブジェクトを選び、構成バックボーン・ビューを選択することで要求されます。このビューは、サブエリア・バックボーンを表示します。構成バックボーン・ビューは、動的に作成されたものであっても事前定義のものであってもかまいません。

動的に作成された構成バックボーン・ビューの場合、GMFHS は、次の処理を用いて、選択したオブジェクトに関連するバックボーン・オブジェクトを探します。

- View_Information_Object_Class オブジェクトを探す。
- View_Information_Object_Class オブジェクトの RelFieldNamesA フィールドを照会する。基本的な GMFHS データ・モデルの場合では、このフィールドが

BackboneConnPP フィールドを指定します。 RelFieldNamesA フィールドはユーザー修正可能であって、他の値が入っている可能性があるので、注意します。

- BackboneConnPP フィールドには、 SNA バックボーンの一部であるすべてのオブジェクトを指すポインターが入っている。

この処理は、バックボーン・オブジェクトの全リストが識別されるまで、選択したオブジェクトに関連するバックボーン・オブジェクトごとに繰り返されます。

定義済み構成バックボーン・ビューの場合、要求は GMFHS に渡されます。GMFHS は、選択したリソース・オブジェクトの ContainedInView フィールドを照会します。このフィールドは、このリソースが現在定義されているあらゆる事前定義ビューを指し示します。これらのビュー・オブジェクトごとに、GMFHS はオブジェクトが作成されたクラスを検出して、そのビュー・タイプを判別します。Configuration_Backbone_View_Class オブジェクトごとに、 GMFHS は、指定されたビュー・オブジェクトの ContainsObjects フィールドを照会して、ビューに入れるオブジェクトのリストを獲得します。

動的に作成されたものと事前定義のものの両方の構成バックボーン・ビューの場合、オブジェクトのリストは、GMFHS 接続処理に渡されます。この処理の説明については、 118 ページの『オブジェクトの接続処理』を参照してください。

より詳細なビュー: より詳細なビューは、選択したオブジェクトの次に低い層の子リソースを表示する動的に作成されたビューです。より詳細なビューには、次の 4 タイプがあります。

- より詳細な論理ビュー
- より詳細な物理ビュー
- 構成子 II ビュー
- 構成子 III ビュー

選択したリソースについては、そのリソース・タイプによってこれらのビューのうち 1 つまたは複数を表示することができます。

これらのビューのうち、オブジェクトのないビューが生じると、そのビューはワークステーションに戻されません。ビューが作成されないと、ビューが見当たらない旨のメッセージがワークステーションで表示されます。

以降のトピックで、GMFHS が 4 タイプの詳細なビューを作成する方法を説明します。

より詳細な論理ビュー: より詳細な論理ビューは、動的に作成することも事前定義することもできます。より詳細な論理ビューがオープンされると、要求が GMFHS に渡されます。ビューに入れるオブジェクトを判別するために、GMFHS は次のことを行います。

- 選択したオブジェクトの ContainsLogical フィールドを照会して、オブジェクトのリストを取得するめに照会するフィールドの名前を検索します。基本的な GMFHS データ・モデルの場合では、このフィールドが ComposedOfLogical フィールドを指定します。 ComposedOfLogical フィールドには、次に低い層の選択したオブジェクトを構成するオブジェクトのリストが入ります。
- オブジェクトのリストを、GMFHS の接続処理に渡す。この処理の説明については、 118 ページの『オブジェクトの接続処理』を参照してください。

より詳細な物理ビュー: より詳細な物理ビューは、動的に作成することも事前定義することもできます。より詳細な物理ビューがオープンされると、要求が GMFHS に渡されます。ビューに入れるオブジェクトを判別するために、GMFHS は次のことを行います。

- 選択したオブジェクトの ContainsPhysical フィールドを照会して、オブジェクトのリストを取得するめに照会するフィールドの名前を検索します。基本的な GMFHS データ・モデルの場合では、このフィールドが ComposedOfPhysical フィールドを指定します。 ComposedOfPhysical フィールドには、次に低い層の選択したオブジェクトを構成するオブジェクトのリストが入ります。

  オブジェクトのリストを、GMFHS の接続処理に渡す。この処理の説明については、 118 ページの『オブジェクトの接続処理』を参照してください。

構成子 II ビュー: 構成子 II ビューは、選択した論理装置オブジェクトに定義された子のサブセットを表示する、動的に作成されたビューです。 GMFHS は、選択したオブジェクトの子のサブセットを探すために次の処理を使用します。

- View_Information_Object_Class オブジェクトを探す。
- View_Information_Object_Class オブジェクトの RelFieldNamesA フィールドを照会する。このフィールドは、第 1 レベル子のリストを判別するときに照会する、フィールドのリストを指定します。

この処理は、子オブジェクトの全リストが識別されるまで、選択したオブジェクトの子オブジェクトごとに繰り返されます。オブジェクトのリストは、GMFHS の接続処理に渡されます。この処理の説明については、 118 ページの『オブジェクトの接続処理』を参照してください。

選択したオブジェクトに、RelFieldNamesA フィールドが指定するフィールドが 1 つまたは複数ある場合は、子がなくてもビューは表示されます。この場合、表示されるのは選択したオブジェクトのみです。このビューは、ルート・ノードとしての選択オブジェクトの放射状レイアウトで表示されます。

以下の SNA トポロジー・マネージャー・リソース・クラスは、このビュー・タイプを用いて、選択したオブジェクトに接続された LU タイプのオブジェクトを表示します。

- appnEN
- appnNN
- crossDomainResource
- interchangeNode
- logicalLink
- logicalUnit
- luGroup
- migrationDataHost
- snaNode
- t5Node

構成子 III ビュー: 構成子 III ビューは、選択した定義グループ・オブジェクトに定義された子のサブセットを表示する、動的に作成されたビューです。 GMFHS は、選択したオブジェクトの子のサブセットを探すために次の処理を使用します。

- View_Information_Object_Class オブジェクトを探す。

- View_Information_Object_Class オブジェクトの RelFieldNamesA フィールドを照会する。このフィールドは、第 1 レベル子のリストを判別するときに照会する、フィールドのリストを指定します。

この処理は、子オブジェクトの全リストが識別されるまで、選択したオブジェクトの子オブジェクトごとに繰り返されます。オブジェクトのリストは、GMFHS の接続処理に渡されます。この処理の説明については、118 ページの『オブジェクトの接続処理』を参照してください。

選択したオブジェクトに、RelFieldNamesA フィールドが指定するフィールドが 1 つまたは複数ある場合は、子がなくてもビューは表示されます。この場合、表示されるのは選択したオブジェクトのみです。このビューは、ルート・ノードとしての選択オブジェクトの階層レイアウトで表示されます。

以下の SNA トポロジー・マネージャー・リソース・クラスは、このビュー・タイプを用いて、選択したオブジェクトに接続された定義グループ・オブジェクトを表示します。
- t5Node
- interchangeNode
- migrationDataHost
- appnEN
- appnNN
- definitionGroup

例外ビュー: 例外ビューは事前定義ビューです。各ビューは、RODM の Exception_View_Class で作成されたオブジェクトによって表されます。 NetView 管理コンソール・グラフィック・データ・サーバーまたは NMC サーバーが GMFHS とのセッションを確立するときに、このクラスのあらゆるオブジェクトが照会され、NMC ツリー・ビューに表示されます。例外ビューの追加もしくは削除を行う際に、このビューのリストは自動的に更新されます。表示されるビュー名は、Exception_View_Class オブジェクトの MyName フィールドの値です。

例外ビューのオブジェクトの展開処理は、ビュー・オブジェクトにビューの各リソースへのリンクが含まれていないため、他の事前定義のビューとは異なります。例外ビューの場合は、例外ビューに入れることができる候補オブジェクトのリスト、およびそのリストに規則正しく適用される一連のフィルターを定義することで、オブジェクトの展開が行われます。これらのフィルターによって、リストは、例外ビューで表示したいオブジェクトだけを含むまでに絞り込まれます。例えば、例外ビューに NCP をすべて定義し、その設定を、注意が必要な問題をもつものに限ってビューに表示されるように行うことができます。

例外ビューがオープンされると、候補オブジェクトのリストを決定する要求が GMFHS に渡されます。候補オブジェクトのリストは、最初に Exception_View_Class オブジェクトの ExceptionViewName フィールドを照会することで検出されます。この場合、GMFHS は RODM の ExceptionViewList フィールドに対して、そのフィールドの値についての位置指定要求を出します。候補として定義されるオブジェクトは、すべてこの位置指定要求によって戻されます。

Exception_View_Class オブジェクトの ExceptionViewFilter フィールドには、このリストの絞り込みに使用するフィルターが入ります。例えば、これらのフィルターを

使用すると、現在中断状態かマークのついたオブジェクト、あるいは状況が問題とは考えられないオブジェクトをフィルターで除くことができます。このようにして、問題プログラム状態にあるリソースのリストが作られます。その後、オブジェクトのリストは、たとえ空であっても NetView 管理コンソールに渡されて表示されます。

GMFHS は、すべてのオープン例外ビューを最新状態に保ちます。これは、リソースの ExceptionViewList で指定されたビューがオープンかどうかを判別することで行われます。 GMFHS は、各ビューのフィルターとリソースを比較した後、オープンしている例外ビューとの間でリソースを追加するか削除するかを判別します。

オブジェクトの接続処理

オブジェクト判別処理によってビュー内のオブジェクトのリストの判別が行われた後、リストはオブジェクトの接続処理に渡されます。ここで GMFHS は、リストされたオブジェクトをビュー内で相互接続する方法を決めなければなりません。GMFHS は、リストされたオブジェクトごとに、以下の処理を順次実行することでこれを行います。オブジェクトごとに、GMFHS は以下のことを行います。

- View_Information_Object_Class オブジェクトを探す。
- RelFieldNames*x* フィールドを照会する。このフィールドで、オブジェクトの照会するフィールドを指定します。
- オブジェクトのこれらのフィールドを照会する。
- 照会要求により戻されたオブジェクト・リストと、接続処理に渡された最初のオブジェクト・リストを比較する。両リストに含まれているオブジェクトは、すべて接続されています。
- ビューを NetView 管理コンソール に渡す。

注:

1. 例外ビューの場合、GMFHS はこの処理を使用しません。オブジェクトはすべてグリッドで表示され、それらのオブジェクト間には接続関係がありません。
2. GMFHS は、ノードは他のノードに接続されることを判別した場合は、 2 つのノード間にヌルのコネクター・リンクを挿入します。
3. リンクにエンドポイントとしての実ノードがない場合は、 GMFHS がヌルのコネクター・ノードを挿入します。

例外ビューのオブジェクトおよび基準を定義する

例外ビューを定義するには、次のタスクを完了する必要があります。

1. 例外ビュー・オブジェクトを作成し、例外であるとみなす基準を定義する。このステップには、例外ビュー候補リスト (ビューで表示するオブジェクトを最終的に定義する) に適用するフィルターが備えられています。
2. 例外ビューの候補である RODM のオブジェクトを定義する。

すべての例外ビューは、NetView ホストで定義します。これらのビューを NetView 管理コンソール からカスタマイズすることはできません。

サンプル DUIFDEXV、例外定義ビューには、4 つの例外ビュー・オブジェクトを作成して、 GMFHS_Managed_Real_Objects_Class と GMFHS_Aggregate_Objects_Class

の両方についての 2 つの ExceptionViewList 値を設定する例があります。サンプル DUIFDEXV のプロローグには、 GMFHS オブジェクトの例外ビューを定義する方法が記載されています。

例外基準を定義する

一定の例外ビューおよびリソースの例外を構成する内容を定義し、したがって、オブジェクトを例外ビューに入れる時点を決めることができます。次のフィールドは、リソースを例外ビューに表示するタイミングを判別するのに使用します。

- オブジェクトの UserStatus フィールドの値
- オブジェクトの DisplayStatus フィールドの値
- オブジェクトの ResourceTraits フィールドの値
- Exception_View_Class オブジェクトの ExceptionViewFilter フィールド

オブジェクトの UserStatus フィールドを使用することによって、オブジェクトをオペレーターの入力または自動化プログラムに基づいて例外ビューで表示するかどうかを指定することができます。例えば、オペレーターは自らが作業を行うオブジェクトにマークすることができます。また、マークされたオブジェクトを例外ビューから除く選択を行うことができます。あるいは、自動化ルーチンは、障害リソースのリカバリーを試みる場合、オブジェクトの進行中自動化ビットを設定することができます。そして、これらのオブジェクトは例外ビューから除く選択を行うことができます。 ExceptionViewFilter は、各例外ビューについてのこれらの UserStatus 値の処理をカスタマイズするのに使用します。

オブジェクトの DisplayStatus フィールドには、オブジェクトを例外ビューに配置するかどうかを決める際に使用する、基本的な状況情報が格納されます。例えば、DisplayStatus 値が 129 (適合) ならば、おそらくオブジェクトを例外ビューに表示したくないでしょう。 DisplayStatus 値が 130 (不良) に変われば、おそらくオブジェクトを表示したいでしょう。しかし、DisplayStatus 値が 132 (不明) ならば、表示したいオブジェクトもあれば、したくないオブジェクトもあります。

NetView は、オブジェクトおよびクラスの DisplayStatus を例外もしくは非例外にマップする、サンプル表、DUIFSMT を提供します。このマッピングのことを、オブジェクトの例外状態といいます。

```
DUIFSMT  CSECT
         DUIFSMTE CLASS=APPNNN,                                        C
               XCPT=(UNSAT,UNKWN,DS152,DS153,DS154,DS155,DS156,DS157,DSC
               158,DS159,MEDUN,LOWUN)
         DUIFSMTE CLASS=INTERCHANGENODE,                               C
               XCPT=(UNSAT,UNKWN,DS152,DS153,DS154,DS155,DS156,DS157,DSC
               158,DS159,MEDUN,LOWUN)
         DUIFSMTE CLASS=MIGRATIONDATAHOST,                             C
               XCPT=(UNSAT,UNKWN,DS152,DS153,DS154,DS155,DS156,DS157,DSC
               158,DS159,MEDUN,LOWUN)
         DUIFSMTE CLASS=T5NODE,                                        C
               XCPT=(UNSAT,UNKWN,DS152,DS153,DS154,DS155,DS156,DS157,DSC
               158,DS159,MEDUN,LOWUN)
         DUIFSMTE CLASS=APPNTRANSMISSIONGROUP,                         C
               XCPT=(UNSAT,UNKWN,DS152,DS153,DS154,DS155,DS156,DS157,DSC
               158,DS159,MEDUN,LOWUN)
         DUIFSMTE CLASS=APPNTRANSMISSIONGROUPCIRCUIT,                  C
               XCPT=(UNSAT,UNKWN,DS152,DS153,DS154,DS155,DS156,DS157,DSC
               158,DS159,MEDUN,LOWUN)
         DUIFSMTE CLASS=T4NODE,                                        C
               XCPT=(UNSAT,UNKWN,DS152,DS153,DS154,DS155,DS156,DS157,DSC
               158,DS159,MEDUN,LOWUN)
         DUIFSMTE CLASS=GMFHS_Managed_Real_Objects_Class,              C
               XCPT=(UNSAT,DS152,DS153,DS154,DS155,DS156,DS157,DS158,DSC
               159,MEDUN,LOWUN)
         DUIFSMTE CLASS=ALL,                                           C
               XCPT=(UNSAT,DEGRD,SDGRD,DS152,DS153,DS154,DS155,DS156,DSC
               157,DS158,DS159)
LAST     DUIFSMTE END
```

図 29. サンプル表 DUIFSMT

DisplayStatus の解釈方法は、DUIFSMT 表を修正してカスタマイズすることができます。詳細については、123 ページの『例外ビュー用の DisplayStatus マッピング・テーブルをカスタマイズする』を参照してください。

RODM データへのアクセスを可能にする RODM ユーザー・メソッドを作成して、表をオーバーライドすることもできます。詳細については、130 ページの『例外ビューの DisplayStatus メソッドを作成する』を参照してください。

注: オブジェクトの例外状態は、どの実在オブジェクトを障害のあるリソースに対する NMC により検出された障害のあるリソースのビューに含めるかを決定するために使用される基準の 1 つです。例外状態にマップする実在オブジェクトだけが、 NMC により検出された障害のあるリソースのビューに含まれます。詳細については、112 ページの『NMC により検出された障害のあるリソースのビュー』を参照してください。

オブジェクトの ResourceTraits フィールドには、 DisplayStatus の解釈方法の値、およびすべての UserStatus ビットの状態が入ります。オブジェクトの ResourceTraits フィールドは、例外ビューを作成するときに使用され、オブジェクトが例外ビューへ入れられる基準を満たした時点を判別します。

オブジェクトの ExceptionViewFilter フィールドは、 Exception_View_Class のすべてのオブジェクトに定義されます。このフィールドは、オブジェクトが例外ビューで表示されるときに置かれていなければならない状態を定義します。
ExceptionViewFilter フィールドの値は、 ResourceTraits フィールドに反映されたリソース・オブジェクトの DisplayStatus および UserStatus フィールドの値と比較さ

れます。 ExceptionViewFilter フィールドと ResourceTraits フィールドの値が一致すると、オブジェクトは例外と見なされ、定義済みの例外ビューに入れられます。ExceptionViewFilter カスタマイズのすべての説明については、 『ExceptionViewFilter フィールドを定義する』を参照してください。

例外ビューの候補を定義する

次のフィールドは、オブジェクトを表示できる例外ビューを定義するのに使用されます。

- Exception_View_Class オブジェクトの ExceptionViewName フィールド
- オブジェクトの ExceptionViewList フィールド

ExceptionViewName フィールドには、作成した Exception_View_Class オブジェクトの固有の名前が入ります。定義する例外ビューごとに Exception_View_Class オブジェクトを 1 つ作成し、かつ各オブジェクトの名前は個々別々でなければなりません。

リソース・オブジェクトの ExceptionViewList フィールドには、ExceptionViewNames のリストが入ります。リソースが例外状態になったときに、このリソースの表示を行いたい各例外ビューの ExceptionViewName を指定する必要があります。リソースは複数の例外ビューで表示することができるので、 ExceptionViewList フィールドには名前のリストを入れることができます。

オープンの例外ビューで表示すべきリソース・オブジェクトを作成する場合は、以下のいずれかのタスクが必要です。

- ExceptionViewList フィールドをヌル値から候補ビューのリストに変更する。
- 例外ビューをクローズしてから、再オープンする。

オープンの例外ビューにある RODM からリソース・オブジェクトを削除したい場合は、リソース・オブジェクトを削除する前に、ExceptionViewList から ExceptionViewName を取り外します。リソース・オブジェクトを ExceptionViewList から除去する前に RODM から削除しても、 GMFHS は削除されたオブジェクトの更新を送れないため、リソース・オブジェクトはクローズされるまでビュー内に残ります。

SNA トポロジー・マネージャーによって管理される SNA リソースの場合は、オブジェクトの作成時に NetView によって ExceptionViewList フィールドが設定されます。 NetView プログラムは、オブジェクトのクラスに基づいてこのフィールドの値を判別します。クラスの例外ビューへのデフォルトのマッピングは、FLBEXV 表をカスタマイズして変更することができます。 FLBEXV 表のカスタマイズの詳細については、「*IBM Tivoli NetView for z/OS SNA トポロジー・マネージャー インプリメンテーション・ガイド*」を参照してください。

ExceptionViewFilter フィールドを定義する

ExceptionViewFilter フィールドは、オブジェクトが例外ビューに入る際に置かれていなければならない状態を定義するのに使用します。フィールドには 5 つの値があり、それぞれ異なる状況フィルターを表します。フィルター 1 は DisplayStatus 用、残りの 4 つのフィルターは UserStatus 用です。

ExceptionViewFilter のデフォルトは X'4000' (ビット値 '0100 0000 0000 0000') です。これは以下のことを示します。

- ビューの候補は、例外状態にあるオブジェクトのみである。例外状態にあるオブジェクトは、ResourceTraits フィールドの値が XCPT であるオブジェクトです。
- UserStatus には、フィルター操作は行われない。

これは、オブジェクトは、例外状態にマップされる場合、その UserStatus に関係なく例外ビューで表示されることを意味します。 ExceptionViewFilter のデフォルト値は、クラス・レベルかオブジェクト・レベルで変更することができます。

DisplayStatus フィルター: DisplayStatus に関係なくすべてのオブジェクトを例外ビューの候補と考えたい場合は、 DisplayStatus の ExceptionViewFilter を 0 (ゼロ) に設定します。例外状態にあるオブジェクトだけを例外ビューの候補と考えたい場合は、 DisplayStatus の ExceptionViewFilter の設定を 1 のままにします。これはデフォルト値です。

シャドー・オブジェクトには DisplayStatus フィールドがありませんので、モニター可能なオブジェクトとは考えられません。しかし、ExceptionViewFilter フィールドの DisplayStatus のフィルターを 0 (ゼロ) に設定すると、シャドー・オブジェクトはビューの候補になります。シャドー・オブジェクトは、ビュー・オブジェクトの ExceptionViewFilter フィールドで指定されたすべての基準を守り、シャドー・オブジェクトの ExceptionViewList フィールドにはビューの ExceptionViewName が入っていなければなりません。

UserStatus フィルター: 例外ビューからフィルターで除く UserStatuses を示すには、 ExceptionViewFilter に UserStatus フィルターを設定します。例えば、“mark” の UserStatus をもつオブジェクトをフィルターで除きたい場合は、ExceptionViewFilter フィールドのマーク UserStatus フィルターをビット値 X'01' に設定します。マークのない オブジェクトすべてをフィルターで除きたい場合は、ExceptionViewFilter フィールドのマーク UserStatus フィルターをビット値 X'10' に設定します。

以下の UserStatus のビットがオンの場合は、オブジェクトは例外ビューで表示されません。

- X'02' (モニターなし)
- X'40' (集約が中断状態)

これは、これらのビットは、例外ビューから自動的にフィルターにかけられるため、フィルターをかけることができないことを意味します。

NetView 管理コンソール で「List Suspended Resources」を用いて、集約から中断状態になっているオブジェクトを判別します。

表 19 に、 ExceptionViewFilter フィールドの代替値および結果としての例外ビューの例を記載します。

表 19. ExceptionViewFilter フィールド値および結果としてのビューの例

値	ビューのオブジェクト
'0000 0000 0000 0000' (X'0000')	DisplayStatus または UserStatus に関係なくビューに定義されたすべてのオブジェクト。

表 19. ExceptionViewFilter フィールド値および結果としてのビューの例 (続き)

値	ビューのオブジェクト
'0101 0000 0000 0000' (X'5000')	**マークのない** ビューに定義された例外状態のすべてのオブジェクト。マークつきオブジェクトは、すべてフィルターでビューから除かれます。
'0110 0000 0000 0000' (X'6000')	マークつきのビューに定義された例外状態のすべてのオブジェクト。**マークのない** オブジェクトは、すべてフィルターでビューから除かれます。

例外ビュー用の DisplayStatus マッピング・テーブルをカスタマイズする

DisplayStatus 値のマッピングは、表 DUIFSMT を用いてカスタマイズすることができます。この表は、DUIFSMTE マクロによって作成されたステートメントから構成されます。

テーブルをカスタマイズするには、サンプル DUIFSMT の DUIFSMTE ステートメントを、希望する DisplayStatus マッピングを反映するように変更してから、サンプル CNMSJH13 を用いて次のことを行います。

- 表をアセンブルし、リンク・エディットして、ロード・モジュールを作成する。
- DisplayStatus 変更メソッドを最新表示する。
- RODM のすべての実オブジェクトおよび集合オブジェクトの、 DisplayStatus マッピングの再計算を起動する。

新しい状況を即時に例外ビューに使用できるように、DisplayStatus のマッピングを再計算します。オブジェクトの DisplayStatus が変更されるまで再計算したくない場合は、サンプル CNMSJH13 の次のステートメントからコメントを外します。

```
OP DUIFRFDS INVOKED_WITH;
```

124 ページの図 30 は、DUIFSMTE マクロの構文です。 *class_name* に値 ALL を使用して、 DUIFSMT テーブルに組み込まれていないクラスにデフォルト値を指定します。

マクロ形式は、 124 ページの図 30 に記載されています。

DUIFSMTE

Class:

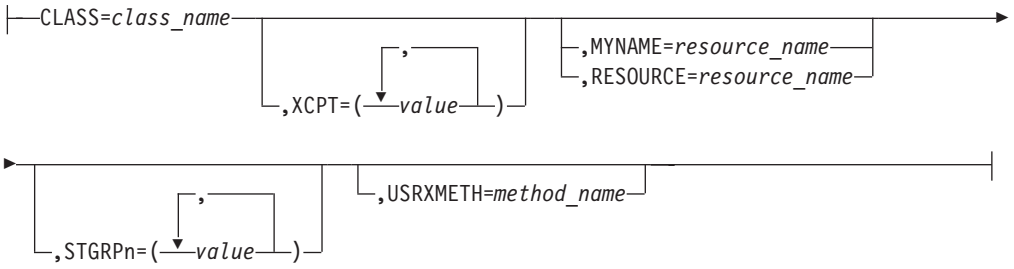


図 30. マクロ DUIFSMTE 構文

キーワードは、複数指定することができますが、どれも複数回指定することはできません。

ここで、

CLASS=*class_name*

DisplayStatus マッピングのカスタマイズを行う、RODM のクラスの名前です。DUIFSMT 表に組み込まれていないクラスにデフォルト値を指定したい場合は、*class_name* に値 ALL を使用します。

クラスのオブジェクトのすべての DisplayStatus マッピングをカスタマイズするには、そのクラスのステートメントが 1 つ必要です。特定のオブジェクトまたはオブジェクトのグループの DisplayStatus マッピングをカスタマイズするには、複数のステートメントが必要です。 *class_name* の値が同じステートメントには、それぞれ RESOURCE または MYNAME キーワードに異なる値が必要です。

注: RODM 名は、大小文字を区別します。

SNA トポロジー・マネージャーに管理されるクラスの場合は、クラスに別名値を使用することができます。 表 20 は、入力することができる別名とそれに対応する実際のクラス名 (RODM に認識されていて、ともに DUIFSMTE マクロにより受け入れられる) のリストです。

表 20. RODM クラス名の別名

クラスの別名	クラスの MyName 値
APPNEN	1.3.18.0.0.1821
APPNNN	1.3.18.0.0.1822
APPNTRANSMISSIONGROUP	1.3.18.0.0.1823

表 20. RODM クラス名の別名 (続き)

クラスの別名	クラスの MyName 値
APPNTRANSMISSIONGROUPCIRCUIT	1.3.18.0.0.2058
CROSSDOMAINRESOURCE	1.3.18.0.0.2281
CROSSDOMAINRESOURCEMANAGER	1.3.18.0.0.2278
DEFINITIONGROUP	1.3.18.0.0.2267
INTERCHANGENODE	1.3.18.0.0.1826
LENNODE	1.3.18.0.0.1827
LOGICALLINK	1.3.18.0.0.2085
LOGICALUNIT	1.3.18.0.0.1829
MIGRATIONDATAHOST	1.3.18.0.0.2155
PORT	1.3.18.0.0.2089
T2-1NODE	1.3.18.0.0.1843
T4NODE	1.3.18.0.0.1844
T5NODE	1.3.18.0.0.1845
VIRTUALROUTINGNODE	1.3.18.0.0.1845

例外ビューの処理の詳細については、 132 ページの『マルチシステム・マネージャーの例外ビューの処理のインプリメント』を参照してください。

XCPT=*value*
例外状態にあると考えられる、オブジェクトの DisplayStatus 値を指定します。値は、複数指定することができますが、どれも複数回指定することはできません。 DisplayStatus 値をもつオブジェクトは、 UserStatus および ExceptionViewList 基準も一致すると、例外ビューに追加されます。

注: XCPT が指定されていないか、XCPT の値がヌルの場合、オブジェクトは、唯一の組み込み例外オブジェクトに定義された例外ビューには組み込まれません。 130 ページの図 34 の CrossDomainResourceManager は、ExceptionViewFilter が X'4000' の例外ビューでは表示されません。

以下は、考えられる **XCPT** 値です。

DEGRD
DisplayStatus 値が 133 (低下) であるオブジェクトを指定します。

INTER
DisplayStatus 値が 131 (中間) であるオブジェクトを指定します。

LOWSA
DisplayStatus 値が 145 (低程度に適合) であるオブジェクトを指定します。

LOWUN
DisplayStatus 値が 161 (低程度に不良) であるオブジェクトを指定します。

MEDSA
DisplayStatus 値が 144 (中程度に適合) であるオブジェクトを指定します。

MEDUN
DisplayStatus 値が 160 (中程度に不良) であるオブジェクトを指定します。

SATIS DisplayStatus 値が 129 (適合) であるオブジェクトを指定します。

SDGRD
DisplayStatus 値が 134 (重大劣化) であるオブジェクトを指定します。

UNKWN
DisplayStatus 値が 132 (不明) であるオブジェクトを指定します。

UNSAT
DisplayStatus 値が 130 (不良) であるオブジェクトを指定します。

顧客専用に予約された考えられるユーザー定義の DisplayStatus 値は、16 あります。以下は、**XCPT** の考えられるユーザー定義の値です。

DS136 ユーザー定義の DisplayStatus 値が 136 であるオブジェクトを指定します。

DS137 ユーザー定義の DisplayStatus 値が 137 であるオブジェクトを指定します。

DS138 ユーザー定義の DisplayStatus 値が 138 であるオブジェクトを指定します。

DS139 ユーザー定義の DisplayStatus 値が 139 であるオブジェクトを指定します。

DS140 ユーザー定義の DisplayStatus 値が 140 であるオブジェクトを指定します。

DS141 ユーザー定義の DisplayStatus 値が 141 であるオブジェクトを指定します。

DS142 ユーザー定義の DisplayStatus 値が 142 であるオブジェクトを指定します。

DS143 ユーザー定義の DisplayStatus 値が 143 であるオブジェクトを指定します。

DS152 ユーザー定義の DisplayStatus 値が 152 であるオブジェクトを指定します。

DS153 ユーザー定義の DisplayStatus 値が 153 であるオブジェクトを指定します。

DS154 ユーザー定義の DisplayStatus 値が 154 であるオブジェクトを指定します。

DS155 ユーザー定義の DisplayStatus 値が 155 であるオブジェクトを指定します。

DS156 ユーザー定義の DisplayStatus 値が 156 であるオブジェクトを指定します。

DS157 ユーザー定義の DisplayStatus 値が 157 であるオブジェクトを指定します。

DS158 ユーザー定義の DisplayStatus 値が 158 であるオブジェクトを指定します。

DS159 ユーザー定義の DisplayStatus 値が 159 であるオブジェクトを指定します。

STGRPn=*value* (n は 1 から 8 までの数字)
状況グループの集約に対して DisplayStatus 値のグループを指定します (166 ページの『状況グループ』を参照)。複数の値を指定することができますが、 1 つの状況グループに対して同じ値を複数回指定することはできません。実オブジェクトの DisplayStatus 値が状況グループ内の DisplayStatus 値と一致する場合、親集合オブジェクトに状況グループが定義されていれば、その親集合オブジェクトには、すべて同じ状況グループの DisplayStatus 値が割り当てられます。集合オブジェクトに対する状況グループ内で複数の DisplayStatus 値が定義されている場合は、最初の DisplayStatus 値が使用されます。

グループには、1 (高) から 8 (低) までの優先順位が付けられています。STGRPn について、実オブジェクトまたは集合オブジェクトのいずれにもキーワードが指定されていないかヌルである場合は、その状況グループの状況がオーバーライドされることはありません。

可能な STGRPn の値は、XCPT キーワードにリストされた値と同じです。

RESOURCE=*resource_name*
これらの値が適用される特定のリソースまたはリソースのグループの DisplayResourceName。リソース名の終わりにワイルド・カード文字 * (アスタリスク) を使用して、リソースのグループを指定することができます。リソース名に組み込まれているワイルド・カード文字 * は、使用できません。詳細については、 128 ページの『DisplayStatus マッピングのリソース名を指定する』を参照してください。

注: 同じ DUIFSMTE ステートメントに、 RESOURCE キーワードと MYNAME キーワードの両方を指定することはできません。

MYNAME=*resource_name*
これらの値が適用されるリソースまたはリソースのグループの MyName。リソース名の終わりにワイルド・カード文字 * (アスタリスク) を使用して、リソースのグループを指定することができます。リソース名に組み込まれているワイルド・カード文字 * は、使用できません。

注: 同じ DUIFSMTE ステートメントに、 MYNAME キーワードと RESOURCE キーワードの両方を指定することはできません。

USRXMETH=*method_name*

このクラスのオブジェクトについて起動する RODM ユーザー・メソッドの名前。このメソッドを指定すると、DisplayStatus マッピングをオーバーライドする場合があります。詳細については、 130 ページの『例外ビューの DisplayStatus メソッドを作成する』を参照してください。

END

このキーワードは、表処理を終了します。 DUIFSMTE END は、表のソースの最後のステートメントでなければなりません。

使用上の注意:

1. サンプル DUIFSMT では、DUIFSMTE は 10 桁目から始まらなければなりません。キーワードは、 DUIFSMTE に続く桁にコーディングできます (DUIFSMTE とキーワードはスペースで区切る)。
2. ステートメントが 71 文字を超える場合は、72 桁目に継続文字を入れて、次の行の 16 桁目からステートメントを続けます。
3. 同じ *class_name* および *resource_name* 値の複数のステートメントを入力すると、最初のステートメントが使用されて、他のステートメントは無視され、警告メッセージが出されます。

クラスのデフォルト値

DUIFSMT 表に定義されていないすべてのクラスにデフォルト値を指定するには、*class_name* に値 ALL を使用します。例えば、次のようになります。

```
DUIFSMTE CLASS=ALL,XCPT=(DEGRD,INTER,SDGRD,UNSAT)
```

これらの値は、他のステートメントによってオーバーライドされない限り、すべてのクラスに適用されます。必要なのは、CLASS=ALL に指定する値とは異なる特定のクラスをコーディングすることだけです。

DisplayStatus マッピングのリソース名を指定する

クラス内の特定のリソースもしくはリソース・グループの DisplayStatus マッピングを指定することができます。リソース名を指定するには、DUIFSMTE マクロの RESOURCE もしくは MYNAME キーワードを使用します。リソースのグループを指定するときは、リソース名の終わりにアスタリスク (*)、つまりワイルド・カード文字を使用することができます。リソース名にワイルド・カード文字を組み込むことはできません。

特定のリソースをカスタマイズする場合は、そのリソースのステートメントを、そのクラスで一致する他の総称ステートメントの前にコーディングします。 (使用上の注意 3 を参照。) 例えば、DisplayResourceName が RALV4 で MyName が DECNET.RALV4 の場合の、 GMFHS_Managed_Real_Objects_Class のリソースがあるとします。リソース DECNET.RALV4 が不良状況で XCPT にマップしたいが、そのクラスの他のリソースには同じようにしない場合は、リソースのステートメントをまず 129 ページの図 31 のようにコーディングします。

```
DUIFSMTE CLASS=GMFHS_Managed_Real_Objects_Class,              C
      RESOURCE=RALV4,XCPT=(UNSAT)
DUIFSMTE CLASS=GMFHS_Managed_Real_Objects_Class,              C
      XCPT=(INTER)
DUIFSMTE CLASS=ALL,                                           C
      XCPT=(UNSAT,UNKWN)
```

図 31. リソースをカスタマイズする

図 31 で、 2 番目の DUIFSMTE ステートメントが最初の DUIFSMTE ステートメントの前にコーディングされていた場合、リソース DECNET.RALV4 および GMFHS_Managed_Real_Objects_Class の他のすべてのオブジェクトは、中間状況のときに限り例外にマップします。

RESOURCE キーワードの規則は、 SNA トポロジー・マネージャーのカスタマイズ表の RESOURCE キーワードの規則と同じです。詳細については、「*IBM Tivoli NetView for z/OS SNA トポロジー・マネージャー インプリメンテーション・ガイド*」を参照してください。

図 32 は、同じクラスに MYNAME キーワードと RESOURCE キーワードの両方をコーディングする例を説明しています。 MyName が DECNET.RALV4 で DisplayResourceName が RALV4 の場合の GMFHS_Managed_Real_Objects_Class のリソース・オブジェクトがあるとします。 DUIFSMTE エントリーを 図 32 のようにコーディングすると、リソースは、DUIFSMTE エントリーの 3 つのすべてに一致します。しかし、ステートメントをコーディングする順序は重要であるため、最初の DUIFSMTE エントリーが例外状態に一致する DUIFSMTE エントリーです。このオブジェクトは、DisplayStatus が中間の場合に限り例外です。

```
DUIFSMTE CLASS=GMFHS_Managed_Real_Objects_Class,              C
      MYNAME=DECNET.*,                                        C
      XCPT=(INTER)
DUIFSMTE CLASS=GMFHS_Managed_Real_Objects_Class,              C
      RESOURCE=RALV*,                                         C
      XCPT=(SATIS)
DUIFSMTE CLASS=ALL,                                           C
      XCPT=(UNSAT)
DUIFSMTE END
```

図 32. 同じ DUIFSMTE エントリーの MYNAME および RESOURCE キーワードの例

DisplayStatus マッピングをカスタマイズする例

このトピックでの例は、DisplayStatus の例外状態へのマッピングを理解しやすくする目的で用意されました。最初の例 (130 ページの図 33 に表示) では、以下の条件を前提としています。

- 例外ビューの DisplayStatus が不良もしくは不明の場合の、 t4Node (1.3.18.0.0.1844) クラスのオブジェクトをすべて表示したい。 (クラス名には 124 ページの表 20 での別名を使用します。)
- 例外ビューの DisplayStatus が不良、中間、もしくは不明の場合の、 appnEN (1.3.18.0.0.1821) クラスのオブジェクトをすべて表示したい。 (クラス名には 124 ページの表 20 での実際の MyName 値を使用します。)

- 例外ビューの GMFHS_Aggregate_Objects_Class のオブジェクトを、その DisplayStatus 値が重大劣化の場合に、すべて表示したい。
- 他のすべてのクラスのオブジェクトについて、それらの DisplayStatus が不良もしくは重大劣化の場合に限り例外ビューに入れたい。

図 33 は、前掲の条件を用いて DisplayStatus マッピング表のコーディングを示しています。 4 番目のステートメントでデフォルトを設定しています。

```
DUIFSMTE CLASS=T4NODE,XCPT=(UNSAT,UNKWN)
DUIFSMTE CLASS=1.3.18.0.0.1821,XCPT=(UNSAT,INTER,UNKWN)
DUIFSMTE CLASS=GMFHS_Aggregate_Objects_Class,XCPT=(SDGRD)
DUIFSMTE CLASS=ALL,XCPT=(UNSAT,SDGRD)
```

図 33. DisplayStatus マッピング表のコーディング例 1

2 番目の例 (図 34 に表示) では、以下の条件を前提としています。

- RODM の他のフィールドの値に基づいて、 t2-1Node のオブジェクトを例外ビューで表示するかどうかを判別する、CUSTMTH1 という名前の RODM メソッドを作成しています。
- crossDomainResourceManager クラスのオブジェクトを ExceptionViewFilter 値が X'4000' である例外ビューに表示したくない。
- DisplayResourceName が USIBMNT.NCPPU1 の appnEN クラスのオブジェクトを、その状況に関係なく例外ビューに表示したい。ユーザー定義の DisplayStatus 値は、定義されていません。
- DisplayResourceName が USIBMNT の SNA ネットワーク ID 部分をともなう appnEN クラスのオブジェクトを、その状況が適合でない場合に例外ビューに表示したい。ユーザー定義の DisplayStatus 値は、定義されていません。

図 34 は、前掲の条件を用いて DisplayStatus マッピング表のコーディングを示しています。

```
DUIFSMTE CLASS=T2-1NODE,USRXMETH=CUSTMTH1
DUIFSMTE CLASS=CROSSDOMAINRESOURCEMANAGER
DUIFSMTE CLASS=APPNEN,                                          C
      RESOURCE=USIBMNT.NCPPU1,                                  C
      XCPT=(DEGRD,INTER,SATIS,SDGRD,UNKWN,UNSAT,MEDSA,MEDUN,LOC
      WSA,LOWUN)
DUIFSMTE CLASS=APPNEN,                                          C
      RESOURCE=USIBMNT.*,                                       C
      XCPT=(DEGRD,INTER,SDGRD,UNKWN,UNSAT,MEDSA,MEDUN,LOWSA,LOC
      WUN)
```

図 34. DisplayStatus マッピング表のコーディング例 2

例外ビューの DisplayStatus メソッドを作成する

オブジェクト独立のメソッドをコーディングして、DUIFSMT 表から得られる処理に加えて、特別レベルの DisplayStatus 例外処理を備えることができます。サンプル・ユーザー・メソッド、DUIFCUXM は、この目的で用意されています。ユーザー・メソッドを作成するときは、このサンプルを参照してください。

DUIFSMT 表の USRXMETH キーワードでメソッド名を指定する場合は、指定したオブジェクトの DisplayStatus が変わるごとに、このメソッドが非同期に起動されます。このメソッドは、RODM メソッドの指針に従っていなければなりません。RODM メソッド作成の詳細については、389 ページの『第 13 章 RODM メソッドの作成』を参照してください。

メソッドは、DUIFCRDC メソッドから非同期に起動され、 DisplayStatus の変更が生じたオブジェクト ID に渡されます。以下は、このメソッドの入力パラメーターです。

```
Smallint   Total_length;
Smallint   Data_Type;
Smallint   Data_Length;
ObjectID   Resource_Object_ID;
Integer    Requested_exception_status;
```

ユーザー・メソッドは非同期であるため、ユーザー・メソッドが制御を得た時点では、これを進めた元の条件が該当しない場合があります。したがって、メソッド DUIFCRDC からユーザー・メソッドに渡される事前照会のフィールド値はありません。

タイミングの問題とエラー処理の問題が生じるおそれがありますので、注意してください。例えば、DUIFSMT から例外状態をマップすると、オブジェクトは例外ビューに加えられますが、ユーザー・メソッドは同じオブジェクトの例外状態を変更することができるので、それはすぐに除かれます。ユーザー・メソッドでのエラーは、ユーザー・メソッドによって解決されなければなりません。 RODM での非同期のエラー処理の詳細については、343 ページの『第 11 章 RODM を使用するアプリケーションを作成する』を参照してください。

ユーザー・メソッドから予期しない結果を受け、それが起動されたものではないと推定される場合は、ユーザー・メソッドのインストールが誤っていた可能性があります。この場合、RODM はトランザクション情報ブロックに、戻りコードと理由コードを出します。このエラーは、カスタマイズ・ファイルに設定されている LOG_LEVEL および MLOG_LEVEL の値に従って、 UAPI トレース・エントリーとして RODM ログに書き込まれます。ログ・エントリーには、次の情報が含まれています。

- 戻りコード: 8
- 理由コード: 81
- 機能 ID: 1416 (オブジェクト独立メソッドを起動する)
- データ: ユーザー・メソッド名

注: ユーザー・メソッドのインストールをテストするときは、 RODMVIEW を用いてそれを起動することができます。

ユーザー・メソッドは、RODM の情報を含むすべての基準を受け入れ、オブジェクトの例外状態を判別します。例外状態が判別されたら、 IBM 提供のメソッド DUIFVCFT をユーザー・メソッドから起動して、指定したオブジェクトの ResourceTraits フィールドに状況を実装する必要があります。

例 1: オブジェクトの例外状態を XCPT に変更します。

1. ユーザー・メソッドから、Requested_exception_status=1 をメソッド DUIFVCFT に渡す。

2. DUIFVCFT は、ResourceTraits フィールドを XCPT に変更する。

例 2: オブジェクトの例外状態を NOXCPT に変更します。

1. ユーザー・メソッドから、Requested_exception_status=0 をメソッド DUIFVCFT に渡す。
2. DUIFVCFT は、ResourceTraits フィールドを NOXCPT に変更する。

いずれの場合も、ResourceTraits フィールドでの設定の結果、オープンの例外ビューとの間でオブジェクトの追加もしくは削除が行われます。この判別は、メソッド DUIFVCFT によって行われます。

Requested_exception_status の埋めこみが DUIFVCFT を起動したときに限られるのを別にすれば、メソッド DUIFVCFT への入力パラメーターは、ユーザー・メソッドへの入力と同じです。ユーザー・メソッドが、入力オブジェクトの例外状態を変える必要があると判断した場合に限り、 DUIFVCFT を起動します。

高水準のリソース障害のために障害があるとマークされたビューからのリソースをフィルター操作する場合にも、ユーザー・メソッドを作成することができます。メソッド DUIFCUX2 は、この機能を実行するサンプル・メソッドとして提供されます。

マルチシステム・マネージャーの例外ビューの処理のインプリメント

例外ビューは、オブジェクトの DisplayStatus または UserStatus フィールドの値によってフィルターに掛けられる、オブジェクトのグラフィック・リストです。マルチシステム・マネージャー・オブジェクトの例外ビューの処理を使用可能にすると、障害のあるリソースを適切な時に認識できるようになります。

例外ビューの処理をインプリメントするには、次のようにします。

1. NetView の一部の DUIFSMT を、サンプル FLCSSMT からのステートメントを組み込むように修正する。 DUIFSMT はアセンブラー部で、%INCLUDE ステートメントをサポートしません。したがって、手動でファイルを編集することによって、これらのステートメントを DUIFSMT に組み込まなければなりません。

   サンプル FLCSSMT は、マルチシステム・マネージャー・オブジェクトおよびクラスの DisplayStatus を例外または非例外にマップするサンプル表です。FLCSSMT は CNMSAMP データ・セットに入っています。
2. NetView JCL サンプル CNMSJH13 を実行して、DUIFSMT をアセンブルおよびリンク・エディットする。これを行うことにより、次のような結果になります。
   - 表をアセンブルし、リンク・エディットして、ロード・モジュールを作成する。
   - DisplayStatus 変更メソッドを最新表示する。
   - RODM のすべての実オブジェクトおよび集合オブジェクトの、DisplayStatus マッピングを再計算する。
3. マルチシステム・マネージャーの例外ビュー・ファイルを修正する。

   マルチシステム・マネージャーの例外ビュー・テーブルは、 RODM オブジェクトがマルチシステム・マネージャーによって作成されるときに RODM オブジェクトが関連付けられる例外ビューの名前をリストします。

すでにマルチシステム・マネージャーの例外ビュー処理をインプリメントしている場合は、既存のマルチシステム・マネージャーの例外ビュー・テーブルを修正します。

マルチシステム・マネージャーの例外ビュー処理をまだインプリメントしていない場合は、 NetView 開始プロシージャーで定義された DSIPARM DD 連結からアクセス可能なデータ・セットにサンプル FLCSEXV をコピーします。サンプル・ファイルをご使用の環境での適切な名前に名前変更します。サンプル FLCSEXV は CNMSAMP データ・セットに入っています。

FLCSEXV には、すべてのマルチシステム・マネージャー実オブジェクト・クラスのサンプル例外ビュー・ステートメントが含まれています。それぞれのマルチシステム・マネージャー機能ごとにセクションがあります。集合オブジェクトの例外ビューを追加することができます。 Exception_View_Class にオブジェクトを作成することも可能で (例としてサンプル FLCSDM6 を参照)、その後 Exception_View_Class オブジェクトの MyName フィールドを EXVWNAME キーワードの値として使用します。

ステートメントは、すべてサンプル内でコメント化されています。特定のオブジェクト・クラスで例外ビュー処理を実行したい場合、そのオブジェクト・クラスに関連付けられているステートメントのコメント化を解除します。

FLCSEXV は %INCLUDE ステートメントをサポートしません。テーブルの構文に関する情報については、サンプル FLCSEXV の前書きを参照してください。

4. CNMSTUSR または C*xx*STGEN にある (MSM) COMMON.FLC_EXCEPTION_VIEW_FILE ステートメントのマルチシステム・マネージャー例外ビュー・テーブルの名前を指定します。
5. マルチシステム・マネージャー・データ・モデルは、 NetView サンプル CNMSJH12 を使用してロードします。各サンプルの前書きには、マルチシステム・マネージャーとともに出荷されるデータ・モデル・メンバーの簡略説明が含まれています。

   FLCSEXV 内の各セクションは、データ・モデル・サンプルと相関しています。f

   JCL サンプル CNMSJH12 では、該当する機能データ・モデル・サンプルのステートメントのコメント化を解除してください。

機能	データ・モデル・サンプル
IP	FLCSDM6I
LAN ネットワーク・マネージャー	FLCSDM6L
Open	FLCSDM6O
TMR	FLCSDM6T

必要とする情報	参照先
例外ビュー処理	*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

必要とする情報	参照先
DUIFSMT	*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

リソース位置指定機能

リソース位置指定機能を使用すると、オペレーターはリソースが入ったビューの名前が不明な場合でもリソースを表示することができます。オブジェクトが RODM で検出されると、複数のタイプのビューを検索して、作成することができます。

リソース位置指定機能が選択されると、要求が GMFHS に渡されます。 GMFHS は、LocateName フィールドおよび DisplayResourceName フィールドに対して、大文字バージョンのエントリーについての位置指定要求を出します。どちらかのリストのオブジェクトについて、要求されたビューが作成されます。 LocateName フィールドはタイプ IndexList で、複数の値をもつことができることに注意してください。したがって、オブジェクトに複数の別名をもち、そのどれを使用してもオブジェクトを探すことができます。位置指定は大文字ストリングであり、したがって LocateName の値も大文字でなければならない点に留意してください。 DisplayResourceName フィールドの値は、大文字であってはなりません。

再帰的ビューを制限する

若干のタイプのビューを作成する際にも、GMFHS は、多数のオブジェクトを照会して、ビューに属するオブジェクトのすべてを探します。この結果、ビューが多数のオブジェクトを抱えることになって、使用不能になることもあります。 HopCount フィールドを使用すると、GMFHS が照会するオブジェクト数を制限することができます。例えば、HopCount フィールドの値を 3 に設定すると、選択したオブジェクトから GMFHS が照会できるのは 3 レベルまでのオブジェクトに限られます。 GMFHS にすべてのオブジェクトを照会させたい場合は、HopCount フィールドの値を 0 (ゼロ) に設定します。

オープン・ビューを最新表示する

ビューの作成に使用されたオブジェクトもしくは接続性フィールドが RODM で変更になったとき、 GMFHS はワークステーションにビュー変更通知を送信します。これは、ビューの作成方法を制御するオブジェクトもしくはクラスのフィールドだけでなく、すべての接続フィールドにインストールされる、通知メソッド、 DUIFVNOT によって行われます。このメソッドは、データ・モデルのロード時に、サンプル FLBTRDME によってインストールされます。 FLBTRDME は、オブジェクト独立メソッド、DUIFVINS を呼び出し、これが各フィールドに DUIFVNOT をインストールします。

通知メソッドは、クラスのオブジェクトから継承されますので、注意してください。 GMFHS が DUIFVNOT をインストールするフィールドの全リストについては、サンプル FLBTRDME を参照してください。

メソッド DUIFVINS は、データ・モデルに追加される新しいクラスもしくは接続フィールドごとに実行する必要があります。メソッド DUIFVINS の説明については、569 ページの『DUIFVINS: ビュー細分性インストール・メソッド (DUIFVNOT)』を参照してください。

制御スパンをビューに適用する

このセクションでは、スパンで制限されたビューを作成する際に、 GMFHS がスパン権限の検査に使用するリソース名およびビュー名を判別する仕組みについて説明します。

このセクションでは、NGMFVSPN および CTL 属性を頻繁に参照します。これらは、RODM 属性ではありません。これらの属性は、 DSIPRF データ・セット内の NetView オペレーター・プロファイルか、 RACF® などのシステム許可機能 (SAF) プロダクトの USER プロファイルにある NETVIEW セグメントで定義されます。これらの属性に関する詳細については、「*IBM Tivoli NetView for z/OS* セキュリティー・リファレンス」を参照してください。

スパンは、ビューおよびビュー内のリソースを参照するオペレーターの操作を制限するために使用することができます。制御スパンをビューに適用するには、次のようにしてください。

- NGMFVSPN 属性を使用して、各オペレーターについてビューおよびビュー内のリソースに対するスパン検査を行うかどうかを指定する。
- NetView スパン表を使用して、ビューおよびビュー内のリソースをスパンに定義する。
- CTL 属性を使用して、このオペレーターに対してスパン検査を行う必要があるかどうかを指定する。

リソースおよびビューを NetView スパン表内のスパンに定義する方法については、「*IBM Tivoli NetView for z/OS* セキュリティー・リファレンス」を参照してください。

ビューおよびビュー内のリソースの制限にスパンを使用するには、 RODM がビューおよびリソースの判別に使用する命名規則をあらかじめ理解しておく必要があります。リソース名およびビュー名は、 NetView スパン表内では、リソース ID およびビュー ID として表されます。これらの ID (ワイルドカード文字を含む場合もあります) は、ビューの作成処理中に GMFHS によって使用される名前と正確に一致していなければなりません。このセクションでは、リソース名およびビュー名を判別するための GMFHS 規則について説明します。

ビュー

106 ページの『オブジェクトの展開処理』で説明したように、GMFHS によって作成されたすべてのビューは、事前定義されたものか、動的に作成されたものかのいずれかに分類することができます。 GMFHS は、ビューが事前定義されているか動的であるかに応じて異なるプロシージャーを使用してビュー名を判別します。

事前定義ビューをスパンに定義する

事前定義ビューは、ユーザーによって定義されます。各事前定義ビューは、RODM 内のビュー・オブジェクトによって表されます。以下のタイプのビューは、RODM に事前定義することができます。

- ネットワーク・ビュー
- 例外ビュー
- 構成対等機能ビュー
- 構成バックボーン・ビュー
- 構成論理ビュー
- 構成物理ビュー
- より詳細な論理ビュー
- より詳細な物理ビュー

ネットワーク・ビュー、例外ビュー、および構成対等機能ビューは、事前定義することしかできず、RODM によって動的に作成されることはありません。上記のリスト内の他のビューは、事前定義することも、動的に作成することもできます。

NetView スパン表内のスパンに事前定義ビューを定義する場合は、ビュー ID がビュー・オブジェクトの MyName 属性と等しくなければなりません。事前定義ビューをスパンに定義する方法について、次の例を使って考えてみます。ネットワーク・ビューが RODM に事前定義され、 MyName フィールドが `MY_NETWORK_VIEW` と等しいとします。 NGMFVSPN 属性の *span_level* の位置で、ビュー名に対するスパン許可を検査するように指定されていると、 GMFHS は、ビューを要求しているオペレーターにビュー名 `MY_NETWORK_VIEW` に対するスパン許可があるかどうかを検査します。

以下のステートメントが NetView スパン表で定義されている場合、スパン SPAN1 を開始しているオペレーターは、そのビューにアクセスすることができます。

```
SPANDEF SPAN=SPAN1,VIEW=MY_NETWORK_VIEW;
```

もう 1 つの方法として、`MY_NETWORK_VIEW` ビュー名に一致するワイルドカード文字を使用して、 SPANDEF ステートメントを定義することもできます。次に例をいくつか示します。

- `SPANDEF SPAN=SPAN1,VIEW=*VIEW;`
- `SPANDEF SPAN=SPAN1,VIEW=M*;`
- `SPANDEF SPAN=SPAN1,VIEW=*NETWORK*;`

動的に作成されたビューをスパンに定義する

動的に作成されたビューは RODM 内のビュー・オブジェクトによっては表されません。動的に作成されたビューを NetView スパン表内のスパンに定義する場合は、ビュー ID が、選択されたリソースの DisplayResourceName フィールド (ただし、ビューのタイプを指定する 3、4 文字のサフィックスが追加されたもの) と等しくなければなりません。

以下のタイプのビューは、GMFHS によって動的に作成することができます。

ビュー・タイプ	**サフィックス**
構成バックボーン	-BAK
構成子	-CHD
構成子 II (より詳細な LU)	-MLU

構成子 III (より詳細な定義グループ)	
	-MDF
構成論理	-LOG
構成論理/物理	-LP
構成親	-PAR
構成物理	-PHY
高速パス	-FP
より詳細な論理	-MDL
より詳細な物理	-MDP

注: ハイフンはサフィックスの一部です。

次の例では、動的に作成されたビューをスパンに定義する方法を示します。DisplayResourceName フィールドが `MyAggResource` と同じになっている集合リソースに、 NMC により検出された障害のあるリソースのビューが選択されているとします。 NGMFVSPN 属性の span_level の位置で、ビュー名に対するスパン検査が指定されていると、 GMFHS は、ビューを要求しているオペレーターにビュー名 `MyAggResource-FP` に対するスパン許可があるかどうかを検査します。

別の例として、 DisplayResourceName フィールドが `NETA.NCP1` と等しい実リソースに、構成親ビューが選択されている場合を考えてみます。 NGMFVSPN 属性の span_level の位置で、ビュー名に対するスパン検査が指定されていると、GMFHS は、ビューを要求しているオペレーターにビュー名 `NETA.NCP1-PAR` に対するスパン許可があるかどうかを検査します。

スパンにビュー (特に動的に作成されたビュー) を定義する場合は、ワイルドカード文字を使用すると便利です。ワイルドカード文字の詳細については、「*IBM Tivoli NetView for z/OS* セキュリティー・リファレンス」を参照してください。

スパンへのビューの定義例

スパンにビューを定義する方法を理解するために、以下に例を示します。この例では以下を前提としています。

- CTL=SPECIFIC が、ビューを要求しているオペレーターに定義されている。
- NGMFVSPN の *span_level* 位置で、ビュー名に対するスパン検査が指定されている。
- ビューを要求しているオペレーターがスパン SPAN1 を開始している。
- 例で定義されているもの以外は、ビュー名と一致するスパン表に SPANDEF ステートメントが定義されていない。

例 1: スパンにビュー ID を定義する SPANDEF ステートメントが NetView スパン表内に存在しません。 NetView スパン表内の SPANDEF ステートメントによってスパン SPAN1 に 1 つまたは複数のビュー ID が定義するまでは、オペレーターは、どのビューもオープンすることはできません。

例 2: 動的に作成されたビューは、それらが選択されたリソースからビュー名を得るため、リソース ID をそのリソースの名前に基づいてスパンに定義することができます。例えば、ネットワーク A 内のリソース名がすべて文字 `NETA` で始まっており、以下のステートメントが NetView スパン表で定義されているとします。

- `SPANDEF SPAN=SPAN1,VIEW=NETA*;`

スパン SPAN1 を開始しているオペレーターは、 NETA.NCP-FP、NETA_NETWORK_VIEW、NETA.HOST-MDL、 NETA など、ビュー名が NETA で始まるビューをすべて表示することができます。

例 3: リソース名によるオペレーターの制限が不可能な場合でも、ビューへのアクセスをビュー・タイプによって制限できることがあります。例えば、オペレーターが NMC により検出された障害のあるリソースのビューまたはより詳細なビューだけを見られるようにするには、 NetView スパン表内で以下のステートメントを定義します。

- `SPANDEF SPAN=SPAN1,VIEW=(*-FP,*-MD*);`

スパン SPAN1 を開始しているオペレーターは、 NMC により検出された障害のあるリソースのビューまたはより詳細なビューを表示できるようになります。

例 4: ネットワーク A 内のリソースで生成されているものを除くすべての、NMC により検出された障害のあるリソースのビューに対するスパン権限をオペレーターに与えるには、 NetView スパン表内で以下のステートメントを定義します。

- `SPANDEF SPAN=SPAN1,VIEW=(*-FP<NETA*-FP>);`

スパン SPAN1 を開始しているオペレーターは、 DisplayResourceName が文字 NETA で始まるリソースによって生成されたものを除く、 NMC により検出された障害のあるリソースのビューを表示できるようになります。

例 5: より詳細なビューを除くすべてのビューに対するスパン権限をオペレーターに与えるには、 NetView スパン表内で以下のステートメントを定義します。

- `SPANDEF SPAN=SPAN1,VIEW=*<*-M*>;`

スパン SPAN1 を開始しているオペレーターは、タイプがより詳細なビューであるものを除く、すべてのビューを表示できるようになります。

例 6: ビュー名は、最大文字数の 32 文字で切り捨てられます。DisplayResourceName フィールドが 32 文字を超えたリソースがあるとします (例えば DisplayResourceName 値が NETWORKA.OPCENTER22.OPERATOR.SHIFT1)。このリソースを選択して構成親ビューを要求すると、作成される動的ビュー名は、 NETWORKA.OPCENTER22.OPERATOR.SHIFT1-PAR になります。しかし、ビュー名は 32 文字に切り捨てられるため、 NETWORKA.OPCENTER22.OPERATOR-PAR になります。DisplayResourceName が 32 文字であっても、 32 文字のビュー名にはサフィックスも含めなければならないため、名前は切り捨てられます。サフィックスがビュー名から切り捨てられることはありません。

SPANDEF 定義によっては、この切り捨てのためにスパン表内に問題が生じる場合があります。例えば、モニター責任があるオペレーターのシフトを示すようにリソースのグループの DisplayResourceName を設定しているとします。また、SHIFT1 リソースとして指定されたすべてのリソースに対するスパン権限をオペレーターに与えるために、 NetView スパン表内で以下のステートメントを定義しました。

- `SPANDEF SPAN=SPAN1,VIEW=*SHIFT1*;`

ビュー名 NETWORKA.OPCENTER22.OPERATOR-PAR は、この SPANDEF ステートメントと一致しないため、オペレーターはこのビューを表示することはできません。

DisplayResourceName 値の長さを 28 文字未満に設定するか、または DisplayResourceName の切り捨てられた文字を参照しないように SPANDEF ステートメントを定義しなければなりません。

リソース

NGMFVSPN 属性の *span_level* 位置でリソース名のスパン検査を指定している場合は、ビューを要求しているオペレーターについて開始されたスパンに認可されているリソースだけがビューに表示されます。 NetView スパン表内のスパンにリソース ID を定義する前に、 GMFHS によって使用されるリソース名を知り、スパン権限を判別しておく必要があります。

ビューに表示することができ、かつ、シャドー・オブジェクトではないリソースはモニター可能です。例えば、クラス GMFHS_Monitorable_Objects_Parent_Class の下の GMFHS データ・モデルで定義されたすべてのリソースは、モニター可能なオブジェクトです。 RODM のモニター可能なオブジェクトには、必ず以下のフィールドがあります。

- MyName
- DisplayResourceName
- UserSpanName

RODM でオブジェクトを作成する場合は MyName フィールドに値を割り当てることができますが、オブジェクトの作成後に MyName 値を変更することはできません。

DisplayResourceName フィールドは、割り当てたり修正したりすることができます。このフィールドは、NetView 管理コンソール ビューに表示されるリソース名を作成するために使用します。

DisplayResourceName は、GMFHS メソッドの DUIFCLRT で設定することができます。このメソッドは、リソース・オブジェクトの DisplayResourceType フィールドを、 Display_Resource_Type_Class のオブジェクトの Resources フィールドにリンクするために使用します。メソッドが起動されたときに DisplayResourceName がヌルになっていると、メソッドは、DisplayResourceName フィールドの値を MyName フィールドと等しい値に設定します。メソッドが起動されたときに DisplayResourceName がヌルでない場合は、 DisplayResourceName は変更されません。

注: マルチシステム・マネージャー、SNA トポロジー・マネージャー、およびその他のユーザー・アプリケーションは DisplayResourceName を変更することができます。

UserSpanName フィールドを作成したり、変更したりすることもできます。マルチシステム・マネージャーは、他のユーザー・アプリケーションと同じように、UserSpanName フィールドを変更することができます。マルチシステム・マネージャーのこのフィールドの使用方法については、「*IBM Tivoli NetView for z/OS* マルチシステム・マネージャー ユーザーズ・ガイド」を参照してください。

RODM でシャドー・オブジェクトとして定義された SNA オブジェクト、すなわち GMFHS_Shadow_Objects_Class で定義された SNA オブジェクトには、UserSpanName フィールドがありません。 RODM ベース・ビューとワークステーシ

ョン・ベース・ビューの間の整合性を保証するには、 MyName フィールドだけを使用してシャドー・オブジェクトのスパン権限を判別します。シャドー・オブジェクトに対して DisplayResourceName フィールドが定義され、この名前がビューに表示される場合でも、この名前はスパン権限の判別に使用されません。

RODM の使用方法によっては、任意のリソース・オブジェクトのこれらの各フィールドに異なる値を割り当てることができます。例えば、ネットワークに任意のワークステーションを定義する場合、 MyName フィールドを *netid.resource_type.real_resource_name* として定義し、このフィールドを使用してネットワーク内のリソースをトレースすることができます。

その後、そのワークステーションの DisplayResourceName を、ワークステーションを所有するユーザーの *userid* として定義することができます。DisplayResourceName 値がリソース ID としてビューに表示されるため、オペレーターは、障害のあるリソースが位置するオフィスを判別しやすくなります。

また、ワークステーションを含むネットワークの *netid* を UserSpanName として定義することができます。これにより、すべてが同じ *netid* にあるワークステーションのグループを定義するために UserSpanName を使用することができます。

GMFHS は、以下の論理を使用してビュー内のリソースのスパン権限を判別します。

- リソースがシャドー・オブジェクトである場合、スパン権限の判別には常に MyName フィールドを使用する。
- リソースがシャドー・オブジェクトでない場合は、以下の論理を使用する。
  - UserSpanName に値が存在する場合は、スパン権限の判別に UserSpanName フィールドを使用する。
  - UserSpanName に値が存在しないが DisplayResourceName に値が存在する場合は、スパン権限の判別に DisplayResourceName フィールドを使用する。
  - UserSpanName または DisplayResourceName に値が存在しない場合は、スパン権限の判別に MyName フィールドを使用する。

スパンを使用してビュー内のリソースを制限する例

ビュー内のリソースを制限する方法を理解するために、以下に例を示します。これらの例では、以下のことを想定しています。

- CTL=SPECIFIC が、ビューを要求しているオペレーターに定義された。
- NGMFVSPN の span_level 位置でリソース名に対するスパン検査が指定されている。
- ビューを要求しているオペレーターがスパン SPAN1 を開始した。
- リソース名に一致するスパン表に、他の SPANDEF ステートメントが定義されていない。

注: CHARVAR フィールドの長さがゼロ (0) である場合は、ヌルであると見なされます。 MyName、DisplayResourceName、および UserSpanName は、すべて CHARVAR フィールドです。

例 1: DisplayResourceName および UserSpanName が両方ともヌルである場合は、MyName フィールドはリソースに対するスパン権限を判別します。例えば、RODM

内のモニター可能なリソースの MyName 値が DECNET.RALV4 であるとします。 DisplayResourceName および UserSpanName はヌルです。 NetView スパン表で以下のステートメントを定義します。

- `SPANDEF SPAN=SPAN1,RESOURCE=DECNET.RALV4;`

これによって、スパン SPAN1 を開始しているオペレーターは、ビュー内にリソース DECNET.RALV4 を表示することができます。

例 2: UserSpanName がヌルであり、 DisplayResourceName に値がある (つまり DisplayResourceName がヌルでない) 場合は、 DisplayResourceName フィールドはリソースに対するスパン権限を判別します。例えば、RODM 内のモニター可能なリソースの MyName 値が DECNET.RALV4、また DisplayResourceName 値が RALV4 であるとします。 UserSpanName はヌルです。 NetView スパン表で以下のステートメントを定義します。

- `SPANDEF SPAN=SPAN1,RESOURCE=RALV4;`

スパン SPAN1 を開始しているオペレーターは、ビュー内にこのリソースを表示することができます。 DisplayResourceName がヌルでなく、リソースがシャドー・オブジェクトではないため、 DisplayResourceName フィールドがスパン権限を判別します。

この場合、リソース定義でワイルドカードを使用すると便利です。直前のステートメントの代わりにステートメントが NetView スパン表で定義されている場合、スパン SPAN1 を開始しているオペレーターは、 DisplayResourceName 値が RALV4 であってもそうでなくても、このリソースを表示することができます。
DisplayResourceName がヌルである場合は、スパン権限の判別に MyName 値 DECNET.RALV4 が使用されます。例えば、次のようになります。

- `SPANDEF SPAN=SPAN1,RESOURCE=*RALV4;`

例 3: DisplayResourceName は、ビューに表示するリソース名を作成するために使用されます。 DisplayResourceName 値は、ビュー内に表示するリソースを記述するには便利ですが、スパン権限を判別するには不便な場合があります。この値は、UserSpanName フィールドを設定することによってオーバーライドすることができます。この場合にも DisplayResourceName はビューに表示されますが、スパン権限には UserSpanName 値が使用されます。

例えば、RODM 内のモニター可能なリソースが次のようになっているとします。
- MyName 値が DECNET.RALV4
- DisplayResourceName 値が RALV4
- UserSpanName 値が BUILDING500.RALV4

この例では、NetView スパン表に以下のステートメントを定義します。

- `SPANDEF SPAN=SPAN1,RESOURCE=BUILDING500.*;`

スパン SPAN1 を開始しているオペレーターは、リソース DECNET.RALV4 をビューに表示することができます。

ここで、前記のステートメントの代わりに NetView スパン表に以下のステートメントの 1 つが定義されたと仮定します。
- `SPANDEF SPAN=SPAN1,RESOURCE=DECNET.RALV4;`

- `SPANDEF SPAN=SPAN1,RESOURCE=RALV4;`

この場合、オペレーターは、リソースに対するスパン権限を拒絶されます。 UserSpanName に値があるため、これを使用してリソースに対するスパン権限を判別します。 UserSpanName に値がある場合には、 DisplayResourceName および MyName はスパン権限の判別に使用されません。

有用なヒント

場合によっては、リソース、ビュー、およびスパンの定義が予想どおりの結果を生みません。以下のセクションでは、予期しない状態が起こったときにそれをデバッグするのに使用できる有用なヒントをいくつか説明します。

ビュー・リスト内のビューがオペレーターの制御スパン内にない

制御スパンがビュー・レベルでビューに適用される場合は、すべてのビューは、オープンされる前 (ほとんどの場合はビュー・リストに入れられる前) に、スパン検査が行われます。ビュー・リスト内のどのビューも当該オペレーターの制御スパン内にない場合は、 NGMFVSPN 値に基づいて情報メッセージが出され、ビュー・リストが返されない理由が示されます。

ビュー内のリソースがオペレーターの制御スパン内にない

制御スパンがリソース・レベルでビューに適用される場合は、ビューがオープンされる前に、ビュー内のすべてのリソースについてスパン検査が行われます。ビュー内のどのリソースも当該オペレーターの制御スパン内にない場合は、情報メッセージが出され、ビューがオープンされない理由が示されます。

選択されたオブジェクトがオペレーターの制御スパン内にない

当該オペレーターの制御スパン内にないリソースについてリソース位置指定が要求された場合は、情報メッセージが出され、ビューがオープンされない理由が示されます。

同様に、オープン・ビュー内で選択されているが、当該オペレーターの制御スパン内にないリソースについてビュー (より詳細なビューなど) が要求された場合も、情報メッセージが出され、そのビューがオープンされない理由が示されます。この状態は、以下に該当する場合にのみ発生します。

- オペレーターが、NetView スパン表内で定義されたリソースに対応するスパンを停止した。
- オペレーターが開始したスパンにこれ以上リソースが定義されないように、NetView スパン表が変更された (その後更新された)。

NetView スパン表が変更されたり、スパンが開始または停止したりすることが原因で、リソースがオープン・ビューから取り除かれることはありません。オープン・ビューが更新されると、初めてこのような変更が行われます。

NGMFVSPN 属性を変更する

NMC オペレーターのプロファイルで割り当てられた NGMFVSPN 属性は、その NMC オペレーターのセッションの間だけ有効になっています。変更された NGMFVSPN 属性が検索されるのは、 NetView オペレーターがサインオフして新規

の NGMFVSPN 属性でサインオンし直し、 NetView オペレーターがサインオンし直した後で NMC オペレーターがサインオフしてサインオンし直した場合だけです。

このような制約事項があるために、NGMFVSPN 属性を変更しても、オープンしている NetView 管理コンソール ビューは影響を受けません。オペレーターが再びサインオンすると、すべての NetView 管理コンソール ビューが更新されます。

RACF を RODM セキュリティーに使用する

RODM のセキュリティーに RACF を使用している場合は、 NetView ドメイン名が RACF に定義されており、最低限でも RODM のセキュリティー・レベル 2 になっていることを確認してください。これらのセキュリティー要件が満たされていない場合、RODM 照会が失敗し、スパン権限エラーとなります。

制御スパンを設定オペレーター状況およびクリア・オペレーター状況に適用する

制御スパンは、設定オペレーター状況アクションおよびクリア・オペレーター状況アクションの以下のサブセットに適用されます。

- Marker (マーカー)
- Suspended (保留、手動でクリア)
- Suspended (保留、自動でクリア)

制御スパンに対するアクセス・レベル UPDATE(U) をオペレーターが持つ場合は、スパン内で選択されたリソースに対するマーカー・アクションまたは中断アクションが完了し、オペレーターの要求に応じてオペレーター状況が設定またはクリアされます。アクセス・レベル UPDATE (U) は、制御スパン内のリソースに対するマーカー・アクションおよび中断アクションに必要です。

リソースを含む制御スパンに対するアクセス・レベル READ(R) しかオペレーターが持たない場合、またはオペレーターがアクセスしたスパン内にリソースがない場合は、選択されたリソースに対するマーカー・アクションまたは中断アクションは無視されます。

シャドー・オブジェクトも含め、 VTAM リソースに対するマーカー・アクションまたは中断アクションに対しては、コマンドと同様の方法でスパン検査が行われます。 NetView スパン表を使用しているのであれば、 RODM オブジェクトのマーカー・アクションおよび中断アクションのスパン検査には、 UserSpanName、DisplayResourceName、および MyName フィールドの階層が利用されます。

マーカー・アクションおよび中断オプションは、制御スパンのオプションではありません。制御スパンが実装される場合は、オペレーターに対するアクティブのスパンには、マーカー・アクションおよび中断アクションを受信しているリソースへの UPDATE (U) アクセスを含めておかなければなりません。

- UserSpanName、DisplayResourceName、および MyName フィールドの階層に関する詳細については、 139 ページの『リソース』を参照してください。
- リソースを保護するためのスパンの使用の詳細については、「*IBM Tivoli NetView for z/OS* セキュリティー・リファレンス」を参照してください。

ポリシーをビューに適用する

NMCSTATUS ポリシー定義を使用すると、NMC ビュー内のリソースに対して時間スケジュールを定義できます。これらのスケジュールとともにビューに対してポリシーが適用され、ビュー内の 1 つ以上のリソースの表示可能状況が NMC コンソールでいつ使用不可になるか、またはビュー内の 1 つ以上のリソースの集約がいつ中断されるかを指定します。

NMCSTATUS ポリシー定義が処理されると、CHRON タイマーが設定されて、ポリシーがアクティブになる時と非アクティブになる時を指定します。各ポリシー定義は、リソースおよびアクションのグループが、指定された時間枠にそのリソースのグループに適用されるように指定します。

開始タイマーがポップすると、ポリシーはアクティブになります。 NMCSTATUS ポリシー・コードは Aggregate_Collection_Class 内に RODM オブジェクトを作成し、ポリシー定義を表します。これは RODM Collection Manager を起動し、GMFHS_Aggregate_Objects_Class 内に集合オブジェクトを作成して、Aggregate_Collection_Class 内のオブジェクトの RODM フィールド値に基づくリソース・オブジェクトのコレクションを表します。コレクションに属するリソースは、 AggregateParent/AggregateChild フィールドおよび ComposedOfLogical/IsPartOf フィールドを経由して集合体にリンクされます。ポリシー定義で指定されたアクションは、コレクション内のすべてのリソースに適用されます。

終了タイマーがポップすると、ポリシーは非アクティブになります。 NMCSTATUS ポリシー・コードは、Aggregate_Collection_Class から RODM オブジェクトを削除します。これは RODM Collection Manager を起動し、ポリシーに属するリソース・オブジェクトのコレクションを表す、 GMFHS_Aggregate_Objects_Class 内の対応する集合オブジェクトを削除します。コレクションと一致するリソース・オブジェクトはどれも、コレクションから除去されます。ポリシー定義に基づいて、状況更新が再開され、中断されていたリソースは再開されます。リソース・オブジェクトが別のアクティブ・ポリシーに属する場合、それはコレクションから除去されません。詳細については、146 ページの『複数のポリシーに属するリソース』を参照してください。

RODM でポリシー定義を表す

各アクティブ・ポリシーは、Aggregate_Collection_Class 内のオブジェクトによって、RODM で表されます。 NMCSTATUS キーワードからの値を使用して、オブジェクトの RODM フィールドを設定します。以下はオブジェクトのキー・フィールドのリストで、値がポリシー定義から取られる方法を示しています。

MyName オブジェクトの名前は、ポップされる CHRON タイマーのタイマー・ハンドルを連結することによって作成され、ポリシー定義の名前とともにポリシーの開始を示します。例えば、タイマー・ハンドル NMC1 がポリシー定義 POLICY1 の開始タイマーである場合、RODM オブジェクトの MyName フィールドは、NMC1POLICY1 に設定されます。

CollectionSpec1

リソースのコレクションを指定する RODM Collection Manager の言語は、 CLASS、MYNAME、および RESOURCE キーワード、または BLDVIEWSSPEC キーワードあるいは COLLECTIONSPEC キ

ーワードから生成されます。 CollectionSpec1 には 32K のデータが含まれます。値が 32K より大きい場合、追加データが、必要に応じて RODM フィールドの CollectionSpec2、 CollectionSpec3、または CollectionSpec4 に保管されます。このそれぞれのフィールドにも 32K のデータが含まれ、 GMFHS データ・モデルに定義されます (DUIFSTRC)。

RequestFlags
ポリシーに適用するアクションを示します。キーワード SUSPENDAGG=YES が指定される場合、アクションはコレクション内のすべてのリソースを中断します。キーワード STOPUPDATE=YES が指定される場合、アクションはコレクション内のリソースに対して、 NMC コンソールでのシステム状況の更新を使用不可にします。両方のアクションを同じリソースのコレクションに適用することができます。

CollectionLocateName
'NMCSTATUS' の値がこの索引付けされたリストに追加され、オブジェクトがポリシー定義を表すように指定します。

例 1: 午前 6 時に RODM オブジェクトが Aggregate_Collection_Class に作成され、以下の例に示すフィールド値を持ちます。タイマー・ハンドルは NMC1 です。

```
Policy definition:
  NMCSTATUS POLICY1
   CLASS=(GMFHS_Managed_Real_Objects_Class)
   TIME=(06.00.00,18.00.00)
   STOPUPDATE=YES

RODM field values:
  MyName='NMC1POLICY1'
  CollectionSpec1='|GMFHS_Managed_Real_Objects_Class|MyName|*|.CONTAINS.'
  RequestFlags='80000000'x
  CollectionLocateName='NMCSTATUS'
```

例 2: 午前 6 時に RODM オブジェクトが Aggregate_Collection_Class に作成され、以下の例に示すフィールド値を持ちます。タイマー・ハンドルは NMC1 です。

```
Policy definition:
  NMCSTATUS POLICY2
   CLASS=(GMFHS_Managed_Real_Objects_Class)
   RESOURCE=(RALV4)
   TIME=(06.00.00,18.00.00)
   STOPUPDATE=YES
   SUSPENDAGG=YES

RODM field values:
  MyName='NMC1POLICY2'
  CollectionSpec1='|GMFHS_Managed_Real_Objects_Class|
      DisplayResourceName|RALV4|.EQ.'
  RequestFlags='C0000000'x
  CollectionLocateName='NMCSTATUS'
```

例 3: 午前 6 時に RODM オブジェクトが Aggregate_Collection_Class に作成され、以下の例に示すフィールド値を持ちます。タイマー・ハンドルは NMC1 です。

```
Policy definition:
  NMCSTATUS POLICY3
   CLASS=(GMFHS_Managed_Real_Objects_Class)
   MYNAME=(DEC*)
   TIME=(06.00.00,18.00.00)
   SUSPENDAGG=YES
```

```
RODM field values:
  MyName='NMC1POLICY3'
  CollectionSpec1='|GMFHS_Managed_Real_Objects_Class|MyName|DEC*|.CONTAINS.'
  RequestFlags='40000000'x
  CollectionLocateName='NMCSTATUS'
```

例 4: 午前 6 時に RODM オブジェクトが Aggregate_Collection_Class に作成され、以下の例に示すフィールド値を持ちます。タイマー・ハンドルは NMC1 です。

```
FILE1 contains the following BLDVIEWS statements:
  Majnode=NETA.A01M,
  Type=XCA

Policy definition:
  NMCSTATUS POLICY4
   BLDVIEWSSPEC=(QSAMDSN,USER.INIT(FILE1))
   TIME=(06.00.00,18.00.00)
   STOPUPDATE=YES

RODM field values:
  MyName='NMC1POLICY4'
  CollectionSpec1='|1.3.18.0.0.3315.8.3.7|MyName|1.3.18.0.2.4.6=*;
                   1.3.18.0.0.2032=*;1.3.18.0.0.2032=XCA.NETA.A01M|.CONTAINS.'
  RequestFlags='80000000'x
  CollectionLocateName='NMCSTATUS'
```

例 5 午前 6 時に RODM オブジェクトが Aggregate_Collection_Class に作成され、以下の例に示すフィールド値を持ちます。タイマー・ハンドルは NMC1 です。

```
DDFFILE2 is a data definition file allocated with command
  ALLOCATE FILE(DDFFILE2) DATASET(USER.INIT(FILE2)) SHR

DDFFILE2 contains the following BLDVIEWS statements:
  NONSNA=*

Policy definition:
  NMCSTATUS POLICY5
   BLDVIEWSSPEC=(QSAMDD,DDFFILE2)
   TIME=(06.00.00,18.00.00)
   STOPUPDATE=YES

RODM field values:
  MyName='NMC1POLICY5'
  CollectionSpec1='|GMFHS_Managed_Real_Objects_Class|MyName|*|.CONTAINS.'
  RequestFlags='80000000'x
  CollectionLocateName='NMCSTATUS'
```

複数のポリシーに属するリソース

リソースは、複数のポリシー定義に定義できます。リソースが属するアクティブ・ポリシーの数は、カウンター・フィールドに保存されます。それぞれの表示可能リソース・オブジェクトには、定義済みの 2 つのカウンター・フィールドがあります。

PolicyCtrSU このリソースが属するアクティブ・ポリシーの数を表します。ここでリソースに対して適用されるアクションは、*更新の停止* です。

PolicyCtrSA このリソースが属するアクティブ・ポリシーの数を表します。ここでリソースに対して適用されるアクションは、*集約の中断* です。

これらのフィールドがあるために、アクションは他のアクティブ・ポリシーに属するリソースから決して除去されません。リソースがポリシーから除去されると、該当するカウンターが 1 減少します。カウンターがゼロになると、アクションはリソ

ースから除去されます。カウンターがゼロでない場合、リソースは他のアクティブ・ポリシーに属し、アクションはそのまま残ります。

例 1: POLICY1 は、土曜日に状況の更新がリソース ABC に送信されないことを指定します。 POLICY2 は、午前 8 時から午前 10 時の毎日 (土曜日を含む) に、状況の更新が A で始まる実リソース (例えば、RESOURCE=A*) に送信されないことを指定します。

```
Policy definitions:
  NMCSTATUS POLICY1
   CLASS=(GMFHS_Managed_Real_Objects_Class)
   RESOURCE=(ABC)
   DAYOFWEEK=(SAT)
   TIME=(00.00.00,23.59.59)
   STOPUPDATE=YES
  NMCSTATUS POLICY2
   CLASS=(GMFHS_Managed_Real_Objects_Class)
   RESOURCE=(A*)
   TIME=(08.00.00,10.00.00)
   STOPUPDATE=YES
```

1. 土曜日の午前 0 時にタイマーがポップされて、POLICY1 がアクティブになります。リソース ABC の PolicyCtrSU フィールドが 1 増加します。リソース ABC の PolicyCtrSU=1 および状況の更新はリソースに送信されません。
2. 土曜日の午前 8 時にタイマーがポップされて、POLICY2 がアクティブになります。コレクション内のすべての実リソース A* の PolicyCtrSU フィールドが 1 増加します。リソースは両方のコレクションに属するため、リソース ABC は PolicyCtrSU=2 になります。リソースの PolicyCtrSU=1 は、POLICY2 コレクションにのみ属しています。状況の更新は、PolicyCtrSU フィールドがゼロでないリソースのいずれにも送信されません。
3. 土曜日の午前 10 時、タイマーがポップされて、POLICY2 が非アクティブになります。コレクション内のすべての実リソース A* の PolicyCtrSU フィールドが 1 減少します。リソースは依然 POLICY1 コレクションに属しているため、リソース ABC は PolicyCtrSU=1 になります。リソースの PolicyCtrSU=0 は、POLICY2 コレクションにのみ属しています。状況の更新はこれらのリソースに送信されますが、リソース ABC には送信されません。
4. 土曜日の午後 11 時 59 分にタイマーがポップされて、POLICY1 が非アクティブになります。リソース ABC の PolicyCtrSU フィールドが 1 減少します。リソース ABC は PolicyCtrSU=0 になります。これで状況の更新が送信されます。

例 2: POLICY1 は、土曜日にリソース ABC の集約が中断されるように指定します。 POLICY2 は、午前 8 時から午前 10 時の毎日 (土曜日を含む) に、 A で始まる実リソース (RESOURCE=A*) の集約が中断されるように指定します。

```
Policy definitions:
  NMCSTATUS POLICY1
   CLASS=(GMFHS_Managed_Real_Objects_Class)
   RESOURCE=(ABC)
   DAYOFWEEK=(SAT)
   TIME=(00.00.00,23.59.59)
   SUSPENDAGG=YES
  NMCSTATUS POLICY2
   CLASS=(GMFHS_Managed_Real_Objects_Class)
   RESOURCE=(A*)
   TIME=(08.00.00,10.00.00)
   SUSPENDAGG=YES
```

1. 土曜日の午前 0 時にタイマーがポップされて、POLICY1 がアクティブになります。リソース ABC の PolicyCtrSA フィールドが 1 増加します。リソース ABC は PolicyCtrSA=1 になり、リソース ABC の集約が中断されます。
2. 土曜日の午前 8 時にタイマーがポップされて、POLICY2 がアクティブになります。コレクション内のすべての実リソース A* の PolicyCtrSA フィールドが 1 増加します。リソースは両方のコレクションに属するため、リソース ABC は PolicyCtrSA=2 になります。リソースの PolicyCtrSA=1 は、POLICY2 コレクションにのみ属しています。 PolicyCtrSA フィールドがゼロでないリソースの集約は、いずれも中断されます。
3. 土曜日の午前 10 時、タイマーがポップされて、POLICY2 が非アクティブになります。コレクション内のすべての実リソース A* の PolicyCtrSA フィールドが 1 減少します。リソースは依然 POLICY1 コレクションに属しているため、リソース ABC は PolicyCtrSA=1 になります。リソースの PolicyCtrSA=0 は、POLICY2 コレクションにのみ属しています。これらのリソースの集約は中断されなくなりますが、リソース ABC に関しては中断されます。
4. 土曜日の午後 11 時 59 分にタイマーがポップされて、POLICY1 が非アクティブになります。リソース ABC の PolicyCtrSA フィールドが 1 減少します。リソース ABC は PolicyCtrSA=0 になります。リソースの集約は中断されなくなります。

例 3: NMC オペレーターは、ポリシーによって現在集約が中断されているリソースの集約を再開できます。中断フラグをセットすることによって、また NMC から中断フラグをクリアすることによって、アクティブなポリシーをオーバーライドします。ただし、PolicyCtrSA フィールドは、リソースがコレクションに追加、またはコレクションから除去される場合にのみ、増加および減少します。この例では、POLICY1 は土曜日にリソース PC1 の集約が中断されるように指定します。POLICY2 は、午前 8 時から午前 10 時の毎日 (土曜日を含む) に、リソース PC1 の集約が中断されるように指定します。オペレーターはリソースの中断フラグの値を変更できます。ただし、ポリシーはアクティブおよび非アクティブになるときに中断フラグの更新を継続します。

```
Policy definitions:
  NMCSTATUS POLICY1
   CLASS=(GMFHS_Managed_Real_Objects_Class)
   RESOURCE=(PC1)
   DAYOFWEEK=(SAT)
   TIME=(00.00.00,23.59.59)
   SUSPENDAGG=YES
  NMCSTATUS POLICY2
   CLASS=(GMFHS_Managed_Real_Objects_Class)
   RESOURCE=(PC1)
   TIME=(08.00.00,10.00.00)
   SUSPENDAGG=YES
```

1. 土曜日の午前 0 時にタイマーがポップされて、POLICY1 がアクティブになります。リソース PC1 の PolicyCtrSA フィールドが 1 増加します。リソース PC1 は PolicyCtrSA=1 になり、リソース PC1 の集約が中断されます。
2. 土曜日の午前 8 時にタイマーがポップされて、POLICY2 がアクティブになります。リソース PC1 の PolicyCtrSA フィールドが 1 増加します。リソースは両方のコレクションに属するため、リソース PC1 は PolicyCtrSA=2 になります。リソースの集約は中断されたままになります。

3. 土曜日の午前 10 時、タイマーがポップされて、POLICY2 が非アクティブになります。リソース PC1 の PolicyCtrSA フィールドが 1 減少します。リソースは依然 POLICY1 コレクションに属しているため、リソース PC1 は PolicyCtrSA=1 になります。リソースの集約は中断されたままになります。
4. 土曜日の午後 3 時に、NMC オペレーターはリソース PC1 の中断フラグをクリアします。 PolicyCtrSA は未変更のままですが (依然として 1 に等しい)、リソースの集約は中断されなくなります。
5. 土曜日の午後 11 時 59 分 59 秒にタイマーがポップされて、POLICY1 が非アクティブになります。リソース ABC の PolicyCtrSA フィールドが 1 減少します。リソース ABC は PolicyCtrSA=0 になります。この例では、中断フラグはすでにクリアされていますが、クリアされていなければクリアされ、リソース PC1 の集約は中断されなくなります。

NMC オペレーターがリソースの中断フラグの値を変更できますが、ポリシーはアクティブおよび非アクティブになるときに中断フラグの更新を継続します。

例 4: ポリシーは、リソースの集約が中断されて、状況を受け取らないように指定できます。このような状況では両方のカウンターが使用され、各アクションごとにリソースが属するアクティブ・ポリシーの数を監視します。この例では、POLICY1 は土曜日に状況の更新がリソース PC1 に送信されないように指定します。 POLICY2 は、土曜日の午前 8 時から午後 5 時までリソース PC1 の集約が中断されるように指定します。 POLICY3 は、土曜日の午後 2 時から午後 4 時まで状況の更新がリソース PC1 に送信されず、リソース PC1 の集約が中断されるように指定します。

```
Policy definitions:
  NMCSTATUS POLICY1
   CLASS=(GMFHS_Managed_Real_Objects_Class)
   RESOURCE=(PC1)
   DAYOFWEEK=(SAT)
   TIME=(00.00.00,23.59.59)
   STOPUPDATE=YES
  NMCSTATUS POLICY2
   CLASS=(GMFHS_Managed_Real_Objects_Class)
   RESOURCE=(PC1)
   DAYOFWEEK=(SAT)
   TIME=(08.00.00,17.00.00)
   SUSPENDAGG=YES
  NMCSTATUS POLICY3
   CLASS=(GMFHS_Managed_Real_Objects_Class)
   RESOURCE=(PC1)
   DAYOFWEEK=(SAT)
   TIME=(14.00.00,16.00.00)
   STOPUPDATE=YES
   SUSPENDAGG=YES
```

1. 土曜日の午前 0 時にタイマーがポップされて、POLICY1 がアクティブになります。リソース PC1 の PolicyCtrSU フィールドが 1 増加します。カウンター・フィールド値は、PolicyCtrSA=0 および PolicyCtrSU=1 です。状況の更新は、リソース PC1 に送信されなくなります。
2. 土曜日の午前 8 時にタイマーがポップされて、POLICY2 がアクティブになります。リソース PC1 の PolicyCtrSA フィールドが 1 増加します。カウンター・フィールド値は PolicyCtrSA=1 および PolicyCtrSU=1 です。状況の更新は依然リソース PC1 に送信されず、リソースの集約も中断されます。

3. 土曜日の午後 2 時にタイマーがポップされて、POLICY3 がアクティブになります。両方のカウンター・フィールドが 1 増加します。カウンター・フィールド値は PolicyCtrSA=2 および PolicyCtrSU=2 です。状況の更新は依然リソース PC1 に送信されず、リソースの集約は中断された状態のままです。
4. 土曜日の午後 4 時にタイマーがポップされて、POLICY3 が非アクティブになります。両方のカウンター・フィールドが 1 減少します。カウンター・フィールド値は PolicyCtrSA=1 および PolicyCtrSU=1 です。状況の更新は依然リソース PC1 に送信されず、リソースの集約は中断されたままです。
5. 土曜日の午後 5 時にタイマーがポップされて、POLICY2 が非アクティブになります。リソース PC1 の PolicyCtrSA フィールドが 1 減少します。カウンター・フィールド値は PolicyCtrSA=0 および PolicyCtrSU=1 です。状況の更新は依然リソース PC1 に送信されません。リソースの集約は中断されなくなります。
6. 土曜日の午後 11 時 59 分 59 秒にタイマーがポップされて、POLICY1 が非アクティブになります。リソース ABC の PolicyCtrSU フィールドが 1 減少します。カウンター・フィールド値は PolicyCtrSA=0 および PolicyCtrSU=0 です。これで状況の更新はリソース PC1 に送信されます。

ポリシーにより集約が中断されているリソース

スケジュール済みのポリシー定義により実リソースの集約が中断されていると、リソースはポリシーを表すコレクションに追加され、GMFHS 内で以下のことが生じます。

- リソースの中断フラグがセットされます。
- リソースの中断フラグの注釈が *Scheduled* に設定されます。
- リソースの PolicyCtrSA に 1 が追加されます。

ポリシー定義により実リソースの集約が再開すると、リソースはポリシーを表すコレクションから除去され、 GMFHS 内で以下のことが生じます。

- リソースの中断フラグがクリアされます。
- リソースの中断フラグの注釈がクリアされます。
- リソースの PolicyCtrSA から 1 が引かれます。

中断フラグは、注釈の値が ″Scheduled″ で、オペレーター ID *GMFHS* によって設定された場合にのみクリアされます。

ポリシー定義が SUSPENDAGG=YES および STOPUPDATE=NO を指定する場合、影響を受けるリソースは *Scheduled* システム状況に変更されません。リソースの集約は中断されますが、システム状況の更新は受け取り続けます。

NMC オペレーターは、中断フラグの設定をオーバーライドできます。詳細については、 146 ページの『複数のポリシーに属するリソース』を参照してください。

集合体を使用した集約の中断

集合体の集約が中断されるとき、集合体そのものの集約は中断されません。代わりに、現在集合体に状況を報告しているすべての実オブジェクトの集約が中断されます。 GMFHS 内で以下のことが生じます。

- 実リソースの中断フラグがセットされます。
- 実リソースの中断フラグの注釈が *Scheduled* に設定されます。

- 実リソースの PolicyCtrSA に 1 が追加されます。
- 集合体の子の中断フラグがセットされます。
- 集合体の子の中断フラグの注釈が *Scheduled* に設定されます。

子の中断フラグは、ポリシーによって影響を受ける集合体とその集合体に状況を報告する実リソースの間の、 AggregateChild/AggregateParent パス内の集合体にも設定されます。ただし、子の中断フラグの注釈フィールドは、これらの中間集合リソースの場合、 *Scheduled* に設定されません。

集合体の集約が再開されるとき、集合体そのものが再開されるのではありません。代わりに、現在集合体に状況を報告しているすべての実オブジェクトの集約が再開されます。 GMFHS 内で以下のことが生じます。
- 実リソースの中断フラグがクリアされます。
- 実リソースの中断フラグの注釈がクリアされます。
- 実リソースの PolicyCtrSA から 1 が引かれます。
- 集合体の子の中断フラグがクリアされます。
- 集合体の子の中断フラグの注釈がクリアされます。

例: AGGPOLICY は、土曜日に集合リソース AGG1 の集約が中断されるように指定します。

```
Policy definitions:
  NMCSTATUS AGGPOLICY
   CLASS=(GMFHS_Aggregate_Objects_Class)
   RESOURCE=(AGG1)
   DAYOFWEEK=(SAT)
   TIME=(00.00.00,23.59.59)
   SUSPENDAGG=YES
```

1. 土曜日の午前 0 時にタイマーがポップされて、AGGPOLICY がアクティブになります。集合リソース AGG1 がコレクションに追加され、アクション (集約の中断) がリソースに適用されます。集合体の集約を中断することは、状況を現在集合体に報告しているすべての実リソースの集約を中断するためのショートカット要求になります。それぞれの実リソースの PolicyCtrSA フィールドが 1 増加します。集合体そのものは中断されないので、集合体の PolicyCtrSA フィールドは更新されません。
2. 土曜日の午後 11 時 59 分 59 秒にタイマーがポップされて、AGGPOLICY が非アクティブになります。集合リソース AGG1 がコレクションから除去され、アクション (集約の中断) が各リソースから除去されます。集合体の集約の中断を解除することは、状況を現在集合体に報告しているすべての実リソースの集約を再開するためのショートカット要求になります。それぞれの実リソースの PolicyCtrSA フィールドが 1 減少します。集合体そのものは中断されず、また中断解除できないので、集合体の PolicyCtrSA フィールドは更新されません。

ポリシーがアクティブになった後に追加の実リソースが集合体 AGG1 に状況を報告し始めた場合、それらはポリシー定義 AGGPOLICY によって中断されません。アクションは、コレクションのメンバーに対してのみ適用されます。コレクションのメンバーである集合体 AGG1 に対するアクションによってのみ、実リソースは中断され、再開されます。

ポリシーによりリソースに送信されないシステム状況の更新

システム状況の更新が起きると、リソースの DisplayStatus フィールドが新しい状況によって更新されます。 DisplayStatus フィールドの変更により、オープンしている NMC ビューにリソースが表示される場合、リソースの更新が行われます。

スケジュール済みのポリシー定義によりシステム状況の更新がリソースに送信されなくなると、リソースはポリシーを表すコレクションに追加されます。これがリソースが属する唯一のアクティブ・ポリシーである場合、GMFHS 内で以下のことが生じます。

- PolicyDisplayStatus フィールドは、DisplayStatus フィールドの現行値に設定されます。
- DisplayStatus フィールドが *Scheduled* に設定されます。
- オープンしている NMC ビューにリソースが表示される場合、システム状況の更新はリソースに *Scheduled* を送信します。
- リソースの PolicyCtrSU フィールドに 1 が追加されます。

アクティブ・ポリシーに属している間にこのリソースで受け取られたシステム状況の更新はすべて、 DisplayStatus フィールドではなく PolicyDisplayStatus フィールドに保存されます。こうして、システム状況の更新は NMC に送信されません。

システム状況の更新が再開されると、リソースはポリシーを表すコレクションから除去されます。 GMFHS 内で以下のことが生じます。

- リソースの PolicyCtrSU フィールドから 1 が引かれます。
- リソースの PolicyCtrSU フィールドが 0 の場合、 DisplayStatus フィールドは PolicyDisplayStatus フィールドの現行値に設定されます。これにより NMC は更新され、リソースの状況は *Scheduled* 状況からその現行のシステム状況に変更されます。
- リソースの PolicyCtrSU フィールドがゼロより大きい場合、 DisplayStatus フィールドは *Scheduled* のままとなり、システム状況の更新はすべて PolicyDisplayStatus フィールドに保存されます。 STOPUPDATE=YES が指定されていると、リソースがポリシーを表すコレクションに属している間、更新は送信されません。

追加情報

NMCSTATUS ポリシー定義を含むポリシー・ファイルの作成およびロードの詳細については、「*IBM Tivoli NetView for z/OS アドミニストレーション・リファレンス*」を参照してください。

RODM Collection Manager の詳細については、「*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。

NMCSTATUS ポリシー定義を処理するために必要なタスクの詳細については、「*IBM Tivoli NetView for z/OS インストール: グラフィカル・コンポーネントの構成*」を参照してください。

特定の RODM フィールドの詳細については、「*IBM Tivoli NetView for z/OS データ・モデル・リファレンス*」を参照してください。

集約の概念

このセクションでは、ネットワーク・リソースの集約について説明します。ネットワーク・リソースのトポロジーは RODM によって管理されます。集合リソースを含むネットワーク・リソースは、 GMFHS によって集められる情報に基づいて NetView 管理コンソール ビューに表示されます。

集約の概要

集約とは、集合オブジェクトの状況を作成、接続、および更新する処理のことです。*集合オブジェクト* は、実オブジェクトの集合を表します。*実オブジェクト* は、実際のリソースを表します。集合オブジェクトは、実際の物理デバイスには対応しません。集合オブジェクトは、それに関連する実オブジェクトに関する 2 つのタイプの情報を提供します。

- 障害のあるリソースに対する高速パスのビューに関する接続情報。この種のビューに関する詳細については、112 ページの『NMC により検出された障害のあるリソースのビュー』を参照してください。
- 一連の規則に基づいて実オブジェクトのグループを表す、単一の DisplayStatus (状況ともいう) の表示。

集合オブジェクトおよび実オブジェクトは、両方とも RODM 内の任意のクラスの下に存在することができます。 GMFHS は、ResourceTraits フィールドを使用して、オブジェクトが集合オブジェクトであるか実オブジェクトであるかを判別します。 ResourceTraits フィールドは、データ・タイプ INDEXLIST であり、複数の値を入れることができます。すべての値は、ブランクで埋められて 8 文字にされます。 GMFHS、SNA トポロジー・マネージャー、およびマルチシステム・マネージャー・データ・モデルは、実クラスおよび集合体クラス両方のクラス・レベルで ResourceTraits フィールドを設定します。集合オブジェクトが作成されると、値 AGG が ResourceTraits フィールドに設定され、このオブジェクトが集合オブジェクトであることが示されます。同様に、実オブジェクトが作成されると、値 REAL が ResourceTraits フィールドに設定され、このオブジェクトが実オブジェクトであることが示されます。 1 つのオブジェクトの ResourceTraits フィールドに両方の値が入ることはありません。つまり、1 つのオブジェクトが実オブジェクトと集合オブジェクトの両方になることはできません。

154 ページの図 35 において、ラベル ***A*** が付いたオブジェクトは集合オブジェクトを、またラベル ***R*** が付いたオブジェクトは実オブジェクトを表します。

オブジェクトの*集約レベル* は、現行の集合オブジェクトを含め、1 つの集約パスを通過した集合オブジェクトの数です。実オブジェクトの集約レベルは常に 0 です。例えば、 154 ページの図 35 において、R4 の集約レベルは常に 0 です。 A34 の集約レベルは、 R10→A41→A34→A22→A12 のパスでは 2 であり、 R9→A34→A22→A12 のパスでは 1 です。 A35 の集約レベルは常に 1 です。

集合体子をもたない集約階層にオブジェクトがある場合は、*集約パス* は、AggregationParent フィールドを使用して、集約階層の固有の経路を定義します。パスは、各レベルにオブジェクトを 1 つずつ含み、このパスにある現行のオブジェクトに集合体親がなくなるまで続きます。例えば、 154 ページの図 35 において、R8→A32→A21→A12 は集約パスを形成します。 R8→A33→A22→A12 は、同じオブジェクトで開始あるいは終了する別の集約パスを形式化します。

集合体子 は、 AggregationChild フィールドによってリンクされた実オブジェクトまたは集合オブジェクトです。このリンクは、直接または間接のいずれかになります。直接子とは、オブジェクトの AggregationChild フィールドに直接リンクされた実オブジェクトまたは集合オブジェクトです。間接子とは、オブジェクトの直接子から開始し、集約階層を通って AggregationChild リンクのチェーンに従って到達することができる実オブジェクトまたは集合オブジェクトです。例えば、図 35 において、 A21 の直接子は R3、R4、A31、および A32 です。 A12 の間接子は R9 です。 A22 の間接子は、R8、R9、R10、R11、R12、R13、および A41 です。

集合体親 は、 AggregationParent フィールドによってオブジェクトにリンクされた集合オブジェクトです。このリンクは、直接または間接のいずれかになります。直接親とは、オブジェクトの AggregationParent フィールドに直接リンクされた任意の集合オブジェクトです。間接親とは、オブジェクトの直接親から開始し、集合階層を通って AggregationParent リンクのチェーンに従って到達することができる集合オブジェクトです。例えば、図 35 において、R1 の直接親は A11 および A12 です。 A34 の直接親は A22 です。 R11 の間接親は A12 です。 A41 の間接親は A22 および A12 です。

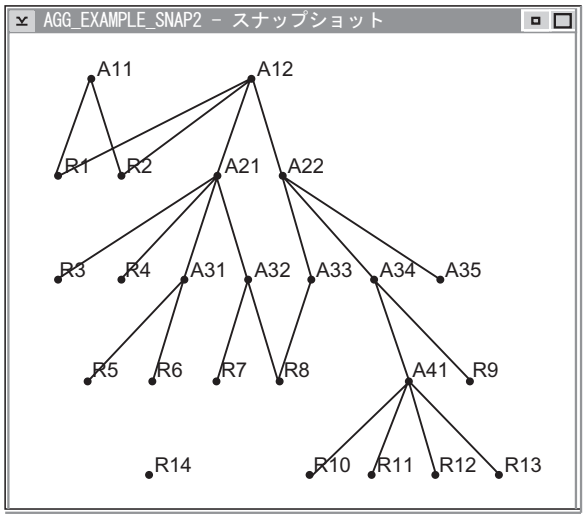


図 35. 実オブジェクト (R) と集合オブジェクト (A) を使用した集約の例

集約階層の作成

集約階層 は、集合オブジェクトとその基礎となる実オブジェクトのトポロジーです。集約階層は、オブジェクトの AggregationParent フィールドおよび AggregationChild フィールドを使用して作成されます。

実オブジェクトは集約階層の一部ですが、少なくとも 1 つの集合オブジェクトが RODM に作成されるまでは集約階層は存在しません。 図 35 は集約階層の例の 1 つです。集約階層は、以下の規則によって定義されます。

- 階層内の各パスについて、パスの最下位の子は、実オブジェクトにも集合オブジェクトにもなることができる。*最下位の子* は集合体子をもたない実オブジェクト

または集合オブジェクトです。したがって、0 個以上の集約パスがこの子から開始します。例えば、 154 ページの図 35 において、R2、R7、および A35 は最下位の子の例です。

- 階層内の各パスについて、パスの最上位の親は集合オブジェクトでなければならない。*最上位の親* は、集約親をもたない集合オブジェクトです。したがって 1 つ以上の集約パスで終了します。例えば、 154 ページの図 35 において、A11 および A12 は最上位の親の例です。実オブジェクトは、集約階層の一部と見なされる集合体親を少なくとも 1 つもっているため、最上位の親になることはできません。例えば、 154 ページの図 35 において、 R14 は集合体親がないため、集約階層の一部ではありません。
- 実オブジェクトが集合体親になることはできない。
- 集約階層のレベルの数は制限されない。集約階層のレベルの数は、階層内で最長の集約パスのレベルの数と等しくなります。

  注: 集約優先順位機能では、9 レベルの集約に制限されています。詳細については、 160 ページの『集約優先順位』を参照してください。
- 1 つのオブジェクトは複数の集合オブジェクトの直接子になることができ、また、1 つの集合オブジェクトは複数の直接子をもつことができる。 R1 は A11 と A12 の両方の直接子です。 R3、R4、A31 および A32 は A21 の直接子です。
- GMFHS が集約を正しく操作できるようにするためには、集約階層にループがあってはならない。集合オブジェクトがそれ自体の親であるときには、*集約階層のループ* が存在します。例えば、A12 は A33 の子になることができません。これでは結果的にパスが A12→A33→A22→A12→A33→A22... となり、無限にループしてしまいます。
- 複数のパス上のオブジェクト間に親子関係が存在することができる。各パスにおいて、子は、親に対する固有のオブジェクトと見なされます。例えば、 154 ページの図 35 において、R8 および A12 は、 R8→A32→A21→A12 および R8→A33→A22→A12 の 2 つの同じ集合体パスに属しています。 A12 から見ると、R8 は同一の特性をもつ 2 つの別個の実オブジェクトです。
- 集約階層内のすべてのオブジェクトが相互に連結されている必要はない。例えば、集約階層の別のサブセットが 154 ページの図 35 と同様の階層を形成するオブジェクトから構成することはできますが、この階層の 2 つのサブセット間には共通のオブジェクトはありません。階層サブセットは、一緒に全体の集約階層を形成します。

RODM での集約階層の構築

オブジェクトは、いつでも集約階層にリンクしたり、リンクを解除したりすることができます。集約階層は、AggregationParent および AggregationChild の 2 つの RODM フィールドを使用して作成します。これらのフィールドの詳細については、「*IBM Tivoli NetView for z/OS データ・モデル・リファレンス*」を参照してください。 これらのフィールドの RODM タイプは OBJECTLINKLIST です。すべてのオブジェクトについて、AggregationParent フィールドは、すべての直接親のオブジェクトに対するリンクを含みます。 AggregationChild フィールドは、すべての直接子のオブジェクトに対するリンクを含みます。

156 ページの図 36 において、 R2 の AggregationParent フィールドには、2 つのオブジェクト A11 および A12 に対するリンクが含まれています。 A22 の

AggregationParent フィールドには、 1 つのオブジェクト A12 に対するリンクが含まれています。 A22 の AggregationChild フィールドには、 3 つのオブジェクト A33、A34、および A35 に対するリンクが含まれています。

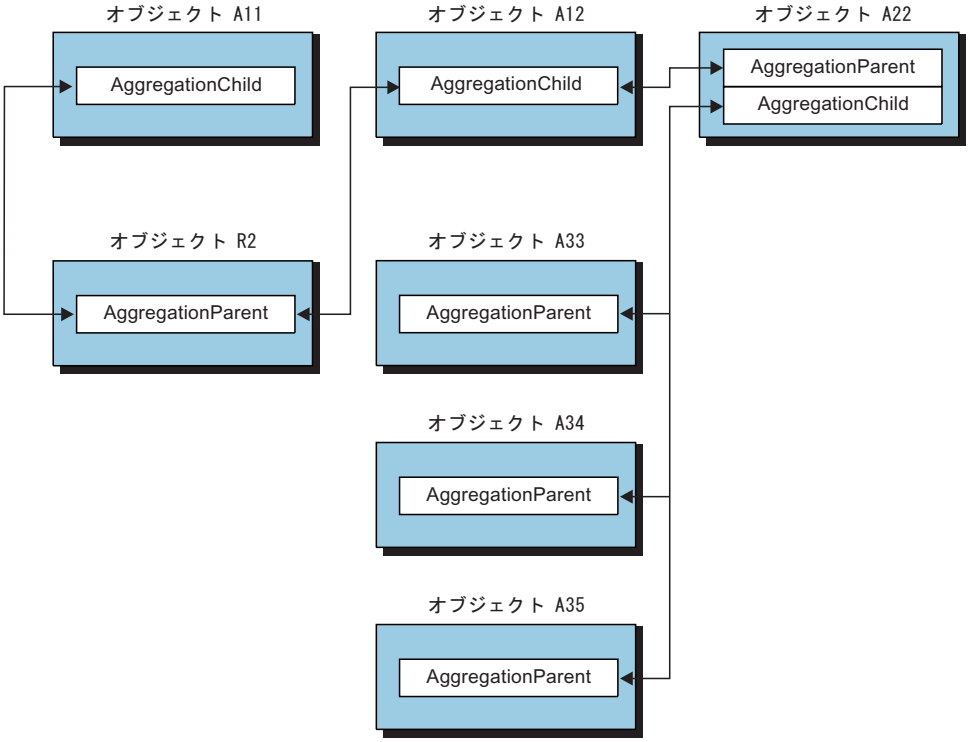


図 36. AggregationChild フィールドと AggregationParent フィールドの間のリンク

GMFHS が集約を正しく処理できるようにするためには、 2 つのオブジェクトの AggregationParent フィールドおよび AggregationChild フィールドのリンクまたはリンク解除は、メソッド DUIFCUAP で実行しなければなりません。操作が DUIFCUAP メソッドを使用せずに行われても、 RODM がこの操作を防いだり警告を出したりすることはありません。しかし、このメソッドが使用されていないと、リンクまたはリンク解除されている子オブジェクトから上にある集合オブジェクトすべての状況値を正しく計算することができません。メソッド DUIFCUAP によって、集約階層のループを防ぐこともできます。集約階層ループが集約階層で発生した場合は、GMFHS の動作は予測できません。メソッド DUIFCUAP の使用方法に関する詳細については、 560 ページの『DUIFCUAP: 集約パス更新メソッド』を参照してください。

RODM メソッドおよび通知を使用している場合は、集約階層をいつでも変更することができます。階層のセクション全体をリンクおよびリンク解除することができます。例えば、 154 ページの図 35 において、A34 を A22 からリンク解除して A31 にリンクすることができます。この手順では、以前と同じオブジェクトが A11 の下にあるため、 A11 には影響しません。しかし、階層の変更に伴って A21、A31、および A22 の下にあるオブジェクトの論理グループが変更され、これらの集合オブジェクトの状況も変わる場合があります。メソッド DUIFCUAP を使用してリンクまたはリンク解除を行うと、 GMFHS はこれらの階層の変更を動的に操作します。

注: A34 をリンク解除して再リンクするまでの時間の長さによっては、一時的に A12 の状況が変更される場合があります。

状況の更新

集約は、AggregationChild への最初の AggregationParent のリンクが発生したときから、 AggregationChild からの最後の AggregationParent のリンク解除が発生したときまでの集約階層で実行されます。集合の主な目的は、集約階層にあるすべての集合オブジェクトの状況を常に正確な状態に保つことです。集合オブジェクトの状況は、集合オブジェクトの下にある実オブジェクトの子すべての状況を収集してから、集合オブジェクトおよび実オブジェクトの両方で定義された RODM フィールドを使用して収集された一連の集約規則を実行することによって判別されます。

状況の集約への影響

集合体親の状況値に影響を与えるのは、実オブジェクトの子の状況のみです。子集合オブジェクトの状況は、親集合オブジェクトが実際のエンティティーを示していないため、これらのオブジェクトの状況には影響しません。例えば、 154 ページの図 35 において、実オブジェクトの子である R10、 R11、 R12、および R13 の状況は、集合オブジェクト A41 および A34 に影響します。しかし、オブジェクト A41 の状況は集合オブジェクト A34 には影響しません。

集約処理を以下に要約します。

1. 集約階層の集合オブジェクトの状況に影響するイベントが発生する。詳細については、 163 ページの『集約処理を開始するイベント』を参照してください。
2. 集合オブジェクトに影響するすべての実オブジェクトの状況を収集する。
3. 『実オブジェクトの DisplayStatus を使用する』の説明に従って集合オブジェクトの状況を計算する。
4. 集合オブジェクトの状況が変更された場合は、その状況を更新する。
5. ステップ 1 に戻り、次のイベントを待つ。

実オブジェクトの DisplayStatus を使用する

集約処理の間に多くの RODM フィールドが使用されますが、中でも DisplayStatus フィールドがこの処理の中心です。 『状況の集約への影響』にリストした集約処理のステップ 3 では、以下のように DisplayStatus フィールドを使用します。

- XCPT グループに影響する子の数をカウントする。
- XCPT グループに影響する各オブジェクトについて、オブジェクトの状況に基づいて、多くの状況グループにさらにオブジェクトを分類する。
- 各状況グループ内のオブジェクトの子の数をカウントする。
- 162 ページの『集約の規則』にリストした集合規則を XCPT グループおよび状況グループのカウントに適用し、集合オブジェクトの状況を判別する。
- 集合オブジェクトの状況が変更された場合は、その状況を更新する。

XCPT グループおよび状況グループ: 実オブジェクトは、状況値に応じて、 XCPT グループの 0 から 8 の状況グループのメンバーになります。これらのグループは、実オブジェクトの集合オブジェクトの状況への影響に優先順位を付けたりこれらの影響を定義したりする方法を提供します。 8 つの状況グループは、それぞれ STGRP1 (状況グループ 1) から STGRP8 までです。

グループに定義された状況値のいずれかと実オブジェクトの状況が一致した場合は、その実オブジェクトは XCPT グループ、状況グループ、またはその両方のメンバーです。各グループに定義された状況値はカスタマイズすることができます。XCPT および状況グループの状況値の定義に関する詳細については、123 ページの『例外ビュー用の DisplayStatus マッピング・テーブルをカスタマイズする』を参照してください。

XCPT グループは、例外ビューの処理および集約の処理に使用されます。集約の処理の場合は、集合オブジェクトの下の各実オブジェクトの状況を使用して、実オブジェクトが例外 (XCPT) 状態と非例外 (NOXCPT) 状態に分類されます。 XCPT 状態にあるすべての実オブジェクトは、XCPT グループにカウントされます。 XCPT グループおよび状況グループに関する詳細については、118 ページの『例外ビューのオブジェクトおよび基準を定義する』を参照してください。

注: 実オブジェクトを 8 つの状況グループにさらに分類するには、実オブジェクトも XCPT グループにカウントしなければなりません。

例: 154 ページの図 35 において、集合体 A41 は、実オブジェクト子 R10、R11、R12、および R13 をもっています。以下の DUIFSMTE ステートメントが DUIFSMT 表内にコーディングされているとします。

```
DUIFSMTE CLASS=R10s_Class,MYNAME=R10,                              C
         XCPT=(UNSAT,INTER,DS136,DS137,DS142,DS143),               C
         STGRP1=(UNSAT,INTER),STGRP2=(DS136,DS142),                C
         STGRP6=(DS137,DS158,UNSAT)
DUIFSMTE CLASS=R11s_Class,MYNAME=R11,                              C
         XCPT=(UNSAT,LOWSA,LOWUN,DS140),                           C
         STGRP3=(LOWSA,LOWUN),STGRP5=(DS140)
DUIFSMTE CLASS=R12s_Class,MYNAME=R12,                              C
         XCPT=(INTER,LOWSA,DS154,DS158),                           C
         STGRP1=(DS158),STGRP4=(LOWSA),STGRP6=(DS154),             C
         STGRP8=(INTER,DS158)
DUIFSMTE CLASS=R13s_Class,MYNAME=R13,XCPT=(UNKWN),STGRP8=(UNKWN)
```

図 37. 表 DUIFSMT 内の DUIFSMTE ステートメントの例

また、オブジェクトの実際の状況値は以下のとおりとします。
- R10 は UNSAT
- R11 は DS140
- R12 は DS158
- R13 は UNKWN

この例では、4 のリソースはすべて例外状態にあり、XCPT グループにカウントされます。 R10 は状況グループ 1 および 6 のメンバー、 R11 は状況グループ 5 のメンバー、R12 は状況グループ 1 および 8 のメンバー、 R13 は状況グループ 8 のメンバーです。集合オブジェクト A41 の場合は、以下のようになります。
- XCPT グループの中に 4 の実オブジェクトがある。
- 状況グループ 1 および 8 の中に 2 の実オブジェクトがある。
- 状況グループ 5 および 6 の中に 1 の実オブジェクトがある。
- 状況グループ 2、3、4、および 7 の中には実オブジェクトがない。

注:

1. すべての DUIFSMTE マクロ定義について、各状況グループに定義された状況値は、 XCPT グループに定義された状況値のサブセットでなければなりません。 XCPT グループの状況値でない状況グループの状況値を定義しようとすることもできますが、集約状況の計算には影響しません。
2. 158 ページの図 37 の最初の DUIFSMTE ステートメントでは、状況値 DS158 を STGRP6 に定義しています。 DUIFSMTE ステートメントによってこのように指定することはできますが、 DS158 が XCPT グループの中にないため、状況 DS158 は STGRP6 に対してカウントされません。
3. XCPT グループ内の状況値は、いずれの状況グループ内でも状況値として定義する必要はありませんが、実オブジェクトは、状況グループに影響を与えることなく XCPT グループに影響を与えることができます。

中断状態のリソース: 実オブジェクトは、AggregationParent および AggregationChild フィールドを実際に変更せずに集約階層から一時的に除去することができます。この論理的な除去を「オブジェクトを*中断状態* にする」といいます。オブジェクトを中断するには、以下の手法を用いることができます。

- NMC を使用すると、「Resource Properties」ウィンドウからリソースの中断フラグをセットすることができ、また「List of Suspended Resources」ウィンドウから中断リソースをクリアすることもできます。詳細については、NMC オンライン・ヘルプを参照してください。
- RODMView を使用して、UserStatus フィールドを RODM で直接設定する。詳細については、「*IBM Tivoli NetView for z/OS データ・モデル・リファレンス*」を参照してください。

実オブジェクトは、何らかの理由でオペレーターによって中断することができます。ほとんどの場合、オブジェクトによって表される実リソースの問題を解決しているときに、オブジェクトを中断状態にします。オブジェクトは、集約階層に論理的に戻されると、*再開* するように要求されます。

GMFHS は、中断しているリソースの数を追跡するために、SuspendedCount フィールドを使用します。実リソースは、以下のいずれかのアクションが行われた場合には、集合体親に状況を提供しません。

- UserStatus フラグの中断フラグがオンになっている。
- AggregationPriorityValue フィールドの値が -1 (無視) になっている。
- AggregationPriorityValue フィールドの値が -2 (リソース・タイプ・デフォルト) になっている。 DefaultAggregationPriorityCopy フィールドには、実オブジェクトの DisplayResourceType フィールドにリンクされた Display_Resource_Type_Class オブジェクトの DefaultAggregationPriorityValue フィールドの値のコピーが入っています。 DefaultAggregationPriorityCopy フィールドの値が -1 (無視) であって、AggregationPriorityValue フィールドの値が -2 (リソース・タイプ・デフォルト) である場合、このリソースは集約のための状況計算には関係しません。

注: AggregationPriorityValue または DefaultAggregationPriorityValue フィールドの値を -1 (無視) に設定しても、 UserStatus フィールドの中断フラグは影響を受けません。これらのアクションは相互に独立していて、一方によって他方が発生することはありません。

集合体親の状況を計算する

各実オブジェクト子の状況を XCPT グループと状況グループに分類して、特定の集合オブジェクトの各グループにある実オブジェクト子の数をカウントしてから、独立メソッドを使用して集合オブジェクトの状況を計算します。その後、集約の規則を使用して、各メソッドによって出力された矛盾する状況の結果を解決します。

集約しきい値: 集合体親の状況は、XCPT グループ・カウントとしきい値の大小関係に基づいて判別されます。 3 つのしきい値が、すべての集合オブジェクトにおける RODM フィールドとして定義されています。この値を、重大度の順に以下にリストします。

- ThresholdDegraded (重大度の下限)
- ThresholdSeverelyDegraded
- ThresholdUnsatisfactory (重大度の上限)

しきい値は、集合オブジェクトに対する XCPT グループのカウントがしきい値以上である場合に満たされます。 ThresholdSeverelyDegraded 値は ThresholdUnsatisfactory 値以下でなければならず、 ThresholdDegraded 値は ThresholdSeverelyDegraded 値以下でなければなりません。

これらのフィールドに有効な値は、「*IBM Tivoli NetView for z/OS データ・モデル・リファレンス*」に記載されています。値は以下のとおりです。

- 値 `-2` は、 Display_Resource_Type_Class オブジェクト (DefaultThresholdDegraded、DefaultThresholdSeverelyDegraded、DefaultThresholdUnsatisfactory のいずれか) からのデフォルト・フィールドの値を使用してしきい値を定義することを示します。デフォルト値は -1、0、またはすべての正数にすることができます。これらのデフォルト値が実際のしきい値の代わりに直接使用されます。
- しきい値フィールドの値が `-1` である場合は、集合オブジェクトについてのこのしきい値の計算ができないことを示します。
- しきい値フィールドの値が `0` である場合は、オブジェクトを、集合体親の XCPT グループのカウントにかかわらず、常にしきい値の状況に変更することを示します。複数のしきい値に値 0 がある場合は、最も優先順位が高いしきい値が有効になります。
- 正数は、XCPT グループのカウントがこの数値以上でなければならず、集合オブジェクトをしきい値の状況値に変更することを示します。この条件を満たす最も高い優先順位のしきい値が、この状況に適用するために使用されます。
- しきい値フィールド内の -100 から -200 までの値 (両端の値を含む) は、 XCPT グループのカウントが、(値 + 100)*(集合体に報告する実数の合計)*0.01 以上でなければならないことを示します。実際には、値は現在集合オブジェクトに付加されている実オブジェクトの合計数のパーセンテージになります。

集約優先順位: 集約優先順位によって、実オブジェクトをクリティカル・リソースとして指定することができます。クリティカル・リソースが集合体親の XCPT グループに影響する場合は、優先順位の低いしきい値と自動的に突き合わせられます。XCPT グループに影響するクリティカル・リソースを追加しても、これ以上の影響はありません。最後のクリティカル・リソースが XCPT グループにこれ以上影響しない場合は、優先順位の低いしきい値との突き合わせはこれ以上行われません。

AggregationPriorityValue フィールドはすべての実オブジェクトに定義されており、実オブジェクトをクリティカル・リソースとして定義するために使用されます。このフィールドに有効な値は、「*IBM Tivoli NetView for zOS データ・モデル・リファレンス*」に記載されています。 一般に、値は以下のとおりです。

- 値 -2 は、 Display_Resource_Type_Class オブジェクトの DefaultAggregationPriorityValue フィールドからのデフォルト・フィールドの値を使用して優先順位の値を定義することを示します。デフォルト値は、-1、0、または 1 から 9 までの任意の正数です。これらのデフォルト値が、実際の優先順位の値の代わりに直接使用されます。
- 値 -1 は、実オブジェクトの集約が中断されていることを示します。
- 値 0 は、実オブジェクトがクリティカル・リソースでないことを示します。
- 1 から 9 までの正数は、実オブジェクトがクリティカル・リソースであることを示します。また、この数は、オブジェクトがこの XCPT グループに影響している場合に、このオブジェクトが重要な性質に影響する集約階層に上がるレベルの数を示します。リソースの重要な性質は、集約階層から上に最大 9 レベルまでしか伝わりません。

注: 集約階層は、任意の数のレベルをもつことができます。実オブジェクトは、階層のすべてのレベルにある集合体の XCPT グループにカウントされます。しかし、このオブジェクトもクリティカル・リソースである場合は、重要な性質は、実オブジェクトから上の最大 9 レベルしか伝わりません。したがって、AggregationPriorityValue フィールドで指定されたレベル以下にある集合オブジェクトについて、優先順位が低いしきい値との突き合わせが行われます。

状況グループのカスタマイズ: しきい値による区分けおよび優先順位による集合体の両方によって、親の集合オブジェクトの状況を Unknown (不明)、Satisfactory (適合)、 Degraded (劣化)、SeverelyDegraded (重大劣化)、または Unsatisfactory (不良) の 5 つの事前決定値のいずれかに設定することができます。

集合オブジェクトの実際の状態をカスタマイズするには、8 つの状況グループを使用します。状況グループのカスタマイズは集合優先順位によく似ていますが、集約階層に関する 9 レベルまでの制限はありません。

状況グループのカスタマイズでは、集合体親の最終的な状況が 5 つの事前決定値以外の値になるようにカスタマイズすることができます。特定の状況グループのメンバーである実オブジェクトは、すべてカウントされます。カウントは、各状況グループについて行われます。状況グループ内の実オブジェクトの数がゼロより大きい場合は、集合オブジェクトに関する状況グループ定義を使用して集合オブジェクトの状況が判別されます。

状況グループには、STGRP1 (最高) から STGRP8 (最低) までの優先順位が付けられています。複数の状況グループのカウントがゼロより大きく、かつ集合オブジェクトについて合致する状況グループ定義が複数ある場合は、集合オブジェクトに対する最高の優先順位の状況グループ定義における最初の状況値が集合オブジェクトの状況として使用されます。

不明のリソース: 実オブジェクト子の状況値は集合体親の状況値に直接影響しますが、必ずしも XCPT グループには影響しません。集合体親の下にある Unknown (不明) 状況の実オブジェクトの総数は、 Global_Aggregation_Parameters_Class の

UnknownThreshold フィールドの値と比較されます。このしきい値以上であれば、この集合体親の集約処理をさらに行っても無効であり、集合体親の状況が Unknown (不明) になります。

160 ページの『集約しきい値』で定義した 3 つのしきい値とは異なり、この限界値は、1 % から 100 % までの数値です。この割合は、集約にアクティブで参加している (中断されていない) 集合体親の下にある実の子オブジェクトの総数に適用されます。

集約の規則: 中断状態のリソース、不明のリソース、集約しきい値、集約優先順位、および状況グループのカスタマイズは、集合オブジェクトの状況を計算するために使用されます。以下の集約規則は、集約メソッド間の競合を解決するために、リストした順序で使用されます。

1. 中断された実オブジェクト子を集約階層から論理的に除去します。この操作は、中断された実オブジェクトが XCPT および状況グループにカウントされないようにすることによってすでに実行済みになっています。しかし、集合体親の下にある全オブジェクトの合計カウントは、中断状態のリソースの除去を反映するようにここで変更されます。
2. 現行の実オブジェクト子の総数がゼロである場合、または 集合体親に現在リンクされている DisplayResourceType オブジェクトはないがこのオブジェクトからのデフォルトしきい値が必要な場合は、集合オブジェクトの状況が不明に設定されて状況の計算が終了します。
3. 不明状況である実オブジェクト子の割合が不明しきい値より大きい場合は、集合オブジェクトの状況が不明に設定されて状況の計算が終了します。
4. カスタマイズされた状況グループが集合オブジェクトとマッチする場合は、集合オブジェクトは、集合オブジェクトの最高のマッチ状況グループで定義された最初の状況になります。状況の計算は終了します。
5. XCPT 内の実オブジェクト子の数が不良しきい値以上である場合は、集合オブジェクトの状況が不良になって状況の計算が終了します。不良しきい値は、絶対数またはパーセンテージで表すことができます。
6. XCPT グループ内の実オブジェクト子の数が最大劣化しきい値以上である場合は、集合オブジェクトの状況が最大劣化になって状況の計算が終了します。最大劣化しきい値は、絶対数またはパーセンテージで表すことができます。
7. XCPT グループ内の実オブジェクト子の数が劣化しきい値以上である場合は、集合オブジェクトの状況が劣化になって状況の計算が終了します。劣化しきい値は、絶対数またはパーセンテージで表すことができます。
8. クリティカル・リソースである XCPT グループ内にカウントされた実オブジェクト子の数がゼロを超える場合は、集合オブジェクトの状況が劣化になって状況の計算が終了します。任意の実オブジェクト子の AggregationPriorityValue フィールドでは、現行のレベルの集合オブジェクトに対するクリティカル・リソースとしてカウントすることができないので注意してください。
9. 上記の条件のいずれにも該当しない場合は、集合オブジェクトの状況が適合になって状況の計算が終了します。

集約に関する問題

集約は、さまざまな RODM フィールドを使用して行われます。これらのフィールドの中には、ユーザーが変更することができるものや、 GMFHS メソッドでのみ使

用できるものもあります。ユーザーは GMFHS メソッド専用のフィールドを変更してはなりませんが、 RODM はこの変更を規制しません。

不整合は、以下の場合に発生します。

- 各集合オブジェクトに対する内部カウントが等しくない。
- しきい値が集合体親の実オブジェクトの子の総数より大きい。またはしきい値が 160 ページの『集約しきい値』 に定義した制約事項を守っていない。

UserStatus フィールドのインディケーターは、集約の処理中の不整合を表示するためのものです。

UserStatus フィールド

集合オブジェクトの UserStatus フィールドには、ビュー内にオブジェクトのオペレーター状況を設定するために使用される情報が含まれています。 UserStatus フィールドには、集合オブジェクトのオペレーター状況に影響を与える 5 つのビットがあります。

- リソース・マーク・ビット
- しきい値不整合ビット (前述の集合の問題が発生すると設定される)
- 中断状態ビット
- 再開ビット
- 集合体下の中断リソースビット

リソース・マーク・ビット、中断状態ビット、再開ビット、および集合体下の中断リソース・ビットは、オペレーターのアクションの結果として設定されるか、UserStatus フィールドを直接 RODM で設定すること (RODMView を使用するなど) によって設定されます。 しきい値不整合ビットは、不整合が検出されると、集約の処理中に設定されます。

集約処理を開始するイベント

集約の処理は、さまざまなイベントによって開始させることができます。一般に、集約は、集約の処理に使用される RODM フィールドの 1 つに対して行われた変更に基づいて起動されます。例えば、2 つのオブジェクトの AggregationParent および AggregationChild フィールドを使用してリンクが作成されたり、あるいは、集約階層内の実オブジェクトの DisplayStatus が変更されたりする場合です。以下では、集約の処理を起動するイベントについて 1 つ 1 つ説明します。

実オブジェクトの DisplayStatus を変更する: これは、集約処理を開始する最も一般的なイベントです。実オブジェクトの DisplayStatus 値は、 NetView 管理コンソールまたは NetView アラートからの状況変更要求など、さまざまな理由で変更される可能性があります。集約階層のメンバーである実オブジェクトの状況が変更されるたびに、その実オブジェクトのすべての集合体親の状況も変更する必要がある場合があります。

実オブジェクトが自動再開機能によって中断状態となっており、オブジェクトの状況が適合である場合は、オブジェクトは集約階層に論理的に再リンクされてそのオブジェクトの集約が再開されます。

XCPT グループまたは状況グループに対するオブジェクトの影響が変更されておらず、このオブジェクトが他の状況から不明状況に変更されたり不明状況から他の状況に変更されたりしていない場合は、集合体親の状況も変更されません。

メソッド DUIFCUAP を使用してリンクおよびリンク解除を行う: DUIFCUAP メソッドに渡された子オブジェクトおよび親オブジェクトの AggregationParent および AggregationChild フィールドが更新されます。リンクおよびリンク解除の操作には 2 つのオブジェクト (子オブジェクトと親オブジェクト) しか関係しませんが、このアクションによって集約階層の多くの集合オブジェクトの状況値が影響を受ける場合があります。

リンクまたはリンク解除の操作を行うと、直接親集合オブジェクトの状況および直接親集合オブジェクトのすべての親オブジェクトを変更する必要がある場合があります。

AggregationPriorityValue を変更する: 実オブジェクトの AggregationPriorityValue を変更した場合は、実オブジェクトのすべての集合体親の状況を変更する必要がある場合があります。実オブジェクトが集合体親オブジェクトに対する XCPT グループにカウントされない場合は、集合体親の状況は変更されません。
AggregationPriorityValue フィールドの値を変更するには、以下の手法を用いることができます。

- NetView 管理コンソール・ワークステーションを使用する。詳細については、「*IBM Tivoli NetView for z/OS NetView 管理コンソール ユーザーズ・ガイド*」を参照してください。
- NMC を使用する。詳細については、NMC オンライン・ヘルプを参照してください。
- AggregationPriorityValue フィールドを RODM で (RODMView を使用するなど) 直接設定する。詳細については、「*IBM Tivoli NetView for z/OS データ・モデル・リファレンス*」を参照してください。

集合オブジェクトのしきい値を変更する: これらのしきい値を変更する場合は、その特定の集合オブジェクトの状況を変更する必要がある可能性があります。
ThresholdDegraded、ThresholdSeverelyDegraded、および ThresholdUnsatisfactory フィールドの値を変更するには、以下の手法を用いることができます。

- NMC を使用する。詳細については、NMC オンライン・ヘルプを参照してください。
- フィールドを RODM で (RODMView を使用するなど) 直接設定する。詳細については、「*IBM Tivoli NetView for z/OS データ・モデル・リファレンス*」を参照してください。

不明 (unknown) しきい値を変更する: このしきい値を変更した場合は、集約階層のすべての集合オブジェクトの状況を変更する必要があることがあります。
Global_Aggregation_Parameters_Class の UnknownThreshold フィールドの値を変更するには、以下の 2 つの手法を使用することができます。

- UnknownThreshold フィールドを RODM で (RODMView を使用するなど) 直接設定する。詳細については、「*IBM Tivoli NetView for z/OS データ・モデル・リファレンス*」を参照してください。

注: NMC を使用して、UnknownThreshold フィールドの値を変更することはできません。

実オブジェクトを中断状態にする: リソースを中断状態にした場合は、その実オブジェクトの全集合体親の状況を変更する必要があることがあります。実オブジェクトは、ワークステーションでの集約への参加を中断することができます。実オブジェクトの集約への参加を中断するには、以下の手法を用いることができます。

- NMC を使用する。詳細については、NMC オンライン・ヘルプを参照してください。
- UserStatus フィールドを RODM で (RODMView を使用するなど) 直接設定する。詳細については、「*IBM Tivoli NetView for z/OS データ・モデル・リファレンス*」を参照してください。

リソース・タイプのデフォルトを変更する: 実オブジェクトの AggregationPriorityValue フィールドは、実オブジェクトにリンクされた Display_Resource_Type_Class オブジェクトからの DefaultAggregationPriorityValue フィールドの値を優先順位集約に使用することを示すことがあります。集合オブジェクトの ThresholdDegraded、ThresholdSeverelyDegraded、および ThresholdUnsatisfactory フィールドは、集合オブジェクトにリンクされた Display_Resource_Type_Class オブジェクトからのデフォルト・フィールドの値をしきい値集約に使用することを示すことがあります。

これらのデフォルトを使用する実オブジェクトおよび集合オブジェクトの場合も、優先順位の値およびしきい値フィールドが直接オブジェクト上で変更された場合と同様に機能します。主な相違点は、Display_Resource_Type_Class オブジェクトを複数のオブジェクトにリンクすることができるために、複数の実オブジェクトまたは集合オブジェクトを変更することができることです。

ThresholdDegraded、ThresholdSeverelyDegraded、および ThresholdUnsatisfactory フィールドの値を変更するには、以下の手法を用いることができます。

- NMC を使用する。詳細については、NMC オンライン・ヘルプを参照してください。
- フィールドを RODM で (RODMView を使用するなど) 直接設定する。詳細については、「*IBM Tivoli NetView for z/OS データ・モデル・リファレンス*」を参照してください。

メソッド DUIFCLRT を使用してリンクおよびリンク解除を行う: メソッド DUIFCLRT は、実オブジェクトおよび集合オブジェクトを Display_Resource_Type_Class のオブジェクトに関連付けるために使用します。実オブジェクトについては、 Display_Resource_Type_Class オブジェクトからのデフォルト値が使用されている場合は、このメソッドによってオブジェクトの優先順位集約の値が影響を受けます。集合オブジェクトについては、Display_Resource_Type_Class オブジェクトからのデフォルト値が使用されている場合は、このメソッドによって、オブジェクトの Degraded (劣化)、SeverelyDegraded (最大劣化)、または Unsatisfactory (不良) しきい値のすべてが影響を受けます。

実オブジェクトおよび集合オブジェクトでこれらの任意のデフォルトを使用する場合、優先順位の値およびしきい値フィールドが直接オブジェクト上で変更された場合と同様に機能します。

状況マッピング・テーブルを変更する: 状況マッピング・テーブルは、サンプル CNMSJH13 を使用して動的に更新することができます。 XCPT グループまたは任意の 8 つの状況グループの定義を変更することができるため、このサンプルをオプションで使用して、例外ビューおよび集約状況の計算が起動されるように RODM の各実オブジェクトの DisplayStatus 値を更新 (現在の値と同じ値に変更) することができます。

集約メソッド

556 ページの『GMFHS メソッド』に、GMFHS メソッドのリストを示します。記載したメソッドは、それぞれ DUIFCLRT で始まり、いずれも集約に少なくとも間接的に影響します。 DUIFCUAP、DUIFFAWS および DUIFFRAS の 3 つのメソッドは、集合に直接影響します。

メソッド DUIFFAWS および DUIFFIRS は、しきい値が整合していないことがオブジェクトの UserStatus フィールドに示されているとき、または集合オブジェクトの状況が誤っているとオペレーターが判断したときに、集約階層を同期化させるために使用します。 DUIFFRAS は、DUIFFAWS が実行する機能のサブセットを実行します。 DUIFFRAS によって、各集合オブジェクトの既存の XCPT グループおよび状況グループのカウントに基づいて各集合オブジェクトの状況が再計算されます。 DUIFFAWS は、集合オブジェクトの状況を再計算する前に、各集合オブジェクトについて XCPT グループおよび状況グループのカウントを蓄積することによって DUIFFRAS を拡張します。

これらのメソッドの詳細については、556 ページの『GMFHS メソッド』を参照してください。

状況グループ

集合オブジェクトの状況 (DisplayStatus フィールドの値) は、その集合の下の実オブジェクト子の状況に基づいてカスタマイズすることができます。

このカスタマイズには、119 ページの『例外基準を定義する』に記載したサンプル表 DUIFSMT を使用します。実子オブジェクトの状況を集合体親の適切な状況にマップするには、 DUIFSMTE マクロの STGRP*n* キーワード (*n* は 1 から 8) を使用します。 DUIFSMTE マクロおよび DUIFSMT 表の更新方法に関する詳細については、 123 ページの『例外ビュー用の DisplayStatus マッピング・テーブルをカスタマイズする』を参照してください。

STGRPn キーワードは、XCPT キーワードが例外ビューに使用される場合と同様に DisplayStatus 値をグループ化するために使用されます。このグループは、STGRP1 を優先順位最高のグループ、 STGRP8 を優先順位最低のグループとして優先順位に従って編成されます。同じ状況値が複数の状況グループに属することも可能です。実際に、状況値は、すべて任意の状況グループに入れることができます。 DisplayStatus 値は、STGRP*n* キーワードとして登録するためには、 XCPT 値でもなければなりません。

状況グループは、任意の親集合オブジェクトの状況に実オブジェクトの状況をマップするために使用します。状況グループのいずれかに存在する状況値に実オブジェクトを変更する場合、集合オブジェクトの状況値の判別には、対応する親集合オブ

ジェクトすべての状況グループが使用されます。実オブジェクトの状況値が複数のグループにリストされている場合は、その状況値を持っている最高の優先順位グループが使用されます。

以下の条件において、任意の集合体親の状況を判別するために実オブジェクトの例外状態が使用されます。

- 実オブジェクトに状況グループがないか、実オブジェクトの状況値がいずれの状況グループにも含まれない。
- 親集合オブジェクトのマッチ状況グループが定義されていない。

状況グループの使用

以下に、状況グループを使用して集約を操作する場合のその他の操作特性をリストします。

- 集合体親の状況グループ・マッチが起こると、その親の既存の状況をオーバーライドします。状況グループは、以下のいずれか になるまでオーバーライドされたままです。
  1. より高い優先順位の状況グループ・マッチが集合体親に発生する。
  2. 集合体親の現行の最高優先順位の状況グループに影響している最後の実オブジェクトの状況値が、その状況グループにマッチしなくなったか、または実オブジェクトが階層からリンク解除されたか集約から中断された。
- 状況グループ・マッチが発生すると、どのレベルの集合階層にある集合体親の状況値もオーバーライドします。集約優先順位のようなレベル制限はありません。
- 例外ベースの集約を使用すると、中断状態のオブジェクトは状況グループの集約に参加しません。
- 実オブジェクトの集合オブジェクトしきい値が不明状況の場合は、状況グループの集合によってオーバーライドされません。

集合体 DisplayStatus のカスタマイズ例

状況グループを使用して集合オブジェクトの DisplayStatus 値をカスタマイズする場合の例を以下に示します。この例では、以下の条件を仮定しています。

- T4NODE クラスのすべてのオブジェクトが、 DisplayStatus が不良または不明である例外状態の集約に影響を与える。 DisplayStatus が不良である場合には、状況グループ 1 にタグ付けされます。
- 1.3.18.0.0.1821 クラスのすべてオブジェクトが、 DisplayStatus が不良、中間、または不明である例外状態の集約に影響を与える。 DisplayStatus が中間または不明である場合には、状況グループ 2 にタグ付けされます。
- すべての集合オブジェクトに、状況グループ 1 および 2 に対する状況グループ・マッチがある。オブジェクトの T4NODE クラスが不良状況であると、すべての集合体親の状況が DS136 になります。 1.3.18.0.0.1821 クラスのオブジェクトが不良状況または中間状況のいずれかである場合は、この状況が状況グループ 1 のマッチによってオーバーライドされない限り、集合体親の状況は DS137 になります。
- 上記 3 つのクラスのいずれにもないすべてのオブジェクトが、 DisplayStatus が不良または中程度に不良の状況にある例外状態の集約に影響を与える。 DisplayStatus が UNSAT である場合は、状況グループ 3 にタグ付けされます。

マッチする状況グループ 3 の定義がどの集合オブジェクト上にもないため、DisplayStatus が UNSAT である実オブジェクトによって集合体親の状況グループ 3 がオーバーライドされることはありません。

前記の条件を使用して、 図 38 に DisplayStatus マッピング・テーブルをコーディングしました。 4 番目のステートメントでデフォルトを設定しています。

```
DUIFSMTE CLASS=T4NODE,XCPT=(UNSAT,UNKWN),STGRP1=(UNSAT)
DUIFSMTE CLASS=1.3.18.0.0.1821,XCPT=(UNSAT,INTER,UNKWN),       C
         STGRP2=(INTER,UNKWN)
DUIFSMTE CLASS=GMFHS_Aggregate_Objects_Class,XCPT=(SDGRD),     C
         STGRP1=(DS136),STGRP2=(DS137)
DUIFSMTE CLASS=ALL,XCPT=(UNSAT,MEDUN),STGRP3=(UNSAT)
```

図 38. 集合の表示状況のカスタマイズ例

コレクション定義オブジェクトの使用

このセクションでは、コレクション定義オブジェクトの使用方法を説明します。

コレクション定義オブジェクトは、 GMFHS RODM Collection Manager 機能が Network_View_Class および GMFHS_Aggregate_Objects_Class オブジェクトの内容を定義するために使用するものです。コレクション定義オブジェクトは、Network_View_Collection_Class または Aggregate_Collection_Class のいずれかで作成されます。これらのクラスはそれぞれ、Collection_Definition_Class のサブクラスです。 Collection_Definition_Class でオブジェクトを作成しないでください。

コレクション定義オブジェクトによって定義される Network_View_Class および GMFHS_Aggregate_Objects_Class オブジェクトは、コレクション作成オブジェクトと呼ばれます。コレクション作成オブジェクトは、コレクション定義オブジェクト内の情報から GMFHS RODM Collection Manager 機能によって作成されます。RODM は、継続して RODM 内で作成または削除される新規コレクション定義オブジェクトを監視します。これは、対応するコレクション作成オブジェクトを動的に作成します。さらに、既存のコレクション定義オブジェクト上のリソース・コレクションに対する変更も、継続してモニターされます。変更は、対応するコレクション作成オブジェクトに動的に反映されます。

コレクション定義オブジェクト

コレクション定義オブジェクトのフィールドは、以下のものを指定します。

- コレクション作成オブジェクトの RODM MyName。
- Network_View_Collection_Class オブジェクトの場合、 Network_View_Class コレクション作成オブジェクトの Annotation。
- Aggregate_Collection_Class オブジェクトの場合、GMFHS_Aggregate_Objects_Class コレクション作成オブジェクトの DisplayResourceUserData。

- Aggregate_Collection_Class オブジェクトの場合、GMFHS_Aggregate_Objects_Class コレクション作成オブジェクトの DisplayResourceName。
- Aggregate_Collection_Class オブジェクトの場合、GMFHS_Aggregate_Objects_Class コレクション作成オブジェクトの DisplayResourceType。
- Aggregate_Collection_Class オブジェクトの場合、GMFHS_Aggregate_Objects_Class コレクション作成オブジェクトの DisplayResourceOtherData。
- Aggregate_Collection_Class オブジェクトの場合、GMFHS_Aggregate_Objects_Class コレクション作成オブジェクトの DegradedThreshold。
- Aggregate_Collection_Class オブジェクトの場合、GMFHS_Aggregate_Objects_Class コレクション作成オブジェクトの SeverelyDegradedThreshold。
- Aggregate_Collection_Class オブジェクトの場合、GMFHS_Aggregate_Objects_Class コレクション作成オブジェクトの UnsatisfactoryThreshold。
- GMFHS_Aggregate_Objects_Class コレクション作成オブジェクトの Network_View_Class の LayoutType。
- Aggregate_Collection_Class オブジェクトの場合、集合コレクションを処理するために使用する要求固有のフラグ。
- NMC コンソールによって解釈される情報を保持するデータ・フィールド。
- Network_View_Class または GMFHS_Aggregate_Objects_Class に渡して組み込まなければならない、ルールの論理ツリー。

コレクション定義オブジェクトのフィールド

コレクション定義オブジェクトのクラスおよびフィールドの詳細については、「*IBM Tivoli NetView for z/OS データ・モデル・リファレンス*」を参照してください。

コレクション定義オブジェクトのフィールドのほとんどは、コレクション作成オブジェクトの同名のフィールドに直接コピーされます。 RequestFlags、CollectionLocateName、および WizardHints フィールドなどのいくつかのフィールドは、 RODM Collection Manager によってのみ使用されます。これらのフィールドを使用して、コレクション作成オブジェクトのフィールドに値を提供することはありません。

中には、間接的にコレクション作成オブジェクトのフィールドに値を提供するコレクション定義フィールドもあります。 Aggregate_Collection_Class オブジェクトで LayoutType フィールドが指定されると、これは文字ストリングに変換され、ストリング ″RCMLayoutParmViewType″ に付加されます。この連結ストリングは、Layout_Parameters_For_View_Class オブジェクトの名前として使用されます。このオブジェクトはその後、コレクション作成オブジェクトの DetailViewLayoutForSelectedResource フィールドにリンクされます。

同様に、DisplayResourceType フィールドも、 Display_Resource_Type_Class オブジェクトの名前として使用されます。このオブジェクトはその後、コレクション作成

オブジェクトの DisplayResourceType フィールドにリンクされます。CollectionSpec*n* フィールドは、 Network_View_Class コレクション作成オブジェクトの ContainsObjects フィールド、および GMFHS_Aggregate_Objects_Class コレクション作成オブジェクトの AggregationChild および IsPartOf フィールドにデータを取り込むために使用されます。これらのフィールドの使用法についての詳細は、『コレクション仕様の使用』を参照してください。

コレクション作成オブジェクトがすでに RODM に存在する場合、それは削除され、コレクション定義オブジェクトの情報を使用して再作成されます。既存の Network_View_Class または GMFHS_Aggregate_Object_Class オブジェクトを上書きしないように、注意してコレクション作成オブジェクトに名前を付けてください。コレクション作成オブジェクトを重複して作成しないようにするための簡単な方法は、 RODM Collection Manager によって作成されたオブジェクトであることを示すコレクション作成オブジェクト名に接頭部または接尾部を付けることです。

コレクション仕様の使用

コレクション仕様は、コレクション定義オブジェクトの CollectionSpec*n* フィールドに含まれます。これらのフィールドは、フィールドの *n* 数値部の昇順で互いに連結され、完全なコレクション仕様を形成します。最初の CollectionSpec*n* フィールドは、CollectionSpec1 になります。コレクション仕様には、 Network_View_Class コレクション作成オブジェクトの ContainsObjects フィールド、または GMFHS_Aggregate_Objects_Class AggregationChild および IsPartOf フィールドに入るオブジェクトを記述する一連のルールが含まれます。

コレクション仕様のルールは、動的に適用されます。ルールが最初に RODM 現在 Collection Manager 機能によって処理されるときに RODM に存在するオブジェクト、およびルールが最初に処理された後で動的に RODM に追加される、または RODM から削除されるオブジェクトとルールが突き合わせられます。 RODM Collection Manager は、コレクション定義オブジェクトのコレクション仕様で指定したすべてのクラスの全フィールドに RODM 通知を置き、その後にいずれかのオブジェクトのこれらのフィールドの値が変更した場合に通知が行われます。結果として、RODM 内でコレクション作成オブジェクトに影響を与える変更が生じるたびに、 RODM Collection Manager はコレクション作成オブジェクト内のオブジェクトを更新できます。

条件ステートメント

条件ステートメントは互いに論理結合され、コレクション仕様の一部になります。各条件ステートメントは、RODM フィールド、RODM クラス、値 (またはオプション)、値のセット、および操作から成ります。指定されたクラスの各オブジェクトごとに、指定されたフィールドは、操作を使用して値または値のリストと比較されます。操作が正常に比較が行われると、オブジェクトは条件と一致します。正常に比較が行われない場合、条件と一致しません。条件との比較が正常に行われる全オブジェクトのリストが、条件ステートメントの結果です。これらのオブジェクトは、RODM タイプ ObjectList です。

コレクション仕様の最も単純な形式は単一の条件ステートメントで、次の一般的な条項で表されます。

```
{Class/Field} operation {Value}   ==>  list_of_objects
```

特定の Class の各オブジェクトごとに、オブジェクトの Field の値を取って、それを比較操作を使用して Value と比較します。値の比較が正常に行われる場合、オブジェクトを出力 *list_of_objects* に置きます。

{Class/Field} が間接的に Class のすべてのオブジェクトを参照するように、{Value} も、値のセットを参照することができます。それぞれの値が直接コレクション仕様にリストされます。 {Value} に複数の値がリストされる場合、オブジェクトの Field 値は値リストの各値に対して比較されます。値リスト内の 1 つ以上の値は、 *list_of_objects* に追加されるオブジェクトに対して比較が成功しなければなりません。

単一の条件ステートメントは、次の条項によっても表されます。

```
{Value1} operation {Value2}  ==>  list_of_objects
```

Value1 および Value2 のどちらも、単一の値にも値リストにもなれます。 Value1 は、Class 内の各オブジェクトの Field の値を表します。 Value2 は、条件ステートメントで直接指定される値のリストを表します。この汎用構文は、複合条件ステートメントを記述する際に役立ちます。

単純なコレクション仕様の場合、生成される *list_of_objects* は、コレクション作成オブジェクトの ContainsObjects フィールドまたは AggregationChild/IsPartOf フィールドのいずれかのオブジェクト・リストになります。事実上、この *list_of_objects* は、コレクション仕様からの最終出力です。

条件ステートメントの後置表記法

後置表記法を使用して条件ステートメントを表す場合、ステートメントは次のようになります。

```
{Class/Field} {Value} operation ==> list_of_objects
```

または

```
{Value1} {Value2} operation   ==>  list_of_objects
```

後置表記法は、RODM 内のコレクション定義オブジェクトの、実際のコレクション仕様で使用される表記法です。

例えば、以下のような単純なコレクション仕様があるとします。

```
|GMFHS_Managed_Real_Objects_Class|DisplayStatus|132|.EQ.
```

このコレクション仕様は、 GMFHS_Managed_Real_Objects_Class 内の各オブジェクトの DisplayStatus フィールドの値を取り、それを 132 と比較します。値が等しい場合、条件ステートメントを満たす *list_of_objects* にオブジェクトが追加されます。すべてのオブジェクトが比較された後、*list_of_objects* はコレクション作成オブジェクトのオブジェクト・リスト・フィールドに入れられます。

また、条件ステートメントはリーフ仕様とも呼ばれます。リーフ仕様は、2 つの値のリストの比較から、 *list_of_objects* を生成します。これは、コレクション仕様が概念的に表現するツリー処理におけるリーフです。これがリーフであるのは、Value1 および Value2 の演算子がコレクション仕様から他の条件ステートメントの評価によって生成されるのではなく、コレクション仕様 (Value2) またはオブジェクトのフィールド (Value1) から直接由来するからです。

複合条件ステートメント

コレクション仕様のほとんどは、複数の条件ステートメントから成ります。例えば、ネットワーク・ビューの候補と見なされるオブジェクトの場合、その DisplayStatus を 132 に、そして (AND) その MyName を Chihuahua にすることができます。この場合、2 つの条件ステートメントを互いにリンクするために結合子 AND が使用されています。

後置表記法における条件ステートメントを互いにリンクするための構文は、次のとおりです。

```
( {Class/Field}  operation {Value} )    ( {Class/Field}  operation
{Value} ) conjunction  ==>  list_of_objects
```

または

```
 (leaf_specification) (leaf_specification) conjunction  ==>  list_of_objects
```

両方のリーフ使用とも、リストにオブジェクトがない場合でもオブジェクト・リストを生成します。最終的な *list_of_objects* は、結合演算子 (AND または OR) を 2 つのオブジェクト・リストに適用することによって決まります。結合演算子が AND の場合、オブジェクト ID は、生成される *list_of_objects* に存在するために、両方のリストになければなりません。結合演算子が OR の場合、オブジェクト ID は、生成される *list_of_objects* に存在するために、いずれか一方のリストになければなりません。

リーフ仕様は *list_of_objects* に評価されるため、上記構文の汎用形式は次のようになります。

```
(list_of_objects)(list_of_objects) conjunction  ==>  list_of_objects
```

この構文もノード仕様と呼ばれます。ノード仕様は、他の条件ステートメント (オブジェクト・リスト) からの出力を、結合子のオペランドとして使用します。ノード仕様そのものがオブジェクト・リストを生成する条件ステートメントであるため、ここで説明した単純ノード仕様のノード仕様を回帰的に置換することにより、複合条件を無限に作成できます。

例えば、次の複合条件を後置表記法で考えてみます。

```
(a) (b) EQ (c) (d) EQ AND (e) (f) EQ (g) (h) EQ AND OR
```

この例を続けるため、これを複合条件の汎用形式に構築します。まず、(a) (b) EQ はリーフ仕様になります。

```
(leaf_specification) (c) (d) EQ AND (e) (f) EQ (g) (h) EQ AND OR
```

次に、(c) (d) EQ もリーフ仕様になります。

```
(leaf_specification) (leaf_specification) AND (e) (f) EQ (g) (h) EQ AND OR
```

または

```
(list_of_objects) (list_of_objects) AND (e) (f) EQ (g) (h) EQ AND OR
```

次に、(list_of_objects) (list_of_objects) AND がノード仕様の形式になります。

```
(node_specification) (e) (f) EQ (g) (h) EQ AND OR
```

または

```
(list_of_objects) (e) (f) EQ (g) (h) EQ AND OR
```

次に、(e) (f) EQ (g) (h) EQ は、(a) (b) EQ (c) (d) EQ と同一です。

```
(list_of_objects) (leaf_specification) (leaf_specification) AND OR
```

リーフ仕様が関係する複合条件を評価すると、次のようになります。

```
(list_of_objects) (node_specification) OR
```

または

```
(list_of_objects) (list_of_objects) OR
```

この最終的な条件は、ここで説明した汎用構文と一致し、複合条件の最終オブジェクト・リストを生成します。コレクション仕様で使用される後置表記法を実際に評価するために使用するメソッドについての詳細は、 『スタック・モデルの後置処理』を参照してください。

スタック・モデルの後置処理

コレクション仕様は、仮想スタックを使用して、コレクション仕様の条件ステートメントからの中間結果を保持することによって処理されます。リーフ仕様からの出力 (オブジェクト・リスト) はすべて、スタックに追加されます。コレクション仕様内で結合が起きると、スタックに追加される最後の 2 つのオブジェクト・リストがスタックから除去され、結合子がオブジェクト・リストに適用されて、生成されるオブジェクト・リストがスタックに追加されます。この処理は左から右に、コレクション仕様の最後まで継続されます。コレクション仕様の最後に達すると、スタックにオブジェクト・リストが 1 つだけ残されています。そのようにならない場合、コレクション仕様は構文上の誤りがあります。スタックに残されるオブジェクト・リストは、コレクション仕様の最終結果です。これは、コレクション作成オブジェクトの ContainsObjects または AggregationChild/IsPartOf フィールドに直接割り当てられます。

リーフ仕様は後置表記法を使用して処理されますが、演算子への入力 (Value1 および Value2) は、オブジェクト・リストです。スタックには、オブジェクト・リストだけが含まれます。そのため、リーフ仕様は、スタックを使用せずに評価されます。その出力 (オブジェクトのリスト) は、スタックに追加されます。

以下は、172 ページの ページの例を評価する間に起きるスタック操作を示しています。

```
(a) (b) EQ (c) (d) EQ AND (e) (f) EQ (g) (h) EQ AND OR
```

最初に、スタックは空の状態です。コレクション仕様を左から右へ読み取り、リーフ仕様 (a) (b) EQ はオブジェクト・リスト *a_b_objects* に評価され、スタックに追加されます。結果は次のようになります。

スタックに含まれるもの:	*a_b_objects*
残りの仕様:	(c) (d) *EQ AND* (e) (f) *EQ* (g) (h) *EQ AND OR*

(c) は結合子ではないため、次に続くのは別のリーフ仕様でなければなりません。結合または有効なリーフ仕様以外はどれも構文的に誤りとなります。 (c) (d) EQ はオブジェクト・リスト *c_d_objects* に評価され、スタックに追加されます。結果は次の

ようになります。

スタックに含まれるもの:	*c_d_objects*
	a_b_objects
残りの仕様:	*AND* (e) (f) *EQ* (g) (h) *EQ AND OR*

AND は結合子なので、スタックの最初の 2 つのオブジェクト・リスト (この場合は 2 つだけ) は除去され、結合子を使用して評価されて、結果がスタックに追加されます。結合子が評価されるときに、2 つ以上のオブジェクトがスタックにない場合、エラーになります。結果は次のようになります。

スタックに含まれるもの:	*a_b_AND_ c_d_objects*
残りの仕様:	(e) (f) *EQ* (g) (h) *EQ AND OR*

(e) は結合子ではないので、次に続くのは別のリーフ仕様になります。 (e) (f) EQ はオブジェクト・リスト e_f_objects に評価され、スタックに追加されます。結果は次のようになります。

スタックに含まれるもの:	*e_f_objects*
	a_b_AND_ c_d_objects
残りの仕様:	(g) (h) *EQ AND OR*

(g) は結合子ではないので、次に続くのは別のリーフ仕様になります。 (g) (h) EQ はオブジェクト・リスト g_h_objects に評価され、スタックに追加されます。結果は次のようになります。

スタックに含まれるもの:	*g_h_objects*
	e_f_objects
	a_b_AND_c_d_objects
残りの仕様:	*AND OR*

AND は結合子なので、スタックの最初の 2 つのオブジェクト・リストは除去され、結合子を使用して評価されて、結果がスタックに追加されます。結果は次のようになります。

スタックに含まれるもの:	*e_f_AND_g_h_objects*
	a_b_objects AND c_d_objects
残りの仕様:	*OR*

最後に、OR は結合子なので、スタックの最後の 2 つのオブジェクト・リストは除去され、結合子を使用して評価されて、結果がスタックに追加されます。結果は次のようになります。

スタックに含まれるもの:	*a_b_AND c_d_objects_OR_e_f_AND_g_h_objects)*
残りの仕様:	

この時点で、(存在する) スタックにあるオブジェクト・リストは 1 つだけであり、コレクション仕様には 1 つも残されていないはずです。これらのいずれかが真である場合、コレクション仕様は構文的に誤りがあります。最終的なオブジェクト・リストはコレクション仕様の結果であり、コレクション作成オブジェクトにコピーされます。

コレクション仕様の構文

以下は、コレクション仕様のフィールドの構文です。

```
<collection_specification> :: <separator><leaf_specification>  -または-
                                 <separator><node_specification>

<node_specification> ::
   <leaf_specification><separator><leaf_specification><separator>
     <conjunction>  -または-
<leaf_specification><separator><node_specification><separator>
     <conjunction>  -または-
<node_specification><separator><leaf_specification><separator><conjunction>
                    -または-
<node_specification><separator><node_specification><separator><conjunction>

<leaf_specification> ::
   <class_name><separator><field_name><separator><value_list>
   <separator><operator>

<value_list> ::
   <value>  -または-
   <value><separator><value_list>

<class_name> ::
      文字のストリング、最大 64 文字、RODM クラスを指定 (例、NMG_Class)

<field_name> ::
      文字のストリング、最大 64 文字、RODM フィールドを指定 (例、MyName)

<value> ::
      文字のストリング、RODM フィールドの値を指定 (例、CNM01AGT)

<separator> ::
      単一文字、任意の文字値 (例、|)

<operator> ::  .EQ. (等しい)                                          -または-

               .NE. (等しくない)                                      -または-

               .LT. (より小)                                          -または-

               .GT. (より大)                                          -または-

               .LE. (より小か等しい)                                  -または-

               .GE. (より大か等しい)                                  -または-

               .CONTAINS. (少なくとも 1 つを含む)                     -または-

               .CONTAINS=. (大文字小文字を区別し、少なくとも 1 つを含む)  -または-

               .NCONTAINS. (含まない)                                 -または-

               .NCONTAINS=. (大文字小文字を区別し、含まない)

<conjunction> :: .AND.  -または-

                 .OR.
```

コレクション仕様内の個々のトークンを区切る文字は、コレクション仕様の一部として定義されます。 <separator> は、任意の文字です。選択された値はどれも <class_name>、 <field_name>、または <value> に現れる可能性があるので、この文字は、ユーザー定義とすることができます。 NetView 管理コンソール GUI は、デフォルトの区切り文字として垂直バー (|) を使用します。

コレクション仕様の値

リーフ仕様の {Value} は、パターンと見なされます。パターンとは一連の文字のことで、その中には特別な意味を持ち、特定の値または値のセットに対して突き合わせられるものがあります。特殊文字によって、パターンは複数の値を記述することができます。特殊文字を持たないパターンは、1 つの値、つまり厳密にパターン内の文字から成る値しか記述しません。特殊文字をもつパターンは、値のリストと類似しています。この値のリストは、パターンと一致するすべての固有値から成ります。 {Value} は値のリストで、リスト内のそれぞれの値は、特殊文字を持つパターンになれます。

パターンを表現するために、DOS ワイルドカードまたは正規表現を使用できます。正規表現とは、検索パターン内のストリング (またはストリングのグループ) を定義する文字と演算子のセットのことです。また、正規表現にはメタ文字 (特別な意味を持つ文字) も含まれます。パターンのデフォルト表記は、DOS ワイルドカードの使用です。パターンが正規表現を使用する場合、パターンの最初の文字を、円記号 (¥) にしなければなりません。パターンが (DOS でも正規表現でも) 特殊文字を使用しない場合、パターンは、比較演算の単一の固有値に解決します。

DOS ワイルドカードを使用し、DOS ワイルドカードの最初の文字が円記号 (¥) の場合、正符号 (+) を使用してそれをエスケープしなければなりません。つまり、 `+¥value` は、 DOS ワイルドカード値 `¥value` として解釈されます。また、DOS ワイルドカードを使用し、DOS ワイルドカードの最初の文字が正符号の場合、正符号をもう 1 つ付けてそれをエスケープしなければなりません。つまり、`++value` は、DOS ワイルドカード値 `+value` として解釈されます。エスケープ文字としての正符号は、それが値の最初の文字である場合、またその後にもう 1 つの正符号または円記号 (¥) が続く場合にのみ有効です。

DOS パターンの特殊文字に、アスタリスク (*) および疑問符 (?) があります。アスタリスクは、パターン内でゼロ個以上の文字と一致します。疑問符 (?) は、パターン内の任意の 1 文字と一致します。 DOS パターンの特殊文字は、パターン内のどこででも使用できます。パターン `*re?*om*` は、`re` の前に他の文字がゼロ個以上あり、 `re` の後に少なくとも 1 文字以上あり、 `om` までにゼロ個以上の文字があり、その後に、ストリングの最後にゼロ個以上の文字が続くストリングと一致します。

DOS ワイルドカード文字を使用するパターンは、常にそれが比較されているストリング全体と一致しなければなりません。この例において、パターンが文頭と文末にアスタリスクの付かない `re?*om` の場合、一致するストリングは `re` で始まり、`om` で終わらなければなりません。これは、正規表現の機能の仕方と少し異なります。

正規表現は、より複雑なパターン・マッチングで使用されます。コレクション仕様内の DOS パターンは、値に対してパターンを付き合わせる前に RODM Collection Manager によって正規表現に変換されます。パターン・マッチングはすべて、RODM Collection Manager によって、正規表現を使用して行われます。正規表現の

パターンは、入力ストリングのサブストリングに適用されます。サブストリングと一致する場合、パターンは入力ストリング全体と一致したものと見なされます。正規表現が入力ストリングのサブストリングと一致するため、脱字記号 (^) メタキャラクターが変換済み DOS ワイルドカード・パターンの先頭に追加され、ドル記号 ($) メタキャラクターが同じ変換済み DOS ワイルドカード・パターンの最後に追加されて、ストリング全体のマッチングの DOS ワイルドカード制約を強化します。

正規表現の最も単純な形式は、特別な意味を持たない文字のストリングです。以下の文字には、特別な意味があります。これらの文字は、拡張された正規表現を形成するために使用されます。

. (ピリオド)
ピリオドは、改行文字以外の任意の 1 文字と一致します。

[string]
大括弧内のストリングは、ストリング内の任意の文字を指定します。つまり、[abc] が他のストリングと比較される場合、a、b、または c を含むストリングと一致します。大括弧内のストリングの文字にハイフンと別の文字が続く場合、 2 つの文字間の現行の照合シーケンス内のすべての文字が、ストリングの一部と見なされることを示します。例えば、[a-z] は、[abc...xyz] と同等であり、別の照合シーケンスでは [aAbBcC...xXyYzZ] と同等です。大括弧内のストリングの先頭に脱字記号 (^) がある場合、大括弧内の文字を否定します。つまり、[^abc] が他のストリングと比較される場合、ストリングに a、b、または c が 1 つでもあると一致しません。

expression[m] または expression[m,] または expression[m,u]
[ ] で囲まれた整数値は、先行する正規表現を適用する回数を示します。 m は最小数、u は最大数です。 u は、256 より小さな数でなければなりません。 m を指定する場合、正規表現を適用する正確な回数を示します。 [m,] は、[m,u] と同等ですが、u の上限がありません。どちらも、m またはそれ以上の表現の出現と一致します。正符号 (+) およびアスタリスク (*) の操作は、それぞれ [1,]、[0,] と同等です。

expression* (アスタリスク)
アスタリスク・シンボルは、ゼロ個以上の任意の文字を示します。例えば、a*e は、99ae9、aaaaae、a999e99 に相当します。

$ (ドル記号)
ドル記号は、ストリングの末尾と一致します。

^ (脱字記号)
脱字記号は、ストリングの先頭と一致します。

¥ (円記号)
円記号は、それに続く任意の文字の特別な意味を無効にして、それによってパターン内で文字そのものが持つ意味として解釈されるようにします。例えば、¥. は、. と一致します。任意の文字に先行する ¥ ではありません。

expression+ (正符号)
正符号 (+) は、1 回以上の文字の出現を指定します。つまり、smith+ern は、例えば、smithhhern と一致します。

(expression)
*、+、または [ ] などの演算子を許容する副次式をグループ分けし、括弧

で囲まれた副次式で機能するようにします。例えば (a*(cb+)*) は、ゼロ個以上の a と、その後にゼロ個以上のパターン c が続き、さらに 1 つ以上の b が続くストリングと一致します。

DOS パターン内のアスタリスク (*) は、正規表現ではピリオドとアスタリスク (.*) になります。 DOS パターン内の疑問符 (?) は、正規表現ではピリオド (.) になります。

DOS パターンはすべて、RODM Collection Manager によって正規表現に変換されると、脱字記号 (^) (ストリングの先頭と一致する) 付きで現れ、ドル記号 ($) (ストリングの末尾と一致する) が付加されます。これによって、文字ごとに一致して、ストリング全体が一致します。

例えば、DOS パターンの *IS?R* は、以下のものと一致します。

- BISTRO
- MISERLY
- MISER

しかしながら、以下のものとは一致しません。

- MISTER
- DISRUPT

正規表現では同じパターンが ¥.*IS.R.* と表現されます。

正規表現のパターン ¥RE[AGLRU]+.E[^A-O]+.*ON は、以下のものと一致します。

- REGULAR EXPRESSION
- REGAL-EXPATRIATION

しかしながら、以下のものとは一致しません。

- REGULATION
- REGENERATION

値およびデータ型

リーフ仕様の {Value} は最初、常に文字ストリングとして解釈されます。 {Value} が比較される {Class/Field} は、実データ型の数値の 1 つです。必要な場合、{Value} (各値が値のリストの場合) は、比較が行われる前に適切なデータ型に変換されます。一般に、DOS ワイルドカードまたは正規表現を使用して表現できるのは文字データ型だけです。パターン・マッチングの特殊文字は、他のデータ型と突き合わせられる {Value} に見つかる場合、リテラル文字として解釈されます。

すべての RODM データ型がリーフ仕様の {Class/Field} エレメントで使用できるわけではありません。以下の表は、それぞれの RODM データ型をリストし、データ型がリーフ仕様で使用できるかどうかを示し、 DOS ワイルドカードまたは正規表現がデータ型で使用できるかどうかを示し、データが文字ストリングから変換されてデータ型と一致する方法を示しています。

RODM データ型	リーフ仕様での使用	ワイルドカード/正規表現の使用	変換
ANONYMOUS	不可	N/A	N/A
ANONYMOUSVAR	可	不可	{Value} に含まれるのは、'0' または '1' の文字だけです。これは、比較の前に実ビット・ストリングに変換されます。
APPLICATIONID	不可	N/A	N/A
BERVAR	可	不可	{Value} に含まれるのは、'0' または '1' の文字だけです。これは、比較の前に実ビット・ストリングに変換されます。
CHARVAR	可	可	なし (文字ストリングとして扱われる)
CHARAVARADDR	不可	N/A	N/A
CLASSID	不可	不可	なし (文字ストリングとして扱われる)
CLASSIDLIST	不可	N/A	N/A
CLASSLINKLIST	不可	N/A	N/A
ECBADDRESS	不可	N/A	N/A
FIELDID	可	不可	{Value} は整数に変換されます。{Value} に浮動小数点変数に変換できない文字が含まれる場合、エラーになります。
FLOATING	可	不可	{Value} は浮動小数点変数に変換されます。 {Value} に浮動小数点変数に変換できない文字が含まれる場合、エラーになります。
GRAPHICVAR	不可	N/A	N/A

RODM データ型	リーフ仕様での使用	ワイルドカード/正規表現の使用	変換
INTEGER	可	不可	{Value} は整数に変換されます。{Value} に整数に変換できない文字が含まれる場合、エラーになります。
INDEXLIST	可	可	なし (IndexList 内の各値は、その実際の型に関係なく CharVar として扱われます。正常に比較を行うには、少なくとも 1 つの値が IndexList に対して比較が成功しなければなりません。)
METHODNAME	不可	N/A	N/A
METHODPARAMETERLIST	不可	N/A	N/A
METHODSPEC	不可	N/A	N/A
OBJECTID	不可	N/A	N/A
OBJECTIDLIST	不可	N/A	N/A
OBJECTLINK	不可	N/A	N/A
OBJECTLINKLIST	不可	N/A	N/A
OBJECTNAME	可	可	なし (文字ストリングとして扱われる)
RECIPIENTSPEC	不可	N/A	N/A
SELFDEFINING	不可	N/A	N/A
SHORTNAME	不可	不可	なし (文字ストリングとして扱われる)
SMALLINT	可	不可	{Value} は短整数に変換されます。{Value} に短整数に変換できない文字が含まれる場合、エラーになります。
SUBSCRIBEID	不可	N/A	N/A
SUBSCRIPTSPEC	不可	N/A	N/A
SUBSCRIPTSPECLIST	不可	N/A	N/A
TIMESTAMP	不可	N/A	N/A

RODM データ型	リーフ仕様での使用	ワイルドカード/正規表現の使用	変換
TRANSID	不可	N/A	N/A

コレクション定義オブジェクトの例

このセクションには、コレクション定義オブジェクトの例が示されています。

例 1:

DisplayStatus フィールドが 129 と等しくない GMFHS_Managed_Real_Objects_Class 内のすべてのオブジェクトを収集し、それをネットワーク・ビューに表示します。垂直バー文字 (|) は、CollectionSpec1 フィールドの区切り文字の役割を果たします。

このコレクションを記述する CDO オブジェクトは、RODM ローダー・ファイルで次のように指定されます。

```
CREATE INVOKER  ::= 0000003;
     OBJCLASS ::= Network_View_Collection_Class;
     OBJINST  ::= MyName = (CHARVAR) 'Example1';
     ATTRLIST
     Annotation ::= (CHARVAR) 'Example1 Annotation',
     LayoutType ::= (INTEGER) 1,
     CollectionSpec1 ::=
       (CHARVAR) '|GMFHS_Managed_Real_Objects_Class|
                              DisplayStatus|129|.NE.';
END;
```

この RODM Collection Manager は、 LayoutType に 1、Annotation に ″Example1 Annotation″ を持つ、 ″Example1″ という Network_View_Class オブジェクトを作成します。コレクション仕様は、単一の条件 (単一のリーフ仕様から成る) を表します。一致するオブジェクト・リストは、 Example1 ビューの ContainsObject フィールドにコピーされます。

例 2:

DisplayResourceOtherData フィールドに最初の 2 文字として CP、最後の 6 文字として Active が含まれる appnTransmissionGroupCircuit クラス (実際のクラス名は、1.3.18.0.0.2058) 内のすべてのオブジェクトと、 AggregationPriorityValue が 1、2、または 3 に等しい appnTransmissionGroupCircuit クラス内のすべてのオブジェクトを収集し、それらを Aggregate に入れます。垂直バー文字 (|) は、区切り文字の役割を果たします。

このコレクションを記述する CDO オブジェクトは、RODM ローダー・ファイルで次のように指定されます。

```
CREATE INVOKER  ::= 0000003;
          OBJCLASS ::= Aggregate_Collection_Class;
          OBJINST  ::= MyName = (CHARVAR) 'Example2';
       ATTRLIST
          DisplayResourceOtherData ::= (CHARVAR) 'Example2 Other Data',
          DisplayResourceUserData ::= (CHARVAR) 'Example2 User Data',
       CollectionSpec1 ::=
            (CHARVAR) '|1.3.18.0.0.2058|DisplayResourceOtherData|
```

```
                                    CP*Active|.CONTAINS=.',
        CollectionSpec2 ::=
          (CHARVAR) '|1.3.18.0.0.2058|AggregationPriorityValue|
                                    1|2|3|.EQ.|.AND.';
    END;
```

RODM Collection Manager は、DisplayResourceOtherData に ″Example2 Other Data″、 DisplayResourceUserData に ″Example2 User Data″ を持つ、 Example2 という GMFHS_Aggregate_Objects_Class オブジェクトを作成します。 Aggregate_Collection_Class オブジェクトで指定されない他のフィールドは、 GMFHS_Aggregate_Objects_Class で作成されるオブジェクトで使用されるデフォルトに設定されます。

コレクション仕様は、CollectionSpec*n* フィールドの両方で表されます。これは、 CollectionSpec1 フィールド、または CollectionSpec2 フィールドのいずれかに完全に配置されます。この例は、2 つのフィールドの連結を示しています。連結後の実際のコレクション仕様は、以下のとおりです。

```
|1.3.18.0.0.2058|DisplayResourceOtherData|CP*Active|.CONTAINS=.|1.3.18.0.0.2058|
AggregationPriorityValue|1|2|3|.EQ.|.AND.
```

このコレクション仕様は、複合条件を表します (これは 2 つのリーフ仕様から成ります)。 DOS ワイルドカードを使用して、 DisplayResourceOtherData 値と一致するオブジェクトを検索します。クラス 1.3.18.0.0.2058 に、オブジェクト ID 1、2、および 3 を持つ 3 つのオブジェクトが存在し、そして対応する DisplayResourceOtherData フィールドに次のものが含まれる場合、

- CPCP-supportedActive
- CP-CP Session Support
- CPCP-supportedNotActive

また、対応する AggregationPriorityValue フィールドに次のものが含まれる場合、

- -1
- 2
- 3

最初のリーフ仕様を評価した後、仮想スタックには次のものが含まれます。

- {1, 3}

{1, 3} は、リーフ仕様を評価することによって生成されるオブジェクト・リストです。 2 番目のリーフ仕様を評価した後、仮想スタックには次のものが含まれます。

- {2, 3}
- {1, 3}

.AND. 結合子により、2 つのオブジェクト・リストはスタックから除去されます。 それらの結合によりリスト {3} が生成され、スタックに追加されます。このオブジェクトは、完全な複合条件の結果です。これは、Example2 オブジェクトの AggregationChild フィールド (DUIFCUAP メソッドを使用) と IsPartOf フィールドの両方にリンクされます。

個々のリーフ仕様で 2 つの異なるクラスを使用する利点はありません。両方のリーフ仕様は、定義によって、共通のオブジェクトを含まないオブジェクト・リストを

生成します。 .AND. 結合子によって要求される結合は、常に空のオブジェクト・リストを生成します。結合子が .OR. の場合、2 つの異なるクラスが許容されます。

例 3:

MyName が TEST、英字種別文字、追加文字の番号、1、数値範囲の文字を連結したものと一致する、 GMFHS_Managed_Real_Objects クラス内のすべてのオブジェクトを収集します。例えば、英字の種別文字が B ではなく、 DisplayStatus が Satisfactory (適合) または Unsatisfactory (不良) のいずれかの ″TESTACPU10″ はその一例です。このリストに、MyName が TEST、英字種別文字、追加文字の番号を連結したものと一致する、 GMFHS_Aggregate_Objects_Class クラス内のオブジェクトを追加します。例えば、英字種別文字が B でない ″TESTACPUALL″ はその一例です。これをネットワーク・ビューに入れてください。

このコレクションを記述する CDO オブジェクトは、RODM ローダー・ファイルで次のように指定されます。

```
CREATE INVOKER  ::= 0000003;
       OBJCLASS ::= Network_View_Collection_Class;
       OBJINST  ::= MyName = (CHARVAR) 'Example3';
       ATTRLIST
       Annotation ::= (CHARVAR) 'Example3 Annotation',
       CollectionSpec1 ::=
          (CHARVAR) '|GMFHS_Managed_Real_Objects_Class|MyName|\^TEST[A-C].*1.$|
                           .CONTAINS.'
                     '|GMFHS_Managed_Real_Objects_Class|MyName|TESTB*|
                           .NCONTAINS.|.AND.'
                     '|GMFHS_Managed_Real_Objects_Class|DisplayStatus|130|
                           .LE.|.AND.'
                   '|GMFHS_Aggregate_Objects_Class|MyName|\^TEST[A-C].*$|
                           .CONTAINS.'
                     '|GMFHS_Aggregate_Objects_Class|MyName|TESTB*|
                           .NCONTAINS.|.AND.|.OR.';   END;
```

以下のオブジェクトが GMFHS_Managed_Real_Objects_Class に存在すると仮定します。

オブジェクト ID	MyName	DisplayStatus
1	TESTACPU10	131
2	TESTACPU11	129
3	TESTBCPU10	130
4	TESTBCPU11	132
5	TESTCCPU10	129
6	TESTCCPU11	132
7	TESTCCPU12	129
8	TESTCCPU12X	130
9	TESTDCPU10	129

以下のオブジェクトが GMFHS_Aggregate_Objects_Class に存在すると仮定します。

オブジェクト ID	MyName
10	TESTAAGGs
11	TESTBAGGS

オブジェクト ID	MyName
12	TESTCAGGS
13	TESTDAGGS

最初のリーフ仕様の式では正規表現が使用されています。 DOS ワイルドカードでは 5 番目の文字が A と C の間にくるように指定できないため、ここでは正規表現が使用されています。最初のリーフ仕様を評価した後、仮想スタックには次のものが含まれます。

```
{1, 2, 3, 4, 5, 6, 7}
```

2 番目のリーフ仕様を評価した後、仮想スタックには次のものが含まれます。

```
{1, 2, 5, 6, 7, 8, 9}
{1, 2, 3, 4, 5, 6, 7}
```

.AND. 結合子は、スタックからこれら 2 つのリストを除去して、次のものに置き換えます。

```
{1, 2, 5, 6, 7}
```

3 番目のリーフ仕様を評価した後、仮想スタックには次のものが含まれます。

```
{2, 3, 5, 7, 8, 9}
{1, 2, 5, 6, 7}
```

.AND. 結合子は、スタックからこれら 2 つのリストを除去して、次のものに置き換えます。

```
{2, 5, 7}
```

4 番目のリーフ仕様を評価した後、仮想スタックには次のものが含まれます。

```
{10, 11, 12}
{2, 5, 7}
```

5 番目 (最後) のリーフ仕様を評価した後、仮想スタックには次のものが含まれます。

```
{10, 12, 13}
{10, 11, 12}
{2, 5, 7}
```

.AND. 結合子は、スタックから上位 2 つのリストを除去して、次のものに置き換えます。

```
{10, 12}
{2, 5, 7}
```

最後に、.OR. 結合子は、スタックからリストを 2 つだけ除去して、次のものに置き換えます。

```
{2, 5, 7, 10, 12}
```

これが、複合条件によって戻される最終的なオブジェクト・リストとなり、例 3 オブジェクトの ContainsObjects フィールドにリンクされます。

NetView Resource Manager の使用

このセクションでは、 NetView Resource Manager (NRM) ビュー、およびそのカスタマイズの仕方について説明します。 NetView Resource Manager を使用すると、リソースの使用率および状況について、NetView タスクを NMC でグラフィカルにモニターおよび管理できます。 1 つの NMC を使用して企業内のすべての NetView をモニターできます。 NetView Resource Manager のセットアップおよび使用についての詳細は、以下を参照してください。

- *IBM Tivoli NetView for z/OS インストール: グラフィカル・コンポーネントの構成*
- *IBM Tivoli NetView for z/OS ユーザーズ・ガイド*

NetView Resource Manager ビュー

NRM がアクティブのとき、 **NetViewTasks** が NetView 管理コンソールのビュー・ツリーに表示されます。これにより、NRM ドメイン集合オブジェクトのビューがオープンされます。このビューから NRM Task 集合オブジェクト・ビューにナビゲートできます。 Task 集合体から、以下の実オブジェクトを持つビューにナビゲートできます。これは、統計モニターを表します。

- Status
- CPU (CPU 使用率)
- STG (ストレージ)
- MSGCT (メッセージ・キュー・カウント)
- MQOUT (出力メッセージ率)
- MQIN (入力メッセージ率)
- I/O (I/O 率)

モニターの値を見るには、「**リソース・プロパティー**」ノートブックをオープンしてください。

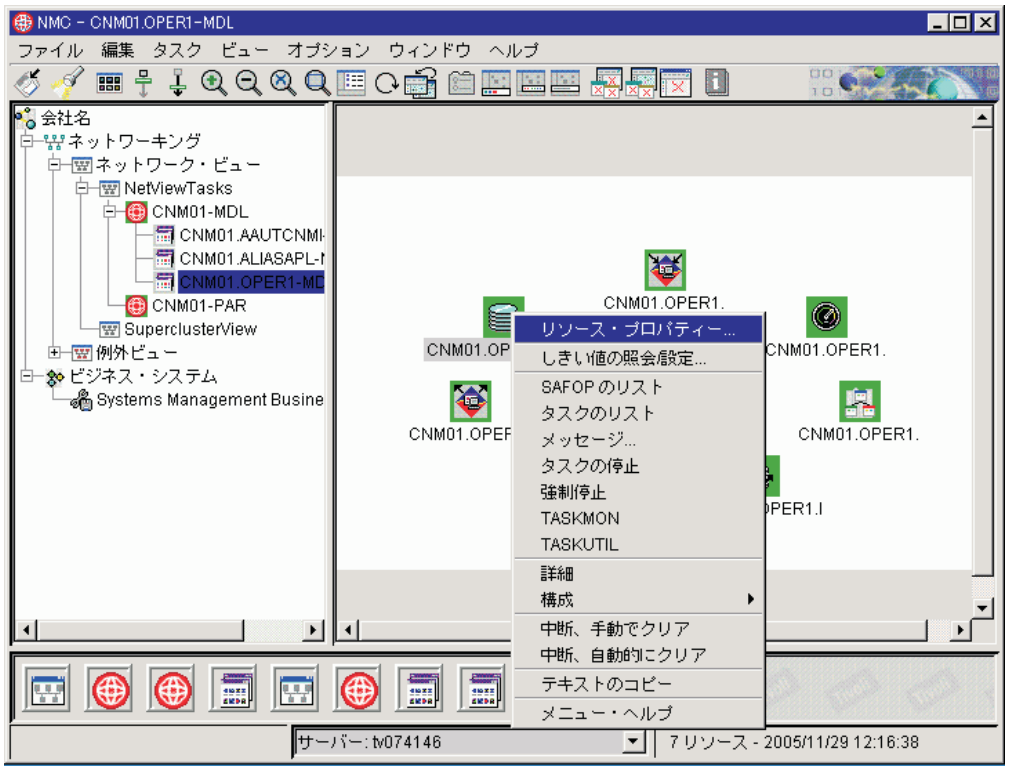


図 39. 「リソース・プロパティー」ノートブック

モニター値は、「**Data 1 (データ 1)**」フィールドにあります。

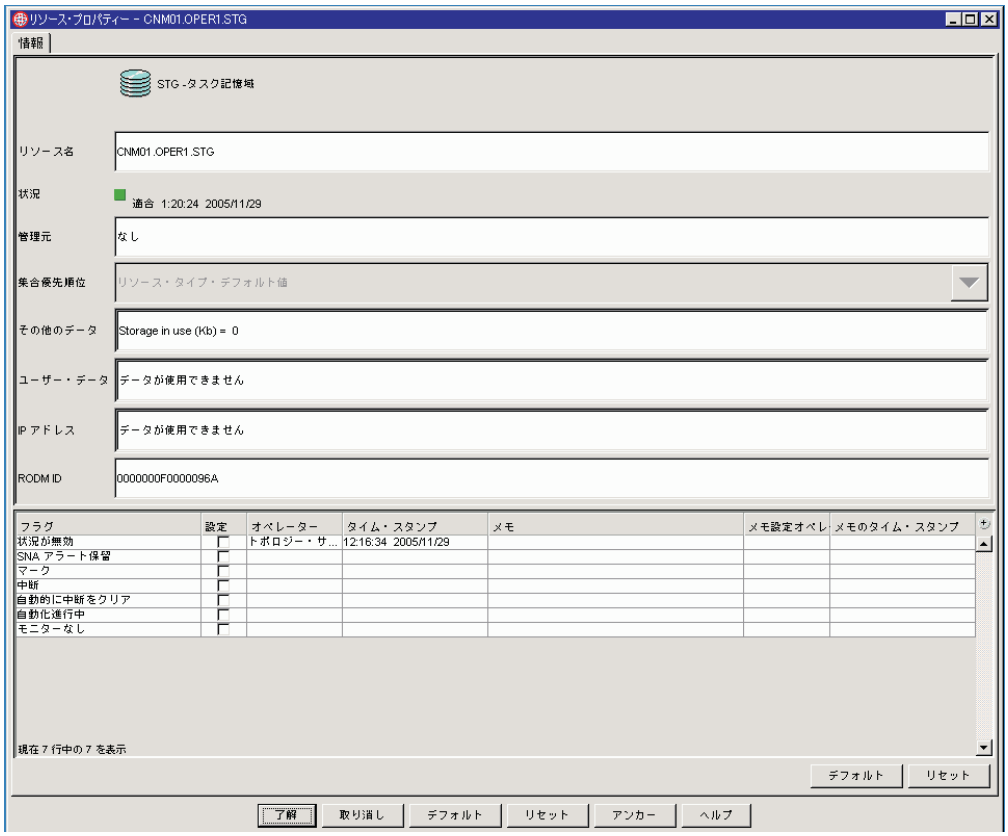


図 40. Data 1 (データ 1) フィールド

このフィールドは、自動的には動的に更新されません。このフィールドを動的に更新させる場合は、 191 ページの『NetView Resource Manager とともに DUIFVINS を使用する』を参照してください。

NRM 実オブジェクトのデフォルトの状況値は、以下のとおりです。

- タスクがアクティブ - Satisfactory (適合)
- タスクが非アクティブ - Unknown (不明)
- タスク状況が不明 - Unknown (不明)
- しきい値 1 に達した - Intermediate (中間)
- しきい値 2 に達した - Medium Unsatisfactory (中程度に不良)
- しきい値 3 に達した - Unsatisfactory (不良)

状況値は、統計モニターを表す各 NRM オブジェクトごとの RODM DisplayStatus フィールドに保管されます。

実オブジェクトの状況値は、カスタマイズ可能です。カスタマイズを行う方法についての詳細は、 NetView Resource Manager 初期設定パラメーター の下の CNMSTYLE にある 表示状況 セクションを参照してください。 NRM 実オブジェクトは、GMFHS_Managed_Real_NRM_Objects_Class クラスにあるため、 DisplayStatus が Unknown (不明) の場合は例外状態にマップしません。 Unknown DisplayStatus を NRM オブジェクトの例外状況にマップする場合は、 191 ページの『NetView Resource Manager で DUIFSMT を変更する』を参照してください。

NetView Resource Manager のオブジェクト情報

NetView Resource Manager 集合オブジェクトは、GMFHS_Aggregate_NRM_Objects_Class クラスにあります。 NRM 実オブジェクトは、GMFHS_Managed_Real_NRM_Objects_Class クラスにあります。すべての NRM オブジェクトは、MyName フィールドに ″NRM″ という接頭部があります。

NetView Resource Manager 用の NMC コマンド・サポート

NetView Resource Manager オブジェクトのすべてでコマンドが使用可能です。使用可能なコマンドは、表 21 に示すように、タスクのタイプに依存します。選択されたオブジェクトで右マウス・ボタンをクリックすることによって、使用可能なコマンドを選択できます。コマンドの結果は、NetView 管理コンソールのコンソール・ログに表示されます。

表 21. サポートされる NMC コマンド

タスク	使用可能コマンド
DSRBS	• DST
LIST SAFOP=opid	• OST • NNT • AOST (Autotask)
LIST taskname	• PPT • Automatic Tasks • DST • OPT • OST • NNT • AOST (Autotask) • HCT
LIST STATUS=TASKS	• NetView Aggregate
LIST STATUS=VOST	• VOST (Virtual OST)
Message	• OST • NNT • AOST (Autotask) • VOST (Virtual OST)

表 21. サポートされる NMC コマンド (続き)

タスク	使用可能コマンド
Query/Set Thresholds[1]	• NetView Aggregate • MAINTASK • PPT • Automatic Tasks • DST • OPT • OST • NNT • AOST (Autotask) • VOST (Virtual OST) • HCT
RECYCLET	• DST • OPT
RESOURCE	• NetView Aggregate
START HCL=hclname[1]	• HCT
START TASK=taskname[1]	• DST • OPT
STOP FORCE=taskname[1]	• DST • OPT • OST • NNT • AOST (Autotask) • VOST (Virtual OST) • HCT
STOP TASK=taskname[1]	• DST • OPT • OST • NNT • AOST (Autotask) • VOST (Virtual OST) • HCT

表 21. サポートされる NMC コマンド (続き)

タスク	使用可能コマンド
TASKMON	• NetView Aggregate • MAINTASK • PPT • Automatic Tasks • DST • OPT • OST • NNT • AOST (Autotask) • VOST (Virtual OST) • HCT
TASKUTIL	• NetView Aggregate • MAINTASK • PPT • Automatic Tasks • DST • OPT • OST • NNT • AOST (Autotask) • VOST (Virtual OST) • HCT

TASK 集合で出されるコマンドは一般に、実オブジェクトで出されるものと同一です。ただし、TASKMON コマンドは例外です。 TASKMON *taskname* は、集合 TASK オブジェクトで出されます。 TASKMON *taskname stat* は、以下で出されます。

- CPU
- STG
- IO
- MQIN
- MQOUT

TASKMON *taskname* は、STATUS および MSGCT オブジェクトで出されます。

注: Automatic Tasks についての詳細は、「*IBM Tivoli NetView for z/OS* ユーザーズ・ガイド」の『Automatic Tasks』セクションでリストされているタスクを参照してください。 DSIWEB および FLBTOPO を除いて、リストされているすべてのタスクが NRM で有効です。

1. これらのコマンドは、 NetView (CNMSCAT2/CNMSAF2) のデフォルトのセキュリティーによって保護されます。

ダイアログとして表示される Query/Set Threshold コマンドにより、有効な NRM しきい値を調査/変更できます。このダイアログは、STATUS オブジェクト以外のすべてのオブジェクトで使用可能です。しきい値の設定は、DEFAULTS/OVERRIDE コマンドでも行えます。ダイアログとして表示される Message コマンドにより、選択済みのオペレーター・タスクにメッセージを送信できます。

NetView Resource Manager で DUIFSMT を変更する

不明のリソース (非アクティブ・タスク) は、デフォルトで、例外状態にあるとは見なされません。 Unknown (不明) の DisplayStatus 値を GMFHS_Managed_Real_NRM_Objects_Class にあるリソースの例外状態にマップするには、 DUIFSMT を使用します。

例:

```
DUIFSMTE CLASS=GMFHS_Managed_Real_NRM_Objects_Class,             C
         XCPT=(UNSAT,DS152,DS153,DS154,DS155,DS156,DS157,DS158,DSC
         159,MEDUN,LOWUN,UNKWN)
```

DUIFSMT をアセンブルし、リンク・エディットするために、CNMSJH13 が提供されています。 DUIFSMT の詳細については、 123 ページの『例外ビュー用の DisplayStatus マッピング・テーブルをカスタマイズする』を参照してください。

NetView Resource Manager とともに DUIFVINS を使用する

NRM モニター値を動的に更新させるには、以下の RODM ローダー・ステートメントをコーディングします。

```
OP DUIFVINS INVOKED_WITH (SELFDEFINING)
   (
    (SMALLINT) 0
    (INTEGER)  7
    (OBJECTID) EKG_Method.DUIFVNOT
    (CLASSID)  GMFHS_Managed_Real_NRM_Objects_Class
    (FIELDID)  GMFHS_Managed_Real_NRM_Objects_Class.DisplayResourceOtherData
);
```

詳細については、 569 ページの『DUIFVINS: ビュー細分性インストール・メソッド (DUIFVNOT)』を参照してください。

NetView Resource Manager のサンプル・ローダー・ファイル

NetView Resource Manager オブジェクト・ビュー、および RODM Collection Manager を利用する集合体のサンプルが利用できます。 RODM Collection Manager は、RODM の内容をアクティブでモニターし、指定する基準に従ってビューおよび集合体を更新する NetView 機能です。サンプル JCL CNMSJH12 の 1 つのセクションは、 NetView Resource Manager オブジェクトの RODM Collection Manager コレクションを作成する、サンプル RODM ローダー・ファイルを提供します。

CNMSJH12 にある指示に従って、以下の例で示される DUIFNRM1 と DUIFNRM2 を含む 2 つの DD ステートメントのコメント化を解除してください。

```
// DD DSN=NETVIEW.V5R3M0.CNMSAMP(DUIFNRM1),DISP=SHR  <-NRM RCM SAMPLE
// DD DSN=NETVIEW.V5R3M0.CNMSAMP(DUIFNRM2),DISP=SHR  <-NRM RCM SAMPLE
```

サンプル DUIFNRM1 には、以下のビューおよび集合体が含まれています。

- ビュー - NRM_OSTs - ログオンしたすべての NetView ユーザー

- ビュー - NRM_CPU_USERS - Non-Satisfactory (不良) CPU ユーザー
- ビュー - NRM_HEALTH - 以下の集合体を含む、 NetView の一般的な状態
  - 集合体 - NRM_HEALTH_CPU - すべての Non-Satisfactory (不良) CPU オブジェクト
  - 集合体 - NRM_HEALTH_IO - すべての Non-Satisfactory (不良) IO オブジェクト
  - 集合体 - NRM_HEALTH_MQS - すべて Non-Satisfactory (不良) MQIN および MQOUT オブジェクト
  - 集合体 - NRM_HEALTH_MESSAGES - すべての Non-Satisfactory (不良) MSG オブジェクト
  - 集合体 - NRM_HEALTH_STORAGE - すべての Non-Satisfactory (不良) STG オブジェクト

これらのビューおよび集合体は、 NetView Resource Manager が現在管理しているすべての NetView からのデータを収集するので、それらは単一のシステムで最も効率的に使用できます。あるいは、『サンプル・ローダー・ファイルのカスタマイズ』の説明に従って、 RODM Collection Manager を使用して基準を変更することによって、単一のシステムを選択するように変更することもできます。

サンプル DUIFNRM2 は、オブジェクトを単一の NetView から選択する 1 つの例です。これには以下のビューが含まれます。

- ビュー - NRM_DSI_TASKS - DSI で始まる A01NV タスク

サンプル・ローダー・ファイルのカスタマイズ

サンプル RODM ローダー・ファイルをロードした後、 NMC コンソールを使用してコレクションを修正できます。アドミニストレーターとして、「**Tasks (タスク)**」->「**RODM Collection Manager**」をクリックして、 RODM Collection Manager GUI をオープンします。その GUI から、コレクションを参照および変更できます。変更を RODM コールド・スタートにわたって永続させるには、コレクションをホストに送信するときに、変更の保管先のデータ・セットまたは区分データ・セット・メンバーを指定します。その後、RODM をコールド・スタートするとき、修正済みコレクションを含むデータ・セットをロードすることにより、 NMC コンソール・ユーザーはそのデータ・セットを使用できるようになります。

第 6 章 アラートおよび解決を処理および受信するための GMFHS のカスタマイズ

この章では、GMFHS がアラートおよび解決を受信して処理する仕組みについて説明します。カスタマイズによる変更の結果としてこの処理が受ける影響についても説明します。 RODM で作成するオブジェクトの名前はアラートによって提供されるリソース名と一致させてください。

アラートと解決の受信およびモニター

GMFHS は、ハードウェア・モニターが自動処理する、すべてのアラートおよび解決用の主ベクトルのコピーを受信することによって、非 SNA リソースの状況および SNA リソースのアラート受信 (イベント通知) ユーザー状況をモニターします。GMFHS は、これらの主ベクトルが報告するリソースを識別します。 GMFHS は、リソースを表す RODM 内のオブジェクトに各状況報告書を関連付けます。

注: GMFHS *の非 SNA ドメイン* は、サービス・ポイント、トランザクション・プログラム、およびエレメント管理システムの任意の有効な組み合わせです。GMFHS の非 SNA ドメインは、 NetView プログラムと非 SNA ネットワークの間のインターフェースとして機能します。

この処理には 7 つのエレメントが関係しており、これらのすべてのエレメントはカスタマイズの影響を受けます。

- GMFHS がハードウェア・モニターから受信するもの
- SNA リソースを表す RODM 内のオブジェクト
- ネットワーク管理ゲートウェイ (NMG) を表す RODM 内のオブジェクト
- 非 SNA ドメインを表す RODM 内のオブジェクト
- 非 SNA リソースを表す RODM 内のオブジェクト
- DUIFEDEF アラート処理
- アラート変換表

GMFHS がハードウェア・モニターから受信するもの

NetView がアラートを受信すると、アラートは DUIFECMV コマンド・プロセッサーが実行される自動化テーブルを介して渡されます。このコマンド・プロセッサーは、GMFHS に情報を送信して、アラートの GMFHS 処理を開始します。 GMFHS によって受信される情報は以下のとおりです。

- 主ベクトルのコピー。
- 階層/リソース・リスト (H/RL) サブベクトルまたは階層名リスト (HNL) サブベクトルの内容から作成されたハードウェア・モニター・リソース階層。
- 主ベクトルの生成元である SNA ドメインの名前。
- DUIFEDEF アラート処理プログラムをバイパスする DUIFECMV に対するオプション・パラメーターのセット。これらのパラメーターは、CLASS、DOMAIN、INDICAT、OBJNAME、STATUS、および GMFHSDOM です。指定する場合は、以下のパラメーターが必要です。

- DOMAIN
- CLASS
- OBJNAME
- INDICAT

パラメーター INDICAT の値が 2 または 4 の場合は STATUS が必要です。GMFHSDOM はオプションです。

GMFHS は、リソース名を探す場合、H/RL または HNL サブベクトルではなく、ハードウェア・モニター・リソース階層を調べます。その論理はこれらの 2 つのサブベクトルの有無によって異なります。

パラメーターを DUIFECMV に指定すると、 GMFHS は、『SNA リソースを表す RODM 内のオブジェクト』、 195 ページの『NMG を表す RODM 内のオブジェクト』、 195 ページの『非 SNA ドメインを表す RODM 内のオブジェクト』、および 197 ページの『非 SNA リソースを表す RODM 内のオブジェクト』で説明する処理をバイパスします。 CLASS、DOMAIN、および OBJNAME はアラートをログに記録するオブジェクトを識別するために使用され、 STATUS は新しいリソース状況に値を指定します。 INDICAT には、実行する状況処理のタイプを指定します。INDICAT に値 1 または 3 を指定すると、 203 ページの『アラート変換テーブル』に示す手順が使用されます。

注: コマンド・プロセッサー DUIFECMV は、自動タスク DUIFEAUT で実行しなければなりません。 DUIFECMV およびそのオペランドの詳細については、NetView オンライン・ヘルプまたは「*IBM Tivoli NetView for z/OS コマンド解説書 第 I 巻*」を参照してください。

SNA リソースを表す RODM 内のオブジェクト

GMFHS はアラートまたは解決用の主ベクトルを受信すると、報告されているリソースが SNA リソースであるか非 SNA リソースであるかを判別しようとします。主ベクトルに H/RL サブベクトルおよび HNL サブベクトルのいずれも含まれていない場合は、 GMFHS は主ベクトルを SNA リソースとして扱います。これらのサブベクトルが両方とも存在し、かつ、ハードウェア・モニター・リソース階層にサービス・ポイント・リソース・タイプ (SP または PUGW) またはトランザクション・プログラム・リソース・タイプ (TP または PUGA) のいずれかが含まれている場合は、リソースは非 SNA リソースでなければなりません。 GMFHS は、 196 ページの『最初のメソッド』を使用してこの非 SNA リソースを処理します。これらのサブベクトルのいずれかが存在するがサービス・ポイント・タイプ (SP または PUGW) が存在しない場合、またはトランザクション・プログラム・リソース・タイプ (TP または PUGA) がハードウェア・モニター階層に含まれていない場合は、報告されているリソースについて SNA リソースと非 SNA リソースの両方の可能性があります。 GMFHS は、 197 ページの『2 番目のメソッド』に示すメソッドを使用します。

報告されているリソースが非 SNA リソースであると GMFHS が判別した場合は、GMFHS は、 197 ページの『非 SNA リソースを表す RODM 内のオブジェクト』に示す手順にしたがって処理を行います。報告されているリソースが SNA リソースであると判別した場合は、 GMFHS は、このセクションで説明する処理を行います。

GMFHS は、主ベクトルの元の SNA ドメイン名と一致する名前をもつ SNA_Domain_Class のオブジェクトを探します。このオブジェクトが検出されない場合は、GMFHS は主ベクトルを除去します。このオブジェクトが検出された場合は、 GMFHS は、SNA_Domain_Class オブジェクトの SNA ネットワーク (SNANet) フィールド、ピリオド (.) 区切り文字、およびハードウェア・モニター・リソース階層の最も右側にあるリソース名を連結した名前をもつ GMFHS_Shadow_Objects_Class のオブジェクトを探します。

例えば、以下のオブジェクトが SNA_Domain_Class で定義されているとします。

```
MyName : A10NV
SNANet : NETA
```

GMFHS が A10NV の元の SNA ドメイン名をもつアラートを受信し、そのアラートがハードウェア・モニター・リソース階層の最も右端の名前として NT69I073 をもつ場合は、 GMFHS_Shadow_Objects_Class で検出されるオブジェクトの名前は、次のとおりです。

```
NETA.NT69I073
```

GMFHS が GMFHS_Shadow_Objects_Class でこのオブジェクトを検出した場合は、このオブジェクトの UserStatus フィールドのイベント通知ビットをオンにし、イベント報告プロトコル・データ単位を作成してログに記録します。

SNA_Domain_Class および GMFHS_Shadow_Objects_Class にオブジェクトを作成する場合は、 SNA ネットワーク、SNA ドメイン、および SNA リソースの名前のオブジェクトの名前をそれらのドメインで調整する必要があります。

NMG を表す RODM 内のオブジェクト

GMFHS がアラートを解決するために 2 番目のメソッドが必要であると判別した場合は、 GMFHS は、アラート処理中に NMG オブジェクトを使用します。 NMG オブジェクトの使用方法は 197 ページの『2 番目のメソッド』で定義しています。

非 SNA ドメインを表す RODM 内のオブジェクト

GMFHS が非 SNA リソースのアラートまたは解決を受信した場合は、 GMFHS は、報告されている非 SNA リソースを含む非 SNA ドメインの識別を最初に判別します。次に、GMFHS はリソース自体を識別しようとします。 GMFHS は、ハードウェア・モニター・リソース階層の情報を検索引数として使用することでこれを行い、 Non_SNA_Domain_Class で定義されたオブジェクトの名前と比較します。この検索の仕組みを理解すると、ハードウェア・モニター・リソース階層に含まれる情報で Non_SNA_Domain_Class オブジェクトに名前を付けるための計画を作成する方法が容易に分かります。

GMFHS は、非 SNA ドメインの識別を判別するために、前述の 2 つのメソッドを使用します。これらのメソッドについては、この章で詳細に説明します。最初のメソッドでは、リソースは非 SNA リソースであると見なされます。このメソッドの適用後に、報告されているリソースの非 SNA ドメインを GMFHS が識別することができなかった場合は、 GMFHS は非 SNA リソースを識別することができないため、主ベクトルを除去します。 2 番目のメソッドでは、非 SNA リソース用ではないアラートは SNA リソースのものであると見なされ、 194 ページの『SNA リソースを表す RODM 内のオブジェクト』に示すステップが使用されます。

Non_SNA_Domain_Class でオブジェクトを定義する場合は、 GMFHS がハードウェア・モニター・リソース階層で検索する情報をユーザーの計画に含めてください。

最初のメソッド

前述のように、このメソッドによる処理の最初では、階層リソース・リスト・サブベクトルまたは階層名リスト・サブベクトルのいずれかがアラートに存在しているかどうか、およびサービス・ポイントまたはトランザクション・プログラムのいずれかまたは両方がアラートに含まれているかどうかを判別します。

サービス・ポイントが検出された場合はサービス・ポイントとして定義された階層エレメントで始め、サービス・ポイントが検出されなかった場合にはトランザクション・プログラムとして定義された階層エレメントで始めて、リソース階層にある名前をすべて連結した名前が GMFHS によって作成されます。 この連結において、名前は、それぞれピリオド (.) によって区切られます。

次に GMFHS は、Non_SNA_Domain_Class のオブジェクトすべての名前をこの連結と比較します。このクラスのオブジェクトは、すべて名前の長さに基づいて降順にソートされています。 GMFHS が現在の連結リストと一致する非 SNA ドメイン・オブジェクトを検出できなかった場合は、右端のオブジェクトが連結から除去され、 Non_SNA_Domain_Class がこの新しい名前について再度探索されます。この処理は、Non_SNA_Domain_Class オブジェクトが一致するか、連結リストにエレメントがなくなるまで続行されます。

例えば、ハードウェア・モニター・リソース階層に以下のリソース名とタイプの対が含まれているとします。

```
Name        Type

NMGPU5       PU
SP010        SP
RING010      RING
PRINTER1     PRTR
```

SP010.RING010 という名前の Non_SNA_Domain_Class にオブジェクトがあります。 GMFHS は、以下に示す順番で、以下に示す名前と正確に一致する名前をもつ Non_SNA_Domain_Class オブジェクトを検索します。

```
SP010.RING010.PRINTER1
SP010.RING010
SP010
```

GMFHS は、現在の連結リストに一致する最初のオブジェクトを処理します。この例では、SP010.RING010.PRINTER1 という名前の非 SNA ドメイン・オブジェクトはありませんが、 SP010.RING010 という名前のオブジェクトがあります。 GMFHS は、報告されたリソースのドメインを表すかのように SP010.RING010 という名前のオブジェクトを操作します。

この例では、SP010 という名前の非 SNA ドメイン・オブジェクトもあります。しかし、ソートされたリストの最初の非 SNA ドメイン・オブジェクトで一致するので、 SP010 の前にある SP010.RING010 で一致します。また、名前は正確に一致しなければなりません。 SP010.RING01 という連結名は SP010.RING010 という非 SNA ドメイン名とは一致しません。

2 番目のメソッド

アラート階層にサービス・ポイントまたはトランザクション・プログラムがない場合は、 GMFHS は、階層の右端のリソースから始め、リソース階層内の名前それぞれを各 NMG_Class オブジェクト名と 1 つ 1 つ比較します。

注: これは、最初のメソッドで使用した連結リストではなく、個々のリソース名です。一致しなかった場合は、アラートは SNA オブジェクトのアラートとして処理されます。

一致した場合は、同じ名前で一致するものを、すべての Non_SNA_Domain_Class オブジェクトについて検索します。一致しなかった場合は、アラートは SNA オブジェクトのアラートとして処理されます。これ以外の場合は、非 SNA ドメイン・オブジェクトで一致が生じています。

例えば、ハードウェア・モニター・リソース階層に以下のリソース名とタイプの対が含まれているとします。

```
Name        Type

NMGPU5       PU
PRINTER2     DEV
```

NMGPU5 という名前の NMG_Class 内にオブジェクトがあり、 NMGPU5 という名前の Non_SNA_Domain_Class 内にオブジェクトがあります。 GMFHS は、以下に示す順番で、以下に示す名前と正確に一致する名前をもつ NMG_Class オブジェクトを検索します。

```
PRINTER2
NMGPU5
```

NMG_Class オブジェクト (この場合は NMGPU5 という名前のオブジェクト) と一致する名前が検索されると、Non_SNA_Domain_Class 内で同じオブジェクト名について検査が行われます。一致した場合は、このドメインは、報告されているオブジェクトを含みます。

Non_SNA_Domain_Class 名が一致しない場合は、リソース・リストおよび NMG_Class において次の名前での検索が続行されないので注意してください。 NMG_Class が最初に一致した場合は、 Non_SNA_Domain 名は、リソース階層エレメントとも一致しているか、アラートが SNA リソース・アラートとして処理されているかのいずれかです。

非 SNA リソースを表す RODM 内のオブジェクト

GMFHS が 195 ページの『非 SNA ドメインを表す RODM 内のオブジェクト』に示すように非 SNA ドメインを検出した場合は、非 SNA リソースを識別しようとします。 GMFHS は、Non_SNA_Domain_Class オブジェクトの AlertProc フィールド内で指定されたロード・モジュールを呼び出すことによってこれを行います。 AlertProc フィールドの詳細については、「*IBM Tivoli NetView for z/OS データ・モデル・リファレンス*」を参照してください。

AlertProc フィールドのでデフォルト値は DUIFEDEF です。サンプル DUIFEDEF は NetView プログラムとともに出荷されます。 DUIFEDEF は、以下を戻します。

- GMFHS が受け入れ可能な 0 個以上のリソース名があるリスト

- 名前が単一の非 SNA リソースのものであるか、複数の非 SNA リソースのものであるかを指定するフィードバック・インディケーター
- これらの非 SNA リソースを含む RODM クラスの名前
- DisplayStatus の値

単一の非 SNA リソース

DUIFEDEF フィードバック・インディケーターによって名前が単一の非 SNA リソースのものであることが指定されている場合は、このリストの各名前に対して、GMFHS は、オブジェクトが検出されるかリストの終わりに到達するまで、DUIFEDEF によって戻されたクラスのオブジェクトを検索します。

最初に検出されたオブジェクトのみに対して、GMFHS は以下を行います。

- DUIFEDEF によって戻された DisplayStatus を判別します。それが存在しない場合は、アラートまたは解決によって報告された状況を GMFHS DisplayStatus に変換します。 DisplayStatus フィールドの詳細については、「*IBM Tivoli NetView for zOS データ・モデル・リファレンス*」を参照してください。
- DUIFEDEF によって戻されたクラスのオブジェクトにこの状況を関連付けます。
- イベント報告プロトコル・データ単位を作成します。
- このプロトコル・データ単位を Dbserver データベースに記録します。

複数の非 SNA リソース

DUIFEDEF フィードバック・インディケーターによって名前が複数の非 SNA リソースに指定されている場合は、 GMFHS は、このリストの各名前に対して、DUIFEDEF によって戻されたクラスのオブジェクトを検索します。検出された各オブジェクトに対して、GMFHS は以下を行います。

- DUIFEDEF によって戻された DisplayStatus を判別します。それが存在しない場合は、アラートまたは解決によって報告された状況を GMFHS DisplayStatus に変換します。
- DUIFEDEF によって戻されたクラスのオブジェクトと報告された状況を関連付けます。
- イベント報告プロトコル・データ単位を作成します。
- このプロトコル・データ単位をログに記録します。

非 SNA ドメインのリソースについて報告するアラートおよび解決は、すべて同じ AlertProc モジュールによって処理されます。変更した非 SNA ドメインに関するアラートおよび解決は、そのドメインの AlertProc モジュールが予想された結果を生成するように常に形式化されます。

DUIFEDEF アラート処理

AlertProc に対する値がない場合、または DUIFEDEF が AlertProc フィールドで命名されている場合は、 DUIFEDEF は、非 SNA リソースまたはアラートや解決に記述されたリソースの可能な名前およびこれらのリソースを含むクラスの名前を提供します。 NetView プログラムとともに提供されるサンプル DUIFEDEF は、単一または複数のリソースについて報告できる LAN からのアラートを検索します。

パラメーター

GMFHS は、以下のパラメーターとともに DUIFEDEF (または AlertProc フィールドで命名された他の任意のロード・モジュール) を実行します。

再入可能作業域へのポインター

AlertProc モジュールは再入可能であり、この作業域を使用します。この同じ作業域が、すべての AlertProc モジュールの間で共用されます。 AlertProc モジュールは、モジュールがこの作業域に保管する情報がモジュール呼び出し後も損なわれていないことを想定することができません。作業域の形式は以下のとおりです。

- GMFHS によって設定された作業域の長さを表すフルワード。これは、AlertProc モジュールで修正してはなりません。
- 以下のフィールドを含むフルワード。
  - GMFHS に戻る前に AlertProc モジュールによって設定された DisplayStatus 値を含む 1 バイト。 DisplayStatus 値とその意味は以下のとおりです。

    値　　意味

    0　　DisplayStatus は判別されませんでした。状況マッピング・テーブルを使用します。

    0 以外　使用する DisplayStatus 値。
  - 予約済みの 2 バイト。
  - GMFHS に戻る前に AlertProc モジュールによって設定された 2 進フィードバック・インディケーターを含む 1 バイト。フィードバック・インディケーターの値とその意味は以下のとおりです。

    値　　意味

    0　　可能な名前は、それぞれ非 SNA リソースを 1 つだけ識別します。GMFHS は RODM を照会して個々の名前を調べ、一致する名前を検出して、その状況を当リソースのみに関連付けます。

    0 以外　可能な名前は、それぞれ別々の非 SNA リソースを識別します。GMFHS は、RODM を照会して個々の名前を探し、検出された各名前についてその状況をリソースに適用します。

    注: NetView V3R1 より前では、2 進フィードバック・インディケーターは 4 バイトでした。この 4 バイトのうち 2 バイトはマイグレーションのために予約されており、 1 バイトは DisplayStatus 値に使用されています。 2 進フィードバック・インディケーターを 0 または 1 に設定します。
- 作業域の開始から最初の可能な名前までのオフセットを含むフルワード。
- 可能なリソース名を含む RODM クラスの名前。クラス名の形式は以下のとおりです。
  - 境界に配置されていない、クラス名の長さを含むハーフワード。
  - RODM クラス名を含む文字ストリング。
- 可能なリソース名のリストの形式は以下のとおりです。
  - 境界に配置されていない、リソース名の長さを含むハーフワード。
  - リソース名を含む文字ストリング。

複数の名前が戻される場合は、名前は境界に配置されずに連結されます。可能な名前のリストは、2 進ゼロを含むハーフワードで終わるため、境界に配置されません。 GMFHS は、最初の可能な名前の長さがゼロのリストを受け入れます。

第 2 再入可能作業域へのポインター

この作業域は、各 AlertProc モジュールに提供される個別の作業域であり、長さが 4088 (X'FF8') バイトです。 AlertProc モジュールが呼び出し間で変更されない情報を保存する必要がある場合は、その情報をこの作業域に保管することができます。

EMDomain フィールドの値

Non_SNA_Domain_Class オブジェクトの EMDomain フィールドは、ドメイン ID を表す値です。この値は、AlertProc モジュールによって候補名リストを作成するために使用することができます。 EMDomain フィールドの詳細については、「*IBM Tivoli NetView for z/OS データ・モデル・リファレンス*」を参照してください。

DomainCharacteristics フィールドの値

Non_SNA_Domain_Class の DomainCharacteristics フィールドは、ドメインによってサポートされている機能を示します。このフィールドの詳細については、「*IBM Tivoli NetView for z/OS データ・モデル・リファレンス*」を参照してください。

構造体の配列へのポインター

各構造体には、主ベクトル内のサブベクトルが記述されます。各構造体の形式は以下のとおりです。

- サブベクトルへのポインターを含むフルワード。主ベクトル内の最後のサブベクトルを指すフルワード・ポインター内では、左端のビットがオンになります。
- フルワードの保全検証フラグ。このフルワードがすべてゼロである場合は、サブベクトルの長さが妥当性検査されます (つまり、ゼロではなく主ベクトルの長さに含まれているかどうかが検査されます)。サブベクトルにサブフィールドがある場合は、サブフィールドの長さは妥当性検査されません。このフルワードにゼロでないものがある場合は、サブベクトルの長さが妥当性検査されます。サブベクトルにサブフィールドがある場合は、サブフィールドの長さも妥当性検査されます。

プロダクト・セット ID サブベクトル (X'10') の構造体の直後に続く組み込みプロダクト ID サブベクトル (X'11') ごとに別々の構造体があります。

ハードウェア・モニター・リソース階層へのポインター

これは、ハードウェア・モニターによって提供され、 H/RL または HNL サブベクトルに含まれたリソース名とタイプの対のテキスト表示を含むリストです。各名前とタイプの対には、左寄せされて右側をブランクで埋められた 8 文字のリソース名、および左寄せされて右側をブランクで埋められた 4 文字のリソース・タイプが含まれます。 GMFHS は、Non_SNA_Domain_Class オブジェクト名を構成する名前に続くハードウェア・モニター・リソース階層の部分を提供します。

196 ページの『最初のメソッド』の例では、GMFHS は、1 つの名前とタイプの対を含むリストを提供しています。

```
PRINTER1PRTR
```

ハードウェア・モニター・リソース階層の長さへのポインター

例では、GMFHS は 10 進値 12 を含むフルワードへのポインターを提供しています。

レジスター 15 に関する規則

DUIFEDEF (または他の任意の AlertProc モジュール) は、以下のようにレジスター 15 に値を戻します。

値　　意味

0　　GMFHS によって提供された第 1 再入可能作業域には、前述のように形式化された 0 個以上の可能なリソース名のリスト、リソースが含まれる RODM クラスの名前、およびオプションでリソースの DisplayStatus の値が含まれます。名前がない場合は、AlertProc モジュールは正常に完了したが、非 SNA リソースを識別しなかったことを表しています。

GMFHS は、作業域のフルワード・フィードバック・インディケーターに従って、名前リストおよび状況を処理します。

0 より大きい場合

GMFHS によって提供された第 1 再入可能作業域に、可能な名前および RODM クラス名をすべて保持するために十分な大きさがありません。レジスター 15 の値は、可能な名前および RODM クラス名をすべて入れるために必要な作業域の長さです。

今回初めて AlertProc モジュールがこのアラートに対してより大きな作業域を要求したのであれば、 GMFHS は、要求を満たせるだけのスペースをさらに獲得して、再度 AlertProc モジュールを呼び出します。それ以外の場合、GMFHS はシステム・エラー概要にエラーを記録し、コンソール・メッセージ DUI3913E を出します。

0 より小さい場合

AlertProc モジュールが呼び出しパラメーターのエラーを検出しました。

GMFHS は、システム・エラー概要にエラーを記録し、コンソール・メッセージ DUI3913E を出します。

デフォルト DUIFEDEF の処理

サブベクトル X'51' とサブベクトル X'5D' の両方とも主ベクトルに存在しない場合は、アラートまたは解決が 1 つの非 SNA リソースについてのみ状況を報告します。 DUIFEDEF は、以下のステップを実行します。

- 1 つまたは 2 つの可能な名前のリストを作成します。
  - 最初の名前は以下を連結したものです。
    - 第 3 呼び出しパラメーターに提供される、後続ブランクを含まない EMDomain フィールド。
    - ピリオド (.) 区切り文字。
    - 後続ブランクを含まず、 (DomainCharacteristics フィールドの値によって指定された場合は) ピリオド (.) によって区切られた、ハードウェア・モニター・リソース階層の全リソース名。 DomainCharacteristics フィールドのこの値に関する詳細については、「*IBM Tivoli NetView for z/OS データ・モデル・リファレンス*」を参照してください。

  - 2 番目の名前は以下を連結したものです。
    - 第 3 呼び出しパラメーターに提供される、後続ブランクを含まない EMDomain フィールド。
    - ピリオド (.) 区切り文字。
    - ハードウェア・モニター・リソース階層内の最後のリソース名。

    2 番目の名前が最初のものと同一である場合は、最初のものしか GMFHS に戻されません。
- 2 進フィードバック・インディケーターに値 0 を戻します。
- RODM クラス名に GMFHS_Managed_Real_Objects_Class の値を戻します。

サブベクトル X'51' またはサブベクトル X'5D' のいずれかが主ベクトル内にある場合は、アラートまたは解決は 1 つまたは複数の非 SNA リソースについて状況を報告します。 DUIFEDEF は、以下のステップを実行します。

- 0 個以上の可能な名前のリストを作成します。
  - 以下のサブフィールドを検索します。
    X'03' - ローカル個別 MAC アドレス
    X'04' - リモート個別 MAC アドレス
    X'06' - リング障害ドメイン記述
    X'08' - 単一 MAC アドレス
    X'23' - ローカル個別 MAC 名
    X'24' - リモート個別 MAC 名
    X'26' - 障害ドメイン名
    X'28' - 単一 MAC 名
  - 検出されたサブフィールドごとに、以下の可能な名前のいずれかを作成します。

    ```
    X'03', X'04', X'08', X'23',
    X'24', X'28'
    ```

    または以下の 2 つの可能な名前を作成します。

    ```
    X'06', X'26'
    ```
  - アドレスを変換し、16 進数を表示します。可能な名前は、それぞれ以下を連結したものです。
    - 後続ブランクを含む第 3 呼び出しパラメーターに提供された EMDomain フィールド。
    - ピリオド (.) 区切り文字。
    - サブフィールド内の名前またはアドレス。 DomainCharacteristics フィールドで要求された場合は、候補名リスト内の全リソース名をピリオドで区切ることができます。 DomainCharacteristics フィールドのこの値に関する詳細については、「*IBM Tivoli NetView for zOS データ・モデル・リファレンス*」を参照してください。
    - 結果の名前がすでにリストにある名前と重複する場合は、その名前はリストに追加されません。
    - 結果のオブジェクト名が RODM で許される最大文字数の 254 より長い場合は、名前はリストに追加されません。

- サブフィールド X'23'、X'24'、 X'26'、または X'28' 内の名前がすべてブランクである場合は、 GMFHS は可能な名前を作成しません。
- サブフィールド X'23'、X'24'、X'26'、および X'28' 内の後続ブランクは可能な名前に含まれません。これらのサブフィールドの組み込みブランクは、可能な名前に含まれます。現在、RODM は組み込みブランクを持つオブジェクト名を許可していないので、 GMFHS は、そのような名前のオブジェクトを RODM で正常に検索することはできません。

- 2 進フィードバック・インディケーターに値 1 を戻します。
- RODM クラス名に GMFHS_Managed_Real_Objects_Class の値を戻します。

例えば、この Non_SNA_Domain_Class オブジェクトの EMDomain フィールドの値が DOMAIN1 であるとします。サブベクトル X'51' またはサブベクトル X'5D' がない場合は、 DUIFEDEF は、フィードバック・インディケーター値 0 と可能な名前を 1 つ戻します。

```
DOMAIN1.PRINTER1
```

ただし、リング障害ドメイン記述サブフィールドを含むサブベクトル X'51' またはサブベクトル X'5D' があり、サブフィールド内のアドレスが X'00101AF1CE74' および X'00101AF1CE0B' である場合は、 DUIFEDEF は、フィードバック・インディケーター値 1 と 2 つの可能な名前を戻します。

```
DOMAIN1.00101AF1CE74
DOMAIN1.00100AF1CE0B
```

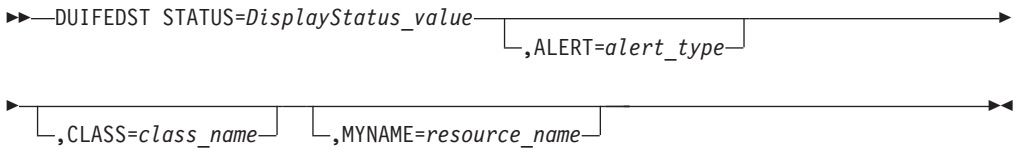
アラート変換テーブル

DUIFEUSR および DUIFEIBM は、再入不能および再利用不能ロード・モジュールに含まれるアラート変換テーブルです。 DUIFEIBM は、ロード・モジュールとしてのみ提供されます。 DUIFEUSR は、ロード・モジュール、アセンブラー・ソース・モジュール、および DUIFEDST というアセンブラー・マクロとして提供されます。

DUIFEIBM には、IBM によって提供されるデフォルト・コード・ポイント変換が含まれます。 DUIFEUSR は空のテーブルとして提供されます。 DUIFEUSR にコード・ポイント変換を追加して、 DUIFEIBM に含まれる対応するコード・ポイント変換をオーバーライドすることができます。

1 つまたは複数の DUIFEDST マクロを DUIFEUSR に追加して、アラート・コード・ポイント変換を定義することができます。マクロの形式は以下のとおりです。

DUIFEDST



ここで、

STATUS=*DisplayStatus_value*

このテーブル項目の NetView DisplayStatus 値です。例えば、DisplayStatus 値 UNSATISFACTORY を割り当てるには、STATUS=UNSATISFACTORY とコーディングします。 STATUS キーワードは必須です。有効な値は、以下のとおりです。

- SATISFACTORY
- UNSATISFACTORY
- INTERMEDIATE
- UNKNOWN
- DS136 (ユーザー肯定 1)
- DS137 (ユーザー肯定 2)
- DS138 (ユーザー肯定 3)
- DS139 (ユーザー肯定 4)
- DS140 (ユーザー肯定 5)
- DS141 (ユーザー肯定 6)
- DS142 (ユーザー肯定 7)
- DS143 (ユーザー肯定 8)
- MEDSA (中程度に適合)
- LOWSA (やや適合)
- DS152 (ユーザー否定 1)
- DS153 (ユーザー否定 2)
- DS154 (ユーザー否定 3)
- DS155 (ユーザー否定 4)
- DS156 (ユーザー否定 5)
- DS157 (ユーザー否定 6)
- DS158 (ユーザー否定 7)
- DS159 (ユーザー否定 8)
- MEDUN (中程度に不良)
- LOWUN (やや不良)

ALERT=*alert_type*

基本アラートまたは総称アラートからの任意の有効なアラート・タイプです。 ALERT キーワードはオプションです。

注: NETCENTER サービス・ポイントは、セッション・プロトコル・アラートにアラート・タイプ X'12' (不明) を使用し、解決をシミュレートします。 NETCENTER サービス・ポイントとの互換性を維持するため、 DUIFEIBM 変換テーブルはアラート・タイプ X'12' に対するコード・ポイント変換を提供しません。アラート・タイプ X'12' に対するコード・ポイント変換を DUIFEUSR 変換テーブルに追加することができます。 NETCENTER を使用しており、かつアラート・タイプ X'12' に対するコード・ポイント変換を追加する場合は、これらのアラートを SATISFACTORY に変換します。 NETCENTER の解決は、すべてこのコード・ポイント変換で指定した状況に変換されます。

CLASS=*class_name*

このテーブル項目に適用される RODM クラスの名前です。 CLASS キーワードはオプションです。

MYNAME=*resource_name*
このテーブル項目に適用されるリソースまたはリソース・グループの MyName です。ワイルドカード文字 (*) をサフィックスとして使用して、リソース・グループを指定することができます。 MYNAME キーワードはオプションです。

GMFHS は、テーブルを順番に検索し、最初にアラートと一致するものを探します。したがって、DUIFEDST マクロを固有性の高い順に指定して、必要な状況処理が行われるようにしてください。

alert_type X'03' (パフォーマンス) が、 'A.B.C' で始まる全リソースについて UNSATISFACTORY の DisplayStatus_value になるように指定するには、以下のステートメントをコーディングしてください。

```
DUIFEDST MYNAME=A.B.C*,ALERT=03,STATUS=UNSATISFACTORY
```

DUIFEDST の最後のステートメントは、以下のようにしなければなりません。

DUIFEDST END

このステートメントは、アセンブラー・ソース・ファイル内の END ステートメントの直前になければなりません。

表 22 に、DUIFEIBM 内に存在するデフォルト・アラート変換を示します。

表 22. DUIFEIBM 内のデフォルト・アラート変換

アラート・タイプ	DisplayStatus 値
01	UNSATISFACTORY
02	UNSATISFACTORY
03	UNSATISFACTORY
04	INTERMEDIATE
0A	INTERMEDIATE
0F	SATISFACTORY
10	UNSATISFACTORY
11	INTERMEDIATE
12	RESERVED
14	INTERMEDIATE
15	INTERMEDIATE

第 3 部 RODM をネットワークの自動化に使用する

第 7 章 自動化コードを作成する

この章では、GMFHS データ・モデルおよび SNA トポロジー・マネージャー・データ・モデルを含む、自動化アプリケーション、および NetView 提供のデータ・モデルとインターフェースをとるメソッドの作成方法を説明します。さらには、それぞれの自動化の必要性を満たすよう NetView 提供のデータ・モデルを拡張する際の、関連規則および考慮事項についても説明します。 RODM に関連する自動化アプリケーションを設計するときは、独自のデータ・モデルを設計することも、NetView 提供のデータ・モデルを使用することもできます。

自動化用 NetView 提供のデータ・モデルを使用する利点

NetView 提供のデータ・モデルを使用せずに独自のデータ・モデルを作成することもできますが、 NetView 提供のデータ・モデルを基本として自動化ルーチンを設計する際の以下の利点を考慮する必要があります。

- NetView 提供のデータ・モデルは、ネットワークのモデル化用に設計されており、使用すると、独自のデータ・モデルを設計してインストールするという、時間と費用がかかる余計なステップを回避できます。
- NetView 提供のデータ・モデルには、DisplayStatus フィールドなど、自動化ルーチンが使用できる多数のフィールドおよびオブジェクトがあります。 NetView 提供のデータ・モデルを用いて RODM でオブジェクトを定義すると、これらのフィールドは NetView コードによって保守されます。フィールドを最新に保つためのコーディングを行う必要がないため、リソースの節約になります。
- NetView 管理コンソールは、 NetView 提供のデータ・モデルの情報を用いて、ネットワークをモニターするワークステーション・オペレーター用のネットワークのビューを動的に作成します。オペレーターは、ビューに表示されたリソースの関係に基づいて問題の原因を推論し、修正アクションを始めるコマンドを出します。自動化にオペレーターと同じデータ・モデルを使用する場合は、ネットワーク・オペレーターのタスク用の自動化ルーチンを設計できるだけでなく、自動化をネットワークの操作と保守に携わる人々に相関させることができます。

RODM ロード・ファイルとして NetView プロダクトとともに提供される GMFHS データ・モデルは、自動化の要件をすべて満たさない場合があります。例えば、ユーザーの自動化コードが、現在 GMFHS データ・モデルでは提供されないリンク・オブジェクトの回線速度フィールドを必要とする場合があります。出荷されたソース・データをユーザーの要件に応じて変更することができます。 GMFHS データ・モデルの変更については、「*IBM Tivoli NetView for z/OS* データ・モデル・リファレンス」を参照してください。この資料には、データ・モデル内のクラスおよびフィールドがすべて記載されています。

GMFHS フィールドの変更についてアプリケーションに通知する

RODM は、データ・モデルのフィールドの値が変更になると、ユーザー・アプリケーションに通知することができます。この通知の設定方法については、 364 ページの『RODM 通知プロセス』を参照してください。個々のオブジェクトのフィールドもしくはクラスのフィールドの通知申請を作成することができます。クラスのフィールドの通知申請を作成すると、そのフィールドがクラスのオブジェクトで変更になったときに、ユーザー・アプリケーションが通知を受けます。

NetView プロダクトは、 RODM で使用する汎用の通知メソッドを提供しています。これらの通知メソッドは、データ・モデルのフィールドに対する変更をユーザー・アプリケーションに通知するのに使用することができます。これらのメソッドでは、フィールドになんらかの変更が加えられたときに通知するか、あるいはフィールドの値が指定した値以上になったときだけ通知するようになっています。まず、オブジェクトもしくはクラスのフィールドに通知メソッドを定義します。次にアプリケーションは、その通知メソッドの通知キューに申請します。これらのメソッドの説明については、 548 ページの『NetView 提供のメソッド』を参照してください。 NetView 提供のメソッドが要求に合わない場合は、独自の通知メソッドを作成することもできます。

自動化に有効なフィールドの 1 つは、DisplayStatus フィールドです。このフィールドはリソースの状況を示します。このフィールドに自動化コードを登録すると、リソースの状況が変更になったときに RODM からコードに通知されます。例えば、リソースの状況が適合から不良に変わる場合は、コードは、これが新しい問題なのかあるいは高レベルの問題がある徴候なのかを判別するために、このオブジェクトの関係やそれに接続する他のオブジェクトの状況を検査することができます。 214 ページの『GMFHS 自動化の例』のプログラムの例で、このタスクを実行します。

RODM は、フィールドが変更になったときに自動化コードに通知するので、自動化コードは、通知から得られた情報を分析して、妥当なアクションをとることができます。

GMFHS 定義のフィールドにアクセスし、変更する

自動化コードは、これらのフィールドの値を判別するために、 GMFHS データ・モデルで定義されたすべてのフィールドにアクセスすることができます。自動化コードは、フィールドによっては変更できるものもあります。コードは、以下の規則に従っていなければなりません。

- クラス・フィールドの値を変更してはなりません。変更するのは、オブジェクト・フィールドの値に限られます。この規則の例外としては、Global_NLS_Parameters_Class の CodePage フィールドと Global_Aggregation_Parameters_Class の UnknownThreshold フィールドがあります。
- 次のクラスに属しているオブジェクトのフィールドの値は変更できません。
  - Agent_Parent_Class
  - Domain_Parent_Class
  - View_Information_Reference_Class
  - View_Information_Object_Class

- どのオブジェクトも、その DefaultAggregationPriorityCopy フィールドの値を変更してはなりません。
- GMFHS_Aggregate_Objects_Class の次のフィールドの値は、変更してはなりません。
  - SuspendedCount
  - TotalRealResourceCount
  - StatusGroupCounts
  - PriorityXCPTCount
  - XCPTCount
  - NOXCPTCount
  - UnknownCount
- 変更メソッドがインストールされる GMFHS データ・モデル・フィールドの場合、自動化コードはメソッドを起動する機能を使用しなければなりません。例えば、EKG_ChangeSubfield 機能ではなく EKG_ChangeField 機能または EKG_ChangeMultipleFields 機能を使用します。変更メソッドが起動されない場合は、集約計算などの操作は実行されません。
- GMFHS は、GMFHS が使用するすべてのフィールドに通知メソッドをインストールして、グラフィック・ワークステーション・ビューを作成します。自動化コードは、通知メソッドがインストールされた GMFHS データ・モデルのフィールドを変更するときは、メソッドを起動する機能を使用しなければなりません。例えば、EKG_LinkNoTrigger 機能ではなく EKG_LinkTrigger 機能を使用します。通知メソッドが起動されないと、GMFHS は、変更のビューをモニターするオペレーターに通知することができません。 GMFHS がフィールドに通知メソッドをインストールしているかどうかを判別するには、特定のフィールド記述を参照してください。
- フィールドによっては、変更が、これらのフィールドの変更用に設計された NetView 提供のメソッドを使用した場合に限られるものもあります。これらのフィールドを変更できるメソッドについては、『GMFHS メソッドを使用する』で説明します。
- 照会メソッドを、GMFHS データ・モデルのフィールドに追加することはできません。
- 変更メソッドは、GMFHS データ・モデルのどの IBM 作成フィールドにも追加してはなりません。データ・モデルに追加するフィールドには、変更メソッドを追加することができます。

GMFHS メソッドを使用する

このセクションでは、自動化のアプリケーションおよびメソッドがアクセスできる GMFHS メソッドの要旨を説明します。各メソッドの入出力パラメーターを含む詳細については、556 ページの『GMFHS メソッド』を参照してください。

DUIFCCAN: すべての注記の消去

DUIFCCAN メソッドを使用すると、それぞれの実オブジェクトおよび集合オブジェクトのトポロジー・コンソールを経由しないで、すべての注釈フィールドを消去す

ることができます。 DUIFCCAN のオペレーター ID は、その注釈が、オペレーターによってではなく、このメソッドによって消去されたことを示すように設定されています。

DUIFCATC: 集約しきい値の変更

これは、GMFHS_Aggregate_Objects_Class の集約しきい値フィールドにインストールされる変更メソッドで、これらのフィールドの値のどれかに変更があれば起動されます。このメソッドは、アプリケーションが直接実行することはありません。しかし、アプリケーションを設計する際に、複数のしきい値が変更されるオブジェクトの場合は、最後の変更以外の変更では常に非トリガー (サブフィールド) 形式の変更要求を使用することを検討してください。これによって、不要な集約計算メソッドを起動しないようにします。

DUIFCLRT: リソース・タイプのリンク

オブジェクト独立メソッドは、 Display_Resource_Type_Class オブジェクトを実オブジェクトおよび集合オブジェクトにリンクします。このメソッドは、GMFHS のネットワーク定義ステートメントの作成時に、 INVOKED_WITH RODM ロード機能プリミティブ・ステートメントを用いて起動されるようになっています。Display_Resource_Type_Class のオブジェクトを GMFHS_Managed_Real_Objects_Class のオブジェクトかその子クラス、または GMFHS_Aggregate_Objects_Class のオブジェクトにリンクまたはリンク解除するアプリケーションの場合は、常にこのメソッドを使用してください。 DUIFCLRT メソッドを使用すると、集合リソースの DisplayStatus は、リンクまたはリンク解除のために必要な場合は必ず再計算されます。このメソッドのパラメーターの説明については、 557 ページの『DUIFCLRT: リソース・タイプ・リンク・メソッド』を参照してください。

DUIFCUAP : 集約パスの更新

このオブジェクト独立メソッドは、RODM ロード機能の INVOKED WITH プリミティブを使用して実行されることを意図しています。集約階層を変更するアプリケーションの場合は、常にこのメソッドを使用してください。このメソッドを使用すると、集合リソースのカウント・フィールドおよび DisplayStatus は、必ず変更での必要に応じて再計算されます。このような変更の後に DUIFFAWS メソッド (集約ウォーム・スタート) を実行しても同じことが行われますが、これは費用がかかり、初期化用のメソッドと考えられていますので、注意してください。

DUIFCUUS: ユーザー状況の更新

この名前付きメソッドは、アプリケーションが GMFHS_Displayable_Objects_Parent_Class 内のオブジェクトの UserStatus フィールドを更新する際に使用することができます。 UserStatus フィールドは直接変更できますが、シャドー・オブジェクトの集約の中断のような、関係のない誤った変更を避けるために、DUIFCUUS メソッドを使用します。

DUIFECDS: 表示状況の変更

この名前付きメソッドは、アプリケーションが GMFHS_Managed_Real_Objects_Class 内の DisplayStatus フィールドを更新する際に使用することができます。このメソッドには、ターゲット・オブジェクトの SourceStatusUpdateTime フィールド値を呼び

出し側が指定した値に照らして検査し、指定された状況がオブジェクトの状況より古い場合は更新が適用されないようにする利点があります。

DUIFFAWS: 集約ウォーム・スタート

このオブジェクト独立メソッドは、処理を続行する前に集合リソースのカウントおよび DisplayStatus の値が正しいことを確認する必要があるアプリケーションによって実行されます。短命パラメーターは必要ありません。

メソッド名が DUIFCUAC のメッセージ DUI4020A を受け取った場合は、このメソッドを実行することが必要な場合があります。これは、集約階層によって伝わる状況に問題があることを示します。 GMFHS CONFIG NETWORK コマンドを用いて GMFHS を再初期化するときは、 DUIFFAWS メソッドを起動します。

このメソッドは、RODM ロード機能プリミティブ・ステートメント OP DUIFFAWS INVOKED_WITH によって起動することもできます。

DUIFFIRS: 初期リソース状況の設定

このオブジェクト独立メソッドは、GMFHS が、 Non_SNA_Domain_Class オブジェクトの ContainsResource フィールドにリンクした実リソース・オブジェクトのすべての DisplayStatus を、そのドメイン・オブジェクトの InitialResourceStatus 値に設定するときに使用します。独自の実リソース DisplayStatus (GMFHS の代わりに) を初期化し保守するアプリケーションでは、このメソッドが役立つことがわかります。

DUIFFRAS: 集合体状況の再計算

このオブジェクト独立メソッドは、どのアプリケーションも、GMFHS_Aggregate_Objects_Class オブジェクトのすべての DisplayStatus 値を再計算させるときに実行することができます。このメソッドは、集合オブジェクトのカウント・フィールドは正しくても、 DisplayStatus は誤りかもしれないと考えられる場合に役立ちます。 DUIFFRAS メソッドには、入力パラメーターは不要です。 DisplayStatus 以外のフィールドが正しくない可能性がある場合は、代わりに DUIFFAWS メソッドを使用してください。

このメソッドも、RODM ロード機能プリミティブ・ステートメント OP DUIFFRAS INVOKED_WITH で起動することができます。

DUIFFSUS: 不明状況の設定

このオブジェクト独立メソッドは、GMFHS が、 Non_SNA_Domain_Class オブジェクトの ContainsResource フィールドにリンクした実リソース・オブジェクトのすべての DisplayStatus を不明値に設定するときに使用します。独自の実リソース DisplayStatus (GMFHS の代わりに) を初期化し保守するアプリケーションでは、このメソッドが役立つことがわかります。

DUIFRFDS : DisplayStatus 変更メソッド DUIFCRDC の最新表示

このオブジェクト独立メソッドは、RODM で定義されている各実リソースおよび集合リソースに関して、 DisplayStatus フィールドを現行の DisplayStatus 値に変更す

るために、任意のアプリケーションによって呼び出すことができます。このメソッドは、DisplayStatus マッピング・テーブル (DUIFSMT) が変更になったときに役立ちます。ネットワークからの状況変更を待って例外ビューの更新を起動する代わりに、メソッド DUIFRFDS を実行して、オブジェクトの例外状態を再計算する状況変更を起こすことができます。これで、適切な例外ビューが更新されます。詳細については、123 ページの『例外ビュー用の DisplayStatus マッピング・テーブルをカスタマイズする』を参照してください。

DUIFVCFT: 例外状態の変更

このオブジェクト独立メソッドは、ユーザー・メソッドがオブジェクトの例外状態を変更するときに呼び出すことができます。ユーザー・メソッドは、 DisplayStatus マッピング・テーブル DUIFSMT の USRXMETH キーワードによって指定されます。サンプル・ユーザー・メソッド DUIFCUXM および DUIFCUX2 は、ResourceTraits フィールド内の値 XCPT または NOXCPT を実 DisplayStatus 変更の処理と同じ方法に設定するために、メソッド DUIFVCFT を実行します。
DUIFVCFT は、次にメソッドを起動して、例外状態の変更によってオープンの例外ビューとの間でオブジェクトの追加もしくは削除が起こるかどうかを判別します。

DUIFVINS: ビュー通知細分性メソッドのインストール

このオブジェクト独立メソッドは、GMFHS がフィールドにビュー通知細分性メソッド、 DUIFVNOT をインストールするときに使用します。このメソッドの説明については、569 ページの『DUIFVINS: ビュー細分性インストール・メソッド (DUIFVNOT)』を参照してください。

使用できない GMFHS メソッド

このセクションで説明されている GMFHS メソッドのほかに、 GMFHS はユーザーのプログラムで使用できない別のメソッドも使用します。使用できない GMFHS メソッドのリストについては、556 ページの『GMFHS メソッド』を参照してください。

GMFHS 自動化の例

このセクションでは、アプリケーションおよびメソッドから構成される自動化の例を紹介します。この目的は、複雑なタスクを自動化する独自のアプリケーションの設定方法を説明することです。この例では、GMFHS_Managed_Real_Objects_Class で定義された DisplayStatus フィールドを使用していますが、この例は、DisplayStatus フィールドを定義したどのオブジェクト・クラスにも適用されます。

自動化アプリケーションは、この例では NetView プロダクトのもとで実行しますが、その独自のアドレス・スペースで実行することもできます。この例は、RODM に接続して、GMFHS_Managed_Real_Objects_Class オブジェクトの DisplayStatus フィールドで値に変更があったときに通知を受けるよう要求しています。この変更は、GMFHS が分析したオブジェクトについてアラートが着信した結果発生します。

この例では、19 ページの『第 2 章 ネットワークを GMFHS に定義する』で説明され、 23 ページの図 7 に図解されているサンプル・ネットワークに入っている 2 つのミニコンピューターのいずれかで状況の変更があった場合に、アプリケーショ

ンが通知を受けるように登録されています。アプリケーションは、これらのリソースのいずれかの状況が不良に変わったと判断すると、 RODM のもとで実行するオブジェクト独立メソッドを実行します。このメソッドは、不良状況のリソースを検出するか、 ParentAccess リンクのないリソースを検出するまで、状況が代わって高水準のリソースになるリソースの ParentAccess フィールドに従います。次にメソッドは、不良状態にある親元リソースを検出したかどうかを稼働中のアプリケーションに通知します。

不良状態にある親元リソースがメソッドによって検出された場合、稼働中のアプリケーションは、アラートは高水準の問題の徴候であると見なし、以後は何も行いません。不良状態にある親元リソースがメソッドによって検出されていない場合、稼働中のアプリケーションは、アラートは新しい問題を表していると見なします。この場合、アプリケーションは NetView ブリッジを経由して新しい問題の問題報告書をオープンするか、問題を回避するしかるべきコマンドを出します。とるべきアクションは、インストール・システムごとに異なるので、コーディングで示されているようにはなりません。

GMFHS 自動化の例の目的は、RODM の自動化の考えられる使い方を図解し、GMFHS インターフェースを使用するコードの書き方を明らかにすることであり、特定の自動化問題の解決策とみるべきではありません。プログラムでは、親子パスでのループの有無を検査しません。プログラムの論理は、高水準のリソースがダウンした場合、低水準リソースのアラートは、その問題の徴候であるか、少なくとも、高水準の問題が解決するまでは手を付けられない問題を表すものという前提に基づいています。この前提は、必ず有効とは限りません。その妥当性は、インストール・システムや関連するネットワーク・リソースによって異なります。この例は、GMFHS オペレーターの作業の自動化と、 GMFHS オペレーターがワークステーションでの構成や状況情報をモニターする際の、推論やアクションを図解しています。

自動化アプリケーションおよびメソッドの例

CNMSNIFF サンプル・アプリケーション・プログラムは、 NetView コマンド行からの、 RODM 名、RODM ユーザー名、および RODM パスワードを受け入れます。次にアプリケーションは、この 3 つのパラメーターを使用して次の機能を実行します。

1. 接続要求を指定された RODM に送ります。
2. DEC ネットワークの DisplayStatus フィールドに申請します。
3. EKGWAIT を出して、DEC ネットワークの DisplayStatus フィールドが変更されるまで待機します。
4. 1 つ以上の DisplayStatus フィールドが変更された場合に、 EKGSNIFF サンプルのオブジェクト独立メソッドを起動します。
5. サンプル・コードは、このステップでの処理は行いません。作業中の自動化アプリケーションを作成していたのであれば、 EKGSNIFF メソッドが処理を終了した後 EKGSNIFF メソッドによって戻される戻りコードおよび理由コードに基づいて、システムに問題の修正もしくは問題レコードの記録を行わせる、適切なコードを作成することができます。
6. EKGWAIT を出して、問題が発生するまで、または RODM が終了するまで待機します。

CNMSNIFF アプリケーションは、C 言語で作成され、 NetView アドレス・スペースで実行します。このアプリケーション例のソース・コードは、 NetView サンプルとして出荷されます。サンプル名は、データ・セット CNMSAMP の CNMS4402 (別名 CNMSNIFF) です。

EKGSNIFF サンプル・オブジェクト独立メソッドは、 CNMSNIFF サンプル自動化アプリケーション・プログラムによって起動されます。 EKGSNIFF メソッドは、ターゲット・オブジェクトのオブジェクト ID をパラメーターとして受け入れます。EKGSNIFF メソッドは、起動されると、ターゲット・オブジェクトとオブジェクトの親の DisplayStatus フィールドを照会します。次にメソッドは、ターゲット・オブジェクトとその親の DisplayStatus フィールドの値に基づいて、トランザクション情報ブロック内の CNMSNIFF 親プログラムに戻りコードと理由コードを戻します。

EKGSNIFF メソッドのソース・コードは、 NetView サンプルとして出荷されます。サンプル名は、データ・セット CNMSAMP の CNMS4403 (別名 EKGSNIFF) です。

第 8 章 RODM 自動化プラットフォームを使用する

この章では、RODM 自動化プラットフォームについて概説します。 *RODM 自動化プラットフォーム* は、RODM を用いた自動化を容易にする一連の NetView サービスです。

RODM 自動化プラットフォームに関する詳細は、「*IBM Tivoli NetView for z/OS 自動操作ガイド*」に記載されています。 本書でも、自動化プラットフォームの使い方を示した広範囲な RODM 自動化シナリオを掲載します。

RODM 自動化プラットフォーム・サービス

RODM 自動化プラットフォームは、以下のサービスから構成されます。

- DSIQTSK タスク
- ORCONV コマンド
- EKGSPPI メソッド
- CNMQAPI サービス・ルーチン
- DSINOR サービス・ルーチン
- ORCNTL コマンド

DSIQTSK タスクは、RODM アドレス・スペースとの連絡専用です。これは、EKGSPPI からコマンド要求を受け取り、コマンドを自動タスクにディスパッチします。 NetView アドレス・スペースから管理する各 RODM は、DSIQTSK に定義しておく必要があります。

NetView 自動化テーブル、コマンド・リスト、およびアプリケーションは、ORCONV コマンドによって、フィールドの値を変更し、メソッドをトリガーする要求を RODM に出すことができます。 ORCONV コマンドの場合、DSIQTSK タスクはコマンドが出される NetView で実行し、かつ RODM は DSIQTSK タスクに定義されていなければなりません。

EKGSPPI NetView 提供のメソッドは、プログラム間インターフェースを用いて、RODM から NetView プロダクトの DSIQTSK タスクにコマンドを送ります。 EKGSPPI メソッドの説明については、553 ページの『EKGSPPI: NetView へのコマンドの送信』を参照してください。

CNMQAPI サービス・ルーチンは拡張 API であり、これを使用すると、より少ないプログラミング作業で、NetView アドレス・スペース内のアプリケーションが RODM の機能を実行できるようにすることができます。 CNMQAPI は、高水準言語 PL/I および C と併用することができます。 CNMQAPI を用いると、アプリケーションは、 RODM がチェックポイント要求を処理する間に要求を出せるようになります。 CNMQAPI は、要求をキューに入れ、チェックポイント処理が完了するとそれを RODM に送ります。 CNMQAPI の構文については、「*IBM Tivoli NetView for z/OS Programming: PL/I and C*」を参照してください。

DSINOR アセンブラー言語マクロは、NetView アドレス・スペースで実行するアセンブラー・アプリケーション用に CNMQAPI のような API を備えています。DSINOR の構文については、「*IBM Tivoli NetView for z/OS Programming: Assembler*」を参照してください。

ORCNTL コマンドは、DSIQTSK タスクに定義された RODM に関する管理詳細を管理します。詳細については、NetView オンライン・ヘルプの ORCNTL コマンドを参照してください。

サンプル自動化コード

NetView プロダクトでは、一部の RODM 自動化プラットフォーム・サービスについて、使用方法を習得する際に使用できるサンプル・コードが提供されています。このサンプル・コードは、以下の NETVIEW.V5R3M0.CNMSAMP サンプル・ライブラリーに入っています。

CNMS4230
このサンプルには、PL/I 言語でプログラミングする際の CNMQAPI サービス・ルーチンの使用法が記載されています。

CNMS4260
このサンプルには、C 言語でプログラミングする際の CNMQAPI サービス・ルーチンの使用法が記載されています。

CNMS4290
このサンプルには、DSINOR アセンブラー言語マクロの使用法が記載されています。

第 4 部 RODM を使用したアプリケーション・プログラミング

第 9 章 RODM 概念を理解する

この章では、RODM データ・キャッシュ、メソッド、およびアプリケーションの構造について説明します。この章は、独自のデータ・モデル、および関連するメソッドとアプリケーションを作成する場合の、 RODM 概念の理解に役立ちます。

この章では、RODM 要約データ・タイプを説明します。 Integer (整数) や MethodSpec のようなデータ・タイプは、RODM に保管されているデータの形式を定義します。

RODM クラス

オブジェクトをグループにまとめる機能、およびオブジェクトのグループをグループ化したり配置したりする機能は、ネットワーク管理に役立ちます。 RODM は、このグループ化の概念をクラス を用いて実現します。クラスは、データ・キャッシュのデータ構造を定義します。

クラスは、グループを表し、そのクラスの下のすべてのクラスおよびオブジェクトのフィールドを定義します。 RODM データ・キャッシュをツリー構造で表示すると、クラスは UniversalClass をクラスの最上位クラスとするツリーの分岐を表します。 225 ページの図 41 は、ツリー構造の例です。

RODM クラスの特性:

- 可能な構造:
  - 子がない
  - クラス子のみ
  - オブジェクト子のみ
  - クラスとオブジェクト子の両方
- クラス子もしくはオブジェクト子の全データ編成を定義する。
- オブジェクトのデータを入れる共用フィールドから構成される。
- 継承されない専用フィールドを組み込む。
- 継承構造を定義する。

クラス名

各 RODM クラスには、 MyName フィールドにクラス名 と呼ばれる文字ストリングがあります。 RODM システム定義のクラス名は、RODM により予約されており、削除することはできません。システム定義の名前は、UniversalClass の場合を除き、すべて `EKG_` で始まります。

EKGCUST の CHARACTER_VALIDATION キーワードは、オブジェクト名 (240 ページの『オブジェクト名』を参照)、フィールド名 (242 ページの『フィールド名』を参照)、およびクラス名に使用される文字に対して、RODM がどの程度の妥当性検査を行うかを指定します。

CHARACTER_VALIDATION(YES) を指定した場合のクラス名の特性

EKGCUST に CHARACTER_VALIDATION(YES) (デフォルト) を指定する場合、以下の特性をもつクラス名が有効です。

- 名前は、PL/I 構文が CHAR(64) VARYING の ShortName データ・タイプに準拠した 1 から 6 文字で構成されます。
- ストリングの先頭文字は、英字もしくは数字でなければなりません。他の文字 (ある場合) は、英字、数字、区切り文字 (_)、アットマーク (@)、番号記号 (#)、あるいはピリオド (.) でもかまいません。
- EKG_ 接頭部は、RODM 作成のクラス用に予約されています。この接頭部を、ユーザーが作成するクラスの名前に使用してはなりません。
- 大小文字の両方の英字が許可され、名前では大小文字が区別されます。
- RODM データ・キャッシュのクラス名は、それぞれに固有です。 RODM は、最大 4,079 のクラスをサポートします。

CHARACTER_VALIDATION(NO) を指定した場合のクラス名の特性

EKGCUST に CHARACTER_VALIDATION(NO) を指定する場合、以下の特性をもつクラス名が有効です。

- 名前は、PL/I 構文が CHAR(64) VARYING の ShortName データ・タイプに準拠した 1 から 6 文字で構成されます。
- 番号記号 (#) はマルチシステム・マネージャー用に予約されているので、先頭文字として使用できません。
- ブランク文字は無効です。
- ヌル文字は無効です。
- EKG_ 接頭部は、RODM 作成のクラス用に予約されています。この接頭部を、ユーザーが作成するクラスの名前に使用してはなりません。
- 大小文字の両方の英字が許可され、名前では大小文字が区別されます。
- RODM データ・キャッシュのクラス名は、それぞれに固有です。 RODM は、最大 4,079 のクラスをサポートします。

システム定義のクラス

RODM がコールド・スタートすると、RODM の初期化が起こり、クラス定義が作成されます。このデータ・モデルは、すべての RODM クラスおよびオブジェクト用に開始点を備えています。これらのシステム定義のクラス によって、ユーザーはそのアプリケーションや RODM そのものに関する情報にアクセスできるようになります。 225 ページの図 41 は、RODM システム定義のクラスおよびその階層を示しています。

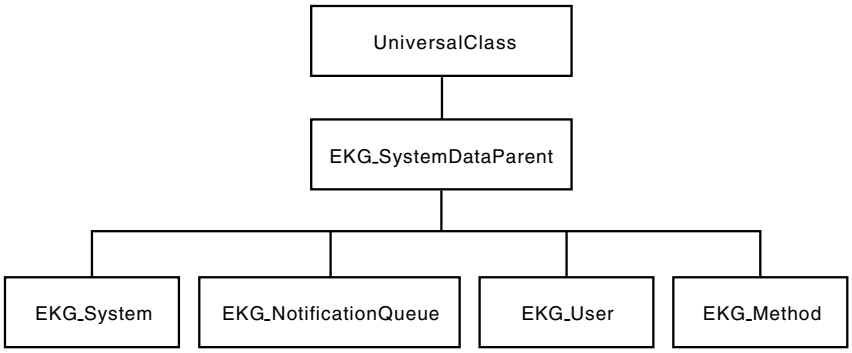


図 41. RODM システム定義のクラス

RODM には、次のシステム定義のクラスがあります。

UniversalClass
RODM データ・キャッシュの継承ツリー構造のルート

EKG_SystemDataParent
システム・データ親クラス、つまりすべての RODM 定義済みシステム・クラスの親クラス

EKG_System
システム・オブジェクト・クラス、つまり RODM の開始時に RODM によって作成されるすべての RODM システム・データ

EKG_User
ユーザー・オブジェクト・クラス、つまりアプリケーションが RODM に接続する際に RODM が作成するフィールドおよびメソッド

EKG_NotificationQueue
通知キュー・オブジェクト・クラス、つまりアプリケーションが通知キューを作成する際に RODM が作成するフィールドおよびメソッド

EKG_Method
メソッド・オブジェクト・クラス、つまりメソッドのインストール時に RODM が作成するフィールドおよびメソッド

以下の 6 つのセクションで、6 つの RODM システム定義のクラスを説明します。6 つのクラスすべてに共通する情報には、以下の内容が含まれています。

- RODM が作成し、アプリケーション・プログラムおよびメソッドがアクセスできるフィールド。
- システム定義のフィールドに RODM が作成するサブフィールド。ユーザー・アプリケーションで、システム定義のクラスのフィールドにサブフィールドを追加することはできません。指定したフィールドには、 EKG_AddNotifySubscription 機能を用いて通知申請を加えることができます。
- notify サブフィールドの指定によって、アプリケーションが通知申請を行うことができるフィールドを識別します。 RODM は、フィールドの値が変更になったときにフィールドに申請されている各アプリケーションに通知します。
- アプリケーションが変更できるのは、書き込みアクセス・フィールドのみです。
- アプリケーションが変更できるのは、オブジェクトのフィールドの値のみです。

UniversalClass

UniversalClass は、RODM 汎用クラス、つまり RODM クラスの階層のルートです。クラスおよびオブジェクトは、すべて汎用クラスの子孫です。 RODM のクラスおよびオブジェクトごとに、UniversalClass のフィールドを継承します。これらのフィールドの内容は継承されず、フィールド定義だけが継承されます。

UniversalClass には親はありません。

表 23 は、UniversalClass のフィールド、各フィールドについてのアクセス、フィールドのデータ・タイプ、およびフィールドごとに定義したサブフィールドの説明です。

表 23. UniversalClass フィールド

フィールド名	アクセス	データ・タイプ	Query	Change	Notify	タイム・スタンプ
MyName	読み取り	ObjectName または ShortName	X			
MyID	読み取り	ObjectID または ClassID	X			
MyPrimaryParentName	読み取り	ShortName	X			
MyPrimaryParentID	読み取り	ClassID	X			
WhatIAm	読み取り	列挙型の Integer	X		X	
MyClassChildren	読み取り	ClassIDList	X		X	
MyObjectChildren	読み取り	ObjectIDList	X		X	

UniversalClass フィールドは次のとおりです。

MyName
オブジェクトまたはクラスの名前。このフィールドのデータ・タイプは、オブジェクトにフィールドが作成される際の ObjectName、およびクラスにフィールドが作成される際の ShortName です。クラス名もしくはオブジェクト名は、クラスもしくはオブジェクトを作成する際に指定します。

MyID
RODM により割り当てられるオブジェクトもしくはクラスの数値 ID。 RODM にクラスまたはオブジェクトを作成する際に、 RODM にクラスまたはオブジェクトの名前を指定します。次に、RODM が、クラスまたはオブジェクトに数値 ID を割り当てます。クラスの参照はそのクラス ID で行い、オブジェクトの参照はそのオブジェクト ID で行う方が、名前による参照より効率的です。

MyPrimaryParentName
このオブジェクトのクラスの名前、もしくはこのクラスの親クラスの名前。

MyPrimaryParentID
このオブジェクトのクラスの ID、もしくはこのクラスの親クラスの ID。

WhatIAm
このフィールドは、オブジェクトまたはクラスのタイプを示します。有効な値は以下のとおりです。

値	意味
1	オブジェクト
2	子のないクラス
3	オブジェクト子のあるクラス
4	クラス子のあるクラス
5	クラス子とオブジェクト子のあるクラス

MyClassChildren
このクラスのクラス子のリスト。これが有効なのは、WhatIAm フィールドの値が 4 または 5 のときです。このフィールドは、クラスにクラス子が無い場合、ヌル値に設定されます。

MyObjectChildren
このクラスのオブジェクト子のリスト。これが有効なのは、WhatIAm フィールドの値が 3 または 5 のときです。このフィールドは、クラスにオブジェクト子が無い場合、ヌル値に設定されます。

EKG_SystemDataParent クラス

EKG_SystemDataParent は、すべての RODM システム・データの親クラスです。

EKG_SystemDataParent クラスにより、 RODM が作成するシステム・データ・クラスおよびオブジェクトに、名前付きの親が与えられます。これにより、システム定義のクラスが、 UniversalClass で定義された他のすべてのクラスから分離されます。

EKG_SystemDataParent の親は UniversalClass です。

SystemDataParent は、そのフィールドのすべてを UniversalClass から継承します。EKG_SystemDataParent のすべてのフィールドは、読み取りアクセス専用です。

EKG_System クラス

EKG_System クラスは、EKG_SystemDataParent クラスの子で、 RODM のすべてのシステム・データが入っています。

コールド・スタート時、RODM は、 EKG_System クラスおよび EKG_System クラスのオブジェクトを 1 つ作成します。このオブジェクトには、この RODM のシステム・データが入ります。

RODM のウォーム・スタート時、RODM は EKG_System フィールドの大部分を更新します。 EKG_TransSegment および EKG_WindowSize フィールドには、最後のチェックポイント時にそこに入っていた値が保存されます。このクラスに追加したユーザー定義フィールドもしくは申請があれば、そこにも最後のチェックポイント時にそこに入っていた値が保存されます。

RODM の開始時に、EKG_System のフィールドのいくつかの初期値が、 RODM カスタマイズ・ファイルから読み取られます。 RODM カスタマイズ・ファイルについては、「*IBM Tivoli NetView for z/OS アドミニストレーション・リファレンス*」を参照してください。

表 24 は、EKG_System クラスのフィールド、フィールドごとのアクセス、データ・タイプ、およびフィールドごとのサブフィールドの説明です。

表 24. EKG_System フィールド

フィールド名	アクセス	データ・タイプ	Query	Change	Notify	タイム・スタンプ
EKG_Name	読み取り	CharVar				X
EKG_APIVersion	読み取り	Integer				
EKG_ReleaseID	読み取り	CharVar				
EKG_ExternalLogState	書き込み	列挙型の Integer			X	X
EKG_LastCheckpointID	読み取り	TransID			X	X
EKG_LastCheckpointResult	読み取り	SelfDefining			X	X
EKG_LastAsyncError	読み取り	AnonymousVar			X	
EKG_AsyncTasks	読み取り	Integer				
EKG_ConcurrentUsers	読み取り	Integer				
EKG_PLI_ISA	読み取り	Integer				
EKG_SSBChain	読み取り	Integer				
EKG_TransSegment	読み取り	Integer				
EKG_WindowSize	読み取り	Integer				

フィールド定義は、次のとおりです。

EKG_Name
RODM 名。このフィールドには、この RODM の名前が入ります。 RODM は、このフィールドの timestamp サブフィールドを、RODM が開始した時刻に設定します。

EKG_APIVersion
API バージョン。このフィールドには、この RODM がサポートする最新の API レベルが入ります。

EKG_ReleaseID
リリース・レベル。サービス用に、RODM は、形式 *product_acronym* バージョン・リリース のバージョンおよびリリースを識別するストリングを生成します。このフィールドの現行値は、RODMN530 です。 値 RODMN530 は、Tivoli NetView for z/OS V5R3 を示します。

EKG_ExternalLogState
外部ログの管理状態 (ログまたはログなし)。このフィールドを変更することで、 RODM ログへのログ記録を動的に制御することができます。有効な値は、以下のとおりです。

値	意味
1	ログ
2	ログなし

このログが適用されるのは、外部ファイル・データ・セットに対してのみです。外部ログがいっぱいのとき、RODM は 2 次ログが割り振られていれば自動的にそちらに切り替えます。割り振られていなければ、RODM は 1 次ログに上書きします。

EKG_LastCheckpointID
最後の正常なチェックポイント操作のトランザクション ID。正常なチェックポイントで更新されたのはこのフィールドだけであるため、ユーザー・アプリケーションは、このフィールドに申請して正常なチェックポイントの通知を求めることができます。アプリケーションは、このフィールドの timestamp サブフィールドを照会して、最後の正常なチェックポイントの時刻を求めることができます。ウォーム・スタート操作の間に、 RODM はウォーム・スタート前からチェックポイント・ファイルに入っている最後のトランザクション ID に、このフィールドを初期化します。

EKG_LastCheckpointResult
表 25 に見られる SelfDefining 値。取り消されたチェックポイントを含む、最後のチェックポイントの試みの状況およびトランザクション ID を示します。

チェックポイントがチェックポイント MODIFY コマンドによって要求されると、 RODM はこのフィールドを現在のトランザクション ID によって更新します。そうでない場合のトランザクション ID は、要求側ユーザー API のトランザクション ID です。

ユーザー・アプリケーションは、EKG_LastCheckpointResult システム・フィールドに申請して、チェックポイント試行の完了通知を求めることができます。アプリケーションは、return_code および reason_code についてフィールドを照会し、試行の成否と、失敗のときはその理由を判別することができます。アプリケーションは、このフィールドの timestamp サブフィールドを照会して、最後のチェックポイントを試みた時刻を求めることもできます。

表 25. EKG_LastCheckpointResult システム・フィールド

オフセット	長さ	タイプ	用途	パラメーター
000	2	Integer	—	SelfDefining の長さ
002	2	Integer	—	データ・タイプ ID
004	4	Integer	出力	Return_code
008	2	Integer	—	データ・タイプ ID

表 25. EKG_LastCheckpointResult システム・フィールド (続き)

オフセット	長さ	タイプ	用途	パラメーター
010	4	Integer	出力	Reason_code
014	2	Integer	—	データ・タイプ ID
016	8	TransID	出力	Transaction_ID

EKG_LastAsyncError
RODM で発生した最後の非同期エラー。アプリケーションは、このフィールドに申請して、 RODM 内で発生する非同期エラーの通知を求めることができます。非同期エラーが発生すると、RODM はエラーについて作成されたログ・レコードのコピーをこのフィールドに入れます。 RODM は、レコードを実際に RODM ログに書き込むことも、書き込まないこともあります。

非同期エラーは、非同期に実行する RODM 機能もしくはメソッド内のエラーです。 EKG_MessageTriggeredAction 機能を用いて実行される機能は、非同期に実行します。メソッドも非同期に実行することができます。

RODM は、EKG_User クラスに EKG_LastAsyncError フィールドも定義します。 EKG_System の EKG_LastAsyncError には、RODM のユーザーの最後のエラーが入ります。 EKG_User の EKG_LastAsyncError には、 EKG_User 下の特定のオブジェクトによって定義された RODM のユーザーの最後のエラーが入ります。

EKG_AsyncTasks
非同期タスクの最大数。このフィールドでは、同時にアクティブとなる非同期タスクの最大数を指定します。

このフィールドは、ウォーム・スタートとコールド・スタート時に RODM カスタマイズ・ファイルの ASYNC_TASKS オペランドから埋め込まれます。

EKG_ConcurrentUsers
並列ユーザーの最大数。このフィールドでは、RODM アドレス・スペース内で同時に実行する活動トランザクションをもてるユーザーの最大数を指定します。

このフィールドは、ウォーム・スタートとコールド・スタート時に RODM カスタマイズ・ファイルの CONCURRENT_USERS オペランドから埋め込まれます。

EKG_PLI_ISA
PL/I 初期ストレージ域。このフィールドでは、PL/I 環境ごとに事前に割り振られている初期ストレージ域のサイズを指定します。

このフィールドは、ウォーム・スタートとコールド・スタート時に RODM カスタマイズ・ファイルの PLI_ISA オペランドから埋め込まれます。

EKG_SSBChain
SSB チェーンのサイズ。このフィールドでは、同時に存在できる同名のシステム状況ブロック (SSB) の数を指定します。これらの項目には、RODM 活動レコードが入ります。

このフィールドは、ウォーム・スタートとコールド・スタート時に RODM カスタマイズ・ファイルの SSB_CHAIN オペランドから埋め込まれます。

EKG_TransSegment
変換セグメントのサイズ。このフィールドでは、RODM 変換セグメントのサイズ (100 万バイト) を指定します。変換セグメントは、内部 RODM 表の記憶に使用します。

このフィールドは、コールド・スタートのときのみに RODM カスタマイズ・ファイルの TRANS_SEGMENT オペランドから埋め込まれます。

EKG_WindowSize
データ・ウィンドウのサイズ。このフィールドでは、RODM データ・ウィンドウのサイズを指定します。データ・ウィンドウは、RODM データの記憶に使用します。

このフィールドは、コールド・スタートのときのみに RODM カスタマイズ・ファイルの WINDOW_SIZE オペランドから埋め込まれます。

EKG_User クラス

EKG_User は、RODM を使用するアプリケーション・プログラムのクラスです。このクラスは、RODM に接続したアプリケーション・プログラムを表すオブジェクトのフィールドを定義します。アプリケーションは、その EKG_User オブジェクトを照会して、アプリケーション自体に関する情報を入手することができます。

EKG_User の親は EKG_SystemDataParent です。

アプリケーションが RODM に接続すると、 RODM はそのアプリケーションを表す EKG_User クラスのオブジェクトを作成します。アプリケーションが RODM から切断すると、RODM はオブジェクトを削除します。アプリケーションに通知キューもしくは申請が定義されている場合、 RODM は EKG_User のオブジェクトを、そのオブジェクトの EKG_StopMode フィールドの値に基づいて削除します。

RODM の開始時に、EKG_User のフィールドのいくつかの初期値が、 RODM カスタマイズ・ファイルから読み取られます。 RODM カスタマイズ・ファイルについては、「*IBM Tivoli NetView for z/OS アドミニストレーション・リファレンス*」を参照してください。

ウォーム・スタート時、RODM はすべての EKG_User オブジェクトの状況を切断に設定します。次に RODM は、通知キューをもたないオブジェクトがあれば削除します。

EKG_User オブジェクトは、EKG_SystemDataParent クラスおよび EKG_User クラスによって UniversalClass のフィールドを継承します。 EKG_User クラスの MyObjectChildren フィールドを照会して、RODM に接続したアプリケーションのリストを獲得します。

232 ページの表 26 は、EKG_User クラスのフィールド、フィールドごとのアクセス、データ・タイプ、およびフィールドごとに定義されたサブフィールドの説明です。

表 26. *EKG_User* フィールド

フィールド名	アクセス	データ・タイプ	Query	Change	Notify	タイム・スタンプ
EKG_Status	読み取り	列挙型の Integer			X	X
EKG_StopMode	書き込み	列挙型の Integer			X	
EKG_LastAsyncError	読み取り	AnonymousVar			X	
EKG_Uses_Q	読み取り	ObjectLinkList				
EKG_RBOverflowAction	書き込み	列挙型の Integer			X	
EKG_LogLevel	書き込み	Integer				
EKG_MLogLevel	書き込み	Integer				
EKG_MTraceType	書き込み	4 バイトの Integer				

フィールド定義は、次のとおりです。

EKG_Status
現行ユーザー・アプリケーションの状況。 RODM は、状況が変わるつど EKG_Status の timestamp サブフィールドを更新します。 timestamp サブフィールドを照会して、RODM への接続時刻を判別します。有効な値は、以下のとおりです。

値	意味
1	接続
2	切断
3	不明

EKG_StopMode
停止モード。このフィールドでは、ユーザー・アプリケーションの切断時に、RODM がユーザー・アプリケーションに行う処理を指定します。デフォルトのアクションは、すべての通知キューおよびすべての申請の除去です。アプリケーション・プログラムは、このフィールドの設定値を変更して、 RODM が通知キューだけを除去するようにも、なにも除去しないようにも指定することができます。有効な値は、以下のとおりです。

値	意味
1	通知キューおよび申請を除去する。
2	通知キュー・エレメントのみを除去する。
3	通知キューも申請も除去しない。

いずれかのアプリケーションが、キューまたは申請 (あるいはその両方) を保存する設定を行って切断した後、アプリケーションが切断する間にイベントによってこのフィールドを変更するものがあった場合は、この新しい設定値がただちに

効力をもちます。しかし、新しい設定がキューまたは申請 (あるいはその両方) の保存である場合は、アプリケーションが再接続して、新しいキューおよび申請を確立するまで、新しい設定は効力をもつことができません。

申請を除去せずにキューを除去すると、 RODM は通知キューに関連するデータのみを除去することになります。 RODM は EKG_NotificationQueue オブジェクトを保存します。アプリケーションもしくは RODM が、指定されたキューのすべてを除去すると、 RODM はそのキューの EKG_NotificationQueue オブジェクトも除去します。

EKG_LastAsyncError
最後の非同期エラー。ユーザーは、このフィールドに申請して、このユーザー ID が開始したトランザクションに関連する非同期エラーの通知を求めることができます。 RODM は、エラーを記録する際に、エラー・レコードを RODM ログに書き込まない場合でも、エラー・レコードのコピーをこのフィールドに書き込みます。次に RODM は、このフィールドに申請されたユーザーに通知します。

RODM は、EKG_System クラスの EKG_LastAsyncError フィールドも定義します。 EKG_System の EKG_LastAsyncError には、RODM のユーザーの最後のエラーが入ります。 EKG_User の EKG_LastAsyncError には、 EKG_User 下の特定のオブジェクトによって定義された RODM のユーザーの最後のエラーが入ります。

EKG_Uses_Q
通知キュー・オブジェクトへのリンクのリスト。このリストには、このユーザーの通知申請によって指定されたキューごとのリンクが入っています。 RODM は、申請要求に応じてこのリスト内にリンクを作成します。リンクは、ユーザー・オブジェクトの EKG_Uses_Q フィールドと EKG_NotificationQueue オブジェクトの EKG_UsedBy フィールド間です。

EKG_RBOverflowAction
応答ブロックのオーバーフロー・アクションの制御。有効な値は、以下のとおりです。

値	意味
1	保管
2	廃棄

アプリケーションがこのフィールドの値を保存に設定すると、 RODM は応答ブロック・オーバーフロー・データを自動的にバッファー内に収集します。アプリケーションでは、他のデータを照会する前に、オーバーフロー・データをバッファーから入手する必要があります。アプリケーションがこのフィールドの値を廃棄に設定すると、 RODM はすべてのオーバーフロー・データを廃棄します。このフィールドの値が保存から廃棄に変更されると、 RODM は、収集された User_appl_ID 関連のオーバーフロー・データをすべて即時に廃棄します。このフィールドのデフォルト値は保存です。

単一のユーザーがマルチタスキングによって並列トランザクションを実行中に、あるスレッドで応答ブロック・オーバーフローが起こり、別のスレッドでこのフィールドが廃棄に変更になると、オーバーフローを起こすトランザクションは、オーバーフローを示す戻りコードを受け取る場合があります。しかし、オーバーフロー・データは廃棄されます。

EKG_LogLevel
ユーザー API 機能のログ・レベル制御。トランザクション処理の完了後、このパラメーターで、ログ・レコードを作成して、このトランザクションを記録するかどうかが判別されます。ログ制御の基本は、トランザクション戻りコードです。トランザクション戻りコードが EKG_LogLevel 以上ならば、 RODM はログ・レコードを作成します。アプリケーションは、このフィールドに新しい値を指定して、クラスのデフォルト値をオーバーライドすることができます。アプリケーションが指定する値が 0 ならば、RODM は、そのアプリケーションについて、ユーザー API 全体のすべてのトランザクションのログ・レコードを作成します。

RODM は、カスタマイズ・ファイルを読み取って、クラス・レベル・フィールドに割り当てるデフォルト値を決めます。カスタマイズ・ファイルに LOG_LEVEL パラメーターが入っていると、そのパラメーターの値でクラスのデフォルト値が決まります。カスタマイズ・ファイルに LOG_LEVEL の値が入っていない場合は、デフォルト値 8 が使用されます。

EKG_MLogLevel
メソッド API 機能呼び出しトレース用のログ・レベルを指定します。 RODM は、メソッド API 機能呼び出しからの戻りコードが EKG_MLogLevel の値以上であると、ログ・レコードを生成します。

このフィールドは、ウォーム・スタートとコールド・スタート時に RODM カスタマイズ・ファイルの MLOG_LEVEL オペランドから埋め込まれます。

EKG_MTraceType
RODM がメソッドの入り口と出口をトレースするかどうかを指定し、 RODM がトレースするメソッドのタイプを指定します。このフィールドは、ウォーム・スタートとコールド・スタート時に RODM カスタマイズ・ファイルの MTRACE_TYPE オペランドから埋め込まれます。

EKG_MTraceType の先頭 3 バイトは、常に X'000000' です。バイトの右側は、7 ビットのフラグとして使用されます。

ビット ビットが設定されたときの意味
1... .... オブジェクト削除メソッドのトレース
.1.. .... オブジェクト独立メソッドのトレース
..1. .... 名前付きメソッドのトレース
...1 .... 通知メソッドのトレース
.... 1... 変更メソッドのトレース
.... .1.. 照会メソッドのトレース
.... ..1. メソッド出口およびストレージのトレース
....1 メソッド入り口およびストレージのトレース

これらの 7 ビットは任意の組み合わせで設定することができます。トレース・メソッドの入り口ビットとトレース・メソッドの出口ビットがともにゼロの場合は、メソッド・トレースが非活動です。ビットがすべてゼロの場合は、すべてのトレースが非活動です。

RODM は、メソッド入り口もしくはメソッド出口のトレースが指定されているときにログ・レコードを生成します。

各メソッド・オブジェクトの EKG_MTraceFlag フィールドと、さらに EKG_MTraceType の対応するメソッド・タイプのビットによって、メソッドがトレース可能かどうかを指定します。 EKG_MTraceType の対応するメソッド・タイプのビットが設定されるか、関連するメソッド・オブジェクトの EKG_MTraceFlag フィールドが 1 ならば、メソッドはトレースされます。

EKG_NotificationQueue クラス

EKG_NotificationQueue は、通知キュー・クラスです。通知キューは、RODM の通知処理に使用されます。通知の詳細については、364 ページの『RODM 通知プロセス』を参照してください。

親は EKG_SystemDataParent です。

アプリケーションもしくはメソッドは、 EKG_NotificationQueue クラスのオブジェクトを作成して通知キューを作成します。 EKG_CreateObject 機能は、通知キュー・オブジェクトを作成し、ユーザー指定のイベント制御ブロック (ECB) をキュー・オブジェクトに割り当てるよう、 RODM に指示します。キューが作成されると、通知メソッドは通知ブロックをキューに入れることができます。アプリケーションおよびメソッドは、EKG_DeleteObject 機能を用いて EKG_NotificationQueue オブジェクトを削除することで、通知キューを削除することができます。 RODM は、キューを作成する際、すべての通知キューの名前を、アクセス・ブロックからの User_appl_ID で自動的に修飾します。特定の User_appl_ID で作成された通知キューは、それぞれ固有でなければなりません。

表 27 は、EKG_NotificationQueue クラスのフィールド、フィールドごとのアクセス、データ・タイプ、およびフィールドごとに定義されたサブフィールドの説明です。

表 27. EKG_NotificationQueue フィールド

フィールド名	アクセス	データ・タイプ	Query	Change	Notify	タイム・スタンプ
EKG_Status	書き込み	列挙型の Smallint			X	X
EKG_ECBAddress	書き込み	ECBAddress				X
EKG_ECBPostedStatus	読み取り	列挙型の Smallint			X	
EKG_UsedBy	読み取り	ObjectLink				
EKG_SubscribedFromClass	読み取り	ClassLinkList				
EKG_SubscribedFromObject	読み取り	ObjectLinkList				
EKG_Maximum_Q_Entries	書き込み	Integer			X	
EKG_MessagesOnQueue	読み取り	Integer				
EKG_SubscribedForDelete	読み取り	ObjectIDList				

フィールド定義は、次のとおりです。

EKG_Status
通知キューの状況。有効な値は、以下のとおりです。

値	意味
0	非アクティブ
1	アクティブ

状況がアクティブならば、RODM は ECB 値に関係なく通知をこのキューに接続します。 ECB が確立されていないときにキューが項目を蓄積すると、アプリケーションが ECB 値を設定するとただちに RODM は ECB に通知します。

状況が非アクティブならば、RODM は ECB が設定済みであっても、通知を接続しません。このフィールドでのデフォルト値は、以下の状態の場合を除きアクティブです。 User_A は User_B に対する通知キューを作成し、 User_B に対するユーザー・オブジェクトはありません。 RODM は、必要なオブジェクトを作成し、 NotificationQueue オブジェクトの EKG_Status を非アクティブに設定し、ユーザー・オブジェクトの EKG_Status を切断に設定します。

EKG_ECBAddress
ECB のアドレス。これは、通知ブロックがこの通知キューに追加されたときに通知されるオプションの ECB のアドレスです。 ECB は、この通知キューを使用するユーザー・アプリケーションのアドレス・スペースで作成されます。

EKG_ECBPostedStatus
通知状況。有効な値は、以下のとおりです。

値	意味
0	偽
1	真

アプリケーションが通知を受けていて、キューが空でなければ、このフィールドは真に設定されます。キューが空ならば、このフィールドは偽に設定されます。

EKG_UsedBy
このフィールドでは、この通知キューを作成するユーザーを指定します。

EKG_SubscribedFromClass
このフィールドは、この通知キューへの申請を行っているクラスのリストです。このフィールドは片方向リンクです。

フィールドのデータ・タイプは ClassLinkList です。各リスト項目は、ClassID と FieldID から構成されます。 FieldID が参照するフィールドには、申請情報が RecipientSpec データ・タイプの形式で入っています。 RecipientSpec データ・タイプには、アプリケーションが通知キュー・オブジェクトを探す際に使用できる 8 バイトの SubscribeID が入っています。これらのデータ・タイプについては、 257 ページの『要約データ・タイプ参照』を参照してください。

EKG_SubscribedFromObject
このフィールドは、この通知キューへの申請を行っているオブジェクトのリストです。このフィールドは片方向リンクです。

フィールドのデータ・タイプは ObjectLinkList です。各リスト項目は、ObjectID と FieldID から構成されます。 FieldID が参照するフィールドには、申請情報が RecipientSpec データ・タイプの形式で入っています。 RecipientSpec データ・タイプには、アプリケーションが通知キュー・オブジェクトを探す際に使用できる 8 バイトの SubscribeID が入っています。これらのデータ・タイプについては、 257 ページの『要約データ・タイプ参照』を参照してください。

EKG_MessagesOnQueue
現在 EKG_NotificationQueue 上にあるメッセージ数。

EKG_Maximum_Q_Entries
EKG_NotificationQueue で使用できる最大項目数。このフィールドは、読み取られない通知に使用する RODM ストレージの量を制限するのに使用することができます。 EKG_NotificationQueue のメッセージ数が EKG_Maximum_Q_Entries の値に達すると、 RODM はそれ以降のメッセージをキューに入れません。 RODM は、メッセージをキューに入れることができないことを説明した理由コード 158 の戻りコード 4 を通知メソッドに出します。

このフィールドのデフォルトの設定値は、-1 です。これは無限を意味します。

EKG_SubscribedForDelete
このフィールドは、この通知キューにオブジェクト削除の申請を行っているオブジェクトのリストです。

フィールドのデータ・タイプは ObjectIDList です。各リスト項目は、ObjectID から構成されます。これらのデータ・タイプについては、 257 ページの『要約データ・タイプ参照』を参照してください。

EKG_Method クラス

EKG_Method は、すべての RODM メソッドのクラスです。

EKG_Method クラスの親は、EKG_SystemDataParent クラスです。

アプリケーション・プログラムが機能要求のメソッドを参照したり、メソッドを起動するには、メソッドが以下のようになっていなければなりません。

- メソッドを表す EKG_Method クラスのオブジェクトがある
- メモリー内にあるか、メソッドのインストール処理によってメモリー内にロードしなければならない

RODM は、メソッドの検出もしくはロードを行えないと、エラーの戻りコードを生成します。メソッドのインストールの詳細については、 410 ページの『メソッドのインストールおよび解放』を参照してください。

メソッド・オブジェクトが作成されると、そのメソッド名が、ユーザー API とメソッド API 機能の両方に実行可能となります。メソッドには、それがオブジェクト特有のメソッドであるか、オブジェクト独立のメソッドであるかによって、データのアクセスに使用できるさまざまな機能があります。オブジェクト特有のメソッドとオブジェクト独立メソッドの両方であるメソッドを作成することができます。

作成する EKG_Method オブジェクトのオブジェクト名は、インストールするメソッドの名前と同じです。 EKG_QueryEntityStructure 機能を用いて EKG_Method クラスを照会することで、インストール済みのすべてのメソッドを識別することができます。

NetView 提供のヌルのメソッド、NullMeth は、ユーザーがオブジェクトを作成してもインストールされません。このメソッドは RODM 内に作成されます。

RODM メソッド・クラスのオブジェクトは、メソッドの最新表示の際にも使用します。最新表示は、最新表示すべきメソッドのメソッド・オブジェクトの EKG_Refresh フィールドによって示されたメソッドを、 EKG_TriggerNamedMethod

機能を用いて呼び出すことで行われます。最新表示により、古いメソッドのコピーはメモリーから削除され、新しいメソッドのコピーがロードされて、以降のすべての参照に使用されます。

EKG_Method のフィールドはすべて、作成することも、削除することもできます。

表 28 は、EKG_Method クラスのフィールド、フィールドごとのアクセス、データ・タイプ、および適用できる操作の説明です。

表 28. EKG_Method フィールド

フィールド名	アクセス	データ・タイプ	Query	Change	Notify	タイム・スタンプ
EKG_InstallerID	読み取り	CharVar				X
EKG_UsageCount	読み取り	Integer				
EKG_Refresh	読み取り	MethodSpec				
EKG_MTraceFlag	書き込み	Integer				X

フィールド定義は、次のとおりです。

EKG_InstallerID
メソッドのインストールに関連するユーザー ID。 timestamp サブフィールドは、メソッドがインストールされた時点を示します。

EKG_UsageCount
notify、change、および query サブフィールドから、および名前付きメソッドに使用される value サブフィールドからこのメソッドを参照する現在の数。使用カウント、EKG_UsageCount は、 EKG_Method クラスのオブジェクトを削除するときはゼロでなければなりません。 EKG_Method クラスのオブジェクトを最新表示するときの EKG_UsageCount の値には、制限はありません。

EKG_Refresh
メソッド・オブジェクトによって表されるメソッドの最新表示の際に呼び出す必要がある、内部 RODM 最新表示メソッドの名前。アプリケーションが EKG_Refresh value サブフィールドを照会すると、 RODM は MethodSpec データの Object_ID フィールドにヌル値を戻します。

最新表示メソッドが EKG_TriggerNamedMethod API を用いて起動されると、RODM はメソッドの新しいコピーをメソッド・ライブラリーからロードします。最新表示メソッドは、EKG_TriggerNamedMethod 機能ブロックの Method_parms フィールドは使用しません。

メソッドは、notify、change、もしくは query サブフィールドで現在参照されていても最新表示することができます。最新表示操作は、メソッドの新しいコピーをロードするまでは、メソッドが実行されない間を待機します。以降のメソッドの実行は、新しいコピーがロードされるまで中断されます。

EKG_MTraceFlag
特定のメソッド・トレース可能フラグ。このフィールドでは、メソッドがトレース可能かどうかを指定します。有効な値は、以下のとおりです。

値	意味
0	トレースの決定を EKG_MTraceType に任せる。
1	確実にトレースする。

初期値は 0 です。

RODM がこのメソッドをトレースするには、 EKG_User クラスの EKG_MTraceType フィールドによってもトレースが可能になっていなければなりません。

EKG_Method クラスのオブジェクトを削除する: メソッド・オブジェクトの削除では、指定したメソッドが、名前付きの、変更、照会、または通知メソッドとして割り当てられているフィールドもしくはサブフィールドがないかが検査されます。なければ、メソッドは、RODM のアクティブのメソッドから除かれ、対応するロード・モジュールはメモリーから解放されます。

メソッドが、オブジェクト特有のメソッドで、1 つまたは複数のフィールドから参照されている場合は、まずその参照がすべて除かれるまでは削除することができません。メソッドを削除する前に、オブジェクト特有のメソッドに対するこれらの参照を除去するときは、以下のように行います。

- データ・タイプが MethodSpec でオブジェクト特有メソッドを参照するフィールドを、 EKG_ChangeField または EKG_ChangeMultipleFields 機能を使用してヌル値 (NullMeth) に変更する。
- データ・タイプが MethodSpec でオブジェクト特有のメソッドを参照するすべてのサブフィールドを、 EKG_ChangeSubfield 機能を用いてヌル値 (NullMeth) に変更する。
- 通知メソッドの通知申請を、EKG_DeleteNotifySubscription 機能を用いて除去する。

RODM オブジェクト

オブジェクトは、RODM におけるデータの基本単位です。オブジェクトは、クラス別に編成され、最大 254 文字の名前によって表されます。オブジェクトは、DASD 装置もしくはプリンターなどの現実のオブジェクトを表すことができます。オブジェクトは、グラフィカル・ディスプレイ上のビュー、オペレーターのアクセス権限、あるいはアプリケーション・プログラムなどの管理オブジェクトを表すこともできます。オブジェクトは、ローカルに定義されたデータを入れることも、クラスからデータを継承することもできます。

ユーザー・アプリケーションおよびオブジェクト独立メソッドは、EKG_CreateObject 機能を用いてオブジェクトを作成することができます。オブジェクトは、RODM ロード機能を用いて作成することもできます。オブジェクトを作成する際に、オブジェクトの名前とオブジェクトが属するクラスを指定します。RODM は、新しいオブジェクトの数値のオブジェクト ID を戻します。オブジェクトは、オブジェクトが属するクラスで定義された共用フィールドを継承します。

オブジェクト名

各 RODM クラスには、MyName フィールドに**オブジェクト名** と呼ばれる文字ストリングがあります。

それぞれクラスが異なれば、2 つのオブジェクトが同じオブジェクト名をもつことができます。各オブジェクトには、そのクラス名とオブジェクト名の、Class_Name.Object_Name の形式での組み合わせでアクセスすることができます。

RODM システム定義のオブジェクト名は、RODM により予約されており、ユーザーが削除することはできません。

オブジェクトの作成時に名前を指定しない場合は、作成するどのオブジェクトにも RODM がオブジェクト名を割り当てます。 RODM は、EKG*ddddddd* の形式で名前を割り当てます。ここで、*ddddddd* の範囲は 0000000 から 9999999 で、EKG0000001 から始まります。この範囲の値は、RODM でしか使用されません。

EKG_Method クラスもしくは EKG_NotificationQueue クラスのオブジェクトを作成する場合のオブジェクト名は、8 文字に制限されます。 EKG_NotificationQueue クラスの場合、ユーザー ID とオブジェクト名を組み合わせて User_appl_ID.object_name 形式の完全修飾の通知キュー名を作成すると、作成される完全修飾の通知キュー名は、区切り用のピリオドを含め 17 文字に制限されます。

EKGCUST の CHARACTER_VALIDATION キーワードは、クラス名 (223 ページの『クラス名』を参照)、フィールド名 (242 ページの『フィールド名』を参照)、およびオブジェクト名に使用される文字に対して、 RODM がどの程度の妥当性検査を行うかを指定します。

CHARACTER_VALIDATION(YES) を指定した場合のオブジェクト名の特性

EKGCUST に CHARACTER_VALIDATION(YES) (デフォルト) を指定する場合、以下の特性をもつオブジェクト名が有効です。

- 名前は 1 から 254 文字で構成され、その要約データ・タイプは、CHAR(254) VARYING の PL/I 構文に準拠した ObjectName です。
- ストリングの先頭文字は、英字もしくは数字でなければなりません。他の文字 (ある場合) は、英字、数字、または特殊文字の #、 @、 .、,、 :、 ;、 ?、 (、)、 '、 "、 -、 _、 &、 +、 %、 *、 =、 <、 >、および / のいずれでもかまいません。
- 大小文字の両方の英字が許可され、名前では大小文字が区別されます。
- EKG_ 接頭部は、RODM 作成のクラスおよびオブジェクトのために予約されています。この接頭部を、ユーザーが作成するクラスやオブジェクトの名前に使用してはなりません。
- EKG*xxxxxxx* (EKG の後に 7 桁が続く) は、 RODM のみでの使用のために予約されています。この形式を、ユーザーが作成するオブジェクトの名前に使用してはなりません。
- クラスのオブジェクトごとに、固有のオブジェクト名がなければなりません。
- RODM は、最大 2097135 個のオブジェクトをサポートします。

CHARACTER_VALIDATION(NO) を指定した場合のオブジェクト名の特性

EKGCUST に CHARACTER_VALIDATION(NO) を指定する場合、以下の特性をもつオブジェクト名が有効です。

- 名前は 1 から 254 文字で構成され、その要約データ・タイプは、CHAR(254) VARYING の PL/I 構文に準拠した ObjectName です。
- 番号記号 (#) はマルチシステム・マネージャー用に予約されているので、先頭文字として使用できません。
- ブランク文字は無効です。
- ヌル文字は無効です。
- 大小文字の両方の英字が許可され、名前では大小文字が区別されます。
- EKG_ 接頭部は、RODM 作成のクラスおよびオブジェクトのために予約されています。この接頭部を、ユーザーが作成するクラスやオブジェクトの名前に使用してはなりません。
- EKG*xxxxxxx* (EKG の後に 7 桁が続く) は、 RODM のみでの使用のために予約されています。この形式を、ユーザーが作成するオブジェクトの名前に使用してはなりません。
- クラスのオブジェクトごとに、固有のオブジェクト名がなければなりません。
- RODM は、最大 2097135 個のオブジェクトをサポートします。

オブジェクト ID

アクセス時間を極力減らす目的で、 RODM は、オブジェクトにアクセスする別の方法をサポートしています。 RODM では任意のクラスのどのオブジェクトにも、オブジェクトの ObjectID のみに基づいてアクセスすることができます。 RODM は、完全修飾の "class name.object name" を ObjectID に変換し、 ObjectID を完全修飾の "class name.object name" に変換する機能を備えています。

オブジェクトは、下記のいずれかの指定を用いて探すことができます。これらの指定は、探索パフォーマンスの高い順にリストされています。

1. ObjectID
2. ClassID プラス ObjectName
3. ClassName プラス ObjectName

RODM フィールド

すべてのクラスは、共用か専用のいずれかのフィールドからなり、その両方を含むことはありません。フィールドにはフィールド名がなければならず、RODM はフィールド ID を割り当てます。 RODM は、最大 4079 個のフィールドをサポートします。

オブジェクト内のフィールドには、 RODM で定義されたオブジェクト間の関係に関する情報を入れることができます。これらの関係は、RODM のクラスおよびオブジェクトを検査して判別することができます。

フィールド名

各 RODM フィールドには、*フィールド名* と呼ばれる文字ストリング名が入ります。 RODM システム定義のフィールド名は、RODM により予約されており、ユーザーが削除することはできません。 RODM システム定義のフィールドのリストについては、 243 ページの『システム定義のフィールド』を参照してください。

EKGCUST の CHARACTER_VALIDATION キーワードは、オブジェクト名 (240 ページの『オブジェクト名』を参照)、クラス名 (223 ページの『クラス名』を参照)、およびフィールド名に使用される文字に対して、 RODM がどの程度の妥当性検査を行うかを指定します。

CHARACTER_VALIDATION(YES) を指定した場合のフィールド名の特性

EKGCUST に CHARACTER_VALIDATION(YES) (デフォルト) を指定する場合、以下の特性をもつフィールド名が有効です。

- 名前は、データ・タイプが `CHAR(64) VARYING` の PL/I 構文に準拠した ShortName の、 1 から 64 文字で構成されます。
- ストリングの先頭文字は、英字もしくは数字でなければなりません。他の文字 (ある場合) は、英字、数字、区切り文字 (_)、アットマーク (@)、番号記号 (#)、あるいはピリオド (.) でもかまいません。
- 大小文字の両方の英文字を使用することができます。アプリケーションで大文字小文字の変換を行うか否かにかかわらず、フィールド名は RODM 下では大小文字が区別されます。

CHARACTER_VALIDATION(NO) を指定した場合のフィールド名の特性

EKGCUST に CHARACTER_VALIDATION(NO) を指定する場合、以下の特性をもつフィールド名が有効です。

- 名前は、データ・タイプが `CHAR(64) VARYING` の PL/I 構文に準拠した ShortName の、 1 から 64 文字で構成されます。
- 番号記号 (#) はマルチシステム・マネージャー用に予約されているので、先頭文字として使用できません。
- ブランク文字は無効です。
- ヌル文字は無効です。
- 大小文字の両方の英文字を使用することができます。アプリケーションで大文字小文字の変換を行うか否かにかかわらず、フィールド名は大小文字が区別されます。

フィールド ID

RODM は、各フィールドに 4 バイトのフィールド ID を割り当てます。フィールド ID は、フィールドの名前の記号表示です。フィールド ID は、割り当てたり、他のフィールド ID と比較したりすることができます。ユーザー API を経てフィールドにアドレッシングするときは、フィールド名に代えてフィールド ID を使用することができます。フィールド ID を用いて API 経由でフィールドにアドレッシングすると、フィールド名を用いる場合より効率がよくなります。 RODM には、

FieldID をフィールド名に変換する EKG_QueryFieldName 機能と、フィールド名を FieldID に変換する EKG_QueryFieldID 機能が付いています。

RODM 生成の内部 ID が存続するのは、それを処理する速度が文字ストリング名の場合より速いためです。アドレッシングするフィールドを決める場合は、文字ストリング名よりこれらの ID の方が常に優先されます。

例えば、フィールド・アクセス情報ブロックで、 Field_ID と Field_name_length パラメーターの両方がヌルでない場合は、 Field_ID が使用され、Field_name_ptr パラメーターは無視されます。 RODM は、提供された Field_ID が提供されたフィールド名と矛盾していないかどうかを検査しません。フィールド・アクセス情報ブロックの形式およびパラメーターについては、 357 ページの表 37 を参照してください。

フィールドが存在するクラスもしくはオブジェクトに関係なく、フィールド ID によってフィールド名は相互に区別されます。 1 つのクラスまたはオブジェクトのフィールドについて得られるフィールド ID は、クラスもしくはオブジェクトに関係なく、同じ名前のどのフィールドにも再利用することができます。フィールド名には、それに関連するクラスもしくはオブジェクトに関する情報は入りません。しかし、クラスおよびオブジェクトには、それらに含まれるフィールドに関する情報が入ります。

システム定義のフィールド

システム定義のフィールドは、RODM により事前定義されるフィールドで、あらゆるクラスおよびオブジェクトに存在しなければなりません。これらのフィールドおよびその値は継承されることはありません。 RODM はフィールドを作成し、その値の設定はオブジェクトまたはそれが属するオブジェクトを作成もしくは変更するときに行います。アプリケーション・プログラムおよびメソッドは、ユーザー API もしくはメソッド API によってこれらのフィールドの内容を変更することはできません。

システム定義のフィールドの名前は、RODM で予約された名前です。クラス内の他のフィールドを、これらの同じ名前を用いて定義することはできません。

システム定義のフィールドのうち、RODM の実行中に変更されるのは、MyClassChildren、MyObjectChildren および WhatIAm フィールドのみです。したがって、これらは、 notify サブフィールドを作成できる唯一のシステム定義フィールドです。

注: クラスまたはオブジェクトの子の削除を検出するために、これらのフィールドに割り当てられた通知メソッドは、削除されたクラスもしくはオブジェクトにアクセスすることはできません。 RODM は、削除処理を完了した後に通知メソッドを実行します。

あらゆる RODM クラスおよびオブジェクトに、以下のシステム定義のフィールドが入っています。

MyPrimaryParentID
1 次階層の親クラスのクラス ID。オブジェクトの場合、このフィールドにはオブジェクトのクラスのクラス ID が入ります。クラス (汎用クラス以

外) の場合、このフィールドには 1 次階層の親クラスのクラス ID が入ります。汎用クラスは、親のない唯一のクラスです。したがってヌルの MyPrimaryParentID フィールドです。

このフィールドのデータ・タイプは、ClassID です。

MyPrimaryParentName
1 次階層の親クラスの名前。

このフィールドのデータ・タイプは、ShortName です。

MyID フィールドが常駐するオブジェクトもしくはクラスの ID。オブジェクトの場合、MyID の内容はオブジェクト ID です。クラスの場合、MyID の内容はクラス ID です。

このフィールドのデータ・タイプは、オブジェクトの場合は ObjectID、クラスの場合は ClassID です。

MyName
現行のオブジェクトまたはクラスの完全名。オブジェクトの場合、このフィールドにはオブジェクト名が入ります。クラスの場合、このフィールドにはクラス名が入ります。

このフィールドのデータ・タイプは、オブジェクトの場合は ObjectName、クラスの場合は ShortName です。

WhatIAm
オブジェクトまたはクラスのタイプ。

このフィールドのデータ・タイプは Integer で、以下の値をもちます。

1 オブジェクト
2 子のないクラス
3 オブジェクト子のあるクラス
4 クラス子のあるクラス
5 クラス子とオブジェクト子の両方があるクラス

あらゆる RODM クラスに、次の追加のシステム定義フィールドが入っています。

MyClassChildren
このクラスのクラス子のクラス ID のリスト。リスト内の各項目は、1 つの子クラスのクラス ID です。

このフィールドのデータ・タイプは、ClassIDList です。

クラスの作成時、このフィールドの値はヌルに設定されています。その後、このクラスを 1 次の親にするときに、作成時に指定されるクラスの作成と削除によって、項目の追加、設定、このリストからの削除が行われます。

MyObjectChildren
このクラスのオブジェクト子のオブジェクト ID のリスト。リスト内の各項目は、1 つの子オブジェクトのオブジェクト ID です。

データ・タイプは ObjectIDList です。

クラスの作成時、このフィールドの値はヌルに設定されています。その後、このクラスを 1 次の親にするときに、作成時に指定されるオブジェクトの作成と削除によって、項目の追加、設定、このリストからの削除が行われます。

オブジェクトには、MyClassChildren および MyObjectChildren フィールドが作成されることはありません。

RODM サブフィールド

257 ページの『要約データ・タイプ参照』で定義されている RODM データ・タイプによって、 RODM がフィールドについて有効と考える値が制限されます。しかし、ネットワーク管理アプリケーションには、フィールドに関して、単なるその値以上の情報が必要です。フィールドには、持続情報と揮発情報の両方を保管するデータ・キャッシュで役立つ、いくつかにまとまったデータもしくは論理が入っていなければなりません。

フィールドの作成時、RODM はフィールドの value サブフィールドを自動的に作成します。フィールドに明示的に定義されるサブフィールドが他になければ、このフィールドへのどの参照も、フィールドの value サブフィールドへの参照と同じです。

プリンター・オブジェクトの *number_of_waiting_print_jobs* フィールドに保存されている主要な値が、印刷を待機するプリンター・ジョブの数であるとします。この値は揮発性で、値が数時間経過しているものである場合、このフィールドの内容はほとんど使用できません。さらに、印刷待ちのジョブ数と値を入手したときの時刻も保存できるとします。これで、このタイム・スタンプを用いて古いデータを無効にし、現行のデータが必要であることを示すことができます。

問題の解決は、タイム・スタンプだけで行うものではありません。アプリケーションが *number_of_waiting_print_jobs* フィールドの内容を要求するときは、タイム・スタンプの内容と現在時刻を比較し、フィールド内のデータの経過日数に基づいて適切なアクションをとる代わりに、若干の論理がなければなりません。 RODM の設計では、フィールドをいくつかのサブフィールドから構成することができます。これらのサブフィールドは、照会に応答する前に、タイム・スタンプの検査のようなことを自動的に行うように設定できるメソッドを参照することができます。

フィールドに表示できるサブフィールドの固定リストがあります。フィールドに保管したデータが入り、したがってフィールドが存在するときは存在しなければならない value サブフィールドを除き、すべてのサブフィールドはオプションです。下記のリストで、各種サブフィールドとその用途を示します。

value サブフィールドおよび prev_val サブフィールドには、対応するフィールドと同じデータ・タイプがあります。他のすべてのサブフィールドは、サブフィールドの種類に基づいて設定されたデータ・タイプを事前に決めています。各サブフィールドのデータ・タイプは、サブフィールドごとの説明とともに、以下のリストで指定します。サブフィールドの作成時、RODM は、サブフィールドのデータ・タイプの要件に基づいて、それにヌル値を割り当てます。

RODM では、以下のサブフィールドが定義されています。

Value (必須)
フィールドに関連する実際のデータ。 value は、Integer、CharVar、または Floating のような RODM 要約データ・タイプに基づいて定義されます。

データ・タイプは、 257 ページの『要約データ・タイプ参照』で定義されているいずれかのデータ・タイプであって、かつフィールドのデータ・タイプと同じでなければなりません。 value サブフィールドは、フィールドの唯一のシステム定義のサブフィールドです。他のサブフィールドは、すべてオプションです。その有無は、ユーザー API によるクラスのフィールドに対するトランザクションによって決まります。

Query
照会メソッドのメソッド指定 (データ・タイプ MethodSpec)。
- このサブフィールドに値がある場合は、フィールドを照会すると照会メソッドが呼び出されます。
- 照会メソッドは、フィールドからの照会データを修正することができます。

query サブフィールドには、フィールドの内容がフィールドの照会に応答して呼び出し元に戻される前に呼び出されるメソッドが入ります。照会メソッドを定義する場合は、その照会メソッドが照会に応答して値を戻さなければなりません。照会メソッドが照会に応答して値を戻さない場合は、RODM が戻します。

query サブフィールドのデータ・タイプは MethodSpec です。 MethodSpec タイプには、呼び出されるメソッドのオブジェクト ID に加えて、メソッドに渡されるパラメーターのリストが組み込まれています。

パラメーターは、ユーザーが、メソッドによって使用されるように設定したオブジェクトのフィールドを示します。これらのフィールドのパラメーターは、メソッドがサブフィールドにインストールされる際に最も頻繁に設定されます。しかし、これらのパラメーターの一部あるいは全部の設定は、照会メソッドを起動する照会トランザクションが要求される直前に、対応するフィールドに値を割り当てることで行うことができます。

Change
変更メソッドのメソッド指定。
- このサブフィールドに値がある場合は、変更フィールドの要求で変更メソッドが呼び出されます。
- 変更メソッドは、それが定義されているフィールドのデータを修正します。

change サブフィールドは、RODM 外のユーザーからか、別のメソッドによる、EKG_ChangeField または EKG_ChangeMultipleFields 機能要求の要求に応じてフィールドの内容を変更するときに呼び出されるメソッドです。フィールドが変更要求を受け取り、かつ change サブフィールドがある場合、変更要求は、フィールドの値を変更しなければなりません。 RODM は change サブフィールドが定義されているフィールドの値の変更は行いません。

change サブフィールドのデータ・タイプは MethodSpec です。サブフィールドには、メソッドの ID と、メソッドのパラメーターが検出されるオブジェクトのフィールドの位置が組み込まれています。

MyName、MyID、MyPrimaryParentID、MyPrimaryParentName、WhatIAm、MyClassChildren、MyObjectChildren などのシステム定義のフィールドには、change サブフィールドは存在することはできません。

Notify
1 つの通知メソッドもしくは通知メソッドのリストのメソッド指定。

- このサブフィールドに値がある場合は、フィールドを変更すると通知メソッドが呼び出されます。 RODM は、フィールドでの変更が完了すると、通知メソッドを呼び出します。
- 通知メソッドは、申請されたユーザーにフィールドへの変更を通知することができます。

notify サブフィールドには、メソッドと関連するパラメーターのリストが入ります。リストの各メソッドは、ユーザーからの変更要求に応じてフィールドの値が変更されるごとに、 1 回ずつ呼び出されます。リストのメソッドの目的は、状態に変更が生じたときに、他のオブジェクトに通知するか、RODM ユーザーに通知することです。リスト内の各項目のデータ・タイプは、SubscriptSpec です。

サブフィールドのデータ・タイプは、SubscriptSpecList です。メソッド名、オブジェクト・フィールドからのメソッドのパラメーター、および通知を受ける側の記述が、項目ごとに組み込まれます。メソッドが呼び出されると、メソッドの論理により、オブジェクトのデータに基づいていずれかに通知するかどうかが決められます。メソッドが元の申請者に通知することもできるし、メソッドを、別のアプリケーションに通知するか、他の RODM オブジェクトにトランザクションを実行依頼するようにプログラミングすることもできます。通知メソッドが、EKG_QueryObjectName 機能以外のトランザクションを他の RODM オブジェクトに実行依頼できるのは、 EKG_MessageTriggeredAction メソッド API 機能を使用する場合に限られます。

Timestamp

フィールドの value サブフィールドが最後に変更された時刻。 RODM はこのサブフィールドを管理します。このサブフィールドは読み取り専用です。サブフィールドのデータ・タイプは、TimeStamp です。

timestamp サブフィールドの作成および削除は、 EKG_CreateSubfield および EKG_DeleteSubfield 機能を用いて行われます。この定義が行われると、新しい値が古い値と同じであった場合など、フィールドに対するトランザクションの変更が正常に行われるたびに、 RODM は timestamp サブフィールドを更新します。 timestamp サブフィールドは、同じフィールドの value サブフィールドと常に関連しています。フィールドではなく、value サブフィールドに対する変更トランザクションでは、 timestamp サブフィールドは更新されません。 EKG_RevertToInherited 機能を出し、かつフィールドにローカル値と対応するタイム・スタンプが入っている場合には、 time-stamp サブフィールドもその継承された値に戻されます。

Prev_val

value サブフィールドの以前の内容のコピー。 RODM はこのサブフィールドを管理します。このサブフィールドは読み取り専用です。このサブフィールドのデータ・タイプは、value サブフィールドのデータ・タイプと同じです。 prev_val サブフィールドは、システム定義のフィールドに作成することはできません。 prev_val フィールドに入れることができる要約データ・タイプについては、248 ページの『サブフィールドのデータ・タイプ』を参照してください。

prev_val サブフィールドの作成および削除は、 EKG_CreateSubfield および EKG_DeleteSubfield 機能を用いて行われます。この定義が行われると、新しい値が古い値と同じであった場合など、フィールドに対するトランザクションの変更が正常に行われるつど、 RODM は prev_val サブフィールドを更新します。

prev_val サブフィールドは、同じフィールドの value サブフィールドと常に関連しています。フィールドではなく、value サブフィールドに対する変更トランザクションでは、 prev_val サブフィールドは更新されません。
EKG_RevertToInherited 機能を出し、かつフィールドにローカル値と対応する prev_val が入っていれば、 prev_val サブフィールドもその継承された値に戻されます。

サブフィールドのデータ・タイプ

サブフィールドごとに、一定の RODM 要約データ・タイプを使用することができます。要約データ・タイプについては、 257 ページの『要約データ・タイプ参照』で定義しています。

サブフィールド

有効な要約データ・タイプ

Value

- AnonymousVar
- BERVar
- CharVar
- FieldID
- Floating
- GraphicVar
- IndexList
- Integer
- MethodSpec
- ObjectLink
- ObjectLinkList
- SelfDefining
- Smallint
- TimeStamp

Query

- MethodSpec

Change

- MethodSpec

Notify

- SubscriptSpecList

Time Stamp

- TimeStamp

Prev_val

- AnonymousVar
- BERVar
- CharVar
- FieldID
- Floating
- GraphicVar
- IndexList
- Integer

- MethodSpec
- SelfDefining
- Smallint
- TimeStamp

複数値フィールドとオブジェクト間リンク

RODM では、複数値フィールドを用いてオブジェクト間に関係を確立することができます。複数値フィールドは、オブジェクト間での、1 対 1、1 対多数、多数対 1、多数対多数の関係の作成をサポートします。

注: このセッションで説明するリンクは、RODM 定義の関連リンクです。これらのリンクは、RODM データ・キャッシュの 2 つのオブジェクト間で定義されますが、 GMFHS 定義のリンク・オブジェクトによって表されるネットワーク・リンクのような、物理リンクと混同しないようにする必要があります。

EKG_LinkNoTrigger および EKG_LinkTrigger 機能を用いると、ユーザー・アプリケーションおよびメソッドは、2 つのオブジェクト間にリンクを作成することができます。 EKG_UnlinkNoTrigger および EKG_UnlinkTrigger 機能を用いると、ユーザー・アプリケーションおよびメソッドは、2 つのオブジェクト間のリンクを削除することができます。 1 つのオブジェクトをリンクするときは、ObjectLink タイプのフィールドを使用します。 1 つまたは複数のオブジェクトをリンクするときは、ObjectLinkList タイプのフィールドを使用します。あるオブジェクトの ObjectLink フィールドは、常に別のオブジェクトの ObjectLink または ObjectLinkList フィールドにリンクします。あるオブジェクトの ObjectLinkList フィールドは、常に別のオブジェクトの ObjectLink または ObjectLinkList フィールドにリンクします。

システム定義のフィールド間のリンクの場合、 RODM は予約済みデータ・タイプ ObjectID および ObjectIDList を使用します。これらの MyObjectChildren フィールドのようなシステム定義フィールドは、 RODM が管理し、ユーザー・アプリケーションもしくはメソッドが直接変更することはできません。

250 ページの図 42 は、データ・タイプ ObjectLink のフィールドを用いた単一値のリンクと、データ・タイプ ObjectLinkList のフィールドを用いた複数値のリンクを示しています。

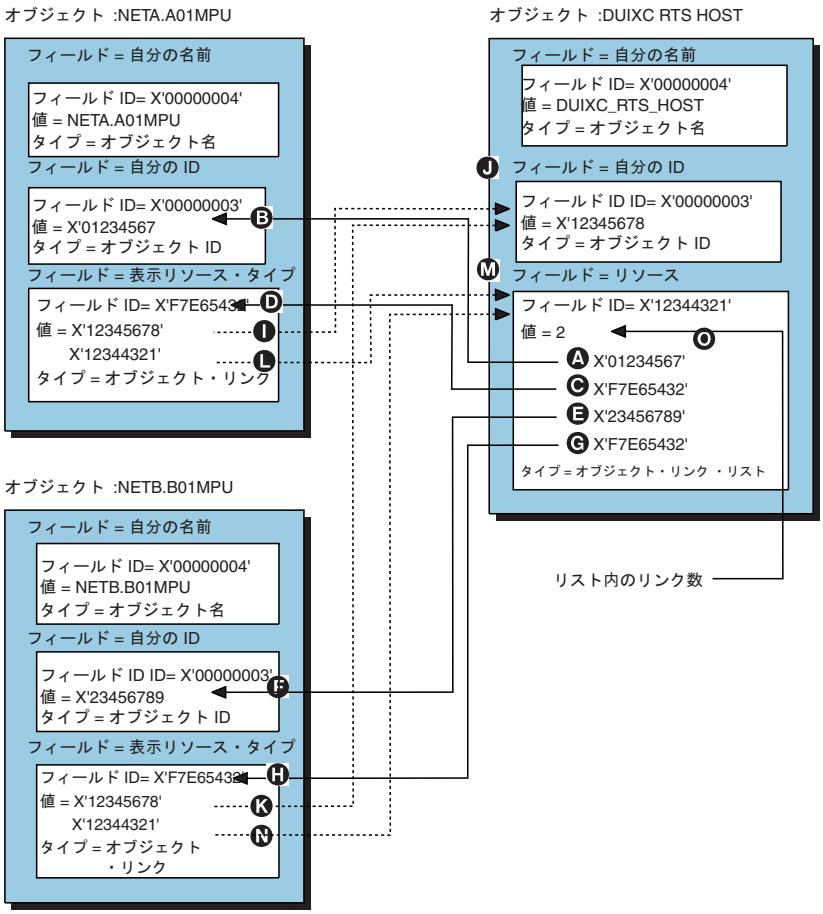


図 42. RODM のオブジェクト間のリンクの例

図 42 には、3 つの RODM オブジェクトが入っています。オブジェクトのうちの 2 つは、ネットワーク内のホスト・プロセッサーを表し、 3 番目のオブジェクトは、オブジェクトのタイプの識別に使用するリソース・タイプのオブジェクトです。 2 つのホスト・オブジェクト、NETA.A01MPU および NETV.B01MPU には、それぞれリソース・タイプ・オブジェクトへの単一値のリンクがあります。リソース・タイプ・オブジェクト DUIXC_RTS_HOST には、 2 つの各ホスト・オブジェクトに対する複数値のリンクがあります。

オブジェクト NETA.A01MPU には、データ・タイプ ObjectLink である DisplayResourceType という名前のフィールドがあります。 DisplayResourceType フィールドには、 (**J**) にリンクされるオブジェクトの ObjectID (**I**)、および (**M**) にリンクされるフィールドの FieldID (**L**) が入っています。

オブジェクト NETB.B01MPU にも、オブジェクト DUIXC_RTS_HOST のフィールド Resources にリンクした DisplayResourceType という名前のフィールドがあります。 DisplayResourceType には、 DUIXC_RTS_HOST (**J**) の ObjectID (**K**) と Resources (**M**) の FieldID (**N**) が入っています。

オブジェクト DUIXC_RTS_HOST には、両方のホスト・オブジェクトにリンクするフィールド Resources があります。 ObjectLinkList フィールド Resources には、それがリンクするオブジェクトの数が入っています (**O**)。 Resources の最初のリスト・エレメントには、オブジェクト NETA.A01MPU (**B**) の ObjectID (**A**) およ

びフィールド DisplayResourceType (D) の FieldID (C) が入っています。Resources の 2 番目のリスト・エレメントには、オブジェクト NETB.B01MPU (F) の ObjectID (E) およびフィールド DisplayResourceType (H) の FieldID (G) が入っています。

EKG_LinkNoTrigger または EKG_LinkTrigger 機能を用いてリンクを作成するときは、このオブジェクトの対とリンクされるフィールドを指定し、 RODM が両方のオブジェクトに ObjectID および FieldID 値を埋め込みます。両方のオブジェクトは、リンクする前に RODM に存在していなければなりません。

リンクおよびリンク解除アクション機能

リンクおよびリンク解除アクション機能は、メソッド API およびユーザー API を経由してユーザーが呼び出すことができます。 EKG_LinkNoTrigger 機能および EKG_LinkTrigger 機能は、2 つのオブジェクトの 2 つのフィールド間にリンクを確立するときに使用します。 EKG_UnlinkNoTrigger 機能と EKG_UnlinkTrigger 機能は、2 つのオブジェクト間のリンクを削除します。これらの機能ごとに、Entity_access_info_ptr および Field_access_info_ptr パラメーターによって指定した 2 つのオブジェクトと 2 つのフィールドが必要です。フィールドは、データ・タイプ ObjectLinkList か ObjectLink のフィールドでなければなりません。機能ブロック形式と他の詳細については、 460 ページの『EKG_LinkNoTrigger、EKG_LinkTrigger − 2 つのオブジェクトをリンクする』および 503 ページの『EKG_UnlinkNoTrigger、EKG_UnlinkTrigger − 2 つのオブジェクトをリンク解除する』を参照してください。

リストであるフィールドもしくはタイプ ObjectLink のフィールドは、リンクおよびリンク解除アクションによってのみ変更されます。これらのアクションには、常にリンクの両端に 1 つずつ、2 つのフィールドが関係します。これらのフィールドには、変更メソッドを定義することができます。これらの変更メソッドは、EKG_LinkTrigger または EKG_UnlinkTrigger 機能によって起動されます。変更メソッドは、EKG_SetReturnCode で戻りコード設定し、リンクもしくはリンク解除を続けられるかどうかを示します。

- 戻りコードが非ゼロの場合は、リンクもリンク解除も行われません。
- いずれかの (あるいは両方の) フィールドに変更メソッドが存在しない場合、RODM は戻りコードがゼロと見なし、リンクもしくはリンク解除は続行します。
- 変更メソッドは存在するが、戻りコードを明示的に設定しなかった場合、 RODM は戻りコードがゼロと見なし、リンクもしくはリンク解除は続行します。

変更メソッドは、フィールドが機能ブロックで表示される順序で起動されます。

対照的に、RODM プログラムは、リンクもしくはリンク解除アクションが要求され、リンクの両端にサブフィールドが存在するときに、リンクの両端で該当する通知メソッドを呼び出します。 2 つのメソッドが呼び出される場合は、最初に呼び出されたメソッドが、希望するアクションを指定した機能ブロックで最上位に指定されるフィールドです。通知メソッドの場合は、最初のリストが処理されてから、他のリストが処理されます。戻りコードが非ゼロであるためにリンクもしくはリンク解除が行われなければ、通知メソッドは起動されません。

リンクおよびリンク解除アクション機能を適用できるのは、 2 つのオブジェクトを一つにリンクする場合に限られます。リンク・アクション機能を用いて、クラスを別のクラスもしくはオブジェクトにリンクすることはできません。オブジェクトは、そのクラスからタイプ ObjectLink のフィールドの存在を継承しますが、オブジェクトがこれらのフィールドについてのそのクラスから継承するのは、ヌル値のみです。同様に、クラスの階層では、タイプ ObjectLink のフィールドの存在は、子クラスによって継承されますが、この種のフィールドはすべて値がヌルです。

リンクされるフィールドのタイプが ObjectLinkList ならば、リンク・アクションによって、リスト内に新たな項目が作られ、その項目は他のオブジェクト - フィールドの対の ObjectID および FieldID が入るように設定されます。データ・タイプ ObjectLinkList のフィールド用に作成されたリンクは、 FIFO または LIFO のような特定のアルゴリズムに従ってフィールド内で配列されるとは限りません。タイプが単純な ObjectLink ならば、そのフィールドの値は、他のオブジェクト - フィールドの対の ObjectID および FieldID が入るように設定されます。リンクは、各オブジェクト・フィールドの対に適用されるため、 2 つのオブジェクト間に両方向リンクを確立します。リンク解除は、このようなリンクを取り外します。リンクおよびリンク解除アクションは、タイプ ObjectLink のフィールドを変更する RODM ユーザーが使用できる唯一のアクションです。

フィールドが単一の ObjectLink ならば、そのフィールドへの照会でタイプ ObjectLink の応答 (8 バイトの ObjectID とその直後に続く 4 バイトの FieldID、計 12 バイト) が出されます。フィールドが ObjectLinkList ならば、ユーザー API かメソッド API によるフィールドの照会で、一列の ObjectLink 項目がユーザーに戻されます。つまり、配列内の各エレメントは、ObjectID および FieldID の 12 バイトの対です。 RODM ユーザーは、ObjectLinkList の項目を個別に照会することはできません。

同じ原則が、MyObjectChildren フィールドの照会に適用されます。この種のフィールドを照会すると、配列内の各エレメントが、 MyObjectChildren フィールドのデータ・タイプ ObjectID のエレメントになるような配列が生じます。配列の長さは、照会されたフィールドのリストの長さと同じです。

リンク・アクション機能で確立されたオブジェクト間のリンクは、両方の対等関係を表すときと、2 次親子関係を表すときに使用されます。 1 次親子関係は必須で、オブジェクトおよびクラスのシステム定義のフィールド MyClassChildren、および MyObjectChildren に組み込まれています。

フィールドに関連するサブフィールド

データ・タイプが ObjectLink または ObjectLinkList のフィールドには、 query サブフィールドを作成することはできません。データ・タイプが ObjectLink または ObjectLinkList でないフィールドの場合、 value サブフィールドは、単一のフィールド項目であり、メソッドを起動しないで、照会し、操作することができます。データ・タイプが ObjectLink または ObjectLinkList であるフィールドの場合、 value サブフィールドは項目の完全なリストから構成され、照会メソッドを起動せずに照会することしかできません。

変更トランザクションは、データ・タイプ ObjectLink または ObjectLinkList のフィールドに適用することはできません。同様に、変更トランザクションは、データ・

タイプが ObjectLink または ObjectLinkList であるフィールドの value サブフィールドに適用することはできません。タイプ ObjectLinkList のフィールドの値の変更の場合に存在するのは、リンクおよびリンク解除機能のみであり、MyObjectChildren フィールドが変更されるのは、子の作成と削除の場合に限られます。

通知メソッドを起動せずにリンクおよびリンク解除アクション機能を実行するため、 RODM プログラムは、EKG_LinkNoTrigger 機能と EKG_UnlinkNoTrigger 機能をサポートしています。

タイプ ObjectLink のフィールドに指定できるサブフィールドは、query、notify、および timestamp サブフィールドです。タイプ ObjectLink および ObjectLinkList のフィールドには、 change サブフィールドを指定することができます。しかし、RODM プログラムがサポートするのは、リスト全体にサブフィールド 1 つのみで、リストの項目ごとの別々のサブフィールドはサポートされません。リストのどれかの項目に変更が加えられると、リスト全体への変更と見なされます。したがって、通知リストがある場合は、リンク (フィールド) のリストのどれかの項目に変更が加えられると、通知リストのメソッドのすべてが呼び出されることになります。

子オブジェクトがデータ・タイプ ObjectLink または ObjectLinkList のフィールドの存在を継承すると、子オブジェクトもそのフィールドを、データ・タイプ ObjectLink または ObjectLinkList フィールドと見なします。しかし、RODM プログラムは、データ・タイプ ObjectLink または ObjectLinkList のフィールドの値の継承はサポートしません。データ・タイプ ObjectLink または ObjectLinkList のフィールドの項目は、データ・タイプ ObjectLink または ObjectLinkList の他のフィールドの項目とは関係ありません。これらは、EKG_LinkNoTrigger 機能もしくは EKG_CreateObject 機能によって一度に 1 つずつ作成され、 EKG_UnlinkNoTrigger 機能または EKG_DeleteObject 機能によって一度に 1 つずつ削除されます。

索引付きフィールド

EKG_Locate 機能は、指定のフィールドに指定した値をもつオブジェクトのオブジェクト ID のリストを検索します。この機能により、アプリケーションは、オブジェクト ID のリストを容易に検索できるようになります。アプリケーションは、照会フィールド機能 (指定のフィールドと値を探す) を用いてユーザーの全データ・モデルを走査するのではなく、希望するフィールドとフィールド値で EKG_Locate 機能を呼び出します。

EKG_Locate 機能によりフィールドを探し出す場合、そのフィールドは public_indexed フィールドとして作成されていなければなりません。 public_indexed フィールドの場合、RODM は、オブジェクト ID の表をフィールド名およびフィールド値別に維持します。これらの表の維持には追加処理が必要であるため、ユーザーは EKG_Locate 機能を利用するフィールドに限って、public_indexed フィールドを作成する必要があります。この例としては、クラスとしての Employees、そのクラスの下のオブジェクトとしての各従業員名、および索引付きフィールドとしての EmployeePhoneNumber というデータ・モデルがあります。この例で、アプリケーションは、データ・モデルのすべてのオブジェクトを照会を実行しなくても、フィールド EmployeePhoneNumber に指定の電話番号が入ったオブジェクトをすべて見つけることができます。

索引付きフィールドのデータ・タイプは、CharVar または IndexList です。 IndexList フィールドは、ObjectID テーブル項目を複数 (リストの各値につき 1 つ) 生成します。 CharVar および IndexList において、EKG_Locate は、Indexed_data_ptr が指し示す 1 つの文字ストリング (最大長 254 バイト) を比較用に受け入れます。

public_indexed フィールドの定義に関するパフォーマンス関連の情報については、547 ページの『索引付きフィールド』を参照してください。

RODM におけるオブジェクトおよびクラスのロック

RODM は、ロックを自動的に制御するようになりました。したがって、以下の機能は不要になりましたが、既存のアプリケーションとの互換性のために使用可能になっています。

- EKG_LockObjectList 機能
- EKG_UnlockAll 機能

上記の機能が必要な既存のアプリケーションを変更する必要はありません。

アプリケーション・プログラム・インターフェースを使用する

このセッションでは、2 つの RODM アプリケーション・プログラム・インターフェースの要旨を説明します。

ユーザー・アプリケーション・プログラム・インターフェース (API)

RODM ユーザー・アプリケーションは、タスクを実行するユーザー API によって RODM データにアクセスする外部プログラムです。この RODM ユーザー・アプリケーションは、 RODM のパラメーター受け渡し規則を満たすことができれば、どの言語によってもコーディングすることができます。ただし RODM は、PL/I および C 言語の場合にしか制御ブロック構造を提供しません。

255 ページの図 43は、ユーザー・アプリケーションが、ユーザー API モジュール EKGUAPI を用いて z/OS 環境の RODM データにアクセスする方法の図解です。完全な RODM アプリケーションのコーディングのステップについては、343 ページの『第 11 章 RODM を使用するアプリケーションを作成する』で説明します。

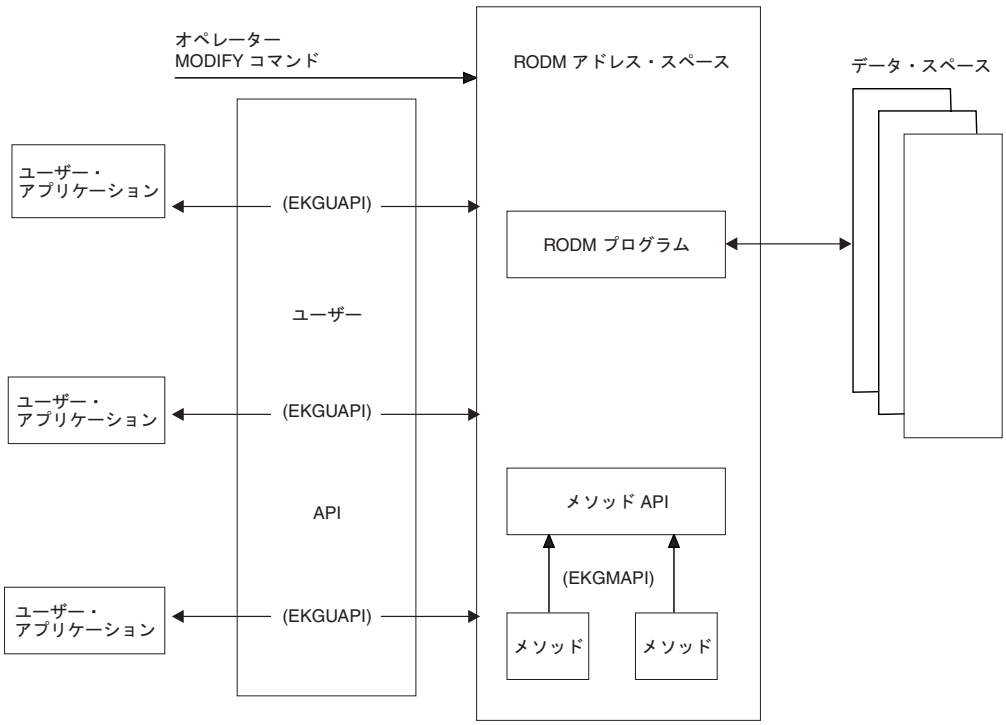


図 43. RODM のシステム構造 (z/OS)

メソッド・アプリケーション・プログラム・インターフェース (API)

メソッド とは、RODM アドレス・スペースに常駐する小さな実行可能プログラムです。メソッドは、ユーザー・アプリケーション、RODM のフィールドへの変更、および他のメソッドによって呼び出すことができ、また RODM の初期化のときに呼び出すことができます。

NetView プログラムは、いくつかの汎用メソッドを提供しており、それぞれの必要に合う場合もありますが、合わない場合は、PL/I または C を使用して独自のメソッドを作成することができます。

図 43 は、メソッドが、メソッド API モジュール EKGMAPI を用いて z/OS 環境の RODM データにアクセスする方法の図解です。 RODM メソッドのコーディングに関連するステップと情報については、 389 ページの『第 13 章 RODM メソッドの作成』で説明します。

RODM 要約データ・タイプ

このセッションでは、RODM データ・タイプの使用方法を説明します。フィールド、サブフィールド、ユーザー API またはメソッド API のフィールドでのデータのタイプ、もしくはメソッドに渡されるパラメーターなど、さまざまなコンテキストで、さまざまなデータ・タイプを使用することができます。

RODM データ・タイプのいくつかは、複合データ・タイプです。これらはプログラム言語の構造に対応しています。これらの複合データ・タイプには、PL/I マクロ宣言および C typedef ステートメントが用意されています。これらの宣言をストレージにマップするときは、コンパイラー生成の埋め込みがないことを確認します。これは、各宣言に UNALIGNED 属性を加えると PL/I で行うことができ、 _Packed 修飾子を用いると C で行うことができます。

データ・タイプのヌル値

RODM プログラムは、データ・タイプごとにヌル値を指定します。以下は、代表的なヌル値の用途です。

- ロケーター・タイプ

  ロケーター・タイプは、他のデータを見つけたり、または指し示すデータです。ヌル値とは、そのデータが*何も指していない* ことを意味します。

- 非ロケーター情報が入っているタイプ

  番号、カウント、あるいはフラグなどの非ロケーター情報が入っているタイプの場合、ヌル値は常に、*ここに情報がない* こと、または*まだ値に設定されていない* ことを意味しています。

RODM プログラムは、フィールドまたはサブフィールドを初めてクラスに作成する際は常に、フィールドまたはサブフィールドの値をフィールドまたはサブフィールドのタイプのヌル値に設定します。クラスまたはオブジェクトがその親クラスからフィールドを継承すると、フィールドの値は親クラスの値に設定されます。

データ・タイプごとのヌル値の指定については、 257 ページの『要約データ・タイプ参照』を参照してください。

データ・タイプ ID

ユーザー・アプリケーションが RODM プログラムにデータを渡す際、RODM プログラムは通常、データとともにデータのデータ・タイプも渡すように要求します。RODM プログラムがアプリケーションにデータを渡すときは、 RODM プログラムは通常、データとともにデータのデータ・タイプを組み込みます。データ・タイプを識別しやすくするため、RODM データ・タイプごとに 10 進数のデータ・タイプ ID があります。

特定データ・タイプのデータ・タイプ ID を見つけるには、 257 ページの『要約データ・タイプ参照』を参照してください。

フィールドのデータのタイプ

アプリケーション・プログラムおよびメソッドでは、 API 呼び出しを出してフィールドを作成するときは、クラスのフィールドごとにデータ・タイプを割り当てなければなりません。 API がフィールドを作成した後、そのフィールドが生きている間はデータ・タイプを変更することができません。

リスト要約データ・タイプは、単一の値ではなく情報のリストを入れるフィールドに指定されます。リスト・データ・タイプは、タイプ IndexList、ObjectLink、

ObjectID、および ClassID のリストの作成に使用できます。このフィールド・タイプを使用すると、クラスおよびオブジェクトの複数対 1 の関係および複数対複数の関係を指定できます。

フィールドに指定できるデータ・タイプによっては、フィールドの性質上制限を受けるものもあります。 RODM プログラムでは、オブジェクトまたはクラスの削除後、 RODM に誤った ID が残らないようにするために、考えられるオブジェクトおよびクラスの関係を制限します。例えば、以下の関係は概念的にはあり得ますが、RODM により禁止されています。

- 1 次階層の親子関係以外のオブジェクトとクラス間の関係。クラス関係は、継承関係でなければなりません。
- EKG_LinkNoTrigger および EKG_LinkTrigger 機能を用いて ObjectLinks によって表される関係以外の、2 つのオブジェクト間の関係。

要約データ・タイプ参照

このセッションでは、RODM プログラムによって定義される要約データ・タイプを説明します。ユーザー・アプリケーションおよびメソッドに、 PL/I 用の EKG1IADT または C 用の EKG3CADT のマクロを組み込みます。このマクロを組み込むと、プログラム内の変数を RODM 機能の使用に必要なデータ・タイプとして宣言することができるようになります。

例えば、RODM 機能ブロックにメソッドの名前を指定する必要がある場合は、受け渡すパラメーターを MethodName 要約データ・タイプとして宣言しなければなりません。 PL/I で ThisMethodName という名前の変数を宣言するには、以下のステートメントを使用します。

```
%include EKGLIB(ekg1iadt);            /* Abstract data declaration  */
DCL ThisMethodName      MethodName;   /* 8-byte char                */
```

C で同じ変数を宣言するには、以下のステートメントを使用します。

```
#include "ekg3cadt.h"                 /* Abstract data declaration  */
MethodName       ThisMethodName;      /* 8-byte char                */
```

タイプごとに変数を宣言する例については、 PL/I の場合は EKG5VDCL、C の場合は EKG6VDCL に記載されています。

以降のデータ・タイプ定義では、データ・タイプによっては予約済みとして指定されているものもあります。これらのデータ・タイプは、フィールド定義を作成する際に指定することはできません。これらのデータ・タイプは RODM プログラムが作成するフィールド用に予約されています。

Anonymous(N) (予約済み)

データ・タイプ ID: 29

説明: データの作成者のみがデータ内容の値を認識しているデータ・バイトの可変長順序。ストリングの最大長は、254 バイトです。実際の長さは暗黙で、このタイプの変数を使用するために定義した場所によって決まります。変数内容の形式は、ユーザー API レベルでは不明です。このタイプを理解しているのは、RODM を使

用し、かつ値を設定するアプリケーション・プログラムもしくはメソッドのみです。この要約データ・タイプは、SelfDefining データ・ストリングでは使用できません。

ヌル値: 不明

PL/I 宣言:

```
% Anonymous = 'CHAR';
```

C 宣言:

```
typedef char Anonymous;
```

AnonymousVar

データ・タイプ ID: 30

説明: 最大 32767 バイトから構成されるデータの可変長ストリング。 2 バイトの長さフィールドと、それに続く長さフィールドで指定されたデータ・バイト数から構成される。このデータ・ストリングは、任意の値の 2 進のデータ・バイトでもかまいません。

変数内容の形式は、ユーザー API レベルでは不明です。この形式を理解できるのは、値を設定するアプリケーション・プログラムもしくはメソッドのみです。

ヌル値: 長さフィールドはゼロです。

PL/I 宣言:

```
% AnonymousVar = 'CHAR(32767) VARYING';
```

C 宣言:

```
typedef    _Packed struct {
             Smallint  Length;
             Anonymous Text[1];
           } AnonymousVar;
```

ApplicationID (予約済み)

データ・タイプ ID: 3

説明: ユーザー・アプリケーション名が入る 8 バイト・トークン。このアプリケーション ID の検査は、それぞれのシステム許可機能によって行われます。文字は 8 バイト内で左寄せされ、右側はブランクで埋め込まれます。このブランクはホスト・システムのコード・ページで定義されます。 S/370 の場合に想定されるコード・ページは、コード・ページ 00500 で、この場合のブランクは X'40' です。

ヌル値: すべてのバイトがブランク (コード・ページ 00500 の場合は、X'40')。

PL/I 宣言:

```
% ApplicationID = 'CHAR(8)';
```

C 宣言:

```
typedef    _Packed struct {
             char     Data_char[8];
           } ApplicationID;
```

BERVar

データ・タイプ ID: 31

説明: BERVar データ・タイプは、BER データを RODM ロード機能に指定します。 RODM は、BER データ形式の一部を検査しますが、そのどれも解釈しません。 RODM によって検査される情報については、以下の説明で明らかにします。

BER データ・タイプの最大長 (ID、長さのバイトおよび内容のバイトを含む) は、32767 を超えてはなりません。 図 44 は、BER データの形式です。

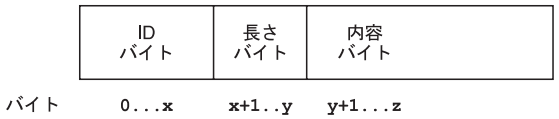


図 44. BER データの形式

RODM は、以下の BER データを検査します。

- **ID バイト。** ID バイトは、長短 2 つの形式をとることができます。形式は、先頭バイトのタグ番号 (ビット 5 から 1) によって決まります。
  - タグ番号が 30 ('11110'b) 以下ならば、ID バイトは短形式で、単一の ID バイトのみが必要です。

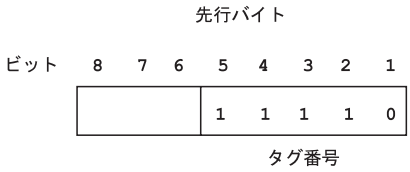


図 45. 短形式の ID バイト

  - 先頭バイトのタグ番号が 31 ('11111'b) ならば、ID バイトは長形式です。長形式の場合は、複数の ID バイトが存在します。先行バイトに続く各バイトのビット 8 は、最後の ID バイトまで 1 に設定されます。最後の ID バイトのビット 8 は 0 (ゼロ) に設定されます。

    260 ページの図 46 は、ID バイトが 3 つの長形式です。

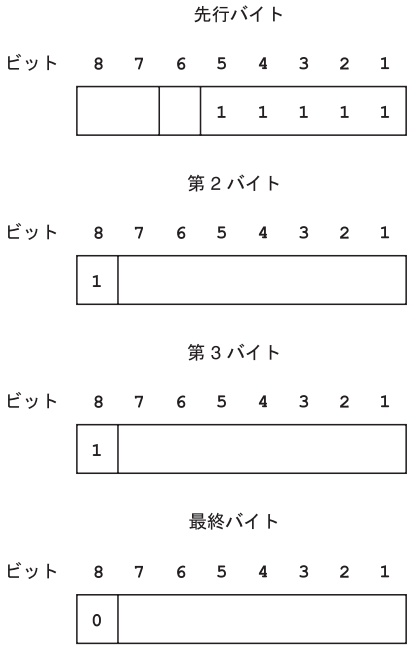


図 46. 長形式の ID バイト

- **長さバイト。** 長さバイトは、内容バイトの長さを指定し、長あるいは短の 2 形式をとることができます。
  - ビット 8 が 0 ならば、長さバイトは短です。この形式のビット 7 から 1 は、無符号の 2 進整数による内容バイトの長さを表します。内容バイトは、短形式の 127 バイト以下でなければなりません。

    図 47 は、値が 86 バイトの短形式の長さバイトです。

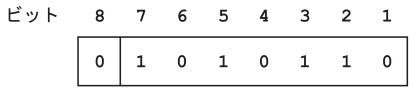


図 47. 短形式の長さバイト

  - ビット 8 が 1 ならば、長さバイトは長です。この形式の場合、ビット 7 から 1 は、長さバイトを構成する後続のバイト数を表し、無符号の 2 進整数です。各後続のバイトは無符号の 2 進整数で、合計すると内容バイトの長さを表します。内容バイトが 127 バイトを超える場合は、長形式を使用しなければなりません。

    261 ページの図 48 は、値が 357 バイトの長形式の長さバイトです。 357 を表すには、2 つの長さバイトが必要です。

図 48. 長形式の長さバイト

ヌル値: 長さフィールドはゼロです。

PL/I 宣言:

```
% BERVar = 'CHAR(32767) VARYING';
```

C 宣言:

```
typedef    _Packed struct {
                Smallint  Length;
                Anonymous Text[1];
           } BERVar;
```

CharVar

データ・タイプ ID: 4

説明: 最大 32767 バイトの可変長文字ストリング。このデータ・タイプの構造は、 2 バイトの長さフィールドと、それに続くストリングの文字です。 CharVar データは、C ストリング・サポートの場合、ユーザーが値を X'00' のヌル・バイトにして任意に終了することができます。 RODM は、文字ストリングを形式化するときは必ずヌル終止符を加えます。例えば、ヌル・バイトで指定された CharVar フィールドにストリング ″RODM″ が入った場合の値は、 X'0004D9D6C4D400' です。ヌル終止符バイトは、CharVar データの長さフィールドには含まれていないので、注意します。

SelfDefining データ・ストリングでの CharVar ストリングの指定については、 269 ページの『SelfDefining』を参照してください。

DBCS (2 バイト文字セット) サポートの場合、DBCS ストリングは、特殊制御文字シフトアウト (X'0E') で始め、制御文字シフトイン (X'0F') で終わらせることができます。各 2 バイト文字は、シフトアウト制御文字とシフトイン制御文字の間に埋め込まれた場合は、 2 バイトとしてカウントされます。さらに、シフトアウトおよびシフトイン文字は、DBCS ストリングの長さに入れられます。有効な 2 バイト文字は、GraphicVar データ・タイプの 2 バイト文字と同じです。 264 ページの『GraphicVar』を参照してください。

ヌル値: 長さフィールドはゼロです。

PL/I 宣言:

```
% CharVar = 'CHAR(32767) VARYING';
```

C 宣言:

```
typedef   _Packed struct {
            Smallint Length;
            char     Text[1];
          } CharVar;
```

CharVarAddr (予約済み)

データ・タイプ ID: 7

説明: 任意の可変長文字ストリングを指すポインター。ポインターは、最大長の要件を暗黙指定するものではありません。

ヌル値: ヌル・ポインター。

PL/I 宣言:

```
% CharVarAddr = 'POINTER';
```

C 宣言:

```
typedef  Pointer  CharVarAddr;
```

ClassID (予約済み)

データ・タイプ ID: 1

説明: RODM にクラスを識別するフルワード整数。 ClassID は、クラスの MyID フィールド、およびクラスとオブジェクトの MyPrimaryParentID フィールドのデータ・タイプにすぎません。

ヌル値: すべてのビットはゼロです。

PL/I 宣言:

```
% ClassID = 'FIXED BINARY(31)';
```

C 宣言:

```
typedef  long  ClassID;
```

ClassIDList (予約済み)

データ・タイプ ID: 2

説明: クラス ID のリスト。これは、クラスの MyClassChildren フィールドのデータ・タイプにすぎません。 ClassIDList の長さフィールドはリスト内のエレメントの数であり、バイト単位の長さではありません。

ヌル値: 長さフィールドはゼロです。

PL/I 宣言:

```
DCL
  1 ClassIDList EKG_BOUNDARY,
    3 Len      Integer,
    3 List(1)  ClassID;
```

注: EKG_BOUNDARY は、UNALIGNED および BASED PL/I 属性の文字置換で、DCL ステートメントを用いたすべての要約データ・タイプの PL/I 定義と一緒に使用します。

C 宣言:

```
typedef    _Packed struct {
             Integer  Length;
             ClassID  List[1];
           } ClassIDList;
```

ClassLinkList (予約済み)

データ・タイプ ID: 6

説明: 4 バイトの長さフィールドと、それに続くリスト (各項目ごとにクラス ID とフィールド ID が連結している) です。 ClassLinkList の長さフィールドはリスト内のエレメントの数であり、バイト単位の長さではありません。各項目は、クラスの MyClassChildren フィールドのシステム・クラス定義に必要な、クラスのいくつかのフィールドへのリンクを指定します。

ヌル値: 長さフィールドはゼロです。

PL/I 宣言:

```
DCL
  1 ClassLinkList EKG_BOUNDARY,
    3 Len      Integer,
    3 List(1),
      5 ClassIdentifier  ClassID,
      5 FieldIdentifier  FieldID;
```

C 宣言:

```
 typedef    _Packed struct {
              Integer    Length;
              ClassLink  List[1];
            } ClassLinkList;
```

ECBAddress (予約済み)

データ・タイプ ID: 8

説明: ECB の 4 バイトのアドレス。 RODM プログラムがイベントの発生時にアプリケーションに通知する際に使用します。 EKG_NotificationQueue クラスにはこのデータ・タイプが必要です。

ヌル値: ヌル・ポインター

PL/I 宣言:

```
% ECBAddress = 'POINTER';
```

C 宣言:

```
typedef  void  *ECBAddress;
```

FieldID

データ・タイプ ID: 26

説明: フィールド ID のフルワード整数。このデータ・タイプは、他のフィールドの ID が入ったフィールドに使用します。

ヌル値: すべてのビットはゼロです。

PL/I 宣言:

```
% FieldID = 'FIXED BINARY(31)';
```

C 宣言:

```
typedef  long  FieldID;
```

Floating

データ・タイプ ID: 9

説明: 汎用の浮動小数点数。この数は、8 バイトで表されます。

ヌル値: すべてのビットはゼロです。

PL/I 宣言:

```
% Floating = 'FLOAT BINARY(53)'
```

C 宣言:

```
typedef  double  Floating;
```

GraphicVar

データ・タイプ ID: 5

説明: 2 バイトの長さフィールドとそれに続く一連の 2 バイト文字として構成される一連のデータ。長さフィールドの値は、16,383 の 2 バイト単位を超えてはなりません。 16 ビットの 2 バイト文字の 1 文字の長さは、2 バイトの 1 単位です。有効な文字には、X'41' から X'FE' の範囲で定義されたデータの先頭バイトと 2 番目のバイトの両方がなければなりません。文字 X'4040' も有効です。 GraphicVar データは、値が X'0000' のヌルの 2 バイトにより終了します。ヌル終止符バイトは、GraphicVar データの長さフィールドには含まれていません。

ヌル値: 長さフィールドはゼロです。

PL/I 宣言:

```
DCL
  1 GraphicVar EKG_BOUNDARY,
    3 Len     Smallint,
    3 Text    CHAR(1);
```

C 宣言:

```
typedef   _Packed struct {
            Smallint Length;
            Smallint Text[1];
          } GraphicVar
```

Integer

データ・タイプ ID: 10

説明: 汎用目的のフルワード整数。

ヌル値: すべてのビットはゼロです。

PL/I 宣言:

```
% Integer = 'FIXED BINARY(31)';
```

C 宣言:

```
typedef  long  Integer;
```

IndexList

データ・タイプ ID: 32

説明: 最大 32767 バイトの複数の値からなるデータの可変長ストリング。データは AnonymousVar データ値のリストで、リスト内の個々のデータ値には、次の特性があります。

- フィールド内では固有でなければならない。
- 最大長は 254 バイト
- 2 バイトの長さフィールドと、それに続く長さフィールド指定のデータ・バイト数から構成される。 AnonymousVar データ・タイプ ID は、値の一部ではありません。

図 49 は、3 つの AnonymousVar 値が入った Indexlist ストリングの例です。

- 00 08 C9 D5 C4 C5 E7 F1 40 40
- 00 06 C9 95 84 85 E7 F1
- 00 08 93 95 C4 C5 A7 C5 C5 C5

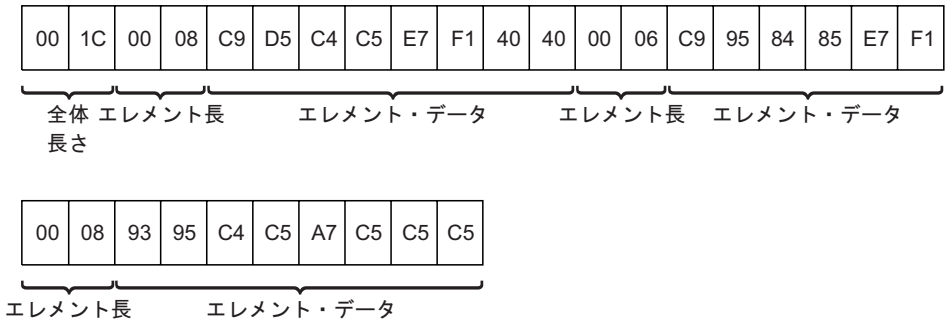


図 49. IndexList フィールドの例

ヌル値: 長さフィールドはゼロです。

PL/I 宣言:

```
% IndexList = 'CHAR(32767) VARYING';
```

C 宣言:

```
typedef    _Packed struct {
             Smallint Length;
             char    Text[1];
           } IndexList;
```

MethodName (予約済み)

データ・タイプ ID: 11

説明: メソッドの名前用の 8 文字のデータ・タイプ。

ヌル値: NullMeth.

PL/I 宣言:

```
% MethodName = 'CHAR(8)';
```

C 宣言:

```
typedef    _Packed struct {
             char     Data_char[8];
           } MethodName;
```

method_parameter_list (予約済み)

データ・タイプ ID: 12

説明: RODM によって保存されて、メソッドに渡される Long-lived パラメーター。最大長は、X'000C' の 2 バイトのヘッダーを除き、254 バイトです。

ヌル値: 長さフィールドはゼロです。

PL/I 宣言:

```
% method_parameter_list =  'SelfDefining';
```

C 宣言:

```
typedef  SelfDefining  method_parameter_list
```

MethodSpec

データ・タイプ ID: 13

説明: オブジェクト特有のメソッド、およびそれを起動した際のそのパラメーターを指定する、メソッド・オブジェクト ID とメソッド・パラメーター・リスト。

ヌル値: ヌルのメソッド・パラメーター・リストに連結した *NullMeth* という名前の予約済みメソッドのメソッド・オブジェクト ID。

PL/I 宣言:

```
DCL
  1 MethodSpec EKG_BOUNDARY,
    3 ObjectIdentifier   ObjectID,
    3 MthdParmList       SelfDefining;
```

C 宣言:

```
typedef    _Packed struct {
             ObjectID      ObjectIdentifier;
             SelfDefining  MthdParmList;
           } MethodSpec;
```

ObjectID (予約済み)

データ・タイプ ID: 14

説明: オブジェクトの MyID フィールドで必要な、オブジェクト ID のダブルワード。

ヌル値: すべてのビットはゼロです。

PL/I 宣言:

```
% ObjectID = 'BIT(64)';
```

C 宣言:

```
typedef    _Packed struct {
             Smallint  Collision_number;
             Smallint  Class_identifier;
             Integer   Object_identifier;
           } ObjectID;
```

ObjectIDList (予約済み)

データ・タイプ ID: 15

説明: 項目が ObjectID であるリスト。クラスの MyObjectChildren フィールドのデータ・タイプ。 4 バイトの長さフィールドとそれに続く、リスト内の項目である ObjectID の連結で構成される一連のデータ。 ObjectIDList の長さフィールドはリスト内のエレメントの数であり、バイト単位の長さではありません。リスト内のオブジェクト ID は、すべて連結し、連続しています。

ヌル値: 長さフィールドはゼロです。

PL/I 宣言:

```
DCL
  1 ObjectIDList EKG_BOUNDARY,
    3 Len      Integer,
    3 List(1)  ObjectID;
```

C 宣言:

```
typedef    _Packed struct {
             Integer   Length;
             ObjectID  List[1];
           } ObjectIDList;
```

ObjectLink

データ・タイプ ID: 16

説明: 別のオブジェクトのフィールドへのリンクを指定する場合の、ダブルワード・オブジェクト ID ならびにフィールド ID。

ヌル値: NULL フィールド ID に連結する NULL オブジェクト ID。

PL/I 宣言:

```
DCL
  1 ObjectLink EKG_BOUNDARY,
    3 ObjectIdentifier   ObjectID,
    3 FieldIdentifier    FieldID;
```

C 宣言:

```
typedef    _Packed struct {
             ObjectID ObjectIdentifier;
             FieldID  FieldIdentifier;
           } ObjectLink;
```

ObjectLinkList

データ・タイプ ID: 17

説明: オブジェクト・リンクのリスト。 4 バイトの長さフィールドとそれに続く、リスト内の項目であるオブジェクト・リンクの連結で構成される一連のデータ。 ObjectLinkList の長さフィールドはリスト内のエレメントの数であり、バイト単位の長さではありません。リスト内のオブジェクト ID は、すべて連結し、連続しています。

ヌル値: 長さフィールドはゼロです。

PL/I 宣言:

```
DCL
  1 ObjectLinkList EKG_BOUNDARY,
    3 Len      Integer,
    3 List(1),
      5 ObjectIdentifier   ObjectID,
      5 FieldIdentifier    FieldID;
```

C 宣言:

```
typedef    _Packed struct {
             Integer     Length;
             ObjectLink  List[1];
           } ObjectLinkList;
```

ObjectName (予約済み)

データ・タイプ ID: 18

説明: オブジェクトの MyName フィールドのデータ・タイプ。名前は、254 文字以下で構成され、1 バイトの X'00' で終了します。 ObjectName データの構造は、2 バイトの長さフィールドとそれに続くストリングの文字です。長さフィールドには、ヌル終止符は含まれません。有効なオブジェクト名については、 240 ページの『オブジェクト名』を参照してください。

ヌル値: 長さフィールドはゼロで、PL/I では、*string* = ’ で設定されます。

PL/I 宣言:

```
% ObjectName = 'CHAR(254) VARYING';
```

C 宣言:

```
typedef    _Packed struct {
             Smallint Name_length;
             char     Name_content[255];
           } ObjectName;
```

RecipientSpec (予約済み)

データ・タイプ ID: 20

説明: 通知メソッドがアプリケーション・プログラムに通知するときに必要とする情報。 8 バイトの ApplicationID、8 バイトの通知キュー SubscribeID、およびデータ・タイプ Anonymous の 8 バイトのユーザー・ワードを含む、一連のデータ。

ヌル値: ヌルのアプリケーション ID、ヌルの SubscribeID、およびヌルの Anonymous(8) ストリングの連結。

PL/I 宣言:

```
DCL
  1 RecipientSpec EKG_BOUNDARY,
    3 User_appl_ID         ApplicationID,
    3 Notification_queue   SubscribeID,
    3 User_word            Anonymous(8);
```

C 宣言:

```
typedef    _Packed struct {
             ApplicationID  User_appl_ID;
             SubscribeID    Notification_queue;
             Anonymous      User_Word[8];
           } RecipientSpec;
```

SelfDefining

データ・タイプ ID: 19

説明: 32767 バイト以下の SelfDefining データ・ストリング。ストリングは、タグ・データ項目の連結です。その各タグ・データ項目の構成は、RODM 要約データ・タイプ ID とそれに続く対応するデータです。予約済みのすべての要約データ・タイプは、 Anonymous(N) データ・タイプ以外の SelfDefining データ・ストリングで使用することができます。

図 50 は、SelfDefining データの形式です。

Self_Defining

図 50. SelfDefining データ・タイプの構文

SelfDefining 構文では、次の変数を使用することができます。

length
2 バイトの長さフィールドそのものを除く、 SelfDefining データ・ストリングの全長を指定する 2 バイトの整数。

identifier
SelfDefining データ・ストリングの ID の直後に続くデータの RODM データ・タイプを指定する、 2 バイトの無符号整数。データ・タイプ ID は、 257 ページの『要約データ・タイプ参照』の RODM データ・タイプ定義で指定されます。

value
identifier によって指定されるデータの値。データ・タイプ ObjectName および ShortName の値の場合、 SelfDefining データ・ストリングにヌル終止符は組み込まれていません。

SelfDefining データ・ストリング内に CharVar を指定するときは、 SelfDefining データ・ストリングの長さフィールドに 1 バイトのヌル終止符を組み込む必要がありますが、 SelfDefining データ・ストリング内の CharVar 指定の長さフィールドには、組み込みません。

図 51 は、10 進値が 1992 の Smallint、値が RODM の CharVar、および値が NETV23 の ApplicationID が入った、 SelfDefining ストリングの例です。

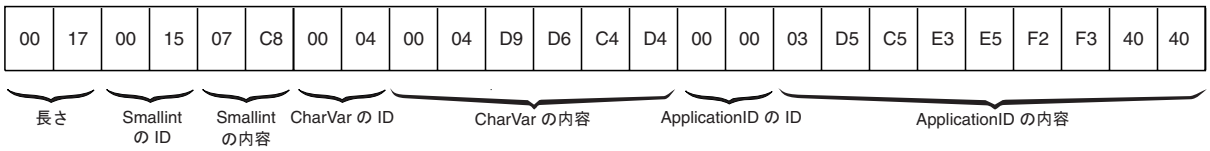


図 51. SelfDefining フィールドの例

ヌル値: 長さフィールドはゼロです。

PL/I 宣言:

```
% SelfDefining = 'CHAR(32767) VARYING';
```

C 宣言:

```
typedef    _Packed struct {
             Smallint  Data_length;
             Anonymous Data_content;
           } SelfDefining;
```

ShortName (予約済み)

データ・タイプ ID: 23

説明: クラスの MyName フィールドおよび任意のオブジェクトまたはクラスの MyPrimaryParentName フィールドのデータ・タイプ。名前は、64 文字以下で構成され、1 バイトの X'00' で終了します。 ShortName データの構造は、2 バイトの長さフィールドとそれに続くストリングの文字です。フィールド名の構成については、 241 ページの『RODM フィールド』を参照してください。

ヌル値: 長さフィールドはゼロで、PL/I では、*string* = ’ で設定されます。

PL/I 宣言:

```
% ShortName = 'CHAR(64) VARYING';
```

C 宣言:

```
typedef    _Packed struct {
             short  Name_length;
             char   Name_content[65];
           } ShortName;
```

Smallint

データ・タイプ ID: 21

説明: 汎用の 2 バイト (ハーフワード) 符号付き整数。

ヌル値: すべてのビットはゼロです。

PL/I 宣言:

```
% Smallint = 'FIXED BINARY(15)';
```

C 宣言:

```
typedef  short  Smallint;
```

SubscribeID (予約済み)

データ・タイプ ID: 22

説明: 8 文字の通知キュー名。フィールドを申請する際に通知キューに関連付けるときに使用します。関連は、申請処理の間に確立されます。文字は 8 バイト内で左寄せされ、右側はブランク (コード・ページ 00500 の場合は X'40') で埋め込まれます。

ヌル値: すべてのバイトがブランク (コード・ページ 00500 の場合は、X'40')。

PL/I 宣言:

```
% SubscribeID = 'CHAR(8)';
```

C 宣言:

```
typedef    _Packed struct {
             char     Data_char[8];
           } SubscribeID;
```

SubscriptSpec (予約済み)

データ・タイプ ID: 24

説明: 通知要求を RODM プログラムに記録する際に使用する、メソッド指定ならびに受信側指定。 SubscriptSpec には、メソッド、メソッド・パラメーター、および予定された通知の受信側に関する情報が組み込まれています。

ヌル値: ヌルの MethodSpec およびヌルの RecipientSpec の連結。

注: SubscriptSpec データ・タイプの一部である MethodSpec データ・タイプは、ObjectID およびメソッド・パラメーター・リストから構成されます。メソッド・パラメーター・リストは自己定義であり、 PL/I 構文では CHAR(254) VARYING です。

SubscriptSpecList (予約済み)

データ・タイプ ID: 25

説明: notify サブフィールドのデータ・タイプ。このデータ・タイプには、SubscriptSpec エレメントのリストが入っています。リストでは、各 SubscriptSpec エレメントが通知申請を表します。 SubscriptSpecList の長さフィールドはリスト内のエレメントの数であり、バイト単位の長さではありません。リスト内の SubscriptSpec エレメントは、すべて連結されて連続しています。

ヌル値: すべてのビットはゼロです。

PL/I 宣言:

```
DCL
  1 SubscriptSpecList EKG_BOUNDARY,
    3 Len      Integer,
    3 Text     CHAR(1);
```

C 宣言:

```
typedef    _Packed struct {
             Integer Length;
             char    Text[1];
           } SubscriptSpecList;
```

TimeStamp

データ・タイプ ID: 27

説明: リリアン・ミリ秒 (8 バイト) で表される時刻の値。リリアン・ミリ秒とは、グレゴリー暦の使用開始を示す 1582 年 10 月 14 日の午前 0 時以降のミリ秒数のことです。時刻の指定範囲は、1582 年 10 月 14 日から 9999 年 12 月 31 日までです。これは、IBM コンパイラー用の共通実行ライブラリーがサポートする時刻形式に似ています。この時刻を共通実行ライブラリー・ルーチンで使用するときは、値を 1000 で割ります。

この時刻形式を生成する場合は、時刻 (TOD) 機構がグリニッジ標準時 (GMT) に設定され、標準の世紀に基づいていることを前提とします。

ヌル値: すべてのビットはゼロです。

PL/I 宣言:

```
% TimeStamp = 'FLOAT BINARY(53)';
```

C 宣言:

```
typedef  double  TimeStamp;
```

TransID (予約済み)

データ・タイプ ID: 28

説明: トランザクション ID は、RODM トランザクションの固有の ID です。

ヌル値: すべてのビットはゼロです。

PL/I 宣言:

```
% TransID = 'CHAR(8)';
```

C 宣言:

```
typedef    _Packed struct {
             char   Content[8];
           } TransID;
```

第 10 章 RODM ロード機能を使用する

この章では、RODM ロード機能を用いて、独自のデータ・モデルとロード・オブジェクト定義を作成する方法を説明します。データ・モデルは、IBM 提供のデータ・モデルを使用しない、新しい RODM アプリケーションを作成する一環として作成します。これを行うには、既存のモデルを修正しても、RODM ロード機能ステートメントを用いてまったく新しいデータ・モデルを作成してもかまいません。

RODM ロード機能を使用すると、データ・モデルを作成し、その初期のデータ値を定義することができます。これを使用すると、RODM プログラムが実行する間に、RODM のクラスおよびオブジェクトの作成、修正、および削除を行うことができます。ロード機能ステートメントが入った順次データ・セットを作成します。ロード機能は、入力データ・セットを読み取り、その情報を RODM データ・キャッシュにロードします。

この章には、次の 5 つのセクションがあります。
- データ・モデル設計時の考慮事項
- RODM ロード機能の概要
- ロード機能ステートメントを使用する
- データ・キャッシュをロードする際の処理
- ロード機能の参照

ロード機能は、RODM の実行中に既存のデータ・モデルを更新するのに使用することができます。ロード機能は、初期化メソッドを用いて実行できるため、 RODM が他のトランザクションを受け入れる前に実行することができます。

データ・モデル設計時の考慮事項

RODM クラスは、子としてのオブジェクト、子としての他のクラス、もしくは子としてのオブジェクトと他のクラスの両方をもつことができます。 276 ページの図 52 に見られるように、親クラスには、新しいクラスもしくは新しいオブジェクトを加えることができます。

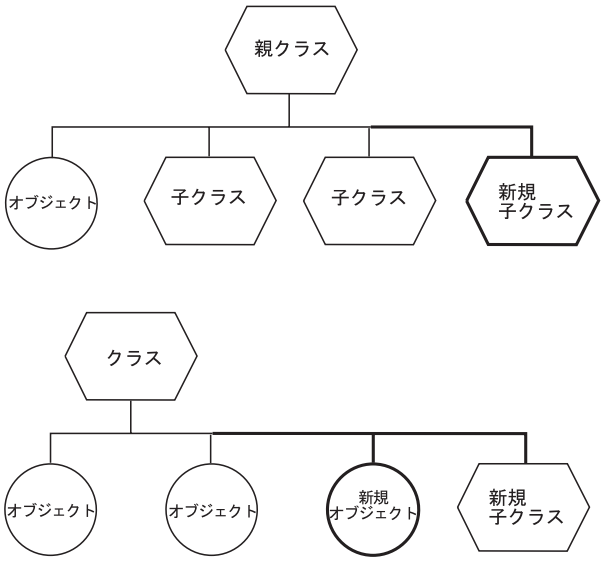


図 52. オブジェクトおよびクラスの追加

RODM ロード機能の概要

RODM ロード機能は、RODM とライブラリーを共用する RODM の一部ですが、RODM ユーザー・アプリケーション・プログラム・インターフェース (API) によってアプリケーション・プログラムのように稼働します。ロード機能ステートメントを用いて、RODM データ・キャッシュ上で動作します。これらのステートメントは、RODM ロード機能への入力として使用される順次ファイルでコーディングします。

ロード機能ステートメント

RODM ロード機能が処理するロード機能ステートメントには、以下の 2 つの異なるレベルがあります。

- 高水準ロード機能ステートメント
- ロード機能プリミティブ・ステートメント

RODM ***高水準*** ロード機能ステートメントは、データ・モデル階層を定義する際に最も一般に使用されるステートメントです。 RODM ロード機能の処理の際、これらのステートメントは、それぞれ 1 つまたは複数の RODM ロード機能プリミティブ・ステートメントに解析されます。次に、これらのプリミティブ・ステートメントは、構文およびアクション用に処理されます。

RODM ロード機能***プリミティブ***・ステートメントは、低水準の構文ステートメントです。これらのステートメントは、高水準ステートメントの処理から RODM ロード機能によって生成されるか、 RODM データ・キャッシュのロードと管理を行う場合の RODM ロード機能への入力として直接使用されます。各プリミティブ・ステートメントは、ユーザー API 呼び出しに密接に対応していますが、場合によっては複数のユーザー API 呼び出しを組み込むこともできます。

さらには、***共通構文エレメント*** があります。これは、RODM 高水準ロード機能構文および RODM ロード機能プリミティブ構文で使用される記述された変数の一群です。

ロード機能の操作

RODM ロード機能には、RODM データ・キャッシュの内容のロード、更新、および妥当性検査ができるようになる、異なる 3 つの操作があります。以下はその 3 つの操作です。

- 解析
- ロード
- 検査

解析 操作は、ロード機能の入力ファイルを処理して、ステートメントのすべての構文をテストします。データ・キャッシュへの変更は行われず、解析操作の際に RODM が実行している必要はありません。ロード機能は、構文エラーが入っているロード機能入力ファイルのステートメントについては、エラー・メッセージを戻します。しかし、解析操作では、値を割り当てたフィールドが存在していない、などの問題のエラーは生じることはありません。

ロード 操作は、ロード機能入力ファイルを解析して、 RODM データ・キャッシュの内容を更新します。ロード機能入力ファイルには、高水準のロード機能ステートメントとロード機能プリミティブ・ステートメントの両方を入れることができます。

RODM ロード機能は、構文エラーが入っているロード機能入力ファイルのステートメントについては、エラー・メッセージを戻します。ロード機能は、構文は正しくても、要求が正常に完了しなかった場合もエラー・メッセージを戻します。例えば、存在しないフィールドに値を割り当てようとすると、ロード機能はエラーを返します。ロード機能は、その処理の一環で、各高水準ロード機能ステートメントをいくつかのロード機能プリミティブ・ステートメントに変換するので、高水準ロード機能ステートメントをコーディングした際に、ロード機能プリミティブについての問題を記述したエラー・メッセージを受け取る場合があります。

ロード操作を実行する前に解析操作を実行して、構文エラーがあれば訂正します。次に、ロード操作を用いて、データ・キャッシュの内容を作成もしくは更新します。データ・キャッシュは、RODM が実行中であれば随時ロード機能を用いて更新することができます。

検査 操作は、ロード機能入力ファイルを解析して、そのステートメントとデータ・キャッシュの内容を比較します。データ・キャッシュへの変更は行われませんが、検査操作を使用するには RODM が実行していなければなりません。検査操作を用いると、データ・キャッシュ内に、指定したクラス、オブジェクト、およびフィールドが存在するかどうかを判別することができます。フィールドに指定の値があるかどうかを判別することもできます。検査操作の詳細記述については、 296 ページの『検査操作を理解する』を参照してください。

RODM データ・キャッシュにロードする

RODM ロード機能入力ファイルを作成した後は、ロード機能を実行して RODM データ・キャッシュにロードする必要があります。 RODM ロード機能は、以下のいずれかとして呼び出します。

- RODM の開始時に実行される初期化メソッド
- プログラムからのモジュール呼び出し
- JCL バッチ・ジョブ

ロードのタイプは、以下の中から選択します。

初期化 メソッド、クラス構造、オブジェクト定義を RODM の開始時にロードします。

構造のみ メソッドとクラス構造定義だけをロードします。すなわち、構造ロードです。

オブジェクトのみ
オブジェクト定義だけをロードします。すなわち、オブジェクト・ロードです。

RODM ロード機能は、ロード機能入力データ・セットの定義に基づくデータ・モデルによって、 RODM データ・キャッシュにロードします。これらのデータ・セットは、以下のラベルが付いた JCL データ定義 (DD) ステートメントによって、RODM ロード機能に識別されます。

EKGIN1 クラス構造定義
EKGIN2 メソッド名テーブル
EKGIN3 オブジェクト定義

RODM データ・キャッシュのロードの詳細については、 281 ページの『RODM データ・キャッシュにロードする処理』を参照してください。

ロード機能ステートメントを使用する

このセクションでは、RODM 高水準ロード機能ステートメントおよび RODM ロード機能プリミティブ・ステートメント、ならびにそれを使用する時点を説明します。 RODM ロード機能は、RODM に以下のことを行わせる RODM ユーザー API を出すときに、これらのステートメントを使用します。

- クラス、フィールド、およびサブフィールドを作成する
- クラス、フィールド、およびサブフィールドを削除する
- フィールドを初期値に設定する
- 階層を定義する親子関係を確立する
- フィールドの値を設定する
- メソッドを起動する

高水準ロード機能ステートメント

このトピックでは、RODM 高水準ロード機能ステートメントを説明します。これらのステートメントのコーディングについては、 313 ページの『RODM 高水準ロード機能ステートメントをコーディングする』を参照してください。

4 つの RODM 高水準ロード機能ステートメントは、以下のとおりです。

MANAGED OBJECT CLASS
クラス定義を追加し、初期値を設定して、RODM データ・キャッシュにデータ・モデルの階層を作成するときに使用する、RODM 高水準ロード機能クラス構造構文。

CREATE
RODM データ・キャッシュにクラスのオブジェクトを作成するときに使用する、 RODM 高水準ロード機能オブジェクト構文。

DELETE
RODM データ・キャッシュからオブジェクトを削除するときに使用する、RODM 高水準ロード機能オブジェクト構文。

SET RODM データ・キャッシュにオブジェクトのフィールドの値を設定するときに使用する、RODM 高水準ロード機能オブジェクト構文。

RODM 高水準ロード機能ステートメントの処理時、各 RODM 高水準ロード機能ステートメントはまず RODM ロード機能プリミティブ・ステートメントに変換されます。例えば、次の MANAGED OBJECT CLASS 高水準ロード機能ステートメントは、 SNANet という名前のフィールドの SNA_Domain_Class という名前の子クラスを、 Domain_Parent_Class という名前のクラスの下に定義します。

```
SNA_Domain_Class             MANAGED OBJECT CLASS;
  PARENT IS Domain_Parent_Class;
  ATTRLIST
   SNANet                                CHARVAR;
END;
```

高水準ステートメントは、RODM ロード機能によって解析され、次の RODM ロード機能プリミティブ・ステートメントに変換されます。

```
OP SNA_Domain_Class HAS_PARENT Domain_Parent_Class;
OP SNA_Domain_Class HAS_FIELD (CHARVAR) SNANet;
```

各 RODM ロード機能プリミティブ・ステートメントは、次に構文およびアクション用に処理されます。 RODM ロード機能プリミティブ・ステートメントの詳細については、『ロード機能プリミティブ・ステートメント』を参照してください。

RODM 高水準ロード機能ステートメント用に生成された RODM ロード機能プリミティブ・ステートメントがエラーを検出すると、その RODM 高水準ロード機能の以降の RODM ロード機能プリミティブ・ステートメントは無視されます。これは、処理中の RODM 高水準ロード機能ステートメントの範囲内で検出されたエラーより後に構文エラーがあっても、検出されないことを意味します。

ロード機能プリミティブ・ステートメント

RODM ロード機能プリミティブは、 278 ページの『高水準ロード機能ステートメント』で説明された RODM 高水準ロード機能ステートメントより低水準にある外部インターフェースです。 RODM ロード機能プリミティブ・ステートメントのコーディング方法については、 322 ページの『RODM ロード機能プリミティブ・ステートメントをコーディングする』を参照してください。

RODM ロード機能プリミティブは、ユーザー生成の入力ファイルから直接来るか、RODM ロード機能によって、入力ファイル内の RODM 高水準ロード機能ステートメントから生成されます。 RODM ロード機能プリミティブ・ステートメントと

RODM 高水準ロード機能ステートメントはともに、同じ RODM ロード機能入力ファイル内で使用することができますが、ロード機能プリミティブは、高水準ステートメント内でコーディングすることはできません。

ロード機能は、プリミティブ・ステートメントを、一度に 1 つずつ順番に処理します。 RODM ロード機能は、プリミティブ・ステートメントのそれぞれをその処理オプションに従って解釈し、しかるべきユーザー API 呼び出しを出して、RODM 機能を実行します。プリミティブは、ユーザー API 呼び出しに密接に対応していますが、場合によっては複数のユーザー API 呼び出しを組み込むこともできます。

高水準もしくはプリミティブのロード機能ステートメントをいつ使用するか

RODM 高水準ロード機能ステートメントは、以下の場合に使用します。

- データ・モデルの初期ロードを実行するとき
- データ・モデルの構造を変更するとき
- RODM ロード機能プリミティブの使用が煩わしい場合に、多数のクラスまたはオブジェクトを RODM データ・キャッシュに追加するとき

RODM ロード機能プリミティブは、クラスの削除、クラスの変更、階層の変更を含むクラス構造の変更を定義する場合、あるいは高水準ステートメントによって希望する機能を実行できないときに使用します。

次の RODM ロード機能プリミティブは、オブジェクトまたはクラスの RODM 高水準ロード機能ステートメントが実行できない機能を実行します。

FORCE_HAS_NO_INSTANCE
: オブジェクトのリンクを解除した後に、無条件にオブジェクトを削除します。

FORCE_NOT_A_CLASS
: リンクに関係なく、無条件にクラスおよびクラスの子を削除します。

HAS_NO_FIELD
: クラス内のフィールドを削除します。

HAS_NO_SUBFIELD
: フィールド内のサブフィールドを削除します。

INVOKED_WITH
: 名前付きメソッドもしくはオブジェクト独立メソッドを起動します。

NOT_A_CLASS
: 子のないクラスを条件付きで削除します。

次の RODM ロード機能プリミティブは、 RODM 高水準ロード機能ステートメントがクラスには実行できない機能を実行します。

注: RODM 高水準ロード機能ステートメントは、これらの機能をオブジェクトには実行することができます。

HAS_VALUE
: クラス内のフィールドの値を定義します。

RODM 高水準ロード機能ステートメント MANAGED OBJECT CLASS は、特定クラスの初期値を定義することができますが、値の変更に使用することはできません。

INHERITS
指定したクラスのフィールドにローカルに定義した値を除き、そのフィールド値をその親から継承した値に戻します。

SUBFIELD_HAS_VALUE
クラス内のサブフィールドの値を定義します。

RODM 高水準ロード機能ステートメント MANAGED OBJECT CLASS によって、クラスに初期化できるのは value サブフィールドのみです。

SUBFIELD_INHERITS
指定したクラスのサブフィールドにローカルに定義した値を除き、そのサブフィールド値をその親から継承した値に戻します。

プリミティブは、構造ロードにもオブジェクトのロードにもコーディングすることができますが、構造のすべてをまず定義してから、オブジェクトを定義しなければなりません。その理由は、親クラスが、必ずそのクラス子またはそのオブジェクト子の作成の前に作成される必要があるためです。

操作を、RODM 高水準ロード機能ステートメントで行うより RODM ロード機能プリミティブで行う方が容易な場合は、 RODM ロード機能プリミティブを使用します。例えば、SNA_Domain_Class というクラスの下に CNM01 というオブジェクトの SNANet という名前のフィールドのフィールド値を、 SET 高水準ステートメントで新しい値に設定することができますが、以下の数行の SET ステートメント構文が必要です。

```
SET  INVOKER   ::= 0001;
     MODE      ::= non-confirmed;
     OBJCLASS  ::= SNA_Domain_Class;
     OBJINST   ::= MyName = (CHARVAR) 'CNM01';
     MODLIST   SNANet ::= (CHARVAR) 'NETC';
END;
```

これに対し、HAS_VALUE プリミティブを使用すると、以下のわずか 1 行の構文でオブジェクトのフィールド値を設定することができます。

```
OP SNA_Domain_Class.CNM01.SNANet HAS_VALUE (CHARVAR) 'NETC';
```

RODM データ・キャッシュにロードする処理

このセクションでは、RODM ロード機能を用いて RODM データ・キャッシュにロードする際に使用する処理を説明します。まず、処理ステップを順番に列挙してから、その順番に従って説明します。

RODM データ・キャッシュにロードするには、以下を行います。

1. インストールするメソッドを識別する
2. クラス構造およびオブジェクト定義を作成する
3. ロードのタイプを決定する
4. RODM ロード機能を実行する
5. 出力リストをチェックする

さらには以下のオプションのステップがあり、これを使用すると、メンバー名およびパラメーターのマッピングを変更できるようになります。
- 制御テーブルを変更する
- パラメーター・マッピング・テーブルを変更する

インストールするメソッドを識別する

クラス構造のロードを、初期ロードもしくはクラス構造の変更の一部として行うと、メソッドのインストールも行うことができます。 RODM アドレス・スペースにインストールするメソッドは、メソッド名テーブル (EKGINMTB) で識別します。 テーブルは、EKGIN2 DD ステートメントによって識別される区分データ・セットのメンバーです。テーブルの形式と他の関連する DD ステートメントについては、 300 ページの『メソッド名テーブル』を参照してください。

RODM ロード機能を実行し、LOAD=STRUCTURE を指定すると、RODM ロード機能は、メソッド名テーブルに指定されたメソッド名ごとに次のステップを実行します。
1. STEPLIB DD データ・セットを検索して、メソッドが使用できるかを確認する
2. メソッド・オブジェクトを作成する
3. メソッドをインストールする

メソッドがインストール済みであるか、メソッド名テーブルに 2 回指定されていると、 RODM ロード機能は次のエラー・メッセージを出します。

```
EKG8568W -

THE METHOD method_name HAS NOT BEEN INSTALLED AS IT ALREADY EXISTS
```

EKGIN2 ファイルがなければなりません。メソッドをインストールしない場合、EKGIN2 ファイルは空のファイルです。メソッドは、ターゲット RODM 始動 JCL の STEPLIB DD ステートメントによって識別されるデータ・セットのいずれかに常駐していなければなりません。

クラス構造およびオブジェクト定義を作成する

以下を行うときは、クラス構造およびオブジェクト定義を入れる順次ファイルを作成します。
- クラス構造およびオブジェクト定義の RODM データ・キャッシュへの初期ロードを行う
- データ・キャッシュのデータ・モデルまたは定義済みオブジェクトの構造に変更を加える

これらの定義は、RODM 高水準ロード機能ステートメントと RODM ロード機能プリミティブから構成されます。 RODM 高水準ロード機能ステートメントおよび RODM ロード機能プリミティブの詳細については、 278 ページの『ロード機能ステートメントを使用する』を参照してください。

データ定義ステートメント・ラベル

RODM ロード機能は、ラベルを付けるロード機能入力定義を収める順次データ・セットもしくは順次データ・セットの連結を宣言する、 DD ステートメントを検出するようになっています。
- クラス構造定義の場合は、EKGIN1

- オブジェクト定義の場合は、EKGIN3

これは、ロード機能の予定ですが、実際には、すべての定義を単一の順次データ・セットもしくは順次データ・セットの連結に入れることもできます。ロードのタイプによってデータ・セットを識別する DD ステートメントの DD 名として、EKGIN1 か EKGIN3 を選びます。ロードのタイプの決め方については、『ロードのタイプを決定する』を参照してください。

この技法は、データ・キャッシュの漸増変更には役立ちますが、『データ・セットの連結』で説明する連結上の注意を守ることはきわめて重要です。

データ・セットの連結

クラス構造およびオブジェクト定義は、いくつかの順次データ・セットに分割してから、これらの定義を入れるデータ・セットを連結することができます。連結でのデータ・セットの順序は重要です。 RODM 高水準ロード機能ステートメントを使用するか、RODM ロード機能プリミティブを使用するかによって、定義を入れるファイルを以下のことが行われるように配列する必要があります。

- RODM ロード機能が、親クラスを、その子を作成する前に作成する
- クラス構造定義が、関連するどのオブジェクト定義よりも前に行われる
- オブジェクトを作成するステートメントが、オブジェクト間にリンクを作成するステートメントの前に処理される

オブジェクト定義は、各データ・セットに 1 つまたは複数のオブジェクト定義が入り、データ・セットがドメイン、サブエリア、あるいは意味のあるものは何でも表せるように、連結することができます。このようにデータ・セットを構成することで、ドメイン用の情報を加えたり、最新表示したりすることができます。

定義の例

RODM は、サンプル・ライブラリーの CNMSAMP という名前の区分データ・セットに 2 つのサンプル・ファイルを備えています。

メンバー	内容
EKGIN1	以下のことを行う目的の、ロード機能ステートメントの例。 • UniversalClass の下にクラスを作成する • サポートされるすべてのデータ・タイプにフィールドを作成する • フィールドに初期値を設定する
EKGIN3	以下のことを行う目的の、ロード機能ステートメントの例。 • 3 つのオブジェクトを作成する • 初期値を設定する

ロードのタイプを決定する

ロード処理のステップは、RODM ロード機能を実行する目的と、実行するロードのタイプによって異なります。 RODM ロード機能は、RODM のコールド・スタートもしくは RODM のウォーム・スタートの際に、初期化メソッドとして実行することができます。 RODM ロード機能は、JCL ジョブを用いて実行することができます。 RODM ロード機能は、アプリケーションからのモジュール呼び出しによって実行することができます。 RODM ロード機能では、以下のロード・タイプが提供されます。

- 初期化ロード
- 構造ロードのみ
- オブジェクト・ロードのみ

初期化ロード

初期化ロードでは、クラス構造、インストールするメソッドの名前、およびオブジェクト定義をロードすることができます。これは、RODM コールド・スタート時に EKGLISLM を呼び出して行われます。

初期化では、次のラベルがついた入力データ用に 3 つの DD ステートメントが必要です。

EKGIN1
クラス構造定義

EKGIN2
メソッド名テーブル

EKGIN3
オブジェクト定義

RODM の初期化が行われると、RODM ロード機能 (EKGLISLM) が起動されて RODM 構造を作成します。この初期ロード・メソッドが、 RODM データ・キャッシュのオブジェクトの値を設定するオブジェクト独立メソッドを実行します。初期ロードが完了した以降の変更は、通常は、定義済みオブジェクトの修正か新しいオブジェクト定義の追加です。

初期ロードでは、RODM ロード機能パラメーターを直接指定することはできません。 RODM は、パラメーター・マッピング・テーブル (EKGPTENU) を使用します。パラメーターのデフォルト値を変更したい場合は、パラメーター・マッピング・テーブルのデフォルト値を変更します。ロード機能が最初に実行される際、ロード機能パラメーターは、そのデフォルト値をパラメーター・マッピング・テーブルから獲得します。しかし、ロード機能は、テーブル内に省略語またはストリング置換があっても無視します。独自のパラメーター・マッピング・テーブルの作成、もしくは RODM のインストールの際にコピーされた表の修正については、301 ページの『パラメーター・マッピング・テーブル』を参照してください。
RODM とともに提供されるパラメーター・マッピング・テーブル EKGPTENU の表示については、 303 ページの図 64 を参照してください。

構造ロードのみ

構造ロード とは、RODM にメソッドとクラス構造のみをロードするロードです。これは、一般に RODM が実行中の JCL もしくはモジュール呼び出しを含むジョブとして行われます。

EKGIN2 データ定義: RODM ロード機能は、ラベルが EKGIN2 のデータ定義ステートメント (そのメンバーのいずれかにメソッド名テーブルが入った区分データ・セットを指定する) を最初に処理します。メソッド名テーブルが入ったメンバーの名前は、RODM が制御テーブル EKGCTABL で見つけます。制御テーブル EKGCTABL、ならびに新規の表を任テーブルに修正もしくは作成する方法については、298 ページの『制御テーブル － EKGCTABL』を参照してください。

メソッド名表の項目ごとに、RODM ロード機能は以下のことを行います。

1. RODM 始動 JCL の STEPLIB DD ステートメントによって識別されるデータ・セットを検索して、メソッドのインストールの有無を調べる。メソッドがインストールされていない場合は、戻りコード 8 と理由コード 81 が戻され、ロード機能はエラー・メッセージを出します。
2. メソッド名テーブル内の項目を該当するクラスの MethodName フィールドに関連付けるロード機能プリミティブを、 RODM ユーザー API 呼び出しに変換する。すなわち、オブジェクトを RODM EKG_Method クラスに加えます。
3. メソッドを RODM アドレス・スペースにロードする。

EKGIN1 データ定義: 初期構造ロードか構造変更かの別なく、構造ロードの間に、RODM ロード機能は、 EKGIN2 データ定義ステートメントの処理が完了した後に、EKGIN1 データ定義ステートメントを処理します。

EKGIN1 は、クラスおよびその親を指定するロード機能入力ステートメントが入っている、順次データ・セットもしくは順次データ・セットの連結を識別します。

RODM ロード機能は、この入力をクラス定義のストリームとして順番に読み取り、すべての RODM 高水準ロード機能ステートメントを RODM ロード機能プリミティブに解析します。次に RODM ロード機能は、ロード機能プリミティブを一連の RODM ユーザー API 呼び出しに変換し、これが RODM データ・キャッシュにクラスを作成します。

データ・セットを連結する場合、EKGIN1 DD ステートメントのデータ・セットの順序は重要です。親クラスが入っているデータ・セットは、その子が入っているデータ・セットの前にロードします。 図 53 は、EKGIN1 DD ステートメントのデータ・セットの連結です。

```
//EKGIN1   DD   DSN=parent.class.input.dataset1,DISP=SHR    (All parent classes)
//         DD   DSN=child.class.input.dataset1,DISP=SHR     (Domain 1 children )
//         DD   DSN=child.class.input.dataset2,DISP=SHR     (Domain 2 children )
//         DD   DSN=child.class.input.dataset3,DISP=SHR     (Domain 3 children )
```

図 53. EKGIN1 のデータ・セットの連結

オブジェクト・ロードのみ

オブジェクト・ロードでは、オブジェクト定義のみをロードすることができます。オブジェクト定義は、RODM の実行中のジョブまたはモジュール呼び出しとしてロードすることができます。オブジェクト・ロードは、ロードのオブジェクト定義が入った順次データ・セットもしくは順次データ・セットの連結を識別するときに、EKGIN3 のラベルがついた DD ステートメントを 1 つ使用します。

データ・セットを連結するときは、オブジェクトを作成するステートメントを、必ずオブジェクト間のリンクを作成するステートメントの前に処理されるようにします。リンクされる両オブジェクトは、リンク・ステートメントが処理されるときは RODM 内になければなりません。連結は、連結データ・セットに関する標準の z/OS 形式で行われます。 286 ページの図 54 は、EKGIN3 DD ステートメントのデータ・セットの連結です。

```
//EKGIN3   DD   DSN=object.instance.input.dataset1,DISP=SHR    (Domain 1)
//         DD   DSN=object.instance.input.dataset2,DISP=SHR    (Domain 2)
//         DD   DSN=object.instance.input.dataset3,DISP=SHR    (Domain 3)
```

図 54. EKGIN3 のデータ・セットの連結

RODM ロード機能を実行する

このトピックでは、RODM ロード機能の呼び出しに加えて、ロード機能実行時の考慮事項について、次の順序で説明します。

- 初期化メソッドとしてのロード機能
- ロード機能のバッチ・ジョブとしての呼び出し
- ロード機能のモジュールからの実行
- ロード機能実行時の考慮事項

RODM ロード機能を実行するには、この機能を初期化メソッド、ジョブ、またはモジュール呼び出しとして実行します。 RODM ロード機能は、データ・モデルの解析、データ・モデルの RODM データ・キャッシュへのロード、あるいはデータ・モデルの検査を行うことができます。

ロードを試みるときは、必ずその前にデータ・モデル定義を解析するようにしてください。こうすると、ロード中のエラーの発生を減らすことができます。このようにすると、これらの定義を RODM データ・キャッシュにロードする前に、ロード機能の入力ステートメント構文のエラーを確認し、訂正することができます。

初期化メソッドとしてのロード機能

NetView とともに提供された初期設定メソッドを使用することも、ユーザー独自の初期設定メソッドを作成することもできます。どちらの場合にも、初期化メソッドを起動するには、メソッドの名前のオブジェクトをユーザーもしくは RODM ロード機能によって EKG_Method クラスに作成しておかなければなりません。

NetView 提供の初期化には、以下の 2 つの部分があります。

EKGLISLM
: EKGIN2 DD ステートメントによって識別されるメソッド名テーブルに定義されているメソッドをロードし、 EKGIN1 DD ステートメントによって識別される順次データ・セットもしくは順次データ・セットの連結のクラス構造をロードしてから、EKGLIILM を起動します。

EKGLIILM
: EKGIN3 DD ステートメントによって識別される順次データ・セットもしくは順次データ・セットの連結のオブジェクト定義をロードします。

EKGLISLM および EKGLIILM は、RODM アドレス・スペースのメソッドとして実行します。これらのメソッドは、RODM が渡してくる環境を使用し、オブジェクト独立メソッドとして稼働します。

コールド・スタート (初期化): RODM を初期化し、データ・キャッシュをコールド・スタートからロードするには、 RODM 始動コマンドの INIT= パラメーターを使用して初期化メソッドの名前を指定します。プログラム (EKGTC000) を実行し、

それが EKGLISLM、つまりロード機能の初期化メソッドを起動し、順番にそれが EKGLIILM を起動します。コールド・スタートには構造ロードが必要であるため、コールド・スタートの場合の RODM 始動コマンドのパラメーターとして INIT=EKGLIILM を指定しないでください。

NetView では、EKGXRODM という名前のサンプル RODM 始動プロシージャーが提供されています。このプロシージャーは初期化ロードを行いますが、この始動プロシージャーを実行する前に、始動プロシージャー JCL に対して次の修正を行ってください。

- STEPLIB DD ステートメントの DSN= パラメーターの *USER.METHODS* の指定を、ユーザー作成のメソッドが入っている区分データ・セットの名前を反映するように変更する。指定がない場合は、コメント化するか、このステートメントを削除します。
- EKGIN1 および EKGIN3 DD ステートメントがクラス構造およびオブジェクト定義を識別しているかを確認する。提供プロシージャーは、定義のコーディング方法の例が入っているデータ・セットを識別します。
- コメント区切り記号を、他のすべての JCL ステートメントから除く。

プロシージャーは、以下の入力を行って実行します。

```
S EKGXRODM,TYPE=C,INIT=EKGLISLM
```

この例で、
- EKGXRODM はプロシージャー名です
- TYPE=C はコールド・スタート操作を指定します
- INIT=EKGLISLM は起動するメソッドの名前を指定します

ウォーム・スタート: ウォーム・スタート時にクラス構造およびオブジェクト定義をデータ・キャッシュにロードするときは、 EKGLISLM を使用することができますが、コールド・スタートと同じく、オブジェクト定義のみをロードするときは、通常 INIT= パラメーターに EKGLIILM を指定します。通常ウォーム・スタートするのは、ネットワーク構成を変更するときか、エラーの結果です。

NetView では、EKGXRODM という名前のサンプル RODM 始動プロシージャーが提供されています。これは、オブジェクト定義ロードの実行に使用します。オブジェクト定義のみをロードする場合は、プロシージャーを実行する前にサンプル・プロシージャーの JCL に次の修正を加えてください。

- 必要に応じて、STEPLIB DD の C ライブラリーを、プロシージャーのヘッディングの注記のようにコメント化する。
- EKGIN3 DD ステートメントが定義を識別するかを確認する。提供プロシージャーは、オブジェクト定義のコーディング方法の例が入ったデータ・セットを識別します。
- EKGLUTB、EKGPRINT および EKGIN3 DD ステートメントのみから、コメント区切り記号を除く。

プロシージャーは、以下の入力を行って実行します。

```
S EKGXRODM,TYPE=W,INIT=EKGLIILM
```

ここで、
- EKGXRODM はプロシージャー名です

- TYPE=W はウォーム・スタート操作を指定します
- INIT=EKGLIILM は起動するメソッドの名前を指定します

ロード機能をバッチ・ジョブとして呼び出す

RODM ロード機能は、バッチ・ジョブとして実行することができます。 RODM ロード機能は、ジョブ実行依頼者の検査済みユーザー ID を RODM に接続するときの User_appl_ID として使用します。検査済みユーザー ID は、システム許可機能から得られます。このユーザー ID には、使用するロード機能ステートメントによって、 3 もしくは 5 の最低 RODM 許可レベルがなければなりません。必要な許可レベルについては、 290 ページの『許可および許可レベル』を参照してください。

ジョブは、以下のロードを行うことができます。
- オブジェクト定義のみ
- メソッドおよびクラス構造定義
- メソッドおよびすべての定義

NetView には、RODM ロード機能をバッチ・ジョブとして実行するサンプル・ジョブとプロシージャーがあります。サンプル・ジョブ EKGLLOAD はプロシージャー EKGLOADP を呼び出し、指定するパラメーターを渡します。 以下のセクションで、RODM をロードできる 3 つの方法のそれぞれについて、 EKGLLOAD サンプル・ジョブの更新方法を説明します。

オブジェクト定義のみをロードする: オブジェクト定義を RODM にロードするときは、サンプル・ジョブ EKGLLOAD をコピーして、更新します。システムの RODM データ・セットの高水準修飾子として NETVIEW.V5R3M0 を使用しない場合は、 EKGLOADP プロシージャーのシステム・レベル修飾子を更新してください。 以下のステップに、EKGLLOAD ジョブによって EKGLOADP プロシージャーに渡されるパラメーターのサンプルの値が示されています。パラメーターごとに、それぞれ独自の値を指定してください。

1. 会計情報によって JOB ステートメントを更新する。
2. RODMNAME に RODM の名前を入れる。
3. EKGIN3 にオブジェクト定義が入ったデータ・セットの名前を入れる。
4. RODM が実行していることを確認し、EKGLLOAD ジョブを実行依頼する。

図 55 は、サンプルの値で更新された EKGLLOAD の一部の行です。

```
//STEP01   EXEC EKGLOADP,
//            RODMNAME=EKGXRODM,
//            EKGIN3=NETVIEW.V5R3M0.CNMSAMP(EKGIN3)
```

図 55. EKGLLOAD サンプルを用いたオブジェクト・ロード・バッチ・ジョブ

メソッド名とクラス構造をロードする: クラスおよびメソッドの定義を RODM にロードするときは、サンプル・ジョブ EKGLLOAD をコピーして、更新します。また、システムで NETVIEW.V5R3M0 を使用しない場合は、 EKGLOADP プロシージャーのシステム・レベル修飾子を更新してください。 以下のステップに、EKGLLOAD ジョブによって EKGLOADP プロシージャーに渡されるパラメーターのサンプルの値が示されています。パラメーターごとに、それぞれ独自の値を指定してください。

1. 会計情報によって JOB ステートメントを更新する。
2. RODMNAME に RODM の名前を入れる。
3. EKGIN1 にクラス定義が入ったデータ・セットの名前を入れる。
4. クラスおよびメソッド・ロードに LOAD=STRUCTURE を指定する。
5. RODM が実行していることを確認し、EKGLLOAD ジョブを実行依頼する。

メソッドは、NETVIEW.V5R3M0.CNMSAMP のメソッド・テーブルに定義されています。 このデータ・セット名を指定する必要はありません。 図 56 は、サンプルの値で更新された EKGLLOAD の一部の行です。

```
//STEP01   EXEC EKGLOADP,
//            RODMNAME=EKGXRODM,
//            EKGIN1=NETVIEW.V5R3M0.CNMSAMP(EKGIN1),
//            LOAD=STRUCTURE
```

図 56. EKGLLOAD サンプルを用いたクラスおよびメソッド・ロード・バッチ・ジョブ

メソッド名とすべての定義をロードする: クラス、メソッド、オブジェクトを EKGLLOAD サンプル・ジョブを用いてロードするには、次の 2 つのオプションがあります。

- まず 288 ページの『メソッド名とクラス構造をロードする』のステップに従ってクラスとメソッドをロードしてから、 288 ページの『オブジェクト定義のみをロードする』のステップに従ってオブジェクトをロードする。
- クラス、メソッド、およびオブジェクト定義のすべてを単一のデータ・セットに入れ、そのデータ・セットを 288 ページの『オブジェクト定義のみをロードする』のステップに従ってロードする。

すべての定義を単一のデータ・セットに入れる代わりに、別々のデータ・セットを連結することができます。 EKGLLOAD ジョブがパラメーターとして渡せるデータ・セットは 1 つのみであるため、これを行うには、EKGLOADP プロシージャーの更新が必要です。

ロード機能をモジュールから呼び出す

RODM ロード機能をモジュールから呼び出すときは、使用している言語に該当するエントリー・ポイントを実行します。 RODM ロード機能は、実行時の呼び出し元プログラムに関連する検査されたユーザー ID を、RODM に接続するときの User_appl_ID として使用します。検査済みユーザー ID は、システム許可機能から得られます。このユーザー ID には、使用するロード機能ステートメントによって、 3 もしくは 5 の最低 RODM 許可レベルがなければなりません。必要な許可レベルについては、 290 ページの『許可および許可レベル』を参照してください。リストが要求されると、リストおよび他の情報が、指定されたデータ・セットに書き込まれ、呼び出し元のモジュールが使用します。

RODM ロード機能モジュールをリンク・エディットするときは、RMODE=24 を指定しなければなりません。

PL/I および C で作成されたモジュールから: PL/I または C で書かれたユーザー・アプリケーション・プログラムが直接 RODM ロード機能を呼び出す場合は、EKGLJOB エントリー・ポイントを呼び出さなければなりません。 EKGLJOB へのリンクは、 305 ページの『z/OS リンクの規則』で説明する z/OS 規則に従わなけ

ればなりません。 RODM ロード機能は、すべてのロード機能をユーザー・アプリケーション・プログラムのタスク制御域環境で実行します。

PL/I または C で作成されていないモジュールから: PL/I または C で書かれていないユーザー・アプリケーション・プログラムが直接 RODM ロード機能を呼び出す場合は、EKGLOTLM エントリー・ポイントを呼び出さなければなりません。 EKGLOTLM エントリー・ポイントは、すべてのロード機能が実行されるタスク制御域環境を作成します。 EKGLJOB の場合と同じリンクの規則を使用します。 305 ページの『z/OS リンクの規則』を参照してください。

RODM ロード機能実行時の考慮事項

RODM ロード機能: RODM ロード機能を実行する際に、アドレス・スペースごとに実行できる RODM ロード機能ジョブは 1 つのみです。 PL/I 実行時ライブラリーがインストールされているか、あるいはジョブの実行依頼もしくは実行の前に使用可能であるかを確認します。 RODM ロード機能は、切断の前に EKG_StopMode フィールドの値を 3 に設定します。 (通知キューも申請も除去しないでください。) この値を使用すると、メソッドが RODM ロード機能によって起動された結果作成された、通知申請、通知キュー、または通知メソッドを除かずに、RODM ロード機能を切断することができます。

RODM プログラム: RODM ロード機能は RODM に接続要求を出してデータ・キャッシュにアクセスするので、 OPERATION=LOAD および OPERATION=VERIFY の場合は、RODM プログラムが実行していなければなりません。 RODM が実行していないと、エラー・メッセージが出されます。

OPERATION=PARSE ならば、RODM は実行している必要はありません。 OPERATION=PARSE の場合、RODM ロード機能は、ロード機能入力ファイルを読み取って、構文エラーを検出するため、それらを解析します。 RODM ロード機能は、RODM に接続機能を出し、RODM のバージョンおよびリリースを照会します。接続機能および照会機能で検出されたエラーは、ジョブ・ログおよび RODM ログに記録されます。ただし、これらのエラーが RODM ロード構文解析操作のエラーと見なされることはありません。パラメーター OPERATION= については、311 ページの『OPERATION』を参照してください。

RODM ロード機能を実行する際に使用する名前が、実行中の RODM プログラムの名前と同じであるかを確認します。 NAME= パラメーターの指定は、実行中の RODM プログラムの名前と同じでなければなりません。パラメーター NAME= については、 311 ページの『NAME』を参照してください。

許可および許可レベル: カスタマイズ・ファイル EKGCUST で *TSTRODM に SEC_CLASS キーワードが設定されていない限り、RODM ロード機能を実行、および RODM ロード機能を呼び出すユーザー・アプリケーション・プログラムを実行するために使用する TSO ID および TSO パスワードは、RODM にアクセスできるようにシステム許可機能によって許可されていなければなりません。

ロード機能を実行する ID には、使用するロード機能ステートメントによって、少なくとも 3 もしくは 5 の許可レベルがなければなりません。 291 ページの表 29 は、ロード機能ステートメント、ステートメント・タイプ、最低許可レベル、およびステートメントに関する追加情報への参照を表しています。

表 29. ロード機能ステートメントおよび最低許可レベル

ステートメント	ステートメント・タイプ	最低許可レベル	参照ページ
CREATE	高水準	3	318
DELETE	高水準	3	319
FORCE_HAS_NO_INSTANCE	プリミティブ	3	323
FORCE_NOT_A_CLASS	プリミティブ	5	324
HAS_FIELD	プリミティブ	5	324
HAS_INSTANCE	プリミティブ	3	325
HAS_NO_FIELD	プリミティブ	5	326
HAS_NO_INSTANCE	プリミティブ	3	326
HAS_NO_SUBFIELD	プリミティブ	5	326
HAS_PARENT	プリミティブ	5	327
HAS_PRV_FIELD	プリミティブ	5	327
HAS_SUBFIELD	プリミティブ	5	328
HAS_VALUE	プリミティブ	3	328
INHERITS	プリミティブ	3	329
INVOKED_WITH	プリミティブ	3	329
IS_LINKED_TO	プリミティブ	3	330
IS_NOT_LINKED_TO	プリミティブ	3	331
MANAGED OBJECT CLASS	高水準	5	316
NOT_A_CLASS	プリミティブ	5	331
SET	高水準	3	320
SUBFIELD_HAS_VALUE	プリミティブ	3	331
SUBFIELD_INHERITS	プリミティブ	3	332

出力リストを検査する

出力リストを理解するには、出力メッセージの形式と出力リストの内容を理解する必要があります。

注: RODM ロード機能によって出されるメッセージの説明については、 NetView のオンライン・ヘルプを参照してください。 RODM ロード機能メッセージは、すべて EKG8 から始まります。

RODM ロード機能を実行すると、異なるタイプの情報から構成される 2 つの出力リストが作成されます。一方のリストは RODM ロード機能によって作成され、EKGPRINT DD ステートメントによって指定されたデータ・セットに書き込まれます。他方のリストは、システム生成の出力で、SYSOUT にあてられます。EKGPRINT DD ステートメントが出力データ・セットとして SYSOUT を指定すると、別々のリストが 1 つの報告書のようになります。

RODM ロード機能の出力リスト

RODM ロード機能により作成されるリストには、日付、現行レベルでの機能の名前、ロード機能の実行時に使用されたオプションのリスト、ロード機能入力、機能

によってとられたアクション、エラー発生時のエコー構文、および END OF JOB メッセージを含むメッセージが含まれます。オブジェクト・ロードのロード機能出力リストの例については、 295 ページの図 59 を参照してください。

EKGPRINT DD ステートメントによって識別されるデータ・セットの内容を表示する際は、使用するソフトウェアとハードウェアが大文字小文字混合で表示できるか確認します。 RODM データでは大/小文字が区別されるので、大/小文字混合でデータを表示しないと、 RODM ロードの検証を妨げることがあります。

構文のすべてを、該当する場合は、実行されたプリミティブの成否を示すメッセージをはさんで、エコーすることも、エラーが検出されたことを示すメッセージ付きで、構文エラーだけをエコーすることもできます。 309 ページで説明する LISTLEVEL パラメーターは、発生する構文エコーのレベルを定義しています。

RODM ロード機能の出力形式

RODM ロード機能出力の形式は、以下の設定に応じて若干異なります。

- 操作のタイプ − PARSE、LOAD、または VERIFY
- ロードのタイプ − STRUCTURE または INSTANCE
- LISTLEVEL オプション − ERRORSYNTAX または ALLSYNTAX

これらのパラメーターの詳細については、 309 ページの『RODM ロード機能パラメーターの構文』を参照してください。

形式の相違点を調べるときは、以下の図を比較します。

- 293 ページの図 57、PARSE 操作の出力例
- 294 ページの図 58、構造ロードの出力例
- 295 ページの図 59、オブジェクト・ロードの出力例

```
OPTIONS USED
------------
OPERATION:PARSE
NAME:RODMNAME
SEV:WARNING
LISTLEVEL:ALLSYNTAX
CODEP:EKGCP500
LOAD:INSTANCE
ROUTECODE:1
INSTANCE  ELEMENTS PROCESSED
   .
   .
   .
--*  DESCRIPTION: SAMPLE STRUCTURE LOAD INPUT FILE                *--
   .
   .
   .
SUPERCLASS                          MANAGED OBJECT CLASS;
PARENT IS                           UNIVERSALCLASS;
ATTRLIST
   FIELD_ANONYMOUSVAR    ANONYMOUSVAR  INITIAL (X'4040'),
   FIELD_BERVAR          BERVAR        INIT(X'810499FF88FF'),
   FIELD_CHARVAR         CHARVAR       INIT ('ANYCHARACTER'),
   FIELD_INDEXCHAR1      CHARVAR       INIT ('INDEXNAME') PUBLIC_INDEXED,
   FIELD_CLASSID         CLASSID,
   FIELD_FIELDID         FIELDID       INIT (SUPERCLASS.FIELD_CHARVAR),
   FIELD_FLOATING        FLOATING      INIT (50.00),
   FIELD_GRAPHICVAR      GRAPHICVAR    INIT ( DBCSDATA ) PRIVATE,
   FIELD_INTEGER         INTEGER       INIT(50) PUBLIC,
   FIELD_OBJECTID        OBJECTID,
   FIELD_OBJECTLINK      OBJECTLINK,
   FIELD_OBJECTLINKLIST  OBJECTLINKLIST,
   FIELD_SMALLINT        SMALLINT      INIT(50),
   FIELD_TIMESTAMP       TIMESTAMP     INIT(X'41B8CCCCCCCCCCD'),
   FIELD_METHODSPEC      METHODSPEC    INIT('EKGNOTF' ((INTEGER) 50)),
   FIELD_SELFDEFINING    SELFDEFINING,
   FIELD_INDEXLIST1      INDEXLIST,
   FIELD_INDEXINDEXLIST1 INDEXLIST     PUBLIC_INDEXED;
END;
BEGIN CLASS SUPERCLASS;* HAS_PARENT UNIVERSALCLASS;* HAS_FIELD (ANONYMOUSVAR)
HAS_VALUE (INTEGER) 50;* HAS_FIELD (OBJECTID) FIELD_OBJECTID;* HAS_FIELD
(OBJECTLINK) FIELD_OBJECTLINK;* HAS_FIELD (OBJECTLINKLIST)
HAS_VALUE (METHODSPEC) ('EKGNOTF' ((INTEGER) 50));* HAS_FIELD (SELFDEFINING)
FIELD_SELFDEFINING;* HAS_FIELD (INDEXLIST) FIELD_INDEXLIST1;*
HAS_INDEXED_FIELD (INDEXLIST) FIELD_INDEXINDEXLIST1;END CLASS *;
   .
   .
   .
END OF JOB    OVERALL RETURN CODE: 00    11:17:15
```

図 57. EKGPRINT への PARSE 操作の出力例

```
OPTIONS USED
------------
OPERATION:LOAD
NAME:RODMNAME
SEV:WARNING
LISTLEVEL:ALLSYNTAX
CODEP:EKGCP500
LOAD:STR
ROUTECODE:1
STRUCTURE ELEMENTS PROCESSED
  .
  .
  .
--*  DESCRIPTION: SAMPLE STRUCTURE LOAD INPUT FILE                *--
  .
  .
  .
SUPERCLASS                          MANAGED OBJECT CLASS;
PARENT IS                           UNIVERSALCLASS;
ATTRLIST
   FIELD_ANONYMOUSVAR    ANONYMOUSVAR  INITIAL (X'4040'),
   FIELD_BERVAR          BERVAR        INIT(X'810499FF88FF'),
   FIELD_CHARVAR         CHARVAR       INIT ('ANYCHARACTER'),
   FIELD_INDEXCHAR1      CHARVAR       INIT ('INDEXNAME') PUBLIC_INDEXED,
   FIELD_CLASSID         CLASSID,
   FIELD_FIELDID         FIELDID       INIT (SUPERCLASS.FIELD_CHARVAR),
   FIELD_FLOATING        FLOATING      INIT (50.00),
   FIELD_GRAPHICVAR      GRAPHICVAR    INIT ( DBCSDATA ) PRIVATE,
   FIELD_INTEGER         INTEGER       INIT(50) PUBLIC,
   FIELD_OBJECTID        OBJECTID,
   FIELD_OBJECTLINK      OBJECTLINK,
   FIELD_OBJECTLINKLIST  OBJECTLINKLIST,
   FIELD_SMALLINT        SMALLINT      INIT(50),
   FIELD_TIMESTAMP       TIMESTAMP     INIT(X'41B8CCCCCCCCCCD'),
   FIELD_METHODSPEC      METHODSPEC    INIT('EKGNOTF' ((INTEGER) 50)),
   FIELD_SELFDEFINING    SELFDEFINING,
   FIELD_INDEXLIST1      INDEXLIST,
   FIELD_INDEXINDEXLIST1 INDEXLIST     PUBLIC_INDEXED;
END;
* HAS_PARENT UNIVERSALCLASS;
EKG8258I - THE HAS_PARENT PRIMITIVE STATEMENT COMPLETED SUCCESSFULLY.
* HAS_FIELD (ANONYMOUSVAR) FIELD_ANONYMOUSVAR;
EKG8258I - THE HAS_FIELD PRIMITIVE STATEMENT COMPLETED SUCCESSFULLY.
EKG8258I - THE HAS_VALUE PRIMITIVE STATEMENT COMPLETED SUCCESSFULLY.
* HAS_FIELD (BERVAR) FIELD_BERVAR;
EKG8258I - THE HAS_FIELD PRIMITIVE STATEMENT COMPLETED SUCCESSFULLY.
EKG8258I - THE HAS_VALUE PRIMITIVE STATEMENT COMPLETED SUCCESSFULLY.
* HAS_FIELD (CHARVAR) FIELD_CHARVAR;
EKG8258I - THE HAS_FIELD PRIMITIVE STATEMENT COMPLETED SUCCESSFULLY.
EKG8258I - THE HAS_VALUE PRIMITIVE STATEMENT COMPLETED SUCCESSFULLY.
* HAS_INDEXED_FIELD (CHARVAR) FIELD_INDEXCHAR1;
EKG8258I - THE HAS_INDEXED_FIELD PRIMITIVE STATEMENT COMPLETED SUCCESSFULLY.
  .
  .
  .
EKG8258I - THE SUBFIELD_HAS_VALUE PRIMITIVE STATEMENT COMPLETED SUCCESSFULLY.
END OF JOB    OVERALL RETURN CODE: 00    13:58:29
```

図 58. EKGPRINT への構造ロードの出力例

```
OPTIONS USED
------------
OPERATION:LOAD
NAME:RODMNAME
SEV:WARNING
LISTLEVEL:ALLSYNTAX
CODEP:EKGCP500
LOAD:INSTANCE
ROUTECODE:1
INSTANCE  ELEMENTS PROCESSED
   .
   .
   .
--*  DESCRIPTION: SAMPLE INSTANCE LOAD INPUT FILE                   *--
   .
   .
   .
CREATE INVOKER  ::= 1;
       OBJCLASS ::= SUBCLASS_2;
       OBJINST  ::= MYNAME = (CHARVAR) 'INSTANCE_4';
       ATTRLIST
           FIELD_ANONYMOUSVAR ::= (ANONYMOUSVAR) X'ABCD',
           FIELD_BERVAR ::= (BERVAR) X'810499FF88FF',
           FIELD_CHARVAR ::= (CHARVAR) 'CHARTEST',
           FIELD_FIELDID ::= (FIELDID) SUPERCLASS.FIELD_INTEGER,
           FIELD_FLOATING ::= (FLOATING) 100.00,
           FIELD_INTEGER  ::= (INTEGER) 100,
           FIELD_SMALLINT ::= (SMALLINT) 100,
           FIELD_TIMESTAMP ::= (TIMESTAMP) X'41B8CCCCCCCCCCCD',
           FIELD_METHODSPEC ::= (METHODSPEC) ('EKGNOTF' ((INTEGER) 100));
END;
SUBCLASS_2 HAS_INSTANCE *;
EKG8258I - THE HAS_INSTANCE PRIMITIVE STATEMENT COMPLETED SUCCESSFULLY.
EKG8258I - THE HAS_VALUE PRIMITIVE STATEMENT COMPLETED SUCCESSFULLY.
EKG8258I - THE HAS_VALUE PRIMITIVE STATEMENT COMPLETED SUCCESSFULLY.
EKG8258I - THE HAS_VALUE PRIMITIVE STATEMENT COMPLETED SUCCESSFULLY.
EKG8258I - THE HAS_VALUE PRIMITIVE STATEMENT COMPLETED SUCCESSFULLY.
EKG8258I - THE HAS_VALUE PRIMITIVE STATEMENT COMPLETED SUCCESSFULLY.
EKG8258I - THE HAS_VALUE PRIMITIVE STATEMENT COMPLETED SUCCESSFULLY.
EKG8258I - THE HAS_VALUE PRIMITIVE STATEMENT COMPLETED SUCCESSFULLY.
EKG8258I - THE HAS_VALUE PRIMITIVE STATEMENT COMPLETED SUCCESSFULLY.
EKG8258I - THE HAS_VALUE PRIMITIVE STATEMENT COMPLETED SUCCESSFULLY.
   .
   .
   .
-- DELETE SUBFIELDS USING THE HAS_NO_SUBFIELD PRIMITIVE --

OP SUPERCLASS.FIELD_CHARVAR HAS_NO_SUBFIELD NOTIFY;
EKG8258I - THE HAS_NO_SUBFIELD PRIMITIVE STATEMENT COMPLETED SUCCESSFULLY.
OP SUPERCLASS.FIELD_CHARVAR HAS_NO_SUBFIELD PREV_VALUE;
EKG8258I - THE HAS_NO_SUBFIELD PRIMITIVE STATEMENT COMPLETED SUCCESSFULLY.
END OF JOB    OVERALL RETURN CODE: 00    13:58:46
```

図 59. EKGPRINT へのオブジェクト・ロードの出力例

ロード機能の参照

このセクションでは、RODM ロード機能の追加参照情報を記載します。以下の事項について説明します。

- ロード機能の検査操作
- データ・タイプの使用法
- ロード機能のデータ・タイプのヌル値
- RODM テーブル:
  - 制御テーブル － EKGCTABL
  - メソッド名テーブル
  - パラメーター・マッピング・テーブル
- 必須およびオプションのデータ定義名
- ロード機能に関する z/OS リンクの規則
- RODM ロード機能の構文:
  - ロード機能の実行に使用するパラメーター
  - 高水準ステートメント
  - プリミティブ
  - 共通構文エレメント

検査操作を理解する

検査操作では、RODM ロード機能の入力ファイルを解析し、ステートメントをデータ・キャッシュの内容と比較します。データ・キャッシュへは変更を加えません。検査操作は、高水準ロード機能ステートメントとロード機能プリミティブ・ステートメントの両方を解析します。ロード機能プリミティブ・ステートメントの方が理解しやすいので、先に説明します。

323 ページの『ロード機能プリミティブの構文および処理ロジック』のロード機能プリミティブ・ステートメントごとの説明には、そのステートメントの検査操作の説明が付いています。検査操作の論理では、ロード機能がステートメントとデータ・キャッシュの内容を比較する方法を説明します。比較が真ならば、ロード機能はゼロの戻りコードを出します。比較が真でなければ、ロード機能はエラー・メッセージを戻します。

例えば、データ・キャッシュ内のあるクラスが別のクラスの親であることを確認する場合は、 HAS_PARENT ロード機能プリミティブ・ステートメントに検査操作を使用することができます。 HAS_PARENT ロード機能プリミティブ・ステートメントの検査操作論理は、ロード機能に指示して、指定の子クラスと親クラスがデータ・キャッシュ内に存在するかどうかを検査します。ロード機能は次に、子クラスの MyPrimaryParentID フィールドが親クラスを指し示しているかどうかを検査します。ロード機能の検査操作を使用するときは、RODM が実行中でなければなりません。

RODM ロード機能は、高水準ロード機能ステートメントを、まずロード機能プリミティブ・ステートメントに変換することで処理します。次にロード機能プリミティブ・ステートメントは、前述の例のように処理されます。

例えば、次の高水準ロード機能ステートメントはロード機能によって処理することができます。

```
ClassA        MANAGED OBJECT CLASS;
PARENT IS     UniversalClass;
ATTRLIST
  Field_1     CHARVAR  INIT('abc'),
  Field_2     CHARVAR  PRIVATE INIT('gsb'),
  Field_3     CHARVAR;
END;
```

検査操作を実行すると、ロード機能はステートメントをロード機能プリミティブ・ステートメントに変換します。ステートメントの先頭 2 行は、次のように変換されます。

```
OP ClassA HAS_PARENT UniversalClass;
```

このロード機能プリミティブ・ステートメントは、最初の例のように処理されます。

フィールド定義リストの各行は、フィールドを作成する 1 つのステートメントに変換され、初期値が与えられていればそれを割り当てる 2 番目のステートメントに変換されます。この例の最初のフィールド定義は、次のように変換されます。

```
OP ClassA HAS_FIELD (CHARVAR) Field_1;
OP ClassA..Field_1 HAS_VALUE (CHARVAR) 'abc';
```

次にロード機能プリミティブ・ステートメントは、323 ページの『ロード機能プリミティブの構文および処理ロジック』での説明のように処理されます。

検査操作をフィールドの値を指定するロード機能ステートメントに使用する際は、値が変わりやすいので注意してください。特定の値について、それに関心がある場合に限り、試験を行います。高水準ロード機能ステートメントの例では、Field_1 の初期値によって、ロード機能が値 abc の Field_1 を試験するステートメントを生成しました。試験を必要とするのがデータ・キャッシュの構造だけならば、検査操作を使用する前に、初期値をフィールド定義から除きます。

CLASSID および OBJECTID データ・タイプを使用する

RODM ロード機能を使用すると、フィールドに CLASSID および OBJECTID データ・タイプを指定できるようになります。しかし、RODM の対応する ClassID および ObjectID 要約データ・タイプは予約されており、SELFDEFINING 変数を除き、これらのデータ・タイプでフィールドを作成することはできません。

CLASSID

RODM ロード機能を用いてタイプ CLASSID のフィールドを作成すると、フィールドは要約データ・タイプが Integer の RODM データ・キャッシュに作成されます。 RODM ロード機能は、指定するクラス名のクラス ID を入手し、クラス ID 値を RODM データ・キャッシュ (タイプ Integer でなければならない) のターゲット・フィールドに入れます。

RODM ロード機能を用いてタイプ CLASSID の値を割り当てる際に、クラス名を指定しますが、指定するクラス名がすでに存在しているかを確認します。 RODM ロード機能を用いてタイプ CLASSID のフィールドは作成しても、初期値を割り当てない場合、フィールドはヌル値で作成されます。

OBJECTID

RODM ロード機能を用いてタイプ OBJECTID のフィールドを作成すると、フィールドは要約データ・タイプが AnonymousVar の RODM データ・キャッシュに作成されます。 RODM ロード機能は、指定するオブジェクト名のオブジェクト ID を入手し、オブジェクト ID 値を RODM データ・キャッシュ (タイプ AnonymousVar でなければならない) のターゲット・フィールドに入れます。

RODM ロード機能を用いてタイプ OBJECTID の値を割り当てる際に、クラス名とオブジェクト名を指定しますが、指定するオブジェクト名とクラス名がすでに存在しているかを確認します。 RODM ロード機能を用いてタイプ OBJECTID のフィールドは作成しても、初期値を割り当てない場合、フィールドはヌル値で作成されます。

RODM ロード機能データ・タイプのヌル値

RODM ロード機能プリミティブと RODM 高水準ロード機能ステートメントで使用するデータ・タイプによっては、ヌル値を指定できるものもあります。この結果、フィールドの値は RODM による定義のように、そのヌル値に設定できるようになります。次のリストは、ヌル値ごとの指定方法を示しています。

```
(ANONYMOUSVAR) X''
(BERVAR) X''
(APPLICATIONID) ''
(CHARVAR) ''
(CHARVARADDR) X'00000000'
(ECBADDRESS) X'00000000'
(GRAPHICVAR) ''
(INDEXLIST) ()
(METHODNAME) 'NullMeth'
(METHODPARAMETERLIST) ()
(OBJECTNAME) ''
(SELFDEFINING) ()
(SHORTNAME) ''
(SUBSCRIBEID) ''
```

制御テーブル – EKGCTABL

EKGCTABL と呼ばれるこの必須制御テーブルに入っているメンバー名は、修正することができます。このテーブルは、必須 DD ステートメントである EKGLUTB DD ステートメントによって識別される区分データ・セットのメンバーです。 RODM では、メンバー名が EKGCTABL のままで、EKGLUTB DD ステートメントによって識別されるデータ・セット内に入るものとします。

EKGCTABL 制御テーブルには、次の 2 つの項目が入ります。

PARAMETER_MAPPING_MEMBER
: パラメーター・マッピング・テーブルが入っている EKGLUTB DD ステートメントによって識別される区分データ・セットのメンバーの名前を指定します。

INSTALL_METHOD_MEMBER
メソッド名テーブルが入っている EKGIN2 DD ステートメントによって識別される区分データ・セットのメンバーの名前を指定します。

図 60 は、制御テーブルの例です。列スケールを入れてあるのは説明のためであって、制御テーブルの一部ではありません。

```
         1         2         3         4         5
1...+....0....+....0....+....0....+.....1...+....0...

PARAMETER_MAPPING_MEMBER:               EKGPTENU
INSTALL_METHOD_MEMBER:                  EKGINMTB
```

図 60. 列スケール付きのサンプル制御テーブル EKGCTABL

必須シンボルの PARAMETER_MAPPING_MEMBER と INSTALL_METHOD_MEMBER は、列 1 から始まらなければなりません。この例では、メンバー名 EKGPTENU と EKGINMTB は列 41 から始まらなければなりません。

他のテーブルと DD 名の関係

300 ページの図 61 は、制御テーブル EKGCTABL、パラメーター・マッピング・テーブル EKGPTENU、メソッド名テーブル EKGINMTB、および DD 名の EKGLUTB と EKGIN2 間の関係を示しています。

図の中で、RODMNAME という RODM の構造を検査するジョブ・ストリームには、 DD ステートメントの EKGLUTB および EKGIN2 があります。 EKGLUTB というラベルの DD ステートメントは、メンバー EKGCTABL と EKGPTENU が入った区分データ・セットの NETVIEW.V5R3M0.CNMSAMP を識別します。 EKGIN2 というラベルの DD ステートメントは、メンバー EKGINMTB が入った区分データ・セットの NETVIEW.V5R3M0.CNMSAMP を識別します。 RODM は、パラメーター・マッピング・テーブルおよびメソッド名テーブルのメンバー名を得るときに制御テーブル EKGCTABL を使用します。

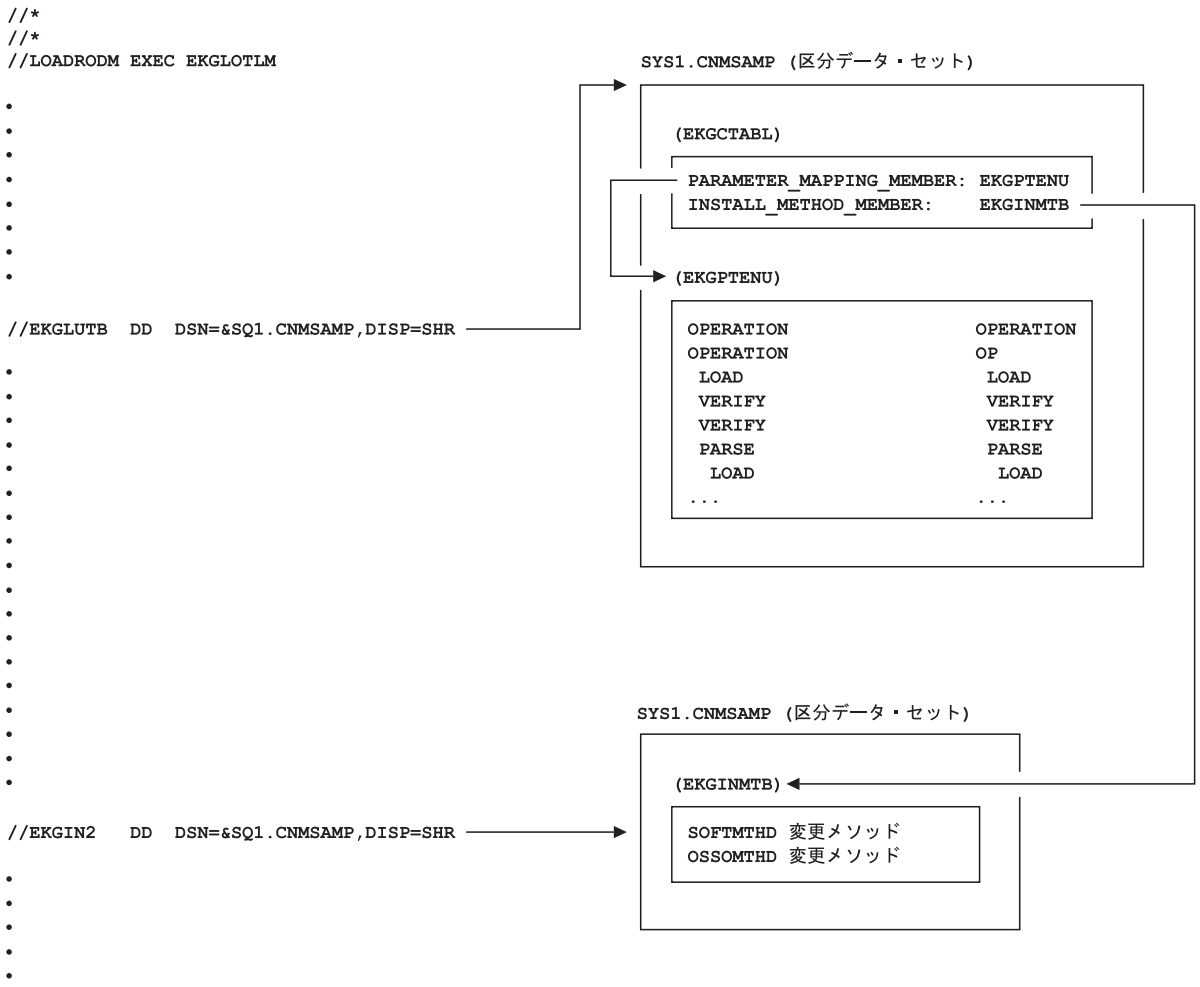


図 61. EKGCTABL、EKGINMTB、EKGPTENU および JCL 間の関係

メソッド名テーブル

メソッド名テーブルには、RODM ロード機能によってインストールさせたいメソッドの名前が入ります。項目 (EKGNOTF) が 1 つだけ入った EKGINMTB という名前のサンプル・ファイルが、サンプル・ライブラリー NETVIEW.V5R3M0.CNMSAMP に入っています。そのファイルをコピーして、修正することも、独自のファイルを作ることもできます。

メソッド名テーブルには、EKGINMTB の名前を使用する必要はありませんが、IBM 提供の制御テーブルでは、メソッド名テーブルに指定されたメンバー名が EKGINMTB であるため、別の名前を使用する場合は、制御テーブル EKGCTABL を修正しなければなりません。制御テーブル EKGCTABL の詳細については、298 ページの『制御テーブル − EKGCTABL』を参照してください。

301 ページの図 62 は、2 つのユーザー作成のメソッドと 7 つの NetView 提供のメソッドを宣言するメソッド名テーブル (EKGINMTB) です。列スケールを入れてあるのは説明のためであって、メソッド名テーブルの一部ではありません。

```
          1         2         3         4         5
1...+..8..1...+....0....+....0....+....0....+....0...

EKGNOTF   NOTIFICATION
EKGNLST   Notify
EKGNEQL   Notify
EKGNTHD   Notify
EKGCTIM   Change method to trigger an OI method
EKGMIMV   Named method to increment a value
EKGSPPI   Object-Independent method
SOFTMTHD  Change Method - (user written)
OSSOMTHD  Change Method - (user written)
```

図 62. 列スケール付きのメソッド名テーブルの形式

メソッド名テーブルの各項目は、1 行から構成されます。列 1 から 8 にはメソッドの名前が入り、列 11 から 80 にはオプションでメソッドのタイプなどのコメントを入れることができます。

RODM メソッド名テーブルのロードをバイパスするには、図 63 に見られるように、制御テーブル EKGCTABL で EKGINMTB を *NONE に置き換えます。列スケールを入れてあるのは説明のためであって、メソッド名テーブルの一部ではありません。

```
         1         2         3         4         5
1...+....0....+....0....+....0....+.....1...+....0...

INSTALL_METHOD_MEMBER:                  *NONE
```

図 63. 列スケール付きのサンプル制御テーブル EKGCTABL

関連 DD ステートメントと制御テーブル

メソッド名テーブルがそのいずれかのメンバーとして入っている、区分データ・セットを宣言する DD ステートメントには、 EKGIN2 のラベルが付けられます。メソッド名テーブルのメンバー名は、 EKGLUTB のラベルが付いた DD ステートメントによって識別される区分データ・セット内の、制御テーブル EKGCTABL にあります。この関連の図については、 300 ページの図 61 を参照してください。

パラメーター・マッピング・テーブル

RODM ロード機能を実行するときは、NAME、OPERATION、CODEPAGE、および LOAD などのパラメーターを指定しなければなりません。 JCL 規則に従って、これらのパラメーターは、EXEC ステートメントの PARM= 部分が括弧内に入ります。パラメーターは、次の形式をとります。

PARM=('*keyword1=keyword_value1,keyword2=keyword_value2***,...')**

パラメーター・マッピング・テーブルは、LRECL が 80 の固定ブロック・テーブルです。このテーブルによって、RODM ロード機能が認識している構文 (内部構文) にストリング置換を使用できるようになります。これらのストリング置換は、省略語でも、各国語へのマッピングでも、あるいはその両方でもかまいません。この結果、RODM ロード機能は他の構文形式に使用できるようになります。

パラメーター・マッピング・テーブル (EKGPTENU) は、 EKGLUTB DD ステートメントによって識別される区分データ・セットのメンバーです。 EKGCTABL 制御ブロックには、パラメーター・マッピング・テーブルのメンバー名が入ります。この関連の図については、 300 ページの図 61 を参照してください。

テーブル EKGPTENU には、列 1 から 30 の内部構文と列 31 から 80 の置換ストリング間に 1 対 1 の関連があります。列 1 から 30 のロード機能のパラメーター・データ (内部構文) については、 309 ページの『RODM ロード機能パラメーターの構文』を参照してください。

以下は、構文の規則です。

- 内部キーワード*項目* は列 1 から、各関連置換ストリング項目は列 31 から始まらなければなりません。
- 内部キーワード*値* は列 2 から、各関連置換ストリング値は列 32 から始まらなければなりません。
- 内部キーワード・デフォルト 値は列 3 から、置換ストリング・デフォルト値は列 33 から始まらなければなりません。
- キーワードごとに、キーワード項目の後にそのキーワードの値項目が付き、その後に順番にそのキーワードのデフォルト値が続きます。

図 64 は、このテーブルの形式の記述で、省略語置換ストリングの例を表しています。列スケールを入れてあるのは説明のためであって、パラメーター・マッピング・テーブルの一部ではありません。

```
         1         2         3         4         5
1...+....0....+....0....+.....1...+....0....+....0...

OPERATION                     OPERATION
OPERATION                     OP
 LOAD                          LOAD
 VERIFY                        VERIFY
 VERIFY                        VER
 PARSE                         PARSE
 PARSE                         PARS
  LOAD                          LOAD
NAME                          NAME
SEVERITY                      SEVERITY
SEVERITY                      SEV
 WARNING                       WARNING
 WARNING                       WARN
 ERROR                         ERROR
 ERROR                         ERR
  WARNING                       WARNING
LISTLEVEL                     LISTLEVEL
LISTLEVEL                     LISTLVL
 ERRORSYNTAX                   ERRORSYNTAX
 ERRORSYNTAX                   ERRORSNTX
 ALLSYNTAX                     ALLSYNTAX
 ALLSYNTAX                     ALLSNTX
  ERRORSYNTAX                   ERRORSYNTAX
CODEPAGE                      CODEPAGE
CODEPAGE                      CODEP
 EKGCP500                      EKGCP500
  EKGCP500                      EKGCP500
LOAD                          LOAD
 STRUCTURE                     STRUCTURE
 STRUCTURE                     STR
 INSTANCE                      INSTANCE
 INSTANCE                      INS
  INSTANCE                      INSTANCE
```

図 64. 列スケール付きサンプル・パラメーター・テーブル EKGPTENU

既存のマッピング・テーブルを修正しても、新しいテーブルを作成してもかまいません。サンプルのロード機能パラメーター・マッピング・テーブルは、RODM とともに提供されるサンプル・ライブラリーのデータ・セット CNMSAMP のメンバー EKGPTENU に入っています。 サンプルをコピーして、コピーに更新があれば行ってください。パラメーター・テーブルの名前を変更するときは、必ず EKGCTABL 制御テーブルを更新してください。

RODM データ定義 (DD) ステートメント

ロード機能を実行するときに使用する DD ステートメントが、データ・セットを宣言します。実行するロードのタイプに該当するデータ・セットの存在を確認します。データ・セット内容が有効であることを確認します。

DD リスト構造 (ロード機能を実行するときにパラメーター・リストを用いて RODM に渡すことができる) を使用すると、 DD 名を必要に合うように変更することができます。 DD リスト構造については、305 ページの『z/OS リンクの規則』で説明します。

STEPLIB (LNKLIST を使用しない場合は必須)

STEPLIB として識別されるデータ・セットは、 RODM ロード機能コードが入っている区分データ・セットでなければなりません。 STEPLIB は、

RODM ロード機能が z/OS LNKLIST にないときは必須 DD ステートメントです。もう 1 つの DD ステートメントは、Language Environment® 実行時ライブラリーを識別する STEPLIB DD ステートメントに連結しなければなりません。 STEPLIB の形式は、すべてのリンク・エディット・データ・セットの標準の DCB (データ制御ブロック) 形式です。

EKGLANG (必須)
EKGLANG DD ステートメントは、RODM ロード機能のメッセージ・ファイルが入っている区分データ・セットを識別します。

EKGLUTB (必須)
EKGLUTB データ定義は、EKGCTABL 制御テーブル・ファイルがそのいずれかのメンバーとして入っている区分データ・セットを識別します。この必須制御テーブルには、 パラメーター・マッピング・テーブルのメンバー名と メソッド名テーブルのメンバー名が入ります。 EKGCTABL 制御テーブルの変更、ならびにパラメーター・マッピング・テーブルとメソッド名テーブルとの関連の詳細については、 298 ページの『制御テーブル − EKGCTABL』を参照してください。

EKGLUTB というラベルの DD ステートメントのデータ制御ブロックは、データ・セットに LRECL=80 と RECFM=FB を指定します。 ブロック・サイズは、80 の倍数でなければなりません。

EKGPRINT (必須)
EKGPRINT データ定義は、RODM ロード機能の出力リストが入るデータ・セットを識別します。このリストには、ロード機能入力、エコー構文、プリミティブ成否の報告書、メッセージとコード、その他の情報が入ります。

SYSOUT への印刷は、順次ファイルもしくは区分データ・セットのメンバーにあてることができます。データ・セットもしくはファイルは、LRECL=80 および RECFM=FB を指定しなければなりません。ブロック・サイズは、80 の倍数でなければなりません。

EKGIN1 (クラス構造定義の場合は必須)
EKGIN1 は、クラス構造定義が入っている、順次データ・セットもしくは順次データ・セットの連結を識別します。クラス構造を定義するデータ・セットは、 LRECL=80 および RECFM=FB を指定するデータ制御ブロックの順次データ・セットでなければなりません。ブロック・サイズは、80 の倍数でなければなりません。 GMFHS データ・モデルを表すクラス構造定義は、サンプル・ライブラリーの CNMSAMP データ・セットのメンバー DUIFSTRC に入っています。

EKGIN2 (クラス構造定義の場合は必須)
EKGIN2 は、メソッド名テーブルがそのいずれかのメンバーとして入っている区分データ・セットを識別します。 EKGIN2 は、LRECL=80 および RECFM=FB を指定するデータ制御ブロックの区分データ・セットでなければなりません。ブロック・サイズは、80 の倍数でなければなりません。 EKGNOTF の項目 (通知メソッド) 1 つをもつ IBM 提供のメソッド名テーブルは、サンプル・ライブラリーの CNMSAMP データ・セットのメンバー EKGINMTB に入っています。

EKGIN3 (オブジェクト定義の場合は必須)
EKGIN3 は、オブジェクト定義が入っている、順次データ・セットもしくは

順次データ・セットの連結を識別します。これらの定義は、ネットワークを定義するときに作成します。 EKGIN3 として連結されるデータ・セットのそれぞれのデータ制御ブロックは、 LRECL=80 および RECFM=FB を指定しなければなりません。ブロック・サイズは、80 の倍数でなければなりません。 19 ページの『第 2 章 ネットワークを GMFHS に定義する』で説明したネットワークを定義するオブジェクト定義は、サンプル・ライブラリーの CNMSAMP データ・セットのメンバー DUIFSNET に例として入っています。

初期化に必要なデータ定義

RODM のコールド・スタートかウォーム・スタートの間に、初期化メソッドを実行する場合は、次のデータ定義名についてのデータ・セットが必要です。

EKGIN1
EKGIN2
EKGIN3
EKGLANG
EKGPRINT
EKGLUTB

構造ロードのみに必要なデータ定義

ジョブ通知かモジュール呼び出しによって、RODM ロード機能を実行し、クラス構造のみをロードして、メソッドをインストールするときは、次のデータ定義名についてのデータ・セットが必要です。

EKGIN1
EKGIN2
EKGLANG
EKGPRINT
EKGLUTB

オブジェクト・ロードのみに必要なデータ定義

ジョブ通知かモジュール呼び出しによって、RODM ロード機能を実行して、オブジェクト定義のみをロードするときは、次のデータ定義名についてのデータ・セットが必要です。

EKGIN3
EKGLANG
EKGPRINT
EKGLUTB

z/OS リンクの規則

306 ページの図 65 は、EKGLJOB へのモジュール呼び出しを使用して RODM ロード機能を実行する際の z/OS リンクの要件です。

レジスター 1 は、3 つまでのパラメーター・アドレスが入っているパラメーター・リストを指し示しています。最初のパラメーター・アドレスは、RODM ロード機能パラメーターを指定する際に使用するパラメーター*構造* を指し示します。 2 番目のパラメーター・アドレスは、3 番目のパラメーター・アドレスが指定されない限りオプションです。これが指定されると、デフォルトの RODM ロード機能 DD 名を変更するときに使用する *DD リスト構造* を指し示します。 3 番目のパラメータ

ー・アドレスはオプションです。これが指定されると、RODM に接続するときに使用されたアクセス・ブロック を指し示します。このパラメーター・リストの最後のアドレスは、高位ビットを ON に設定しなければなりません。

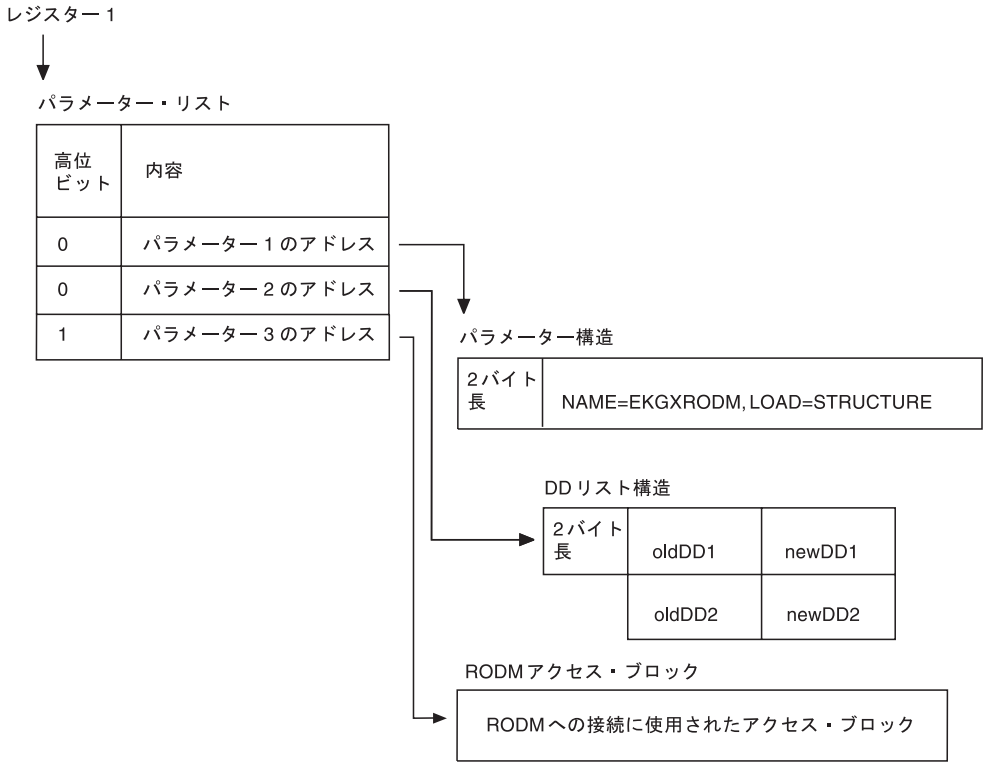


図 65. EKGLJOB へのモジュール呼び出しに必要な z/OS リンクの規則

パラメーター構造

ロード機能に渡されるパラメーターは、パラメーターの長さを指定しなければならない点を除き、 JCL で指定されたパラメーターと同じです。唯一の必須パラメーターは NAME です。パラメーターを指定しない場合のデフォルトは、すべてパラメーター・マッピング・テーブルで指定された値になります。

NAME パラメーターは、アクセス・ブロックを指定すると、無視されます。

パラメーター構造は、2 バイトの固定フィールドと続く文字フィールドから構成されます。固定フィールドには、次の文字フィールドの長さが入っていなければなりません。ロード機能を実行する際の JCL 上の制約から、文字フィールドの長さは 100 バイトを超えてはなりません。文字フィールドには、有効な任意の組み合わせの入力パラメーターを含めることができます。

以下は、16 進形式 (先頭行が 16 進表示、2 行目が EBCDIC) のパラメーター構造の例です。

```
001CD5C1D4C57EC5D2C7E7D9D6C4D46BD3D6C1C47EE2E3D9E4C3E3E4D9C5
    N A M E = E K G X R O D M , L O A D = S T R U C T U R E
```

このパラメーターは、文字フィールドの長さが X'1C' バイトであることを指定します。文字フィールドには、必須の NAME パラメーターと LOAD=STRUCTURE パラメーターが入ります。残りのロード機能パラメーターは、パラメーター・マッピング・テーブルで指定されたデフォルト値にデフォルト解釈されます。

DD リスト構造

DD リスト構造 (指定された場合) は、2 バイトの固定フィールドと、それに続く最大長の制限がない文字フィールドから構成されます。ただし、文字フィールドの長さは 16 の倍数でなければなりません。 DD リスト構造は、データ・セット名またはメンバー名ではなく、DD 名のみの指定に使用されます。

文字フィールドは、各エレメントの長さが 16 (X'10') バイトの DD 名の対の配列から構成されます。先頭 8 バイトはデフォルトもしくは RODM ロード機能で使用された古い DD 名、 2 番目の 8 バイトは RODM ロード機能で使用される新しい DD 名です。この配列の DD 名の対の順序は、任意です。新しい DD 名を指定しない場合は、 303 ページの『RODM データ定義 (DD) ステートメント』で指定されているデフォルトの必須 DD 名が使用されます。

以下は、16 進形式 (先頭行が 16 進表示、2 行目が EBCDIC) の DD リスト構造の例です。

```
0020C5D2C7C9D5F14040E2E3D9E4C3E34040C5D2C7C9D5F34040D6C2D1C5C3E34040

   E K G I N 1    S T R U C T    E K G I N 3    O B J E C T
```

このパラメーターは、DD 名の対が 2 つあること、かつ RODM ロード機能は、EKGIN1 に代えて新しい DD 名 STRUCT、そして EKGIN3 に代えて新しい DD 名 OBJECT を使用することを指定します。

アクセス・ブロック

アクセス・ブロック (指定された場合) は、ユーザー・アプリケーションが RODM に接続する際に使用したアクセス・ブロックです。したがって、RODM に接続済みのユーザー・アプリケーションは、最初に RODM から切断しなくても RODM ロード機能を使用することができます。

アクセス・ブロック・パラメーターを指定する場合は、DD リスト構造も指定しなければなりません。しかし、DD 名を変更したくない場合は、ヌル・ストリングを指定することができます。

RODM ロード機能を呼び出す

RODM ロード機能を呼び出すときは、 306 ページの図 65 に示したリンクの規則に従います。 RODM ロード機能のリンクの規則は、標準の z/OS の方法に従っています。 モジュールの RODM ロード機能へのリンクを定義するときは、ASM および INTER オプションを使用します。 308 ページの図 66 を参照し、次のステートメントを探します。

```
DCL EKGLJOB       OPTIONS(ASM INTER) ENTRY EXTERNAL;
```

308 ページの図 66 は、RODM ロード機能を PL/I プログラムから呼び出す方法の例です。

```
/*********************************************************************/
/* Local Variables                                                   */
/*********************************************************************/
%DECLARE PL1_OR_C FIXED;            /* Flag indicates whether this */
%PL1_OR_C = 1;                      /*  module is IBM PL/1 or C*/
DCL MODULETYPE FIXED INIT(1);       /* Input parm                  */

/*********************************************************************/
/* Declare the parms to pass to RODM LOAD function                   */
/*********************************************************************/
                                    /* Keyword parms for load     */
DCL PARM_STRING    CHAR(100) VARYING ALIGNED;
                                    /* Load DD name mapping       */
DCL DD_STRING      CHAR(160) VARYING ALIGNED;
```

図 66. RODM ロード機能を PL/I プログラムから呼び出す (1/4)

```
/*********************************************************************/
/* Declare the external entry                                        */
/*********************************************************************/
                                    /* This entry is used when    */
                                    /*  calling C or IBM PLI  */
                                    /* modules                    */
DCL EKGLJOB        OPTIONS(ASM INTER) ENTRY EXTERNAL;
                                    /* This entry is used         */
                                    /*  otherwise                 */
DCL EKGLTOLM       OPTIONS(ASM INTER) ENTRY EXTERNAL;
```

図 66. RODM ロード機能を PL/I プログラムから呼び出す (2/4)

```
/*********************************************************************/
/* Assign the value for the parms                                    */
/*********************************************************************/
                                    /* Load function input parms  */
PARM_STRING = 'OPERATION=LOAD,LOAD=INSTANCE,NAME=EKGXRODM';
                                    /* DD name mapping            */
                                    /* Must be multiple of 16     */
                                    /* First 8 bytes specific RODM*/
                                    /*  DD name, and the second 8 */
                                    /*  bytes specifics the DD    */
                                    /*  name user want to use     */
                                    /*  instead.                  */
                                    /* Use OBJECT1 DD name instead*/
                                    /*  EKGIN3 DD name            */
DD_STRING = 'EKGIN3    OBJECT1 ';
                                    /* Use SYSPRINT DD for load   */
                                    /*  messages.                 */
DD_STRING = DD_STRING || 'EKGPRINTSYSPRINT';
```

図 66. RODM ロード機能を PL/I プログラムから呼び出す (3/4)

```
/*********************************************************************/
/* Call load function.                                               */
/*********************************************************************/
IF MODULE_TYPE = PL1_OR_C THEN        /* If it is IBM PL/1 or C */
DO;                                   /* Check DD name mapping      */
  IF LENGTH(DD_STRING) > 0 THEN       /* If yes, pass both parms    */
                                      /* If yes, pass both parms    */
      CALL EKGLJOB(PARM_STRING,DD_STRING);
  ELSE                                /* If no,pass only PARM_STRING*/
      CALL EKGLJOB(PARM_STRING);
END;                                  /* End check DD name mapping  */
ELSE                                  /* Use EKGLOTLM entry point   */
DO;                                   /* Check DD name mapping      */
  IF LENGTH(DD_STRING) > 0 THEN       /* If yes, pass both parms    */
                                      /* If yes, pass both parms    */
      CALL EKGLOTLM(PARM_STRING,DD_STRING);
  ELSE                                /* If no,pass only PARM_STRING*/
      CALL EKGLOTLM(PARM_STRING);
END;                                  /* End check DD name mapping  */
```

図 66. RODM ロード機能を PL/I プログラムから呼び出す (4/4)

RODM ロード機能パラメーターの構文

以降は、アルファベット順の、RODM ロード機能パラメーターの説明と構文です。

構文は、構文図で表します。

CODEPAGE

説明: 入力走査用のコード・ページ。

構文:

CODEPAGE

使用上の注意: 入力走査用のコード・ページ 500 (米国英語) を示すには、CODEPAGE=EKGCP500 とコーディングします。

注: RODM ロード機能がサポートするのは、コード・ページ 500 のみです。

LISTLEVEL

説明: 生成するリストのレベル。エラーのある構文のみをリストすることも、RODM ロード機能への入力として使用されたすべての構文をリストすることもできます。

構文:

LISTLEVEL

使用上の注意: 以下の指定を行う場合:

LISTLEVEL=ALLSYNTAX
生成されたプリミティブ・ステートメントを含む、すべての構文がリストされます。この際、実行された高水準ステートメントおよびプリミティブの成否を示すメッセージが、しかるべき場所に挿入されます。

LISTLEVEL=ERRORSYNTAX
エラーのあるステートメントのみが、*高水準ステートメントから生成されるプリミティブ・ステートメントを除いて、そのエラー・メッセージとともにリストされます。生成されたプリミティブ・ステートメントのエラー・メッセージは、それに関連する高水準ステートメントの後に表示されます。エラーの原因である生成されたプリミティブ・ステートメントは、リストされません。*

LOAD

説明: ロードのタイプ。構造ロードまたはオブジェクト・ロード。

構文:

LOAD

使用上の注意: 以下の指定を行う場合:

LOAD=STRUCTURE
EKGIN1 および EKGIN2 データ定義ステートメントによって識別される、データ・セットからの入力ステートメントのみが使用されます。構造ロードに使用されます。

LOAD=INSTANCE
EKGIN3 データ定義ステートメントによって識別される、データ・セットからの入力ステートメントのみが使用されます。オブジェクト・ロードに使用されます。

クラス構造定義だけでなくオブジェクト定義もロードするときは、
LOAD=STRUCTURE 指定も使用することができます。通常は EKGIN3 DD ステートメントによって識別されるオブジェクト定義を入れるデータ・セットを、EKGIN1 DD ステートメントに連結します。

LOAD=INSTANCE を指定する際に、クラス構造定義をオブジェクト定義に組み込むこともできます。データ・セット連結を用いて、EKGIN3 DD の JCL ステートメントを整理して、通常は EKGIN1 DD によって識別されるクラス構造定義を最初に、続いてオブジェクト定義が処理されるようにします。

NAME

説明: ロードが実行される RODM の名前。これは、構造ロードおよびオブジェクト・ロードには必須パラメーターです。

構文:

NAME

使用上の注意: MYRODM の RODM 名を指定するには、NAME=MYRODM とコーディングします。

NAME パラメーターは、ロードおよび検査操作には必須です。解析操作に NAME を指定すると、RODM ロード機能は指名された RODM に接続しますが、これは必須ではありません。

NAME パラメーターは、初期化メソッド・ロードには必要ありません。 RODM ロード機能を実行しているのは特定の RODM であるため、RODM 名はロード機能に認識されています。

OPERATION

説明: RODM ロード機能が実行する操作。操作パラメーターでは、RODM ロード機能が、ロード機能入力ステートメントを解析して妥当性を調べ、RODM データ・キャッシュにロードし、あるいは他の操作の実行前に定義された内容の存在を検査するように指定することができます。

構文:

OPERATION

使用上の注意: 次のコーディングを行います。

OPERATION=PARSE

RODM ロード機能入力パラメーターが入っているデータ・セットの構文を解析します。 OPERATION=PARSE ならば、RODM は実行している必要はありません。 OPERATION=PARSE の場合、RODM ロード機能は、ロード機能入力ファイルを読み取って、構文エラーを検出するため、それらを解析します。 RODM ロード機能は、RODM に接続機能を出し、RODM のバー

ジョンおよびリリースを照会します。接続機能および照会機能で検出されたエラーは、すべてジョブ・ログおよび RODM ログに記録されます。ただし、これらのエラーが RODM ロード構文解析操作のエラーと見なされることはありません。

OPERATION=LOAD
入力ステートメントを解析してから、データ・キャッシュにロードします。

OPERATION=VERIFY
RODM データ・キャッシュの内容を、解析し検査します。

PARSE も VERIFY も、LOAD 操作は実行しません。

オブジェクト値を割り当てて、失敗させずに実際に存在するオブジェクトを調べたいときは、VERIFY 操作を使用します。 VERIFY の詳細については、296 ページの『検査操作を理解する』を参照してください。

LOAD=STRUCTURE では、EKGIN1 というラベルの DD によって識別されるデータ・セットからの入力ステートメントは解析されますが、 EKGIN2 というラベルの DD によって識別されるデータは解析されません。 LOAD=INSTANCE では、 EKGIN3 というラベルの DD によって識別されるデータ・セットからの入力ステートメントのみが解析されます。これが起こるのは、LOAD、PARSE、または VERIFY 操作の場合です。

ROUTECODE

説明: ローダーが WTO または WTOR マクロを使用してコンソールにメッセージを発行する際に使用する経路コードを定義します。有効値の範囲は 1 から 128 です。デフォルト値は 1 です。

このパラメーターが処理される前に発行可能なメッセージの場合は、ここで設定する値とは無関係に、デフォルトの経路コード 1 が使用されます。

構文:

ROUTECODE

SEVERITY

説明: アプリケーションがクラス構造定義もしくはオブジェクト定義の処理で、エラー (戻りコード 8) を扱う方法。エラー (戻りコード 8) として扱う方法または警告 (戻りコード 4) として扱う方法です。

SEVERITY=ERROR の場合、RODM ロード機能は、ロード機能の入力ステートメントでエラーを検出すると、そのステートメントで処理を終了し、8 の戻りコードを出します。 SEVERITY=WARNING の場合、RODM ロード機能は、ロード機能の入力ステートメントでエラーを検出しても処理を継続し、完了時に 4 の戻りコードを出します。

構文:

SEVERITY

使用上の注意: アプリケーションがクラス構造定義もしくはオブジェクト定義の処理中のエラーをエラーとして扱う場合は、SEVERITY=ERROR とコーディングします。

アプリケーションがクラス構造定義もしくはオブジェクト定義の処理中のエラーを警告として扱う場合は、SEVERITY=WARNING とコーディングします。

構文を解析しているときは WARNING オプションを使用し、ロードしているときは ERROR オプションを使用します。

RODM 高水準ロード機能ステートメントをコーディングする

この参照セクションのトピックでは、RODM 高水準ロード機能ステートメントのコーディング方法を説明します。このトピックでは、高水準ロード機能ステートメントの構文と関連する規則を記載します。

構文は、構文図で表します。

高水準ロード機能ステートメントの構文の規則

このトピックでは、RODM 高水準ロード機能ステートメントに適用される構文の規則について述べます。

入力列: RODM ロード機能は、入力レコードの列をすべてデータとして読み取ります。列 73 から 80 は、順序番号もしくは行番号として使用してはなりません。順序番号もしくは行番号は、コメント (--) 文字を用いてコメントとしてマークを付けた場合は、使用することができます。

区切り文字: 表 30 は、RODM 高水準ロード機能ステートメントに有効な構文の区切り文字の説明です。

表 30. RODM 高水準ロード機能ステートメントの構文の区切り文字

区切り文字	機能
' '	文字ストリングの囲みに使用する。
X'0E' (シフトアウト)	DBCS 混合文字ストリングのデータ・タイプの開始をマークする。
X'0F' (シフトイン)	DBCS 混合文字ストリングのデータ・タイプの終わりをマークする。
-- (ハイフン 2 つ)	コメントの始めか終わりをマークする。

RODM ロード機能では、フリー・フォームの構文を使用することができます。RODM ロード機能では RODM 高水準ロード機能ステートメントの各部分間に 1 つまたは複数のスペースを入れることができるため、スペースを使用してロード機

能の入力データを読みやすくすることができます。例えば、以下の MANAGED OBJECT CLASS 高水準ロード機能ステートメントは、スペースの有効な使い方で読みやすくなっています。

```
Software                                          MANAGED OBJECT CLASS;
  PARENT IS UniversalClass;
  ATTRLIST;
END;
```

引用符付きストリング: 引用符付きストリングは、同じ行で始まり、終わらなければなりません。複数行にわたるストリングを作成する場合は、それを複数行にわたる引用符が付いたパーツに分割します。分割された複数パーツは、RODM ロード機能によって連結されます。例えば、次の 2 行は単一引用符ストリングになります。

```
INIT(' This is the first line of two lines '
     ' that results in one quoted string ' );
```

引用符内に含まれる引用符は、2 つの単一引用符で表されます。例えば、次のようになります。

```
INIT('This is '' a quote '' within a quote. ');
```

引用符は、ストリングの部分としてのキーワードまたはセパレーターを含む、ストリング全体を囲むために使用されます。例えば、次のようになります。

```
INIT(' Create the "MANAGED OBJECT CLASS" now ');
```

2 バイト文字ストリング: RODM ロード機能は、X'0E' シフトアウト文字と X'0F' シフトイン文字間のすべてのデータ値を 2 バイト文字ストリング (DBCS) データとして扱います。これは、通常区切り文字を示すどの 16 進コードも 2 バイト文字ストリング内のデータとして扱われることを意味します。有効な 2 バイト文字は、GraphicVar データ・タイプの 2 バイト文字と同じです。 264 ページの『GraphicVar』を参照してください。

フィールド定義リスト: キーワードが ATTRLIST または MODLIST のフィールド定義リストを指定するときは、リストの各メンバーをコンマで区切り、リストをセミコロンで終わらせます。そうしない場合、RODM ロード機能はリストの各メンバーを別々のステートメントとして扱います。

高水準ステートメントに使用できるデータ・タイプおよびデータ・タイプ値は、RODM で使用が認められているすべてのデータ・タイプおよびデータ・タイプ値です。これらのデータ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。これらのデータ・タイプの値のリストと typed_value ロード機能の共通構文エレメントの構文図については、 340 ページの『typed_value』を参照してください。

コメント: コメントは、始めと終わりを 2 つのハイフン (--) で区切られます。例えば、次のようになります。

```
   -- This is a comment --
```

コメント区切り記号の終わりを指定しなかった場合、コメントの終わりは入力行の終わりと見なされます。 RODM ロード機能は、コメント区切り記号間のテキストをすべて無視します。

高水準ロード機能ステートメントの構文

これは、RODM データ・キャッシュで作成されるデータ・モデル定義の RODM 高水準ロード機能ステートメントをコーディングする際に使用する、構文参照です。RODM 高水準ロード機能ステートメントごとに、その名前、目的、外部構文、構文パラメーター記述、および使用例が入った記述があります。

注: RODM 高水準ロード機能ステートメントの構文は、大小文字を区別します。

このセクションでの RODM 高水準ロード機能ステートメントの使用例は、 図 67 に見られるようにロード機能の入力ステートメント・ストリームのサブセットです。これらのステートメントは、 316 ページの図 68 に記載されている階層の疑似構造を作成して、使用します。この構造および関連するフィールドは、単なる説明用の例であって、RODM の一部ではありません。

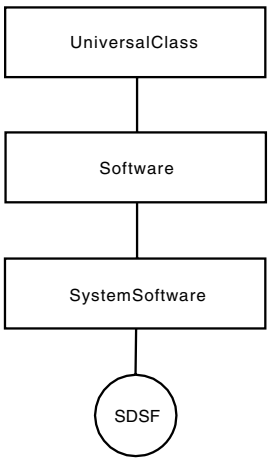


図 67. 階層の疑似構造の例

```
Software                                          MANAGED OBJECT CLASS;
  PARENT IS UniversalClass;
  ATTRLIST;
END;
SystemSoftware                                    MANAGED OBJECT CLASS;
  PARENT IS Software;
  ATTRLIST           -- Field List --
    ProductName                       CHARVAR,
    ProgramNumber                     CHARVAR    INIT('None'),
    LatestPTFNumber                   CHARVAR    INIT('UY12345'),
    CorrespondingAPARNumber           CHARVAR,
    DateApplied                       CHARVAR,
    Priority                          INTEGER    INIT(3),
    UseInHost                         OBJECTLINKLIST;
END;
CREATE    INVOKER    ::= 0000003;
          OBJCLASS   ::= SystemSoftware;
          OBJINST    ::= MyName = (CHARVAR) 'SDSF';
          ATTRLIST
                   ProductName             ::= (CHARVAR)  'SDSF',
                   ProgramNumber           ::= (CHARVAR)  '5697-B82',
                   LatestPTFNumber         ::= (CHARVAR)  'UY12903',
                   CorrespondingAPARNumber ::= (CHARVAR)  'PL45419',
                   DateApplied             ::= (CHARVAR)  '03/01/97',
                   UseInHost               ::= (OBJECTLINKLIST)
                    ('Host_Class'.'HostA'.'UseSystemSoftware')
                    ('Host_Class'.'HostC'.'UseSystemSoftware');
END;
SET       INVOKER    ::= 0000004;
          MODE       ::= non-confirmed;
          OBJCLASS   ::= SystemSoftware;
          OBJINST    ::= MyName = (CHARVAR) 'SDSF';
          MODLIST
                   ProductName             ::= (CHARVAR) 'SDSF V2', REPLACE,
                   ProgramNumber           ::= (CHARVAR) '5697-B82',
                   LatestPTFNumber         ::= (CHARVAR), SET TO DEFAULT,
                   CorrespondingAPARNumber ::= (CHARVAR) ' ',
                   DateApplied             ::= (CHARVAR) '03/01/97',
                   UseInHost               ::= (OBJECTLINKLIST)
                             ('Host_Class'.'HostA'.'UseSystemSoftware'),
                                                    REMOVE VALUE;
END;
DELETE    INVOKER    ::= 0000005;
          OBJCLASS   ::= SystemSoftware;
          OBJINST    ::= MyName = (CHARVAR) 'SDSF';
END;
```

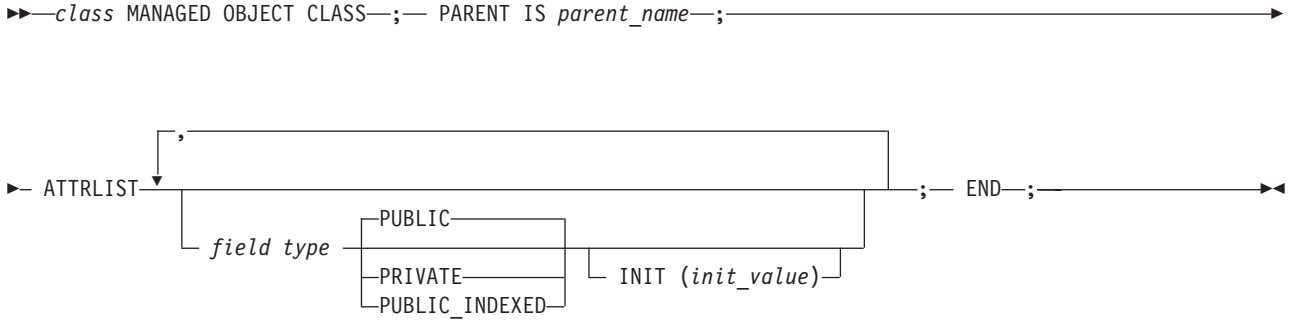
図 68. 疑似構造の高水準入力ステートメント

MANAGED OBJECT CLASS:

目的: MANAGED OBJECT CLASS 高水準ロード機能ステートメントは、階層を定義して、 RODM データ・キャッシュ内にデータ・モデルのクラス構造を作成するときに使用します。

以下の構文は、RODM ロード機能が RODM データ・キャッシュに加えるクラス構造を宣言します。この構文には、値をリセットし、あるいはクラス構造の全部または一部を修正もしくは削除する場合のキーワードは含まれていません。

構文:



キーワードおよびパラメーターの記述:

class 定義するクラスの名前もしくはラベル。

PARENT IS *parent_name*
作成されるクラスの親クラスの名前。

field type
作成されるクラスのデータ・タイプ *type* の名前 *field* でフィールドを作成します。このフィールドの有効なデータ・タイプのリストについては、 340 ページの『type』を参照してください。

PUBLIC|PRIVATE|PUBLIC_INDEXED
フィールドが共用フィールドか、共用索引付きフィールドか、または専用フィールドかを指定します。共用フィールドは、このクラスの子によって継承されますが、専用フィールドは継承されません。共用索引付きフィールドの詳細については、 253 ページの『索引付きフィールド』を参照してください。

INIT (*init_value***)**
フィールド用の初期値の設定。 INITIAL は、INIT の代わりに使用することができます。

例: Software というクラスの子であり、かつ以下のフィールドをもつ SystemSoftware というクラスを指定するとします。

ProductName
ProgramNumber
LatestPTFNumber
CorrespondingAPARNumber
DateApplied
Priority
UseInHost

ProgramNumber というフィールドの初期値が None、 LatestPTFNumber というフィールドの初期値が UY12345、 Priority というフィールドの初期値が 3 であるとします。次の MANAGED OBJECT CLASS ステートメントは、SystemSoftware という名前のクラスを定義します。

```
SystemSoftware                        MANAGED OBJECT CLASS;
  PARENT IS Software;
  ATTRLIST          -- Field List --
    ProductName                       CHARVAR,
```

```
    ProgramNumber                   CHARVAR    INIT('None'),
    LatestPTFNumber                 CHARVAR    INIT('UY12345'),
    CorrespondingAPARNumber         CHARVAR,
    DateApplied                     CHARVAR,
    Priority                        INTEGER    INIT(3),
    UseInHost                       OBJECTLINKLIST;
END;
```

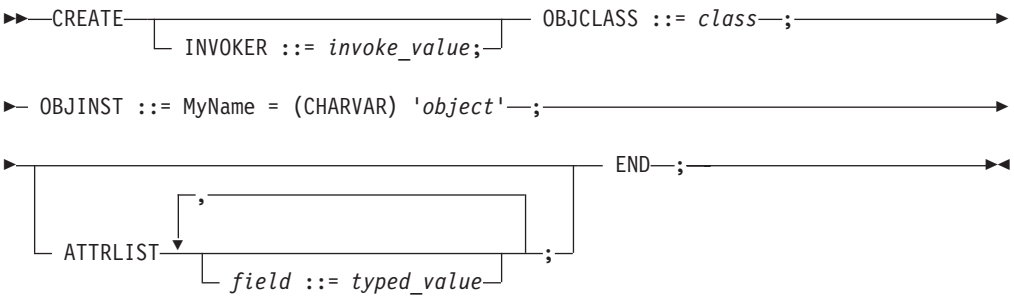
使用上の注意: フィールド定義リストの INIT または INITIAL キーワードに関連する *init_value* を指定するときは、次の規則を順守します。

- すべての値を括弧で囲む。
- 括弧内では文字値を単一引用符で囲む。
- すでに括弧によって結び付いている METHODSPEC および SELFDEFINING のようなデータ・タイプの値には、それ以上に括弧を追加しない。
- 括弧内では非ヌル GRAPHICVAR 値を、シフトアウトおよびシフトイン文字で囲む。
- 括弧内ではヌルの GRAPHICVAR 値を単一引用符で囲む。

CREATE:

目的: CREATE 高水準ロード機能ステートメントは、RODM データ・キャッシュに特定のクラスのオブジェクトを作成するときに使用します。

構文:



キーワードおよびパラメーターの記述:

INVOKER ::= *invoke_value*
ID の値。この値は、RODM ロード機能により無視されますが、定義ファイルの高水準ロード機能ステートメントを数えるときに使用することができます。

OBJCLASS ::= *class*
作成されるオブジェクトの親クラスの名前。

OBJINST ::= MyName = (CHARVAR) *object*
作成されるオブジェクトの名前。

field **::=** *typed_value*
field というフィールドを、値 *typed_value* に設定します。有効なデータ・タイプと値のリストについては、340 ページの『typed_value』を参照してください。

例: SDSF と呼ばれるシステム・ソフトウェアを表すオブジェクトを作成する場合に必要な指定を考えます。 SDSF は SystemSoftware というクラスの子であり、以下のフィールドと値があります。

- 値が SDSF の ProductName
- 値が 5697-B82 の ProgramNumber
- 値が UY12903 の LatestPTFNumber
- 値が PL45419 の CorrespondingAPARNumber
- 値が 03/01/97 の DateApplied
- このオブジェクトを HostA および HostC にリンクする UseInHost フィールド

  注: 正常にリンクするには、HostA および HostC が存在していなければなりません。

以下は、オブジェクト SDSF の作成に必要なステートメントです。

```
CREATE    INVOKER    ::= 0000003;
          OBJCLASS   ::= SystemSoftware;
          OBJINST    ::= MyName = (CHARVAR) 'SDSF';
          ATTRLIST
                     ProductName             ::= (CHARVAR)  'SDSF',
                     ProgramNumber           ::= (CHARVAR)  '5697-B82',
                     LatestPTFNumber         ::= (CHARVAR)  'UY12903',
                     CorrespondingAPARNumber ::= (CHARVAR)  'PL45419',
                     DateApplied             ::= (CHARVAR)  '03/01/97',
                     UseInHost               ::= (OBJECTLINKLIST)
                      ('Host_Class'.'HostA'.'UseSystemSoftware')
                      ('Host_Class'.'HostC'.'UseSystemSoftware');
END;
```

図 69. オブジェクト作成の例

使用上の注意: MyName フィールドは常にオブジェクトの名前を表しているので、CREATE 高水準ステートメントの OBJINST キーワードのパラメーターを指定するときは、フィールドの名前として通常 MyName を指定します。例えば、次のようになります。

```
OBJINST ::= MyName = (CHARVAR) 'SDSF';
```

しかし、オブジェクトのフィールドのもう一つにもオブジェクト名をその値としてもたせたい場合は、 OBJINST 定義で MyName に代えてそのフィールド名を指定します。その場合、MyName フィールドとそのフィールドには同じ値が割り当てられます。例えば、オブジェクトの MyName フィールドと ProductName フィールドの両方の値として SDSF のオブジェクト名を割り当てたい場合は、次のように指定します。

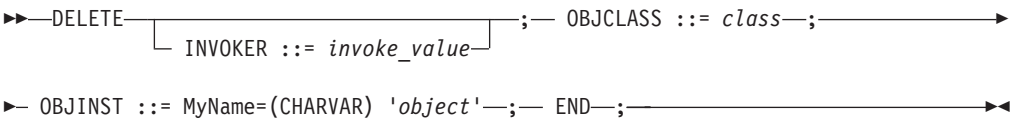
```
OBJINST ::= ProductName = (CHARVAR) 'SDSF';
```

ATTRLIST では、フィールドとして ProductName を繰り返してはなりません。

DELETE:

目的: 高水準ロード機能の DELETE ステートメントは、オブジェクトを RODM データ・キャッシュから削除するときに使用します。

構文:



キーワードおよびパラメーターの記述:

INVOKER ::= *invoke_value*
ID の値。この値は、RODM ロード機能により無視されますが、ロード機能入力ファイルの高水準ロード機能ステートメントを数えるときに使用することができます。

OBJCLASS ::= *class*
削除されるオブジェクトの親クラスの名前。

OBJINST ::= MyName = (CHARVAR) *object*
削除されるオブジェクトの名前。

例: 図 70 は、オブジェクトをデータ・モデルから削除する DELETE ステートメントです。

```
DELETE    INVOKER    ::= 0000005;
          OBJCLASS   ::= SystemSoftware;
          OBJINST    ::= MyName = (CHARVAR) 'SDSF';
END;
```

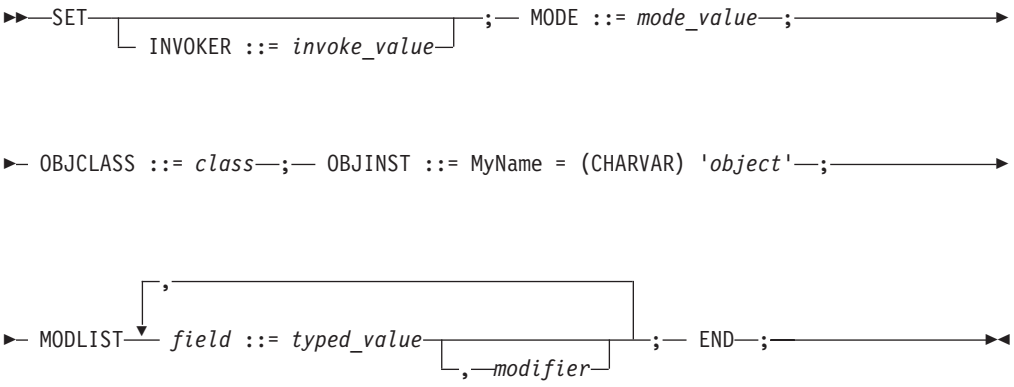
図 70. オブジェクト削除の例

削除されるオブジェクト *SDSF* は OBJINST キーワードのパラメーターとして指定され、オブジェクト *SystemSoftware* の親クラスは OBJCLASS キーワードのパラメーターとして指定されます。

SET:

目的: SET 高水準ロード機能ステートメントは、RODM データ・キャッシュにオブジェクト内のフィールドの値を設定するときに使用します。

構文:



キーワードおよびパラメーターの記述:

INVOKER ::= *invoke_value*
ID の値。この値は、RODM ロード機能により無視されますが、ロード機能入力ファイルの高水準ロード機能ステートメントを数えるときに使用することができます。

MODE ::= *mode_value*
この値は、RODM ロード機能により無視され、常に `non-confirmed` と見なされます。

OBJCLASS ::= *class*
フィールド値が設定されるオブジェクトの親クラスの名前。

OBJINST ::= MyName = (CHARVAR) *object*
フィールド値が設定されるオブジェクトの名前。

field **::=** *typed_value*
field という名前のフィールドが、値 *typed_value* に設定されます。有効なデータ・タイプと値のリストについては、340 ページの『typed_value』を参照してください。

modifier
このパラメーターは、変更のタイプを指定するときに使用します。 *modifier* のとりうる値は、次のとおりです。

値　　説明

ADD VALUE
OBJECTLINK または OBJECTLINKLIST のデータ・タイプの場合に限って、新しいリンクを作成する際に使用します。

REMOVE VALUE
OBJECTLINK または OBJECTLINKLIST のデータ・タイプの場合に限って、既存のリンクを削除する際に使用します。

REPLACE
OBJECTLINK または OBJECTLINKLIST 以外のデータ・タイプの場合に、指定のフィールドの value サブフィールドを新しい値に変更するときに使用します。

SET TO DEFAULT
OBJECTLINK または OBJECTLINKLIST 以外のデータ・タイプの場合に、指定のフィールドの value サブフィールドをデフォルト値に変更するときに使用します。デフォルト値は、親クラスのフィールドの値です。

データ・タイプが OBJECTLINK か OBJECTLINKLIST ならば、デフォルトは ADD VALUE です。他のすべてのデータ・タイプの場合のデフォルトは、REPLACE です。

END SET 高水準ロード機能ステートメントの終わりを識別する、必須キーワード。

例: SystemSoftware というクラスの子である SDSF オブジェクトの値を変更したい、 SET 高水準ロード機能ステートメントを考えます。特に、SDSF のフィールドに以下の変更を行う必要があります。

- ProductName フィールド値を `SDSF V2` に変更する。

- ProgramNumber フィールド値を 5697-B82 に変更する。
- LatestPTFNnumber フィールド値をデフォルト値に変更する。
- CorrespondingAPARNumber フィールド値をブランク・ストリングにリセットする。
- DateApplied フィールド値を 03/01/97 に変更する。
- Host_Class の HostA オブジェクトの UseSystemSoftware フィールドを UseInHost フィールドからリンク解除する。

SDSF オブジェクトのフィールドの値を設定するステートメントは、 図 71 で記載しています。

```
SET       INVOKER    ::= 0000004;
          MODE       ::= non-confirmed;
          OBJCLASS   ::= SystemSoftware;
          OBJINST    ::= MyName = (CHARVAR) 'SDSF';
          MODLIST
                   ProductName             ::= (CHARVAR) 'SDSF V2', REPLACE,
                   ProgramNumber           ::= (CHARVAR) '5697-B82',
                   LatestPTFNumber         ::= (CHARVAR), SET TO DEFAULT,
                   CorrespondingAPARNumber ::= (CHARVAR) ' ',
                   DateApplied             ::= (CHARVAR) '03/01/97',
                   UseInHost               ::= (OBJECTLINKLIST)
                           ('Host_Class'.'HostA'.'UseSystemSoftware'),REMOVE VALUE;
END;
```

図 71. オブジェクトのフィールドの値を設定する例

使用上の注意: OBJECTLINK および OBJECTLINKLIST フィールドの定義の場合、RODM ロード機能は、修正が ADD VALUE ならばリンクを作成し、修正が REMOVE VALUE ならばリンクを削除します。さらに、OBJECTLINK か OBJECTLINKLIST のデータ・タイプを指定するどのフィールドも、その値を括弧で囲みます。

RODM ロード機能プリミティブ・ステートメントをコーディングする

この参照セクションのトピックでは、RODM ロード機能プリミティブ・ステートメントのコーディング方法を説明します。ここでは、構文および処理ロジックを、関連する構文の規則とともに説明します。また、RODM ロード機能プリミティブのグローバル文字の使用法についても説明します。

構文は、構文図で表します。

グローバル文字

RODM プリミティブ・ステートメントの 1 つまたは複数の値を置換するときは、アスタリスク (*) を*グローバル文字* として使用することができます。各グローバル文字は、RODM プリミティブ・ステートメント内の名前、クラス、オブジェクト、フィールド、もしくはサブフィールドの各 1 つの代わりに使用されます。 プリミティブ・ステートメントが RODM 機能に変換されるとき、各グローバル文字は、名前、クラス、オブジェクト、フィールド、もしくはサブフィールドが明示的に指定されたそれまでのプリミティブからの対応する値に換えられます。しかし、グローバル文字はメソッド名の指定には使用することができません。

グローバル文字は、複数使用されるときは、同じ相対位置を用いて以前のプリミティブ・ステートメントからの値を置換します。例えば、次のようになります。

```
OP ClassA  HAS_PARENT    UniversalClass;
OP *       HAS_FIELD     (INTEGER) FieldA_Integer;
OP ClassB  HAS_PARENT    *;
OP *       HAS_FIELD     (CHARVAR) FieldB_CharVar;
```

2 番目のプリミティブ・ステートメントのグローバル文字は、最初のプリミティブからの *ClassA* に置き換わります。 3 番目のプリミティブ・ステートメントのグローバル文字は、最初のプリミティブからの *UniversalClass* に置き換わります。 4 番目のプリミティブ・ステートメントのグローバル文字は、 3 番目のプリミティブからの *ClassB* に置き換わります。最後に、5 番目のプリミティブ・ステートメントのグローバル文字は、それぞれ 3 番目および 4 番目のプリミティブからの *ClassB* および *FieldB_CharVar* に置き換わります。

グローバル文字は、RODM ロード機能プリミティブ・ステートメントを指定する際の簡単な方法となっています。 RODM 処理ロジックは、グローバル文字を使用しても変更されません。グローバル文字は、プリミティブ・ステートメントのグループ化を意味するものではありません。

ロード機能プリミティブの構文の規則

RODM 高水準ロード機能ステートメントの構文と同様、RODM ロード機能プリミティブの各部分も 1 つまたは複数のスペースで区切ることができます。

注: RODM ロード機能プリミティブの構文は大小文字を区別します。

入力列、引用符付きストリング、2 バイト文字ストリング、およびコメントに適用される構文の規則は、 RODM ロード機能プリミティブ構文の場合も RODM 高水準ロード機能構文に指定された場合も同じ規則です。 313 ページの『高水準ロード機能ステートメントの構文の規則』を参照してください。

ロード機能プリミティブの構文および処理ロジック

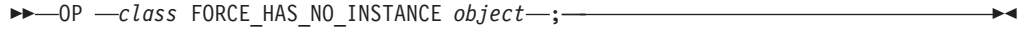
これは、RODM ロード機能プリミティブの構文および処理ロジックについての説明です。 RODM ロード機能プリミティブは、アルファベット順に説明され、RODM ロード機能プリミティブごとに、その名前、意味、外部構文、および実施論理を含む説明が付いています。

FORCE_HAS_NO_INSTANCE:

説明: FORCE_HAS_NO_INSTANCE によって、指定の名前の指定されたクラスの下にオブジェクトは存在しないようになります。オブジェクトへのリンクが存在すると、リンク解除されてから、オブジェクトそのものは削除されます。

このステートメントは、クラス・オブジェクトのすべてのリンクもしくはすべてのオブジェクトを削除する再試行が失敗した後に、オブジェクトを削除できない場合があります。

構文:



class の *object* が存在していれば、削除されます。

PARSE、LOAD および VERIFY の構文論理: 次の構文チェックを行います。
1. *class* が有効な RODM クラス名であるか検査する。
2. *object* が有効な RODM オブジェクト名であるか検査する。

LOAD 論理: 以下のことを行います。
1. *object* を *class* から削除する。
2. リンクされているためオブジェクトを削除できない場合:
   a. クラスの構造を照会する。
   b. すべてのリンク・フィールドを照会する。
   c. リンク付きフィールドごとに、リンクを削除する。
   d. オブジェクト削除要求を再試行する。

VERIFY 論理: *class* の *object* が、存在していないか検査します。

FORCE_NOT_A_CLASS:

説明: FORCE_NOT_A_CLASS によって、指定の名前のクラスは存在しないようになります。クラスのオブジェクトは、存在していれば削除されます。すなわち、オブジェクトへのリンクはすべて除かれ、オブジェクトそのものも削除され、クラスそのものも削除されることを意味します。

構文:

class が存在していれば、削除されます。

PARSE、LOAD および VERIFY の構文論理: *class* が有効な RODM クラス名であるか検査する。

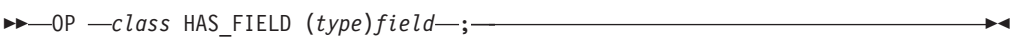
LOAD 論理: 以下のことを行います。
1. *class* を削除する。
2. 子があるためにクラスが削除できない場合は、子を削除し、削除要求を再試行する。
3. オブジェクトがあるためにクラスが削除できない場合は、オブジェクトを削除し、削除要求を再試行する。

VERIFY 論理: *class* が存在していないか検査します。

HAS_FIELD:

説明: HAS_FIELD によって、クラスは指定された共用フィールドを定義 するようになります。

構文:



class は、タイプ *type* の *field* という名前のフィールドをローカルに定義します。

PARSE、LOAD および VERIFY の構文論理: 次の構文チェックを行います。
1. *class* が有効な RODM クラス名であるか検査する。
2. *field* が有効な RODM フィールド名であるか検査する。
3. *type* が有効な RODM ロード機能データ・タイプであるか検査する。

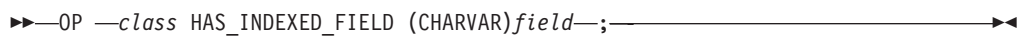
LOAD 論理: *class* が存在するか検査し、 *class* に *type* の *field* を作成します。

VERIFY 論理: *class* が存在し、*field* をローカルに定義し、このフィールドのタイプが *type* と同じであるか検査します。

HAS_INDEXED_FIELD:

説明: HAS_INDEXED_FIELD によって、クラスは指定された共用索引付きフィールドを定義するようになります。

構文:



class は、タイプ CHARVAR の *field* という名前のフィールドをローカルに定義します。

PARSE、LOAD および VERIFY の構文論理: 次の構文チェックを行います。
1. *class* が有効な RODM クラス名であるか検査する。
2. *field* が有効な RODM フィールド名であるか検査する。
3. CHARVAR が有効な RODM ロード機能データ・タイプであるか検査する。共用索引付きにできるのは、CHARVAR フィールドのみです。

LOAD 論理: *class* が存在するか検査し、 *class* に CHARVAR の *field* を作成します。

VERIFY 論理: *class* が存在し、*field* をローカルに定義し、このフィールドのタイプが CHARVAR であるか検査します。

HAS_INSTANCE:

説明: HAS_INSTANCE によって、指定されたクラスの特定のオブジェクトが存在するようになります。

構文:

class には、*object* という名前のオブジェクトがあります。

PARSE、LOAD および VERIFY の構文論理: 次の構文チェックを行います。
1. *class* が有効な RODM クラス名であるか検査する。
2. *object* が有効な RODM オブジェクト名であるか検査する。

LOAD 論理: *class* が存在するか検査し、 *class* の *object* を作成します。

VERIFY 論理: *class* が存在し、オブジェクト *object* をもっているか検査します。

HAS_NO_FIELD:

説明: HAS_NO_FIELD は、指定のフィールドを指定されたクラスから削除します。クラスまたはオブジェクトの子のあるクラスから、フィールドを削除することはできません。また、継承されたフィールドも削除できません。

構文:

field は、それが存在していて、かつクラスにオブジェクトの子がない場合は、*class* の定義から削除されます。

PARSE、LOAD および VERIFY の構文論理: 次の構文チェックを行います。
1. *class* が有効な RODM クラス名であるか検査する。
2. *field* が有効な RODM フィールド名であるか検査する。

LOAD 論理: *field* を *class* から削除します。

VERIFY 論理: *field* が、*class* によって定義されていないか検査します。

HAS_NO_INSTANCE:

説明: HAS_NO_INSTANCE によって、特定クラスの特定のオブジェクトは存在しないようになります。この指定の実行に使用される唯一の命令は、単なる削除です。

オブジェクトは、他のオブジェクトにリンクされている場合は、このプリミティブだけでは削除することはできません。この場合については、 323 ページの『FORCE_HAS_NO_INSTANCE』を参照してください。

構文:

class の *object* は、それが存在していて、かつ他のオブジェクトへのリンクをもたない場合は、削除されます。

PARSE、LOAD および VERIFY の構文論理: 次の構文チェックを行います。
1. *class* が有効な RODM クラス名であるか検査する。
2. *object* が有効な RODM オブジェクト名であるか検査する。

LOAD 論理: *object* を *class* から削除する。

VERIFY 論理: *object* が、*class* に存在していないか検査します。

HAS_NO_SUBFIELD:

説明: HAS_NO_SUBFIELD によって、指定のフィールドに特定のサブフィールドは存在しないようになります。オブジェクトのあるクラスからは、サブフィールドを削除することはできません。また、継承されたフィールドのサブフィールドも削除できません。

構文:

subfield は、それが存在していて、かつクラスにオブジェクトの子がない場合は、*class* の *field* から削除されます。

PARSE、LOAD および VERIFY の構文論理: 次の構文チェックを行います。
1. *class* が有効な RODM クラス名であるか検査する。
2. *field* が有効な RODM フィールド名であるか検査する。
3. *subfield* が有効な RODM サブフィールド名であるか検査する。

LOAD 論理: *subfield* を *class* の *field* から削除します。

VERIFY 論理: *subfield* が、*class* の *field* に定義されていないか検査します。

HAS_PARENT:

説明: HAS_PARENT によって、指定の親の下にクラスが存在するようになります。

構文:

Has_Parent

child_class は、*parent_class* の子でなければなりません。

PARSE、LOAD および VERIFY の構文論理: クラス名が、RODM のクラス名の規則に従っているか検査します。

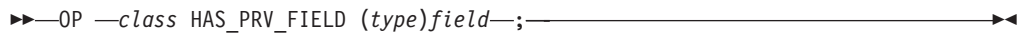
LOAD 論理: *child_class* を、*parent_class* の子として作成します。

VERIFY 論理: *child_class* および *parent_class* が存在しているか、かつ *child_class* の親フィールドが *parent_class* を指し示しているかの両方を検査します。

HAS_PRV_FIELD:

説明: HAS_PRV_FIELD によって、クラスは指定された専用フィールドを定義するようになります。

構文:



class は、タイプ *type* の *field* という名前のフィールドをローカルに定義します。

PARSE、LOAD および VERIFY の構文論理: 次の構文チェックを行います。
1. *class* が有効な RODM クラス名であるか検査する。
2. *field* が有効な RODM フィールド名であるか検査する。
3. *type* が有効な RODM ロード機能データ・タイプであるか検査する。

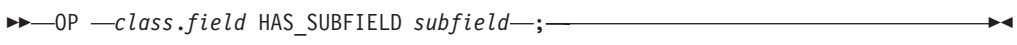
LOAD 論理: *class* が存在するか検査し、 *class* に *type* の *field* を作成します。

VERIFY 論理: *class* が存在し、それが *field* を専用と定義し、このフィールドのタイプが *type* と同じであるか検査します。

HAS_SUBFIELD:

説明: HAS_SUBFIELD によって、クラスのフィールドに指定されたサブフィールドが入るようになります。

構文:



class の *field* には、*subfield* があります。

PARSE、LOAD および VERIFY の構文論理: 次の構文チェックを行います。
1. *class* が有効な RODM クラス名であるか検査する。
2. *field* が有効な RODM フィールド名であるか検査する。
3. *subfield* が有効な RODM サブフィールド名であるか検査する。

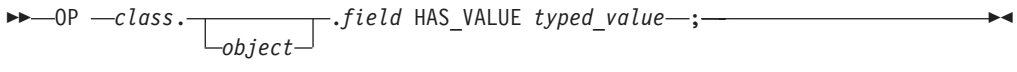
LOAD 論理: *class* が存在し、かつクラスに *field* が存在するか検査して、その *class* の *field* に *type* の *subfield* を作成します。

VERIFY 論理: *class* が存在し、それがローカルに *field* を定義し、このフィールドが *subfield* を定義しているか検査します。

HAS_VALUE:

説明: HAS_VALUE によって、特定のオブジェクトまたはクラスのフィールドに指定された値が入るようになります。

構文:



class の *object* の *field* には、値 *typed_value* が入ります。

class の *field* には、値 *typed_value* が入ります。

PARSE、LOAD および VERIFY の構文論理: 次の構文チェックを行います。
1. *class* が有効な RODM クラス名であるか検査する。
2. *object* が (指定されている場合)、有効な RODM オブジェクト名であるか検査する。
3. *field* が有効な RODM フィールド名であるか検査する。
4. *typed_value* が有効な RODM の型付きの値であるか検査する。

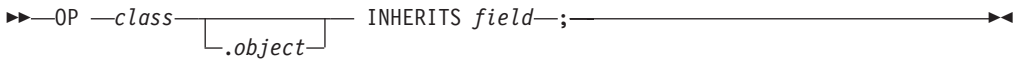
LOAD 論理: *class*、*object*、および *field* が存在しているか検査し、 *class.object* の *field* を *typed_value* によって指定されたタイプおよび値に設定するか、 *class* の *field* を *typed_value* によって指定されたタイプおよび値に設定します。

VERIFY 論理: *class.object* の *field* に *typed_value* によって指定されたタイプおよび値があるかを検査するか、 *class* の *field* に *typed_value* によって指定されたタイプおよび値があるかを検査します。

INHERITS:

説明: INHERITS によって、指定のオブジェクトまたはクラスの指定されたフィールドはローカルに定義されないようになります。

構文:



class の *object* の *field* は、その継承された値に戻されます。

class の *field* は、その継承された値に戻されます。

PARSE、LOAD および VERIFY の構文論理: 次の構文チェックを行います。

1. *class* が有効な RODM クラス名であるか検査する。
2. *object* が (指定されている場合)、有効な RODM オブジェクト名であるか検査する。
3. *field* が有効な RODM フィールド名であるか検査する。

LOAD 論理: *field* を戻します。ローカル値があれば、削除されます。

VERIFY 論理: *field* の値が継承されているか検査します。

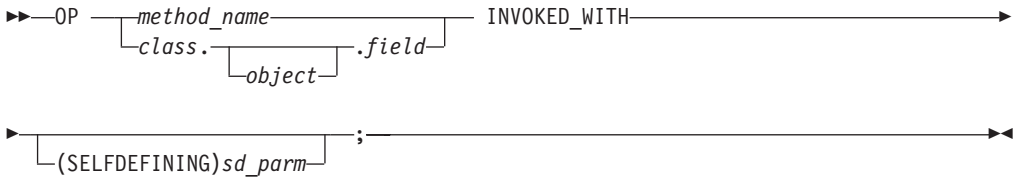
INVOKED_WITH:

説明: INVOKED_WITH は、名前付きのオブジェクト特有のメソッドもしくはオブジェクト独立メソッドを実行します。

sd_parm で最大 8 つのパラメーターを指定できます。

構文:

Invoked_With



class.object.field 名前付きのオブジェクト特有のメソッドは、 *sd_parm* パラメーターによって実行されます。

class..field 名前付きのオブジェクト特有のメソッドは、 *sd_parm* パラメーターによって実行されます。

method_name オブジェクト独立メソッドは、 *sd_parm* パラメーターによって実行されます。

PARSE、LOAD および VERIFY の構文論理: 次の構文チェックを行います。

名前付きのオブジェクト特有のメソッドの場合:

1. *class* が有効な RODM クラス名であるか検査する。
2. *object* が (指定されている場合)、有効な RODM オブジェクト名であるか検査する。
3. *field* が有効な RODM フィールド名であるか検査する。
4. *sd_parm* が有効な SELFDEFINING 値であるか検査する。

オブジェクト独立メソッドの場合:

1. *method_name* が有効な RODM メソッド名であるか検査する。
2. *sd_parm* が有効な SELFDEFINING 値であるか検査する。

LOAD 論理:

名前付きのオブジェクト特有のメソッドの場合は、*class.object.field* によって、あるいは *class..field* によって指定したメソッドを、*sd_parm* で指定されたパラメーターによって起動します。フィールドのデータ・タイプは MethodSpec でなければなりません。

オブジェクト独立メソッドの場合は、 *method_name* を *sd_parm* で指定されたパラメーターで起動します。 *method_name* は、EKG_Method クラスのオブジェクトの名前でなければなりません。

VERIFY 論理: なし。

IS_LINKED_TO:

説明: IS_LINKED_TO によって、2 つのオブジェクトが指定されたフィールドによりリンクされるようになります。フィールドは、タイプ OBJECTLINK または OBJECTLINKLIST のフィールドでなければなりません。

構文:

class_1.object_1 の *field_1* は、 *class_2.object_2* の *field_2* にリンクします。

PARSE、LOAD および VERIFY の構文論理: 次の構文チェックを行います。

1. *class_1* が有効な RODM クラス名であるか検査する。
2. *class_2* が有効な RODM クラス名であるか検査する。
3. *object_1* が有効な RODM オブジェクト名であるか検査する。
4. *object_2* が有効な RODM オブジェクト名であるか検査する。
5. *field_1* が有効な RODM フィールド名であるか検査する。
6. *field_2* が有効な RODM フィールド名であるか検査する。

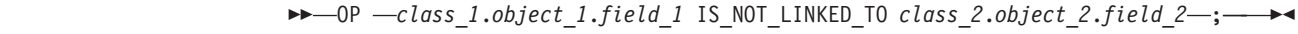
LOAD 論理: *class_1.object_1* の *field_1* は、 *class_2.object_2* の *field_2* にリンクします。

VERIFY 論理: *class_1.object_1* の *field_1* を照会し、 *class_2.object_2* の *field_2* が、照会によって戻されたリンク・フィールドのリスト内にあるか検査します。

IS_NOT_LINKED_TO:

説明: IS_NOT_LINKED_TO によって、2 つのオブジェクトは指定されたフィールドによりリンクされないようになります。

構文:



class_1.object_1 の *field_1* は、 *class_2.object_2* の *field_2* にリンクしません。

PARSE、LOAD および VERIFY の構文論理: 次の構文チェックを行います。
1. *class_1* が有効な RODM クラス名であるか検査する。
2. *class_2* が有効な RODM クラス名であるか検査する。
3. *object_1* が有効な RODM オブジェクト名であるか検査する。
4. *object_2* が有効な RODM オブジェクト名であるか検査する。
5. *field_1* が有効な RODM フィールド名であるか検査する。
6. *field_2* が有効な RODM フィールド名であるか検査する。

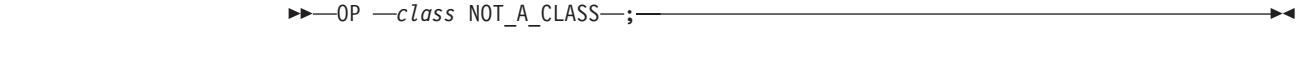
LOAD 論理: *class_1.object_1* の *field_1* の *class_2.object_2* の *field_2* へのリンクを解除します。

VERIFY 論理: *class_1.object_1* の *field_1* を照会し、 *class_2.object_2* の *field_2* が、照会によって戻されたリンク・フィールドのリストにないか検査します。

NOT_A_CLASS:

説明: NOT_A_CLASS によって、指定の名前のクラスは存在しなくなります。この指定の実行に使用される唯一の命令は、単なる削除です。クラスは、オブジェクトがある場合は、このプリミティブだけを削除することはできません。代わりに、FORCE_NOT_A_CLASS を使用するか、オブジェクトを最初に削除しなければなりません。

構文:



class は、存在していて、オブジェクトも子もなければ削除されます。

PARSE、LOAD および VERIFY の構文論理: *class* が有効な RODM クラス名であるか検査する。

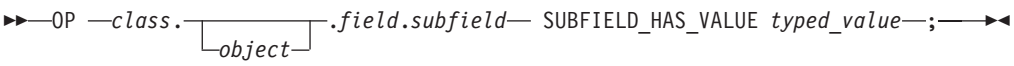
LOAD 論理: *class* を削除する。

VERIFY 論理: *class* が存在していないか検査します。

SUBFIELD_HAS_VALUE:

説明: SUBFIELD_HAS_VALUE によって、サブフィールドに指定された値が入るようになります。

構文:



class の *object* の *field* の *subfield* には、値 *typed_value* が入ります。

class の *field* の *subfield* には、値 *typed_value* が入ります。

PARSE、LOAD および VERIFY の構文論理: 次の構文チェックを行います。
1. *class* が有効な RODM クラス名であるか検査する。
2. *object* が (指定されている場合)、有効な RODM オブジェクト名であるか検査する。
3. *field* が有効な RODM フィールド名であるか検査する。
4. *subfield* が有効な RODM サブフィールド名であるか検査する。
5. *typed_value* が有効な RODM の型付きの値であるか検査する。

LOAD 論理: *class* の *field* の *subfield* を *typed_value* のタイプおよび値に設定するか、 *class.object* の *field* の *subfield* を *typed_value* のタイプおよび値に設定します。

VERIFY 論理: *class* の *field* の *subfield* に *typed_value* のタイプおよび値が入っているかを検査するか、 *class.object* の *field* の *subfield* に *typed_value* のタイプおよび値が入っているかを検査します。

SUBFIELD_INHERITS:

説明: SUBFIELD_INHERITS によって、指定されたオブジェクトまたはクラスの特定のサブフィールドはローカルに定義されないようになります。

構文:

subfield_name は、その継承された値に戻されます。ローカル値があれば、削除されます。

PARSE、LOAD および VERIFY の構文論理: 次の構文チェックを行います。
1. *class* が有効な RODM クラス名であるか検査する。
2. *object* が (指定されている場合)、有効な RODM オブジェクト名であるか検査する。
3. *field* が有効な RODM フィールド名であるか検査する。
4. *subfield* が有効な RODM サブフィールド名であるか検査する。

LOAD 論理: *subfield_name* を戻します。

VERIFY 論理: *subfield_name* の値が継承されているか検査します。

共通構文エレメント

RODM ロード機能プリミティブおよび RODM 高水準ロード機能ステートメントは、クラス名である *class* のような共通構文エレメントを使用します。これらの単純な共通エレメントについては、共通テキストと数字ストリングの説明とともにここで説明します。

これらのエレメントや文字ストリングは、構文図を用いて説明します。

共通構文エレメントの構文

以降に、RODM ロード機能の各共通構文エレメントの説明を行います。

chars:

目的: 文字ストリング。1 つまたは複数の印刷可能な単一バイト文字でも、2 バイト文字でもかまいません。

形式:

Chars

使用上の注意: 2 バイト文字ストリングの場合は、前にシフトアウト文字が付き、終わりはシフトイン文字でなければなりません。

char_literal:

目的: 単一引用符内の文字ストリング。

形式:

Char_Literal

使用上の注意: *char_literal* 内の単一引用符文字 (') を示すには、2 つの単一引用符を続けて記述します。 2 つの単一引用符間にスペースを入れたり、改行したりしないでください。これは、従来からの*二重引用符* の規則です。

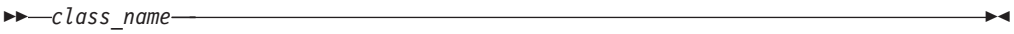
char_literal プリミティブは、複数行にまたがって入力できます。この場合、各行の該当部分を単一引用符内に囲みます。

class:

目的: 有効な RODM クラス名。

形式:

class



使用上の注意: クラス名に非英数字文字が入る場合は、そのクラス名を単一引用符で囲みます。

class_list:

目的: RODM クラス名のリスト。コンマで区切ります。

形式:

class_list

classlink_list:

目的: コンマで区切られたクラス・リンクのリスト。各クラス・リンクは、クラス名、ピリオド、およびフィールド名の連結です。

形式:

classlink_list

dbcs_literal:

目的: シフトアウト文字、1 つまたは複数の有効な 2 バイト文字、およびシフトイン文字の連結。

形式:

DBCS_Literal

パラメーターの説明:

shift-out_char
値 X'0E'。

double-byte_char
印刷可能な 1 文字を表す 4 つの 16 進文字 (2 バイト)。

shift-in_char
値 X'0F'。

使用上の注意: 2 バイト・テキストは、シフトアウトで始まり、シフトインで終わらなければなりません。テキストが複数行にまたがる場合は、各行の 2 バイト・テキストがシフトアウトとシフトインの対の中になければなりません。有効な 2 バイト文字は、GraphicVar データ・タイプの 2 バイト文字と同じです。 264 ページの『GraphicVar』を参照してください。

digits:

目的: 次の 10 進数字の任意の連結。0、1、2、3、4、5、6、7、8、または 9。

形式:

Digits

field:

目的: 有効な RODM フィールド名。

形式:

field

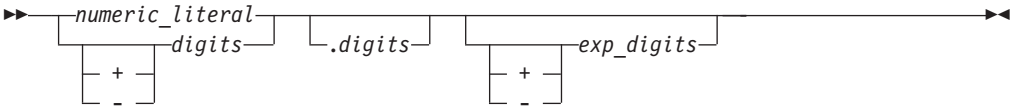
使用上の注意: フィールド名に非英数字文字が入る場合は、そのフィールド名を単一引用符で囲みます。

float_constant:

目的: 浮動小数点定数は、数値リテラル、オプションの 10 進小数部、およびオプションの符号付き浮動小数点指数桁の連結です。

形式:

Float_Constant

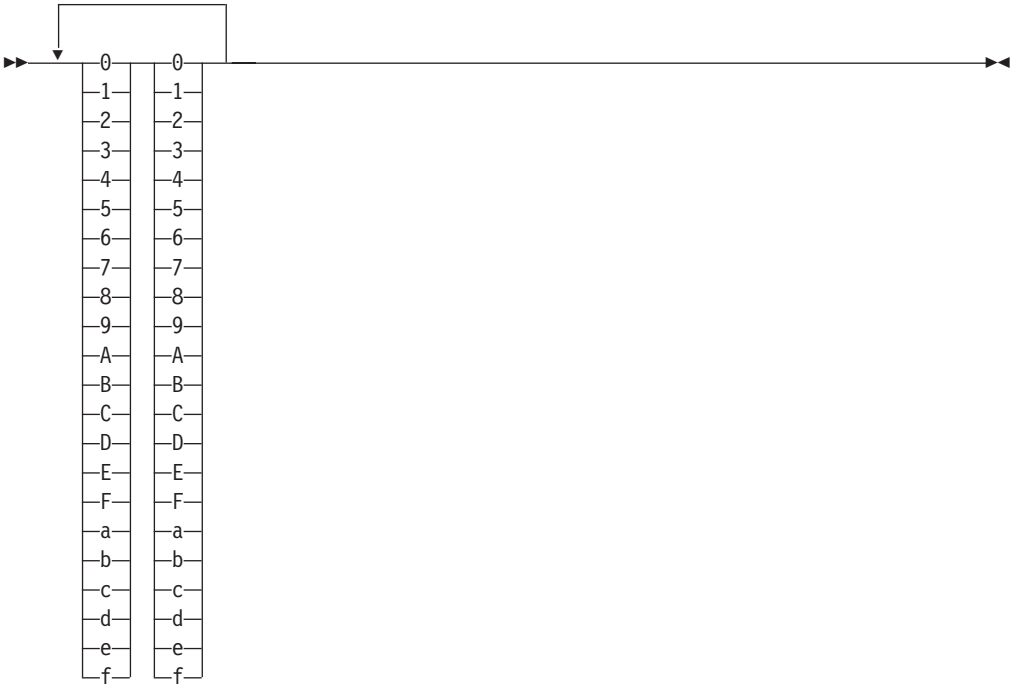


hex_chars:

目的: 16 進文字の対の連結。この場合、各対は 1 バイトを表します。

形式:

Hex_Chars



hex_literal:

目的: 16 進文字の 1 つまたは複数の対。16 進の区切り文字の間にあります。

形式:

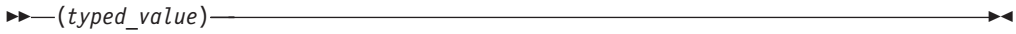
Hex_Literal

il_parm:

目的: INDEXLIST パラメーターは、タイプ指定値のリストです。各タイプ指定値は、ANONYMOUSVAR データ・タイプ値であっても、CHARVAR データ・タイプ値であってもかまいません。しかし、CHARVAR 値は、RODM ロード機能によって ANONYMOUSVAR 値に変換されます。

形式:

ll_Parm

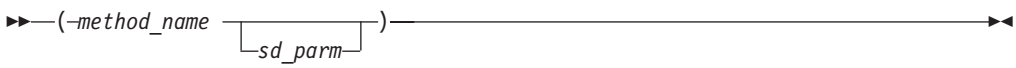


method_spec:

目的: メソッド指定は、括弧内のメソッド名と SELFDEFINING パラメーターの連結です。

形式:

method_spec



numeric_literal:

目的: 数字の符号付きストリング。

形式:

Numeric

object:

目的: 有効な RODM オブジェクト名。

形式:

object

使用上の注意: オブジェクト名に非英数字文字が入る場合は、そのオブジェクト名を単一引用符で囲みます。

objectid_list:

目的: コンマで区切られたオブジェクト ID のリスト。オブジェクト ID は、クラス名、ピリオド、およびオブジェクト名の連結です。

形式:

objectid_list

objectlink_list:

目的: *objectlink_list* は、スペースで区切られたオブジェクト・リンクのリストです。オブジェクト・リンクは、括弧内のクラス名、ピリオド、オブジェクト名、ピリオド、およびフィールド名の連結です。

形式:

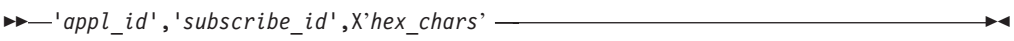
objectlink_list

recipient_spec:

目的: 受信側指定は 2 つの文字リテラルと 1 つのリテラルを連結したもので、これらのリテラルの長さは 3 つとも正確に 8 バイトでなければなりません。

形式:

recipient_spec



使用上の注意: 最初の文字リテラルは *application_id* です。 2 番目の文字リテラルは *subscribe_id* です。どちらかの文字リテラルの長さが 8 バイト未満であると、リテラルは RODM ロード機能によって左寄せされ、長さが 8 バイトになるように右側をブランクで埋め込まれます。 16 進データの長さを 8 バイトにするには、16 進数が 16 桁なければなりません。

sd_parm:

目的: SELFDEFINING パラメーターは型付きの値のリストです。括弧内で、任意にブランクで区切られます。

形式:

sd_parm

subfield:

目的: 事前定義のサブフィールド名。

形式:

subfield

使用上の注意: 以下は、サブフィールド名の定義です。

CHANGE
変更メソッドのメソッド指定

NOTIFY
通知申請を表す申請指定のリスト

PREV_VALUE
フィールドの以前の値

QUERY
照会メソッドのメソッド指定

TIMESTAMP
フィールドに対する最後の変更のタイム・スタンプ

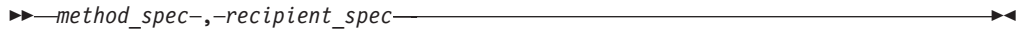
VALUE
フィールドの値

subs_spec:

目的: *subs_spec* は、通知申請指定で、メソッド指定とそれに続く受信側指定から構成され、コンマで区切られます。

形式:

subs_spec



subs_spec_list:

目的: *subs_spec_list* は申請指定のリストです。

形式:

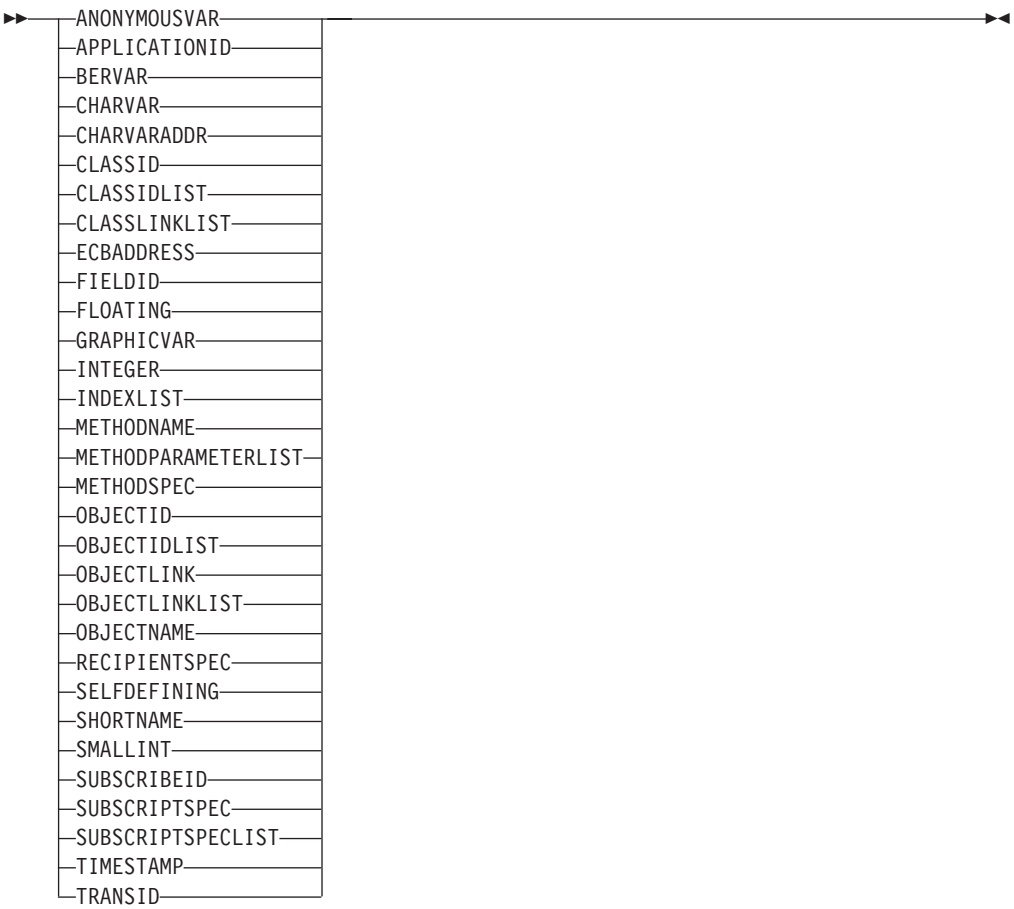
sub_spec&list

type:

目的: 定義済みデータ・タイプ・キーワード。

形式:

type



注:

1. これらのデータ・タイプは、SELFDEFINING データ内でのみ有効です。

APPLICATIONID	CHARVARADDR	CLASSIDLIST
CLASSLINKLIST	ECBADDRESS	METHODNAME
METHODPARAMETERLIST	OBJECTIDLIST	OBJECTNAME
RECIPIENTSPEC	SHORTNAME	SUBSCRIBEID
SUBSCRIPTSPEC	SUBSCRIPTSPECLIST	TRANSID

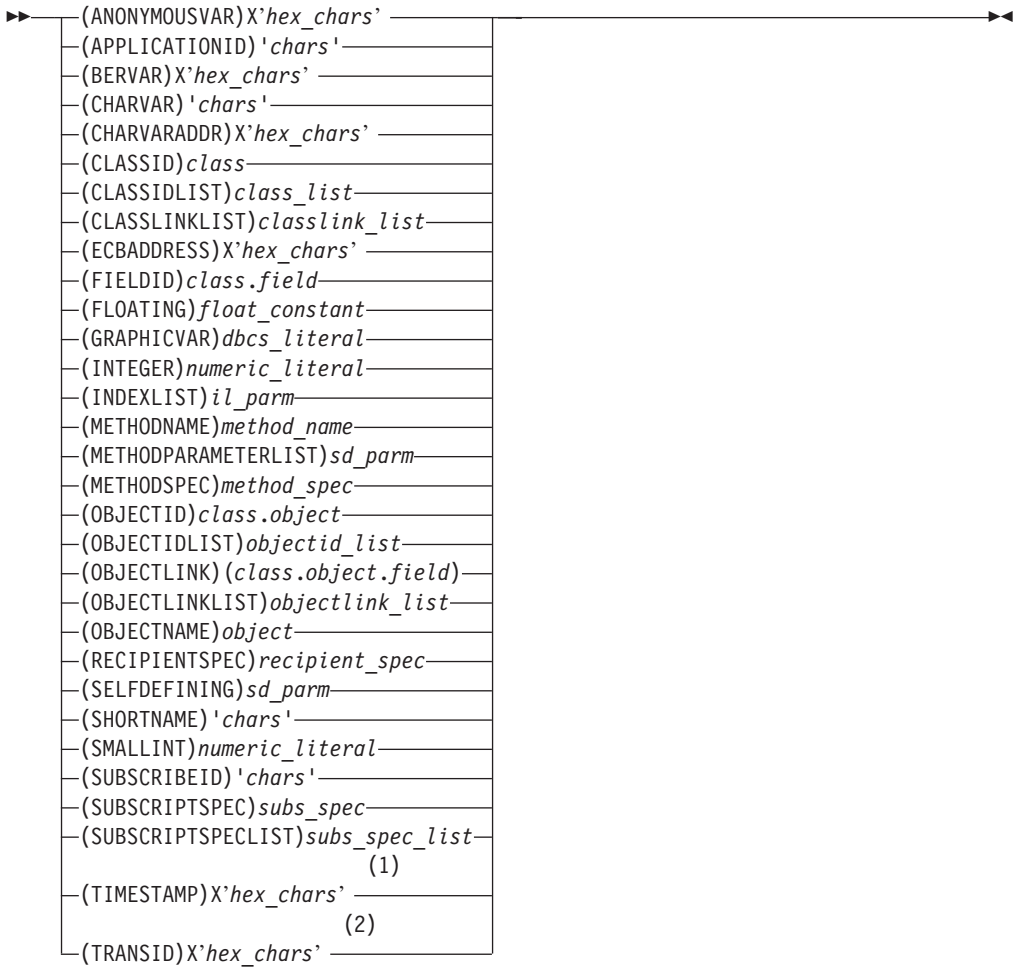
2. CLASSID および OBJECTID における制限については、297 ページの『CLASSID および OBJECTID データ・タイプを使用する』を参照してください。

typed_value:

目的: *typed_value* は、左括弧、タイプ・キーワード、右括弧、およびタイプ・キーワードのデータ・タイプに一致する値の連結です。

形式:

typed_value



注:

1 TIMESTAMP は、ちょうど 8 バイトでなければなりません。

2 TRANSID は、ちょうど 8 バイトでなければなりません。

使用上の注意: データ・タイプによっては、ヌル値を指定することができます。
298 ページの『RODM ロード機能データ・タイプのヌル値』を参照してください。

第 11 章 RODM を使用するアプリケーションを作成する

RODM には、ユーザー・アプリケーション・プログラム・インターフェース (ユーザー API) が備わっています。このユーザー API を使用すると、適切な許可を得たアドレス・スペースから、RODM のアドレス・スペースおよびデータ・スペースに含まれているデータにアクセスできます。このユーザー API を介して、オブジェクトを作成したり、階層に編成したり、削除したりすることができます。このユーザー API を使用することにより、オブジェクトに関連付けられているフィールドの値を照会したり、そのフィールド内の値を変更したりすることもできます。このユーザー API は NetView コマンド処理プログラム、および RODM のパラメーター引き渡し規則に従うプログラム言語で書かれたアプリケーションから呼び出すことができます。 RODM によって PL/I および C の制御ブロック・マッピングが提供されますが、ユーザーは、 344 ページの『レジスター規定』に記載されているどのようなインターフェースを使用するプログラム言語でもアプリケーションを作成できます。

RODM には、メソッド API も備わっています。この API は、多くの機能をユーザー API と共用しています。メソッド API については、 389 ページの『第 13 章 RODM メソッドの作成』で説明します。

NetView プログラムは、一連の汎用メソッドを提供します。これらのメソッドについては、 548 ページの『NetView 提供のメソッド』を参照してください。

ユーザー・アプリケーションによって最良のパフォーマンスが得られるタスク

このセクションでは、どのようなタスクがユーザー・アプリケーションで最良のパフォーマンスを得ることができるのかを説明します。

アプリケーション・プログラムは、以下のことを行うために使用してください。

- リソースの状況変更を RODM データ・キャッシュに知らせる。

  RODM データ・キャッシュは、実世界リソースのモデルとして表示されます。したがって、データ・キャッシュにあるリソース・オブジェクトが、必ず実際のリソース変更状況として更新されるようにしてください。
- データ変更の通知について申請する。

  ユーザー・アプリケーション・プログラムが RODM データ・キャッシュ変更の通知を受け取るには、関係のあるオブジェクトまたはクラスの必須フィールドへの通知申請が必要です。
- データ変更の通知を待って処理する。

  申請先オブジェクトまたはクラスからの通知を待ってそれを処理することは、ユーザー・アプリケーションの役割です。
- オペレーター視点、表示、および照会についてのデータを照会する。

ユーザー・インターフェースを介してユーザーと通信し、 RODM データ・キャッシュのデータへアクセスする必要があるアプリケーション・プログラムは、RODM を介してそのデータを照会しなければなりません。

- リソースを追加または削除する。

  データ・キャッシュの階層修正を必要とするアプリケーション・プログラムは、オブジェクトおよびクラスを操作する RODM を呼び出すことで、これを行うことができます。

- NetView アプリケーションと通信する。

  NetView アプリケーションは、ユーザー API を介して RODM データの照会および変更を行うことができます。 RODM データの照会および変更は、RODMView を使用して行うことも、マルチシステム・マネージャー・アクセス機能を使用して行うこともできます。

ユーザー・アプリケーション・プログラム・インターフェースの使用

RODM へのユーザー API 呼び出しでは、次の 4 つのパラメーターをモジュール EKGUAPI に渡さなければなりません。

- アクセス・ブロック
- トランザクション情報ブロック
- 機能ブロック
- 応答ブロック

機能ブロックは、追加パラメーター (例えば、機能のターゲットを識別するエンティティー・アクセス情報ブロックやフィールド・アクセス情報ブロック) を指すことができます。

図 72 は、典型的なユーザー API 呼び出しを、初めは C で、その次は PL/I で示しています。

```
#include <EKG3CEEP.H>                    /* EKGUAPI declaration for C    /*
EKGUAPI( &access_block,                    /* address of access block      /*
         &transaction_info_block,          /* address of trans info block  /*
         &function_block,                  /* address of function block    /*
         &response_block);                 /* address of response block    /*


%include syslib (EKG1IEEP);                /* EKGUAPI declaration for PL/I /*
CALL EKGUAPI( access_block,                /* access block                 /*
              transaction_info_block,      /* transaction info block       /*
              function_block,              /* function block               /*
              response_block);             /* response block               /*
```

図 72. C および PL/I での典型的なユーザー API 呼び出し

この呼び出しステートメントは、EKGUAPI で示されているコード・セグメントに制御権を移動します。 ユーザーは、アプリケーションのリンク・エディット時に EKGUAPI モジュールを組み込むことができます。

レジスター規定

生成されるコードは、以下の規定に従っていなければなりません。

レジスター 1
access_block、transaction_info_block、function_block、および response_block の各アドレスを示す 4 項目のパラメーター・リストを指します。制御ブロックは 348 ページの図 73 に示されています。

レジスター 13
呼び出し元プログラムの 72 バイト保存域のアドレスが入ります。

レジスター 14
呼び出し元プログラムの戻りアドレスが入ります。

レジスター 15
EKGUAPI モジュールの入り口アドレスが入ります。

使用上の注意

本書では、ヌル・ポインター という用語を使用しています。ヌル・ポインターの値は X'00000000' と定義されています。 PL/I を使用している場合には、この値は組み込み関数 SYSNULL によって提供されます。組み込み関数 NULL は使用しないでください。この関数は X'FF000000' という値を生成します。

コンパイラーによってパラメーター・リストが作成され、ヌルの response_block 値を引き渡せないような、高水準言語からこの呼び出しが行われた場合には、ダミー response_block を指定する必要があります。このダミー response_block は、正しい形式になっている必要があり、長さが少なくとも 8 桁になっていなければなりません。応答ブロックの追加情報については、 359 ページの『応答ブロック』を参照してください。

ユーザー API 呼び出しは同期的に行われます。 EKG_ExecuteFunctionList 機能には、実行される他の機能のリストを指定することができます。機能のリストに、同じオブジェクトを対象とする 2 つの機能が隣接して含まれている場合、そのオブジェクトのロックは、2 つの機能の処理間隔中には解放されません。

RODM アプリケーションは、EKGUAPI の呼び出し時にはキー 8 で実行されている必要があります。 RODM に渡されるすべてのパラメーター・リスト、制御ブロック、その他のデータ域は、キー 8 でアクセス可能なストレージに入っている必要があります。

コンパイルおよびリンク・エディット

アプリケーションは、リンク・エディット・ステップで EKGUAPI モジュールをリンク・エディットすることも、実行時にこのモジュールを動的にロードすることもできます。

EKGUAPI を呼び出す C モジュールのコンパイル

このモジュール内でなんらかの RODM 制御ブロックが参照される場合には、ソース・ファイルにファイル EKG3CINC.H を組み込んでください。このファイルには、RODM のすべての機能と応答ブロック、および RODM 入り口点 EKGUAPI、EKGMAPI、および EKGWAIT の機能プロトタイプ・ステートメントが含まれています。

モジュールで参照される RODM 制御ブロックがなくても、モジュールが EKGUAPI または EKGWAIT を呼び出す場合は、ソース・ファイルにファイル EKG3CEEP.H を組み込んでください。

例:

```
#include "EKG3CINC.H"
  /* or */
#include "EKG3CEEP.H"

void thisproc (void arg)
{
   /* code */
}
```

EKGUAPI を呼び出す PL/I モジュールのコンパイル

このモジュール内でなんらかの RODM 制御ブロックが参照される場合には、ソース・ファイルにファイル EKG1IINC を組み込んでください。このファイルには、RODM のすべての機能と応答ブロック、および RODM 入り口点 EKGUAPI、EKGMAPI、および EKGWAIT の機能プロトタイプ・ステートメントが含まれています。

モジュールで参照される RODM 制御ブロックがなくても、モジュールが EKGUAPI または EKGWAIT を呼び出す場合は、ソース・ファイルにファイル EKG1IEEP を組み込んでください。

ユーザー・アプリケーションに RODM マクロを組み込む場合は、例えば次のように、 MACRO プリプロセッサー・コンパイラー・オプションを指定してください。

```
*PROCESS MACRO;
 thisproc: proc;

%include ekglib(EKG1IINC);
    or
%include ekglib(EKG1IEEP);

 /* code */

end thisproc;
```

EKGUAPI を直接呼び出すモジュールのリンク

ソース・ファイル内の ENTRY ステートメントの前に、 INCLUDE SYSLIB(EKGUAPI) リンク・エディット制御ステートメントを指定する必要があります。

AMODE=31 リンク・エディット・オプションは、必ず指定します。

RMODE=ANY または RMODE=24 リンク・エディット・オプションは、必ず指定します。

次の ENTRY CEESTART ステートメントを指定する必要があります。

```
<module code>

INCLUDE SYSLIB(EKGUAPI)
ENTRY CEESTART
NAME module_name(R)
```

EKGUAPI をロードしてから呼び出すモジュールのリンク

EKGUAPI はロード・モジュールであるため、EKGUAPI をロードしてから呼び出すモジュールには、特別なリンク・エディット制御ステートメントを指定する必要はありません。ただし、EKGUAPI ロード・モジュールは (STEPLIB、JOBLIB、または z/OS リンク・リストを介して)、それをロードするモジュールにアクセスできなければなりません。

制御ブロックの使用

RODM へのすべてのユーザー API 呼び出しでは、 348 ページの図 73 に示すように 4 つのパラメーターを渡します。この図は、ユーザー API 呼び出しと RODM 照会機能要求の制御ブロックとの関係を例示しています。制御ブロックの関係は、ユーザー・アプリケーションから行われる他の RODM 機能要求の場合と似ています。

渡されるパラメーターは、次の制御ブロックを指すポインターです。

アクセス・ブロック
ユーザー API 要求を処理するために必要なユーザー情報が入っています。

トランザクション情報ブロック
API 要求についてのトランザクション情報および状況が入っています。

機能ブロック
RODM データに関して要求されたトランザクションの詳細が入っています。この制御ブロックの内容は、要求されたトランザクションによって異なります。要求されたトランザクションによっては、次の 2 つの情報ブロックを指すポインターが入ることがあります。
エンティティー・アクセス情報ブロック
フィールド・アクセス情報ブロック

応答ブロック
要求されたトランザクションからの出力データが入っています。応答ブロックの形式および特定内容は、要求されたトランザクションのタイプによって異なります。

348 ページの図 73 では、渡される 4 つの制御ブロックおよび関連する 2 つのアクセス情報ブロックが、PL/I に類似した構文で記述されています。同等に編成されたブロックを C でも表現することができます。制御ブロックにおける実際の順序およびオフセット位置は、以下の各制御ブロックの説明で参照されている表の中で指定されています。

図 73. API 照会機能制御ブロックの例

アクセス・ブロック

説明

アクセス・ブロックには、RODM がユーザー API 要求を処理するために必要なユーザー情報が入っています。

機能ブロックの形式

349 ページの表 31 は、アクセス・ブロックの形式を示しています。表ヘッディングの意味は次のとおりです。

オフセット
パラメーターの始まりまでのオフセットを 10 進バイト数で指定します。

長さ パラメーターの長さを 10 進バイト数で指定します。

タイプ パラメーターの RODM データ・タイプを指定します。詳細については、257 ページの『要約データ・タイプ参照』を参照してください。

用途 そのパラメーターが機能へのデータ入力に使用されるのか、機能によるデータ出力に使用されるのかを示しています。

パラメーター名
パラメーターの名前を指定します。

表 31. RODM アクセス・ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	8	character(8)	入力	RODM_name
008	16	Anonymous(16)	入出力	Sign_on_token
024	8	ApplicationID	入力	User_appl_ID

機能ブロック・フィールドの説明

RODM_name
接続するためにこの要求を受け取る RODM の名前は、呼び出し元が RODM_name フィールドに指定しなければなりません。アクセス・ブロックは通常は後続の呼び出しで再利用されるため、ユーザーが RODM_name フィールドを設定するのは、接続要求を出す直前の 1 回だけです。これは、RODM を開始したときに指定した名前です。 RODM 名を判別するには、NetView のオンライン・ヘルプを参照してください。

Sign_on_token
ユーザーを個別に識別するために RODM が使用するトークンです。接続完了時に RODM が設定するデータ構造は、sign_on_token パラメーターに入れて戻されます。

sign_on_token は、ユーザーが RODM に接続するたびに RODM によって設定されます。

User_appl_ID
ユーザー・アプリケーション・プログラムがそれ自体の識別を示すために指定する ID。 APF (許可プログラム機能) 許可プログラムの場合、 User_appl_ID 単体でユーザーが RODM に識別され、そのユーザーの能力が決まります。 APF 許可されていないアプリケーション・プログラムの場合、 User_appl_ID が接続機能ブロックから得られたパスワードと結合されて、ユーザーが RODM によって識別され、そのユーザーの能力が決まります。このフィールドの最大長は 8 バイトで、それよりも短い値は左寄せされ、右側にブランクが埋め込まれます。このストリングに指定できる値は、オブジェクト名の場合と同じです。

例

PL/I および C のサンプル制御ブロックが RODM とともに提供されています。これらの制御ブロックをプログラムに組み込んでください。

表 32. アクセス・ブロックのサンプル名

例	名前
PL/I アクセス・ブロック	EKG1ACCB
C アクセス・ブロック	EKG3ACCB

使用法

Connect 要求の後で出されたユーザー API 呼び出しを RODM が正常に完了させるためには、アクセス・ブロックを完全に初期化する必要があります。 RODM ユー

ザー・インターフェース (EKGUAPI) に対する呼び出しを行うたびに、アクセス制御ブロックを参照または定義しなければなりません。

複数のアプリケーションが RODM データ・キャッシュに同時にアクセスして、各アプリケーションの機能に適したメソッドを起動することができます。アクセス・ブロックの sign_on_token フィールドは、各トランザクションのユーザーを識別するために使用されます。

RODM はユーザー・アプリケーションの許可レベルを検査します。各 RODM 機能には、特定の許可レベルが必要です。

呼び出し元がアクセス・ブロック内で設定するフィールドは、 RODM_name フィールドと User_appl_ID フィールドです。これらのフィールドは、該当のユーザー API が呼び出される直前に、アプリケーションによって 1 回だけ設定されます。EKG_Connect ユーザー API は、sign_on_token フィールドに値を記入します。アクセス・ブロックが接続要求によって確立された後では、アプリケーションでそのブロック内の情報を修正しないようにしてください。

RODM への接続についての詳細は、374 ページの『RODM への接続』で説明しています。

トランザクション情報ブロック

説明

トランザクション情報ブロックには、各 API 要求に関するトランザクション状況情報が入っています。このトランザクション情報ブロックは、RODM 機能要求ごとに必要とされます。

機能ブロックの形式

表 33 は、トランザクション情報ブロックの形式を示しています。表ヘッディングの意味は次のとおりです。

オフセット
パラメーターの始まりまでのオフセットを 10 進バイト数で指定します。

長さ パラメーターの長さを 10 進バイト数で指定します。

タイプ パラメーターの RODM データ・タイプを指定します。

用途 そのパラメーターが機能へのデータ入力に使用されるのか、機能によるデータ出力に使用されるのかを示しています。ダッシュ (−) は、そのパラメーターが機能で使用されることがないこと、または予約されていることを示しています。

パラメーター名
パラメーターの名前を指定します。

表 33. RODM トランザクション情報ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	API_version
004	4	Integer	入力	EPL_blk_len
008	8	TransID	出力	Transaction_ID

表 33. RODM トランザクション情報ブロック (続き)

オフセット	長さ	タイプ	用途	パラメーター名
016	4	Integer	出力	Return_code
020	4	Integer	出力	Reason_code
024	0	Structure	—	EPL_info
024	4	Integer	入力	Lock_level

機能ブロック・フィールドの説明

API_version

API_version フィールドには、API 要求を処理するために RODM が使用する API のバージョンを指定します。このフィールドに指定できる値は、次のとおりです。

0 RODM は API の最新バージョンを使用します。

1 RODM は API のバージョン 1 を使用します。

EPL_blk_len

使用されませんが、互換性のために維持されています。

Transaction_ID

ユーザー・アプリケーションによって開始される各トランザクションには、RODM によって固有のトランザクション ID が割り当てられます。ユーザー・アプリケーション・トランザクションによって起動される同期メソッド・トランザクションのトランザクション ID は、ユーザー・アプリケーションのトランザクション ID と同じです。 transaction_ID フィールドの内容は、他のすべてのトランザクションに対するこのトランザクションの相対的な順序を表しています。このトランザクション ID は、チェックポイント間で RODM に対して行われたすべてのトランザクションのジャーナリングでも使用されます。トランザクションのチェックポイント操作については、本書のチェックポイント通知の登録に関するセクションで説明されています。 440 ページの『チェックポイント制御のコーディング』を参照してください。

Return_code

RODM から戻される戻りコードです。戻りコードのリストについては、 515 ページの『RODM の戻りコードと理由コード』を参照してください。

Reason_code

RODM から戻される理由コードです。理由コードのリストについては、 515 ページの『RODM の戻りコードと理由コード』を参照してください。

EPL_info

使用されませんが、互換性のために維持されています。

Lock_level

使用されませんが、互換性のために維持されています。

例

PL/I および C のサンプル制御ブロックが RODM とともに提供されています。これらの制御ブロックをプログラムに組み込んでください。

表 34. トランザクション情報ブロックのサンプル名

例	名前
PL/I トランザクション情報ブロック	EKG1TRAB
C トランザクション情報ブロック	EKG3TRAB

使用法

戻りコードおよび理由コードのフィールドは、要求された機能の状況に関して RODM がユーザー・アプリケーションと通信するために使用されます。

機能ブロック

説明

RODM データに対して行われるすべてのトランザクションの詳細は、機能ブロックに指定されています。機能ブロックはユーザーによって作成され、希望するトランザクションを要求するために RODM に渡されます。

機能ブロックの形式

各機能ブロックの形式は、427 ページの『機能の解説』にリストしてあります。

機能ブロック・フィールドの説明

機能ブロックの各パラメーターの説明は、507 ページの『機能パラメーターの説明』にリストしてあります。

使用法

各機能ブロックの最初のフィールドには、要求されている機能を指定する 4 バイトの整数が入ります。機能ブロックのそれ以外の部分の形式は、4 バイトの機能 ID に応じて異なります。

機能ブロックの一般的な形式の 1 つでは、クラス、オブジェクト、およびフィールドの仕様が含まれています。 RODM 内のサブフィールドを指定するためのフィールドが機能ブロックに含まれていることもあります。クラスとオブジェクトだけしか指定できない機能ブロックもあります。また、クラスだけしか指定できない場合もあります。

エンティティー・アクセス情報ブロック

説明

エンティティー・アクセス情報ブロック (EAIB) には、 API がクラスまたはオブジェクトにアクセスするために使用する情報が含まれます。 EAIB は、機能ブロックから分離されているので、それ以降の API 呼び出し時に再利用することができます。 EAIB へのポインターは機能ブロックに保管されます。

このアクセス情報は、2 つの異なる形式で利用できます。

- アプリケーションによって提供されるシンボル名
- クラスまたはオブジェクトの作成にシンボル名が使用される場合に RODM が生成する ID。この形式を使用したほうが、情報に速くアクセスできます。

機能ブロックの形式

表 35 は、エンティティー・アクセス情報ブロックの形式を示しています。表ヘッディングの意味は次のとおりです。

オフセット
パラメーターの始まりまでのオフセットを 10 進バイト数で指定します。

長さ パラメーターの長さを 10 進バイト数で指定します。

タイプ パラメーターの RODM データ・タイプを指定します。

用途 そのパラメーターが機能へのデータ入力に使用されるのか、機能によるデータ出力に使用されるのかを示しています。ダッシュ (—) は、そのパラメーターが機能で使用されることがないこと、または予約されていることを示しています。

パラメーター名
パラメーターの名前を指定します。

表 35. RODM エンティティー・アクセス情報ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Anonymous(4)	—	予約済み
004	4	Integer	入力	Naming_count
008	4	ClassID	入出力	Class_ID
012	4	Integer	入力	Class_name_length
016	4	Pointer	入力	Class_name_ptr
020	8	ObjectID	入出力	Object_ID
028	4	Integer	入力	Object_name_length
032	4	Pointer	入力	Object_name_ptr

機能ブロック・フィールドの説明

Naming_count
エンティティー・アクセス情報ブロックの Naming_count フィールドには、このブロック内のどのデータが有効であるのかが示されます。有効な値は、以下のとおりです。

値	意味
0,2	機能のターゲットがクラスまたはオブジェクトのいずれかであって、オブジェクト・アクセス情報とクラス・アクセス情報の両方が有効であることを示します。
1	機能のターゲットがクラスであって、クラス・アクセス情報だけが有効であることを示します。

これらすべての情報の解釈は、354 ページの『使用法』に記載する規則に従って行います。

Class_ID
クラス ID。

Class_name_length
クラス名の長さ。

Class_name_ptr
クラス名を指すポインターです。 PL/I で可変の長さストリングとして CLASS1 CHAR(64) VARYING などの変数を宣言すると、クラス名ポインターは、PL/I V2R3 の Pointeradd 組み込み関数を使用して指定されます。 PL/I の 2 バイト長接頭部ではなく文字データを直接指し示したい場合には、 `class_name_ptr = POINTERADD(ADDR(CLASS1) ,2 )` とコーディングしてください。

Object_ID
オブジェクト ID。

Object_name_length
オブジェクト名の長さ。

Object_name_ptr
オブジェクト名を指すポインターです。 PL/I で可変の長さストリングとして OBJECT1 CHAR(255) VARYING などの変数を宣言すると、オブジェクト名ポインターは、PL/I V2R3 の Pointeradd 組み込み関数を使用して指定されます。 PL/I の 2 バイト長接頭部ではなく文字データを直接指し示したい場合には、 `object_name_ptr = POINTERADD(ADDR(OBJECT1) ,2 )` とコーディングしてください。

例

PL/I および C のサンプル制御ブロックが RODM とともに提供されています。これらの制御ブロックをプログラムに組み込んでください。

表 36. エンティティー・アクセス情報ブロックのサンプル名

サンプル	名前
PL/I エンティティー・アクセス情報ブロック	EKG1ENTB
C エンティティー・アクセス情報ブロック	EKG3ENTB

使用法

この機能ブロック内の function_ID には、使用する機能ブロックを指定します。この機能ブロックは、その機能のためにエンティティー・アクセス情報ブロックを使用するのかどうかを指定するものです。

対応するポインターの長さ値がヌルの場合、この値は、ポインターの値とは無関係にヌル・ストリングを表します。同様に、ヌルのポインター値も、対応する長さの値とは無関係にヌル・ストリングを表します。したがって、ヌル・ストリングはヌルの長さによっても、ヌルのポインターによっても表されます。

名前を指すポインターが使用された場合、このポインターは可変長文字ストリングを指します。文字ストリングの長さはエンティティー・アクセス情報ブロック内のパラメーターとして指定され、エンティティー・アクセス情報ブロック内のポインターは文字データの最初のバイトを直接指し示します。

ID (RODM で生成された内部 ID) が RODM に入っているのは、RODM のほうが文字ストリング名よりも速く処理されるためです。 ID は、アドレス指定対象のクラスまたはオブジェクトを解決する際に、常に文字ストリング名よりも優先されます。以下が適用されます。

- エンティティー・アクセス情報ブロック内で Class_ID と Class_name_length のどちらもヌルの値でない場合、Class_ID が使用されて Class_Name_Ptr は無視されます。呼び出し元によってクラス名と Class_ID の両方が提供された場合、RODM はそれらが矛盾していないかどうかを判別しません。
- Object_ID と Object_name_length がともに非ヌルであって、Naming_count が 1 ではない場合には、Object_ID が使用され、Object_Name_Ptr は無視されます。RODM は、提供されたオブジェクト名と Object_ID が一致しているかどうかを判別しません。
- Naming_count が 1 である場合、RODM によって使用されるのはクラス情報だけです。

オブジェクト ID だけでオブジェクトを十分に突きとめることができます。オブジェクト ID には、そのオブジェクトを含むクラスの識別情報が含まれています。オブジェクト ID が指定されると、 RODM はそれ以外のすべてのオブジェクトおよびクラスに関する情報を無視します。

Object_ID が指定されないときに、実行しようとするトランザクションのターゲットの仕様でオブジェクトを指定する必要がある場合には、 Object_Name を指定しなければなりません。その場合、Class_ID にオブジェクトのクラスを指定するか、あるいは Class_Name_Ptr でクラスの名前を指し示す必要があります。指定されたクラスにその名前のオブジェクトがない場合には、エラーになります。

クラスのフィールドをアドレス指定するトランザクションの場合、オブジェクトは必要ありません。同じ形式がオブジェクトおよびクラス・アクセス情報ブロックにも使用されます。 Object_ID フィールドと Object_name_length フィールドの両方をヌル値に設定し、トランザクションのターゲットがオブジェクトではなくクラスにあることを RODM に警報します。ターゲット・クラスは、Class_ID とともに指定されたクラスまたは Class_Name_Ptr によって指定されたクラスのいずれかです。別の方法として、ユーザーは、Naming_count フィールドの値を 1 に設定し、 RODM によって分析される情報の有効範囲を限定することができます。

制御ブロックは反復して使用するように設計されています。パフォーマンスを向上させるために、制御ブロックを再利用してください。 RODM を使用するアプリケーションの実行中に、ターゲットが異なる類似のトランザクションが反復して要求されることがあります。エンティティー・アクセス情報ブロックの反復使用を単純化するために、 RODM によって以下のアクションが取られます。

- RODM の呼び出し時に Class_ID フィールドがヌルであって、Class_Name_Ptr フィールドがヌルでなく、要求されたトランザクションが正常に (4 以下の戻りコードで) 完了すると、 RODM は Class_ID フィールドにターゲット・クラスのクラス ID を入れます。また、RODM は、エラーの検出前にターゲットがアクセスされたときにエラーによってトランザクションが正常に完了しなかった場合にも、Class_ID にターゲット・クラスのクラス ID を入れます。
- RODM の呼び出し時点で Object_ID フィールドがヌルになっていて、Object_Name_Ptr がヌルになっていない場合に、命名カウントが 1 (この値は、クラス情報だけが使用されることを表します) でなく、要求されたトランザクションが正常に (4 以下の戻りコードで) 完了すると、RODM はターゲット・オブジェクトのオブジェクト ID を Object_ID フィールドに入れます。また、RODM

は、エラーの検出前にターゲットがアクセスされたときにエラーによってトランザクションが正常に完了しなかった場合にも、Object_ID にターゲット・オブジェクトのオブジェクト ID を入れます。

トランザクション要求内でターゲットを指定するために名前が使用されていて、同じエンティティー・アクセス情報ブロックを使用してその要求がその後反復される場合、 ID フィールドはすでに最初のトランザクションで記入されています。したがって、2 番目のトランザクションは、最初のトランザクションよりも速く実行できます。

2 番目のトランザクションが最初のトランザクションと類似しているだけで、同じではないような環境では、2 番目のトランザクションのパフォーマンス向上の程度は、これよりも低下します。例えば、以下の場合には、2 番目のトランザクションのパフォーマンス向上の程度が低くなります。

- 2 番目のトランザクションで、クラスやオブジェクトのフィールドとは無関係に、最初のトランザクションと同じフィールドが指定されている場合。
- 最初のトランザクションと 2 番目のトランザクションのターゲットが同じオブジェクトであるが、最初のトランザクションでオブジェクトを指定するために文字ストリング名が使用されている場合。
- 2 番目のトランザクションで最初のトランザクションと同じクラスが (クラス・フィールドに) 指定されているが、各トランザクションで文字ストリング名を使用して異なるオブジェクトが指定されている場合。エンティティー・アクセス情報ブロックがこのような方法で反復して使用されるときには、そのブロックを使用した後で毎回 ObjectID をヌルに設定しなければなりません。そのようにしないと、再使用時に、ID が文字ストリング名に優先するという規則が適用され、 2 番目のトランザクションが最初のトランザクションと同じターゲット・オブジェクトに送られます。

機能ブロックを再利用し、Class_name または Object_name フィールド (またはポインター) を更新したら、対応する ID フィールド (Class_ID、Object_ID) をヌルにリセットしなければなりません。 ID フィールドがゼロに設定されている場合のみ文字ストリング名が意味を持つので、これを行う必要があります。

フィールド・アクセス情報ブロック

説明

フィールド・アクセス情報ブロック (FAIB) には、 API がフィールドにアクセスするために使用する情報が含まれます。 FAIB は、機能ブロックから分離されているので、次の API 呼び出し時に再利用することができます。 FAIB へのポインターは機能ブロックに保管されます。

このアクセス情報は、2 つの異なる形式で利用できます。

- アプリケーションによって提供されるシンボル名
- クラスまたはオブジェクトの作成にシンボル名が使用される場合に RODM が生成する ID。この形式を使用したほうが、情報に速くアクセスできます。

機能ブロックの形式

表 37 は、フィールド・アクセス情報ブロックの形式を示しています。表ヘッディングの意味は次のとおりです。

オフセット
パラメーターの始まりまでのオフセットを 10 進バイト数で指定します。

長さ パラメーターの長さを 10 進バイト数で指定します。

タイプ パラメーターの RODM データ・タイプを指定します。

用途 そのパラメーターが機能へのデータ入力に使用されるのか、機能によるデータ出力に使用されるのかを示しています。ダッシュ (—) は、そのパラメーターが機能で使用されることがないこと、または予約されていることを示しています。

パラメーター名
パラメーターの名前を指定します。

表 37. RODM フィールド・アクセス情報ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Anonymous(4)	—	予約済み
004	4	Integer	入力	Naming_count
008	4	FieldID	入出力	Field_ID
012	4	Integer	入力	Field_name_length
016	4	Pointer	入力	Field_name_ptr

機能ブロック・フィールドの説明

Naming_count
フィールド・アクセス情報ブロックの naming_count フィールドには、このフィールド・アクセス情報が有効であるのかどうかが示されます。以下は、有効な値です。

値	意味
0	情報は有効
1	予約済み

Naming_count は常に 0 (ゼロ) に設定してください。

Field_ID
フィールド ID。

Field_name_length
フィールド名の長さ。

Field_name_ptr
フィールド名を指すポインターです。

例

PL/I および C のサンプル制御ブロックが RODM とともに提供されています。これらの制御ブロックをプログラムに組み込んでください。

表 38. フィールド・アクセス情報ブロックのサンプル名

例	名前
PL/I フィールド・アクセス情報ブロック	EKG1FLDB
C フィールド・アクセス情報ブロック	EKG3FLDB

使用法

この機能ブロック内の function_ID には、使用する機能ブロックを指定します。この機能ブロックは、その機能のためにフィールド・アクセス情報ブロックを使用するのかどうかを指定するものです。

対応するポインターの長さ値がヌルの場合、この値は、ポインターの値とは無関係にヌル・ストリングを表します。同様に、ヌルのポインター値も、対応する長さの値とは無関係にヌル・ストリングを表します。したがって、ヌル・ストリングはヌルの長さによっても、ヌルのポインターによっても表されます。

名前を指すポインターが使用された場合、このポインターは可変長文字ストリングを指します。文字ストリングの長さは、文字データの最初のバイトを直接指すポインターとともに、フィールド・アクセス情報ブロック内のパラメーターとして指定されます。

ID (RODM で生成された内部 ID) が RODM に入っているのは、RODM のほうが文字ストリング名よりも速く処理されるためです。 ID は、アドレス指定対象のフィールドを解決する際に、常に文字ストリング名よりも優先されます。フィールド・アクセス情報ブロック内で Field_ID と Field_name_length がともにヌルではない場合、Field_ID が使用され、Field_Name_Ptr は無視されます。 RODM は、提供された Field_ID が提供されたフィールド名と矛盾していないかどうかを検査しません。

あるフィールドが、実行するトランザクションのターゲットである場合、そのフィールドをヌルではない Field_ID または Field フィールドで指定する必要があります。指定されたフィールドは、対応するエンティティー・アクセス情報ブロックで指定されたエンティティー (オブジェクトまたはクラス) に関連付けられます。

トランザクション要求内でターゲットを指定するために名前が使用されていて、同じエンティティー・アクセス情報ブロックを使用してその要求がその後反復される場合、 ID フィールドはすでに最初のトランザクションで記入されています。したがって、2 番目のトランザクションは、最初のトランザクションよりも速く実行できます。

制御ブロックは反復して使用するように設計されています。パフォーマンスを向上させるために、制御ブロックを再利用してください。 RODM を使用するアプリケーションの実行中に、ターゲットが異なる類似のトランザクションが反復して要求されることがあります。 RODM は、フィールド・アクセス情報ブロックの反復使用を単純化するために、以下のアクションを行います。

- RODM の呼び出し時に Field_ID がヌルで、Field_name_Ptr はヌルでなく、トランザクションのターゲットがフィールドを必要としている場合、要求されたトランザクションが正常に完了すると、RODM は Field_ID フィールドにターゲット・フィールドのフィールド ID を入れます。

- また、RODM は、エラーの検出前にターゲットがアクセスされたときにエラーによってトランザクションが正常に完了しなかった場合も、Field_ID に値を入れます。

機能ブロックを再利用し、Class_name または Object_name フィールド (またはポインター) を更新したら、対応する ID フィールド (Class_ID、Object_ID) をヌルにリセットしなければなりません。 ID フィールドがゼロに設定されている場合のみ文字ストリング名が意味を持つので、これを行う必要があります。

応答ブロック

説明

RODM の照会要求、照会メソッド、名前付きメソッド、およびオブジェクト独立メソッドによって得られた出力は、応答ブロックに入れて戻されます。応答ブロックの形式および応答ブロックに含まれるデータは、応答を生成したトランザクションの種類によって異なります。

機能ブロックの形式

各応答ブロックの形式を、関連する機能とともに示します。表 39 には、機能ごとに各応答ブロック形式の参照ページを示しています。

表 39. 応答ブロックを使用する機能

応答ブロックを使用する機能	参照ページ
EKG_Locate	463
EKG_QueryEntityStructure	468
EKG_QueryField	470
EKG_QueryFieldID	472
EKG_QueryFieldName	473
EKG_QueryFieldStructure	475
EKG_QueryMultipleSubfields	479
EKG_QueryFunctionBlockContents	476
EKG_QueryNotifyQueue	481
EKG_QueryObjectName	483
EKG_QueryResponseBlockOverflow	485
EKG_QuerySubfield	487
EKG_TriggerNamedMethod	500
EKG_TriggerOIMethod	502
EKG_WhereAmI	506

機能ブロック・フィールドの説明

応答ブロックで使用される各パラメーターの説明は、 507 ページの『機能パラメーターの説明』に列挙されています。

使用法

応答ブロックの基本形式はすべて同じです。

- このメソッドまたはアプリケーションによって設定された Response_block_length フィールドは、提供される応答ブロックの長さをバイト数で示します。
- RODM によって設定される Response_block_used フィールドは、応答ブロックで使用されるストレージの量、またはブロックが小さすぎる場合に必要とされる量を示します。
- ストレージのブロックの形式と内容はトランザクションのタイプによって決まりますが、通常は次の内容で構成されます。
  - 戻されたデータのデータ・タイプ ID を示す Data_type フィールド
  - 機能、または機能によって起動されたメソッドが戻すデータ

呼び出し元が指定した応答ブロックが小さすぎて応答全体を保持できない場合は、以下のいずれかが発生します。

- 提供された応答ブロックが 8 バイト未満である場合、トランザクションはエラーの戻りコードを出してただちに終了します。
- 提供された応答ブロックが 8 バイト以上である場合、トランザクションは RODM によって実行されます。
- 応答ブロックに必要な合計バイト数は、戻された値のデータ・タイプと長さ、および生成される出力の量によって決まります。
- RODM がトランザクションを完了した後に、情報を通常どおりに戻すために十分なスペースが応答ブロックにない場合、RODM は、応答ブロックの Response_Block_Used フィールドが生成された応答の合計サイズを示すように設定します。 RODM はデータのその部分を、Response_Block_Length フィールドで指定されたバイト数に等しいサイズの応答ブロックに保管します。

  RODM は、ユーザー・オブジェクトの EKG_RBOverflowAction フィールドの設定に応じて、次のいずれかのアクションを取ります。

  - そのフィールドで廃棄が指定されている場合には、オーバーフロー・データを消失させます。
  - そのフィールドでオーバーフロー情報を保存するように指定されている場合、RODM はユーザーがその後の呼び出しで検索できるように応答ブロックのオーバーフロー・データを保管します。

  484 ページの『EKG_QueryResponseBlockOverflow － 応答ブロック・オーバーフローを照会する』を参照してください。

  オーバーフロー・データは、オーバーフローの原因となったトランザクションのトランザクション情報ブロック内のトランザクション ID によって識別されます。元の応答ブロックに収まりきらなかったデータを検索するために、このトランザクション ID を EKG_QueryResponseBlockOverflow 機能の Correlation_ID パラメーターで指定する必要があります。機能ブロックに入れて RODM に渡された戻りコードおよび理由コードは、 (応答ブロックが小さすぎたことによる) エラーを示すように設定されています。

  注: EKG_QueryResponseBlockOverflow 機能と EKG_Disconnect 機能を除いて、このトランザクションと同じアクセス・ブロックに関連する追加のトランザクションは、ユーザーが応答ブロック・オーバーフロー・データを検索するまで、RODM に拒否されます。

- 応答ブロックのオーバーフローの原因となっているトランザクションがトランザクションのリストから実行されている場合、リストに残っているトランザクションの実行結果は、後で検索できるようにすべてオーバーフロー・ブロックに入ります。
- すべてのオーバーフロー・データはオーバーフロー・バッファーに入ります。応答ブロック内のデータとこのオーバーフロー・データの連結は、アプリケーションによって行う必要があります。

このブロックの response_block_used フィールドの後の部分は、トランザクション・タイプ、データ・タイプ、およびデータ・リストの長さによって決まります。

名前付きメソッドおよびオブジェクト独立メソッドがトランザクションによって RODM に対して起動されると、これらのメソッドは、応答ブロックを介してそのトランザクションを実行したタスクに戻される、 SelfDefining データ・ストリング (SelfDefining タイプの可変長ストリング) を生成することができます。名前付きメソッドとオブジェクト独立メソッドが起動されると、応答ブロックの変数部分は、それらのストリングを呼び出し元タスクに送達するために使用されます。

名前付きメソッドまたはオブジェクト独立メソッドのために応答ブロックでオーバーフローが発生した場合、そのメソッドは、オーバーフローに関する戻りコードおよび理由コードを受け取ります。しかし、これらのメソッドは、メソッドを起動したプログラムへはこの戻りコードと理由コードを渡さないことがあります。名前付きメソッドまたはオブジェクト独立メソッドが起動された場合には、応答ブロックに入れて戻された Response_block_used パラメーターと Response_block_length パラメーターを常に比較してください。 Response_block_used パラメーターの値が Response_block_length パラメーターの値よりも大きい場合には、オーバーフローが発生しています。

複数のトランザクションが単一のユーザー・アプリケーション ID で同時に実行されている場合、それらのトランザクションのいずれかまたはすべてがオーバーフローの原因となる可能性があります。オーバーフローが起こった場合、EKG_QueryResponseBlockOverflow 機能を呼び出すまでは、 (EKG_Disconnect 機能以外の) ユーザー API 機能を EKGUAPI から使用することはできなくなります。

オーバーフローしたすべての応答ブロックを EKG_QueryResponseBlockOverflow 機能によって検索してからでなければ、(EKG_Disconnect 機能以外の) 他のユーザー API 要求を EKGUAPI から使用することはできません。EKG_QueryResponseBlockOverflow 機能に対するそれぞれの呼び出しで、相関 ID を指定する必要があります。相関 ID は、応答ブロック・オーバーフローの原因となったトランザクションを表すトランザクション ID です。相関 ID を使用することにより、正しいオーバーフロー応答ブロックを戻すことができます。

個々の RODM 機能の説明の多くでは、各種の応答ブロックがさらに詳しく説明されています。

トランザクションのエラー条件

トランザクションの実行中にエラー条件が発生した場合、 RODM はトランザクション情報ブロック内で戻りコードと理由コードを発行します。カスタマイズ・ファイルで設定された LOG_LEVEL および MLOG_LEVEL の値によっては、 RODM ログでもエラーが記録されることがあります。 メソッドが異常終了しないかぎり、実行継続の決定はメソッドが行います。

メソッドは、EKG_SetReturnCode 機能を使用して RODM に戻りコードを出すことができます。 493 ページの『EKG_SetReturnCode − 戻りコードと理由コードを設定する』を参照してください。エラーは RODM ログに記録されることがあり、RODM に対する呼び出しの戻りコードと理由コードは、そのトランザクションが正常に完了しなかったことを示す値に設定されます。

メソッドおよびユーザー・アプリケーションに出された戻りコードと理由コードは、 RODM によって次のように設定されます。

- すべてのユーザー API およびメソッド API のトランザクションに関する最初の戻りコードと理由コードは、0 に設定されます。
- ユーザー API 要求の処理中にメソッドが起動された場合、ユーザー・アプリケーションに対して戻される戻りコードと理由コードは、同期メソッドによって設定されます。同期メソッドが戻りコードを設定しない場合、ユーザー API トランザクションの実行中に RODM がエラーを検出したときには、RODM によって戻りコードが設定されます。
- メソッドは、呼び出し元に戻される戻りコードと理由コードを設定することができます。メソッドに関する現行の戻りコードおよび理由コードは、最初はゼロに設定されています。メソッドは、EKG_SetReturnCode 機能を使用してこの戻りコードと理由コードを変更することができます。現行の戻りコードと理由コードは、このメソッドを起動したメソッドまたは (RODM がこのメソッドを起動した場合には) RODM に戻されます。

  メソッドが EKG_SetReturnCode 機能を使用して新しい戻りコードと理由コードを設定した場合、 RODM は呼び出し元に戻される戻りコードと理由コードを次のように設定します。

  – 新しい戻りコードが現行の戻りコードよりも大きい場合、そのメソッドに関する現行の戻りコードと理由コードは、新しい戻りコードと理由コードによって置き換えられます。
  – 新しい戻りコードが現行の戻りコード以下である場合は、そのメソッドについての現行の戻りコードと理由コードは変更されません。
- メソッドによって設定された戻りコードと理由コードが、それを呼び出したメソッドに戻される場合、呼び出しメソッドの戻りコードと理由コードは、呼び出されたメソッドと同じように設定されます。

RODM は、戻りコードと理由コードを出すだけでなく、エラーに関する追加の診断情報を提供するログ・レコードを記録することもあります。ユーザー API を介して受け渡される各トランザクションには、固有の Transaction_ID が指定されています。 RODM は、この Transaction_ID をアクセス・ブロックに入れて呼び出し元に戻します。メソッド、またはトランザクション内のその他の部分でエラーが発生すると、この Transaction_ID がエラーに関する RODM ログ・レコードに書き込まれ

ます。メソッド API を介して受け渡される各トランザクションには、そのメソッド API を介して出された親トランザクションの Transaction_ID が指定されます。

- メソッドが EKG_SetReturnCode 機能を呼び出したときに戻りコードと理由コードが変更されると、 RODM は、次のことが真である場合に限り、タイプ 3 ログ・レコード (オブジェクト特有メソッドの場合) またはタイプ 4 ログ・レコード (オブジェクト独立メソッドの場合) を作成します。
  - メソッドが同期メソッドである場合の戻りコードは、アプリケーション・プログラムの EKG_User オブジェクト内の EKG_LogLevel フィールドの値よりも大きくなければならず、また、ログ記録が使用可能になっていなければなりません。 EKG_LogLevel フィールドについては、 231 ページの『EKG_User クラス』を参照してください。
  - メソッドが非同期メソッドである場合の戻りコードは、 RODM カスタマイズ・ファイル内の LOG_LEVEL パラメーターよりも大きくなければなりません。 RODM カスタマイズ・ファイルに関する詳細については、「*IBM Tivoli NetView for zOS アドミニストレーション・リファレンス*」を参照してください。
- レベル 1 メソッド (つまり、 EKG_MessageTriggeredAction 機能によって起動された最初の非同期メソッド) から戻された最後の戻りコードと理由コードにより、次のことが決まります。
  - 最後の戻りコードが、非同期メソッドを起動したアプリケーション・プログラムを表すユーザー・オブジェクトの EKG_LogLevel フィールドの値以上である場合は、ログ・レコードが作成されます。
- EKGUAPI を呼び出すユーザー・アプリケーション・プログラムの場合は、次のとおりです。
  - 最後の戻りコードが、アプリケーション・プログラムのオブジェクトの EKG_LogLevel フィールドの値以上である場合、RODM はタイプ 2 のログ・レコードをログに書き込みます。

以下に、戻りコードと理由コードの伝搬の例を示します。

1. ユーザー・アプリケーション・プログラム UA1 は、フィールドを照会するために EKGUAPI を呼び出します。
2. 照会フィールドに照会メソッド・サブフィールドがあるため、メソッド QM1 が起動されます。 QM1 の初期戻りコードおよび理由コードは、ともに 0 です。
3. QM1 は、ターゲット・オブジェクトでなんらかの処理を実行するために、名前付きメソッド NM1 を起動します。 NM1 の初期戻りコードおよび理由コードは、ともに 0 です。
4. NM1 は EKG_SetReturnCode 機能を使用して、戻りコードと理由コードをそれぞれ 4 と 2000 に設定します。
5. QM1 は名前付きメソッドから戻りコード 4 および理由コード 2000 を検索しますが、それらの戻りコードおよび理由コードをユーザー・アプリケーション・プログラムに戻すことはありません。その代わりに、EKG_SetReturnCode 機能を使用して、戻りコードと理由コードをそれぞれ 0 と 3000 に設定します。 QM1 が EKG_SetReturnCode 関数を使用して戻りコードと理由コードを設定しなかった場合、 RODM がユーザー・アプリケーション・プログラムに戻りコード 0 と理由コード 0 を戻しています。

6. ユーザー・アプリケーション・プログラムは、戻りコード 0 と理由コード 3000 を受け取ります。

メソッドを作成する際には、メソッドから戻りコードおよび理由コードを出すことの意味に注意する必要があります。アプリケーションは、メソッドから戻された戻りコードと理由コードがそのメソッドの成功または失敗のみに関連付けられている場合にも、それらのコードを機能の成功または失敗と関連付けて解釈することがあります。例えば、通知申請は、EKG_ChangeField 機能によって正常に変更されたフィールドに割り当てられるものですが、通知メソッドが失敗して戻りコードと理由コードを設定することがあります。この場合、アプリケーションがその戻りコードと理由コードを EKG_ChangeField 機能の障害として解釈し、通知メソッドの障害とは解釈しないことがあります。

RODM 通知プロセス

RODM 通知処理を使用すると、RODM 内の指定されたフィールドの値が変更されたときに、ユーザー・アプリケーションに通知が送られます。通知処理は、フィールドの値が変更されたときに行う必要のあるプロセスを自動化するために使用できます。例えば、特定のネットワーク・リソースがダウンしたときに、それらのリカバリーを自動化することができます。

また、RODM 通知処理は、次のことをユーザー・アプリケーションに通知するためにも使用できます。

- 非同期エラーおよびチェックポイント。 372 ページの『非同期エラー通知』には、エラーの通知とチェックポイント機能が説明されています。ユーザー・アプリケーションでは、 RODM に接続した後でできるだけ速やかに必要な通知をセットアップする必要があります。
- 削除されたオブジェクト。 373 ページの『オブジェクト削除通知』には、削除されたオブジェクトに関する通知が説明されています。アプリケーションは、独自の通知メソッドをインストールする代わりに、 EKG_AddObjDelSubs 機能 (432 ページ) を使用して、削除されたオブジェクトの通知を申請します。

このセクションでは、自動化されたリカバリー・アプリケーションの例を使用して、 RODM 通知処理について説明します。この例では、RODM データ・キャッシュ内のオブジェクトで表される NETRES1、 NETRES2、NETRES3 などのリソースがあるものとします。 DisplayStatus というオブジェクトのフィールドは、そのリソースの状況を表します。このフィールドの値は、別のアプリケーションによって保守されます。また、これらのいずれかのリソースがダウンしたときにそれをリカバリーさせることができる、 RECOVER というユーザー・アプリケーションが作成されているものとします。リソースがダウンするたびに RECOVER アプリケーションが通知を受け取るように、 RODM を設定してください。

RODM 通知処理には、4 つの全般的なステップがあります。
1. 設定
2. 待機
3. 通知
4. 遮断

それぞれの全般的なステップは、RECOVER の例を使用して説明されています。これらのステップの中には、複数の方法で実行できるものがあります。この例では、最も単純な方法を採用していますが、他の方法についても説明しています。

RODM 通知処理には 5 つのエレメントがあります。
- 通知キュー
- 通知メソッド
- notify サブフィールド
- イベント制御ブロック (ECB)
- ユーザー・アプリケーション

設定

RODM 通知処理の最初のステップは、設定です。設定には以下の作業が含まれます。
- RODM へのユーザー・アプリケーションの接続
- notify サブフィールドの作成
- 通知メソッドのインストール
- 通知キューの作成
- フィールドへの申請

この例では、RODM が実行されていて、オブジェクトとそれを保守するアプリケーションが定義されていることを想定しています。通知対象の各オブジェクトのフィールドごとに設定ステップを完了することも、クラス・レベルで通知を設定することもできます。クラス・レベルで設定を行う場合、そのクラスのすべてのオブジェクトについて通知処理が定義されます。

1. RODM を使用するための最初のステップは、RODM への接続です。 RECOVER アプリケーションは、EKG_Connect 機能を使用して RODM への接続を行います。 RODM は RECOVER アプリケーションを表す EKG_User クラスのオブジェクトを作成します。
2. DisplayStatus フィールドに notify サブフィールドが含まれない場合、RECOVER アプリケーションは EKG_CreateSubfield 機能を使用してこのサブフィールドを作成します。このサブフィールドは、DisplayStatus フィールドと同じクラスに作成されます。
3. メソッドは、インストールしてからでなければ使用できません。メソッドのインストールは、RODM 用に指定されたライブラリーにメソッドを配置して、そのメソッドを表す EKG_Method クラスのオブジェクトを作成することによって行います。メソッドのインストール方法は、410 ページの『メソッドのインストールおよび解放』で説明しています。

   この例では、RODM とともに提供される通知メソッドの 1 つを使用しています。フィールドの値が、指定されたしきい値の範囲外になっているときには、EKGNTHD 通知メソッドが起動されます。このしきい値は、EKG_AddNotifySubscription 機能で指定された Long_lived_parm に入れて EKGNTHD に渡されます。

   EKGNTHD 通知メソッドについては、548 ページの『RODM の通知メソッド』で説明されています。 NetView で提供されたメソッドがユーザーの要件に適合しない場合には、独自の通知メソッドを作成することができます。

4. 通知キューとそれに関連するイベント制御ブロック (ECB) を作成してください。ユーザー・アプリケーション RECOVER に通知を行うすべてのオブジェクトについて必要な通知キューは、 1 つだけです。 1 つの通知キューは単一のユーザー・アプリケーションに関連付けられていますが、 1 つのユーザー・アプリケーションが多数の通知キューをもつことが可能です。通知キューは EKG_NotificationQueue クラスのオブジェクトです。

   a. RECOVER が、EKG_CreateObject 機能を使用して EKG_NotificationQueue クラスのオブジェクトを作成します。通知キュー名は、ユーザー・アプリケーション内で固有の名前でなければなりません。この例では、このトランザクションのエンティティー・アクセス・ブロック内でキュー名 RECOVQ をオブジェクト名として指定してください。 RODM は、EKG_NotificationQueue オブジェクトの MyName フィールドを作成するために、ユーザー・アプリケーションの User_appl_ID を指定されたキュー名と連結します。この例では、MyName は RECOVER.RECOVQ に設定されます。 RODM は、EKG_NotificationQueue オブジェクトの EKG_UsedBy フィールドを、ユーザー・アプリケーションを表す EKG_User オブジェクトの EKG_Uses_Q フィールドにリンクします。

   b. ECB の値を 0 (ゼロ) に設定してください。

   c. EKG_ECBAddress フィールドの値として、このキューに使用する ECB のアドレスを指定してください。 RECOVER は、このフィールドの値を設定するために EKG_ChangeField 機能を使用します。 ECB は、ユーザー・アプリケーションのアドレス・スペース内に作成されます。同じ ECB を多数の通知キューで使用できます。

   d. ステップ 4a で作成した通知キュー・オブジェクトの EKG_Status フィールドを 1 (アクティブ) に設定してください。 RECOVER は、このフィールドの値を設定するために EKG_ChangeField 機能を使用します。

   ECB を通知キューと関連付ける必要はありません。アプリケーションは、通知キューに対して時々照会を行い、通知が追加されていないかどうかを調べることができます。しかし、この手法は ECB で提供される非同期通知ほど便利ではありません。

5. 設定の最後のステップは、各オブジェクトのフィールドへの申請です。 RECOVER アプリケーションは EKG_AddNotifySubscription 機能を発行します。この機能は、通知メソッド名 EKGNTHD、通知パラメーター、通知キュー名 RECOVQ、およびユーザー・アプリケーション ID RECOVER を notify サブフィールドに収めます。この機能呼び出しのパラメーターを次のように指定してください。

   Entity_access_info_ptr
   通知申請を作成する対象となるクラスおよびオブジェクトを指定するエンティティー・アクセス・ブロックを指すポインターです。

   Field_access_info_ptr
   DisplayStatus フィールドを指定するフィールド・アクセス・ブロックを指すポインターです。

User_appl_ID
このパラメーターはヌル値に設定してください。 RODM が、この機能呼び出しを出している RECOVER アプリケーションに対応する値を記入します。

Notification_queue
ステップ 4 (366 ページ) で作成した通知キューの名前を指定してください。 この例では、名前として RECOVER.RECOVQ ではなく RECOVQ を入力してください。この名前の User_appl_ID 部分は RODM によって提供されます。

User_word
このオプション・フィールドはブランクにしておくことができます。

Notify_method
通知メソッド EKGNTHD を表す EKG_Method クラスのオブジェクトのオブジェクト ID を指定します。 メソッドがインストールされたものである場合には、ステップ 3 (365 ページ) で EKGNTHD のオブジェクトを作成したときにエンティティー・アクセス・ブロックの Object_ID フィールドに戻した値を指定します。事前インストールされたメソッドの場合、このオブジェクト ID は、そのメソッドの MyName フィールドを照会することによって入手できます。

Long_lived_parm
EKGNTHD が起動されたときに渡されるパラメーターを指定します。このパラメーターには、このメソッドが起動される原因となるしきい値を指定します。これらのパラメーターについては、 548 ページの『RODM の通知メソッド』で説明されています。

申請するフィールドごとに、ステップ 5 (366 ページ) を 1 回ずつ繰り返してください。各オブジェクトに関して EKG_AddNotifySubscription 機能が正常に実行されると、通知処理の設定が完了します。

この例では、フィールドが変更されたときに 1 つのユーザー・アプリケーションに通知を出すことを示していますが、任意の数のアプリケーションに通知を出すことができます。 notify サブフィールドには、通知申請のリストを指定できます。通知対象となるユーザー・アプリケーションごとに、通知処理全体を繰り返してください。

オブジェクトごとに通知申請を作成する代わりに、クラスに関する通知申請を作成することができます。 RODM は、クラスのフィールドがそのクラスのいずれかのオブジェクトについて変更されたときに、そのフィールドの通知メソッド定義を起動します。通知メソッドは、メソッドを起動した特定オブジェクトを突き止めるために、Where Am I (2007) 機能を使用する必要があります。

待機

通知処理が設定された後で、ユーザーのアプリケーションは、RODM から変更を通知されるまで処理を延期します。 EKGWAIT を呼び出すと、指定された ECB、または ECB のリスト内の任意の ECB が RODM によって通知されるまでアプリケーションを待機させることができます。

EKGWAIT は、WAIT 機能を提供するインターフェース・モジュールです。ユーザーのアプリケーションは、ECB 情報を含むパラメーター・リストを指定して EKGWAIT を呼び出します。

この例では、RECOVER は ECB を指定して EKGWAIT を呼び出します。 ECB が通知されると、EKGWAIT は RECOVER に制御を戻します。そして、RECOVER が通知を処理します。

EKGWAIT の呼び出し

RODM では、EKGWAIT の呼び出し方を示すサンプル・コードが提供されます。PL/I のサンプルは EKG5WAIT であり、C のサンプルは EKG6WAIT です。

EKGWAIT はユーザー・アプリケーションだけで使用することができます。EKGWAIT の呼び出しの形式は、次のとおりです。

```
EKGWAIT(Num_ECBs, ECB_Array, Return_code, Reason_code)
```

EKGWAIT インターフェース・モジュールの呼び出しで指定されるパラメーター・リスト内の各パラメーターについて、以下で説明します。このパラメーター・リストは、EKGWAIT が制御を戻すときにユーザー・アプリケーションに情報を渡すために、 EKGWAIT によっても使用されます。

パラメーター名
説明

Num_ECBs (入力)
イベント・リストにおける ECB の数を指定する 2 バイトの小整数。

ECB_Array (入力)
各ポインターに ECB のアドレスが入っているポインターの配列。

Return_code (出力)
戻りコードが入っている 4 バイトの整数。

Reason_code (出力)
理由コードが入っている 4 バイトの整数。 Return_code が 0 の場合、このフィールドには、ECB への通知の対象となった ECB_Array の索引が入ります。

PL/I のコーディング例

図 74 は、PL/I ユーザー・アプリケーションから EKGWAIT を呼び出す場合の例です。

```
%Include SYSLIB(EKG1IEEP); /* EKGWAIT declaration                */

%Dcl n fixed;
%n=3;                      /* Arbitrary max number of ECBs in list*/
```

図 74. PL/I のコーディング例 (1/4)

```
Dcl
  ECB_Array(n) Pointer,    /* Array of ECB pointers              */
  Return_code  fixed bin(31), /* Return code from EKGWAIT        */
  Reason_code  fixed bin(31), /* Reason code from EKGWAIT        */
  Num_ECBs     fixed bin(15), /* Number of ECBs                  */
  POSTED_ECB   fixed bin(31) based,  /* ECB which was posted     */
```

図 74. PL/I のコーディング例 (2/4)

```
  ECB1         fixed bin(31),  /* First ECB                     */
  ECB2         fixed bin(31),  /* Second ECB                    */
  ECBn         fixed bin(31);  /* Nth ECB                       */

ECB_Array(1)=addr(ECB1);   /* Address of ECB1                   */
ECB_Array(2)=addr(ECB2);   /* Address of ECB1                   */
ECB_Array(n)=addr(ECBn);   /* Address of ECBn                   */
Num_ECBs=n;                /* Number of ECBs in list            */
```

図 74. PL/I のコーディング例 (3/4)

```
CALL EKGWAIT(Num_ECBs,ECB_Array,Return_code,Reason_code); /* Wait
                              on list of ECBs                   */

If Return_code = 0 then    /* No errors in WAIT                 */
  Do;
    /*************************************************************/
    /* ECB_Array(Reason_code) is a pointer to the posted ECB.    */
    /*************************************************************/
    ECB_Array(Reason_code)->POSTED_ECB=0;
  End;
```

図 74. PL/I のコーディング例 (4/4)

C のコーディング例

図 75 は、C ユーザー・アプリケーションから EKGWAIT を呼び出す場合の例です。

```
#include "EKG3CEEP.H"      /* EKGWAIT declaration               */
#define n 3                /* Arbitrary max number of ECBs in list*/

int*    ECB_Array[n];      /* Array of ECB pointers             */
int     Return_code;       /* Return code from EKGWAIT          */
int     Reason_code;       /* Reason code from EKGWAIT          */
int     Num_ECBs;          /* Number of ECBs                    */
```

図 75. C のコーディング例 (1/3)

```
int     ECB1;              /* First ECB                          */
int     ECB2;              /* Second ECB                         */
int     ECBn;              /* Nth ECB                            */

ECB_Array[0]=&ECB1;        /* Address of ECB1                    */
ECB_Array[1]=&ECB2;        /* Address of ECB2                    */
ECB_Array[n-1]=&ECBn;      /* Address of ECBn                    */
Num_ECBs=n;                /* Number of ECBs in list             */
```

図 75. C のコーディング例 (2/3)

```
EKGWAIT(&Num_ECBs,ECB_Array,&Return_code,&Reason_code); /* Wait
                              on list of ECBs                     */

if (Return_code == 0) {    /* No errors in WAIT                   */
  /*****************************************************************/
  /* ECB_Array[Reason_code-1] is a pointer to the posted ECB.      */
  /*****************************************************************/
  *ECB_Array[Reason_code-1]=0;
}
```

図 75. C のコーディング例 (3/3)

EKGWAIT 使用上の注意

ECB_Array の目的は、EKG_NotificationQueue オブジェクト内の EKG_ECBAddress フィールドに設定されている ECB アドレスを収めることです。ただし、接続時に RODM に対して識別された Stop_ECB は常に ECB_Array に組み込まれています。これにより、ユーザーは、RODM を停止する必要があるときにいつまでも待機する必要がなくなります。

正常な戻りでは Return_code が 0 になり、Reason_code は、通知された ECB の ECB_Array 内の索引を表す (1 から N までの) 整数値に設定されます。この機能呼び出しから正常に戻ったときには、通知された ECB をただちに消去してください。

この機能呼び出しに ECB アドレスとして 0 が渡されると、警告戻りコードによる即時戻りが行われます。しかし、ECB アドレスが無効な場合には、異常終了または無期限待機が発生する可能性があります。

通知

アプリケーションが申請しているフィールドの値が変更されると、そのアプリケーションの通知メソッドが起動されます。この例では、オブジェクト NETRES3 の DisplayStatus フィールドが変更されると、 RODM が通知メソッド EKGNTHD を起動します。 EKGNTHD は、DisplayStatus の新しい値と、EKG_AddNotifySubscription 機能の Long_lived_parm パラメーターで指定したしきい値を比較します。

新しい値が指定されたしきい値を超えている場合、 EKGNTHD は通知ブロックを通知キュー RECOVQ に入れ、 RODM は RECOVER アプリケーションを ECB に

通知します。通知メソッドは、通知ブロックをキューに入れるために EKG_SendNotification 機能を使用します。 ECB が通知されると、EKGWAIT は RECOVER に制御を戻します。

RODM は、次の条件がすべて当てはまる場合に、ECB に通知キューを通知します。
- 通知キューがある。
- 以前空になっていたキューに通知ブロックが追加される。
- キュー・ポインターの ECB ポインターが有効な ECB を指している。

RODM が ECB に特定の通知キューについて通知した後で、そのキューが完全に処理されて新しいブロックが追加されるか、あるいは通知キュー・オブジェクトにある EKG_ECBAddress フィールドが変更されるまでは、RODM が再びそのキューを ECB に通知することはありません。

RODM に再接続したときに、ユーザー・アプリケーションに関する通知申請と通知キュー・オブジェクトがまだ残っている場合は、ECB に通知を行うことはできません。それぞれの通知キュー・オブジェクトにある EKG_ECBAddress フィールドを現行の ECB アドレスにリセットして、 RODM が ECB に通知を行えるようにしなければなりません。

残りの処理は、アプリケーションによって行われます。

1. ユーザー・アプリケーションは、ECB を 0 に設定することによって ECB を消去します。
2. ユーザー・アプリケーションは、EKG_QueryNotifyQueue 機能を使用して通知キューから通知ブロックを入手します。通知ブロックには、通知の原因となったイベントのタイプを示す Notification_block_type フィールドが含まれています。

   機能呼び出しごとに 1 つのブロックが除去されます。この機能の応答ブロックは、キューに入っている通知ブロックの数を Notification_queue_count パラメーターで表示します。ユーザー・アプリケーションは、通知キューに入っている各ブロックを処理します。 EKG_QueryNotifyQueue 機能は、ユーザー・アプリケーションが接続されているアドレス・スペースから発行する必要があります。

   この例では、RECOVER が通知キュー名 RECOVQ を指定して EKG_QueryNotifyQueue 機能を一度呼び出します。
3. このアプリケーションは、応答ブロックに戻された通知ブロック情報を使用して処理を開始します。この例の RECOVER は、Object_ID パラメーターを使用して、DisplayStatus を変更したリソースを識別します。 RECOVER は、RODM データ・キャッシュから新しい DisplayStatus 値を得るために、 EKG_QueryField 機能を使用することもできます。 RECOVER はその後で適切なコマンドを出して、障害が起こっているリソース NETRES3 を検索します。
4. 通知キューの処理が終了すると、ユーザー・アプリケーションは EKGWAIT を呼び出して、次の通知が行われるまで待機します。

遮断

通知処理では、メモリーとプロセッサーのサイクルを含むシステム・リソースが使用されます。あるオブジェクトについて通知が不要になった場合には、その通知を削除してください。

通知を削除する方法には、次の 2 つがあります。
- 通知キューの削除
- 通知申請の削除

通知キューを使用するすべての通知申請を削除したい場合には、その通知キューを表す EKG_NotificationQueue クラスのオブジェクトを削除してください。 EKG_DeleteObject 機能を使用してください。 RODM は、通知キュー、およびそのキューを指定しているすべての通知申請を削除します。 また RODM は、まだ通知キューに残っている通知ブロックも削除します。

通知キューを使用する通知申請が複数あっても、その申請をすべて削除するのでない場合、削除したい申請ごとに EKG_DeleteNotifySubscription 機能を使用してください。

この例では、保守のために NETRES2 を遮断することにします。 RECOVER が NETRES2 の再始動を試みないようにするために、 EKG_DeleteNotifySubscription 機能を発行し、 Entity_access_info_ptr パラメーターで NETRES2 を指定します。他の通知申請は影響を受けません。

通知キューが削除されると、RODM は EKG_User オブジェクトと EKG_NotificationQueue オブジェクトの間のリンクを削除します。ユーザー・アプリケーションが RODM から切断されたり、切断しないで終了したりしたときには、RODM は、そのユーザー・アプリケーションと関連付けられた通知キューおよび通知申請を削除することができます。オブジェクトを表す EKG_User オブジェクトの EKG_StopMode フィールドでは、 RODM がとるアクションを指定します。 EKG_StopMode フィールドに関しては、 231 ページの『EKG_User クラス』を参照してください。

非同期エラー通知

ユーザー・アプリケーションは、RODM システム定義オブジェクト内のフィールドに申請することにより、同期エラーおよびチェックポイントに関する通知を受け取ることができます。非同期 API 要求、非同期メソッド、または RODM 内部処理の実行中に発生する非同期エラーに関する通知を受け取るようにするには、 EKG_System オブジェクトの EKG_LastAsyncError フィールドに申請してください。ユーザー・アプリケーションが開始したトランザクションのエラーに関する通知だけを受け取るようにするためには、 EKG_User オブジェクトの EKG_LastAsyncError フィールドに申請してください。

これらの通知申請には、NetView 提供のメソッド EKGNOTF を使用することができます。このメソッドについては、 548 ページの『RODM の通知メソッド』を参照してください。ログ・レコードは EKG_LastAsyncError フィールドに割り当てられています。このログ・レコード情報は、EKG_LastAsyncError フィールドへの申請によって作成された通知キュー・ブロックの user_data フィールドに入ります。ユーザー・アプリケーション・プログラムは、EKG_QueryNotifyQueue 機能を呼び出すことによってこの情報を得ることができます。

エラーが発生すると、指定された通知メソッドが起動されます。 EKG_LastAsyncError フィールドに申請を行ったすべてのユーザー・アプリケーションが通知を受けます。

メソッドがユーザー・アプリケーションと非同期で実行された結果としてエラー・メッセージがログに書き込まれると、EKG_LastAsyncError フィールドが変更され、通知メソッドが起動されます。メソッドが設定した戻りコードが、そのユーザーの EKG_LogLevel パラメーターまたは非同期に関して指定された Log_level カスタマイズ・パラメーター以上の値になっている場合、RODM はエラー・ログ・エントリーを書き込みます。

オブジェクト削除通知

特定のオブジェクトが削除されたときにアプリケーションが通知を受け取る必要がある場合、このアプリケーションは、*オブジェクト削除申請* によってこれらのオブジェクトに申請することができます。該当のオブジェクトが削除されると、RODM は通知キューに通知ブロックを入れ、アプリケーションを ECB に通知します。

通知ブロックの形式については、481ページの EKG_QueryNotifyQueue 応答ブロックの説明を参照してください。

RODM 通知処理の 4 つのステップ (設定、待機、通知、遮断) はオブジェクト削除通知にも適用されますが、適用のされ方は少し異なります。

オブジェクト削除通知の設定

オブジェクト削除通知の場合は、365ページで説明している通常の RODM の通知処理とは異なります。

1. RODM に接続します。notify サブフィールドの作成、通知メソッドのインストール、またはフィールドへの申請は行わないでください。
2. ステップ 4 (366 ページ) で説明しているように、通知キューとその ECB を作成してください。
3. 設定の最後のステップは、オブジェクトへの申請です。アプリケーションは、オブジェクトに関するオブジェクト削除申請を作成するために、EKG_AddObjDelSubs 機能を発行します。この機能では、オブジェクト、ユーザー・アプリケーション、および通知キューを指定します。該当のオブジェクトが削除されると、RODM は指定された通知キューに通知ブロックを入れ、ユーザー・アプリケーションを ECB に通知します。この機能呼び出しのパラメーターを次のように指定してください。

   Entity_access_info_ptr
   オブジェクト削除申請を作成しているクラスおよびオブジェクトを指定するエンティティー・アクセス・ブロックを指すポインター。

   User_appl_ID
   このパラメーターはヌル値に設定してください。 RODM が、この機能呼び出しを出しているユーザー・アプリケーションに対応する値を記入します。

   Notification_queue
   ステップ 4 (366 ページ) で作成した通知キューの名前を指定してください。 この名前の User_appl_ID 部分は RODM によって提供されます。

   User_word
   このオプション・フィールドはブランクにしておくことができます。

Long_lived_parm
オブジェクトが削除されると、 RODM はこのオプション・パラメーターの値を応答ブロックの user_area パラメーターに入れます。

申請するオブジェクトごとに、ステップ 3 (373 ページ) を 1 回ずつ繰り返してください。 EKG_AddObjDelSubs 機能が各オブジェクトについて正常に実行されると、削除通知処理の設定は完了します。

オブジェクト削除通知の待機

このステップは 367 ページの『待機』と同じです。

オブジェクト削除通知の通知

アプリケーションが申請しているオブジェクトが削除されると、RODM は通知ブロックをアプリケーションの通知キューに入れ、ECB にアプリケーションを通知します。

このステップの残りの部分は、 370 ページの『通知』で説明したとおりです。

オブジェクト削除通知の消去

オブジェクト削除申請を削除するには、 455 ページの『EKG_DelObjDelSubs － オブジェクト削除申請を削除する』で説明されている EKG_DelObjDelSubs 機能を使用してください。

RODM への接続

どのユーザー API 機能を実行する場合にも、その前に EKG_Connect API 機能を使用して RODM に接続しておかなければなりません。 RODM に接続するときには、ユーザー・アプリケーション ID および接続したい RODM の名前を含むアクセス・ブロックを指定してください。 RODM は、正常に接続された後でアクセス・ブロック内の Sign_on_token フィールドを設定します。この値は、RODM への接続を表すものであり、変更してはなりません。 RODM は、ユーザーが EKG_Connect 以外の API 機能を要求したときに、アクセス・ブロック内の Sign_on_token フィールドの値が無効であることを検出すると、その API 機能要求を拒否し、該当の理由コードを戻します。

RODM は、アプリケーション・ユーザー ID ごとに 1 つの接続だけを許可します。すでに接続されているユーザー・アプリケーション ID への接続を試みると、接続は失敗し、該当の理由コードが戻されます。

オペレーターによって取り消されたアプリケーション、または RODM への接続時に異常終了したアプリケーションは、切断されます。

申請通知キューを除去せずに RODM から切断する場合は、次回の接続時に、通知申請に関連付けられたすべての ECB アドレスを、新しいアドレス・スペース ID を指すようにリセットしなければなりません。

ユーザーのアプリケーションは、仮想記憶間モードで実行されている場合には RODM に接続することはできません。 RODM はこの条件の有無を検査し、エラー理由コードを戻します。

RODM への接続が正常に行われると、RODM は、ユーザー・アプリケーションを表す EKG_User クラスにユーザー・オブジェクトを作成します。このユーザー・オブジェクトにはユーザーのアプリケーション環境が含まれていて、アプリケーションが切断されるまで保存されます。 RODM では、同じユーザー・アプリケーション ID に関して複数の API 要求を並行して実行することができますが、各要求がユーザー・オブジェクト内の情報を使用し、場合によってはその情報を修正することがあります。

RODM への接続の詳細については、 441 ページの『EKG_Connect － RODM に接続する』を参照してください。

RODM からの切断

アプリケーションがすべてのタスクを完了し、それ以上実施すべき API 機能要求がないときには、 RODM の EKG_Disconnect API 機能を使用してそのアプリケーションを切断します。切断した後では、サインオン・トークンは無効になります。その後ユーザーのアプリケーションが別の API 機能要求を実行しようとすると、その API 機能要求が EKG_Connect 機能要求でないかぎり、RODM はエラー理由コードを戻します。

アプリケーションが切断されると、 RODM は、ユーザー・オブジェクト内の EKG_StopMode の値に応じて通知キューの整理を行います。 RODM は、そのユーザー・アプリケーション ID が所有しているすべての通知キュー、キュー・エレメント、および申請を除去することも、通知キュー・エレメントだけを除去して通知キューとをすべて維持することも、また、なにも除去せずすべての通知キュー、キュー・エレメント、および申請を維持することもあります。すべての通知キュー、キュー・エレメント、および申請を除去する場合、RODM はユーザー・オブジェクトも除去します。

注: RODM への接続時に終了したアプリケーションは切断されます。

RODM からの切断の詳細については、 457 ページの『EKG_Disconnect － RODM から切断する』を参照してください。

第 12 章 トポロジー・オブジェクトの相関

この章では、オブジェクト相関機能について説明します。以下の情報が記載されています。
- 相関機能を使用可能にする
- 相関の概念
- ユーザーのオブジェクトを相関に組み込む
- SNA トポロジー・マネージャーおよびマルチシステム・マネージャーのオブジェクトを相関付ける
- 相関機能のカスタマイズ

NetView 管理コンソール (NetView 管理コンソール) は、相関関係がある集合オブジェクトを使用して、以下のことが行えます。
- 相関関係があるリソース間のナビゲート
- 相関関係があるリソースに関する統合データの表示
- 相関関係があるリソースの集合状況のモニター

相関関係があるオブジェクトの使用に関する詳細は、「*IBM Tivoli NetView for z/OS マルチシステム・マネージャー ユーザーズ・ガイド*」を参照してください。

相関機能を使用可能にする

オブジェクト相関は、入力ファイル FLCSDM8 を RODM にロードすることによって使用可能になります。 FLCSDM8 をロードするには、ジョブ CNMSJH12 の以下の行からアスタリスク (*) を取り除きます。

```
//* DD DSN=NETVIEW.V5R3M0.CNMSAMP(FLCSDM8),DISP=SHR  <-CORRELATE SAMPL
```

相関は、あるアプリケーションが、相関の対象となっている RODM オブジェクトのフィールドに有効な値を設定したときに発生します。オブジェクトは、ファイル FLCSDM8 をロードすることによって相関の対象となります。マルチシステム・マネージャーおよび SNA トポロジー・マネージャーは、これらのフィールドの値を自動的に設定します。それにより、相関が発生し、NetView 管理コンソール でビューが表示されるようになります。

マルチシステム・マネージャーのオブジェクト相関を使用可能にする

TMR エージェントにより管理されているリソースのナビゲーションおよびストレージを最適化するには、以下にリストされている順序で GETTOPO コマンドを発行してください。
1. GETTOPO TMERES
2. その他の GETTOPO コマンド

SNA トポロジー・マネージャーのオブジェクト相関を使用可能にする

SNA トポロジー・マネージャーにより管理されているリソースに関して相関を使用可能にするには、初期設定ファイル FLBSYSD を編集して、以下のステートメントの値を YES に変更します。

```
WRITE_CORRELATABLE_FIELDS=NO
```

SNA 相関は PU リソースで行われます。 PU リソースは、*LOCAL* パラメーターを含まない TOPOSNA コマンドから除外されます。相関に組み込むリソースに対して発行される任意の TOPOSNA コマンドで *LOCAL* パラメーターを使用します。

SNA トポロジー・マネージャーが相関係数値を提供するリソースは、 PU 2.1 OS/2 ワークステーションです。 SNA トポロジー・マネージャーが OS/2 ワークステーションをモニターしていない場合には、どの SNA リソースも相関付けることはできません。 SNA リソースの LAN MAC アドレスが分かっている場合には、それらを相関に組み込むことができます。 385 ページの『マルチシステム・マネージャーおよび SNA トポロジー・マネージャーにより作成されたオブジェクトの相関を拡張する』を参照してください。

GMFHS のオブジェクト相関を使用可能にする

GMFHS リソースに関して相関を使用可能にするには、GMFHS_Managed_Real_Objects_Class の以下の 1 つまたは複数のフィールドに値を設定します。

- aIndMACAddress
- Correlater
- iPAddress

RODM ロード入力ファイル FLCSDM8 は、ロードされると、これらのフィールドを GMFHS_Managed_Real_Objects_Class に作成します。

相関の概念

相関機能は、メソッド FLCMCON がインストールされているフィールドの値が変更されると、起動されます。相関は、異なるエージェントにより管理されているリソースを自動的に関連付けます。相関機能は動的に実行され、RODM メソッドを使用して設定されます。相関関係があるオブジェクトは、共通の相関係数値をもっており、これらのオブジェクトを表すために、*相関関係がある集合オブジェクト* が使用されます。 IP アドレスまたは LAN MAC アドレスによって相関が行われる場合、相関付けられた集合オブジェクトは、RODM では aggregateSystem クラス・オブジェクトを使用して表現されます。 Correlater フィールド内の値によって相関が行われる場合には、相関付けられた集合オブジェクトは、GMFHS_Aggregate_Object_Class オブジェクトを使用して RODM で表現されます。

相関関係があるオブジェクト とは、以下のいずれかのフィールドに値が指定されている、相関が使用可能なクラスのオブジェクトのことです。

- aIndMACAddress
- iPAddress
- Correlater

この値が、*相関係数値* です。

クロス相関 という用語は、同じ相関係数値をもつ複数の実オブジェクト間の関係を表すために使用されます。例えば、以下を前提として考えてみます。

- 相関機能が使用可能である。
- インターネット・ホストおよび NetWare サーバーが含まれているワークステーションを使用する。
- リソースは RODM のオブジェクトによって表され、各オブジェクトの iPAddress フィールドの値が 9.37.65.43 である。

これらの 2 つのオブジェクトは、同じフィールドに関して同一の値をもつので、クロス相関であるといえます。

相関メソッド

相関機能は、以下の RODM メソッドによって実現されます。

メソッド FLCMCONI

メソッド FLCMCONI は、相関をサポートするクラスにメソッド FLCMCON をロードする初期設定メソッドです。メソッド FLCMCONI はメソッド FLCMCON にパラメーターを渡すので、 RODM ロード入力ファイル DUIFSTRC の代わりに使用されます。

メソッド FLCMCON

メソッド FLCMCON は、相関を使用可能にするクラスの特定のフィールドにロードされる通知メソッドです。相関が使用可能なクラス、およびメソッド FLCMCON がロードされているフィールドを判別するには、 RODM ロード・ファイル FLCSDM8 をブラウズします。 FLCMCON は FLCMCOR を実行します。

メソッド FLCMCOR

メソッド FLCMCOR は、相関関係がある集合オブジェクトを作成および更新するオブジェクト独立メソッドです。

これらのメソッドのロードおよびカスタマイズは、RODM ロード・ファイル FLCSDM8 を使用して行われます。詳細については、377 ページの『相関機能を使用可能にする』、および 386 ページの『相関機能のカスタマイズ』を参照してください。

相関で使用可能なオブジェクト

サンプル FLCSDM8 をロードすると、マルチシステム・マネージャー、 SNA トポロジー・マネージャー、および GMFHS データ・モデルを使用するユーザーのアプリケーションにより管理されているリソースに関して、相関が自動的に使用可能になります。 どのクラスが自動的に使用可能になっているのかを判別するには、RODM ロード・ファイル FLCSDM8 をブラウズします。メソッド FLCMCON がロードされているすべてのクラスが自動的に使用可能になります。

例えば、次のようにコーディングすると、マルチシステム・マネージャー IP フィーチャーによって作成された、 internetRouter クラスのオブジェクトで、IP アドレスによる相関が使用可能になります。

```
OP FLCMCONI INVOKED_WITH (SELFDEFINING)
(
 (OBJECTID) EKG_Method.FLCMCON
 (CLASSID) '1.3.18.0.0.3330'-- internetRouter
 (FIELDID) '1.3.18.0.0.3330'.'iPAddress'
 (CLASSID) '1.3.18.0.0.6464'
 (CLASSID) 'GMFHS_Managed_Real_Objects_Class'
);
```

相関のタイプ

相関には、2 つのタイプがあります。
- ネットワーク・アドレス相関
- フリー・フォーム相関

ネットワーク・アドレス相関

ネットワーク・アドレス相関は、LAN メディア・アクセス制御 (MAC) またはインターネット・プロトコル (IP) アドレスを使用して行われます。

ネットワーク・アドレスに基づいて相関にオブジェクトを組み込むには、以下のいずれかのフィールドに値を設定します。
- aIndMACAddress (1.3.18.0.0.5263)
- iPAddress

相関は、12 文字の MAC アドレス (例えば、10004BF00943) を使用します。 14 文字の MAC アドレスがサポートされていますが、最後の 2 文字 (リンク・サービス・アクセス・ポイント) が取り除かれます。

有効な IP アドレスは、数字と、数字を区切る 2 つ以上のピリオド (.) から構成されます。

フリー・フォーム相関

フリー・フォーム相関は、フリー・フォーム・ストリング値を使用して行われます。フリー・フォーム・ストリングにおける相関は、ストリング値と一致する表示名をもつ相関関係があるオブジェクトを作成します。

オブジェクトをフリー・フォーム相関に組み込むには、Correlater フィールドの値としてストリングを設定します。以下に有効な値の例をあげます。
- Accounting
- PresidentsOffice
- Building201
- London

複数パーツのストリング値を Correlater フィールドに入力することもできます。複数パーツのストリングを入力すると、 381 ページの図 76 に示すように、相関関係があるオブジェクトを、相関関係がある集合オブジェクトの階層にリンクすることができます。

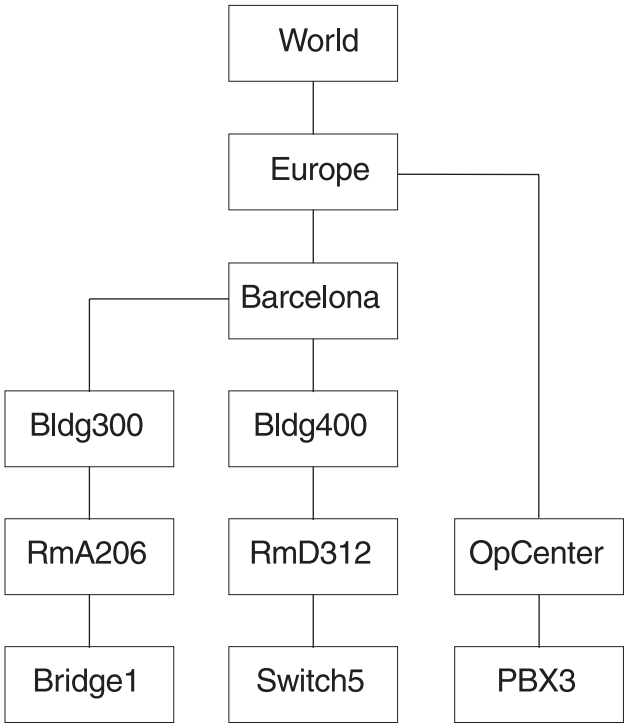


図 76. 複数のフリー・フォーム値でのオブジェクトの相関付け

相関によって図 76 のオブジェクトを作成できるようにするためには、以下の値を設定してください。

- Bridge1 Correlator = 'RmA206 Bldg300 Barcelona Europe'
- Switch5 Correlator = 'RmD312 Bldg400 Barcelona'
- PBX3 Correlator = 'OpCenter Europe World'

これにより、組織または地理的な構造に基づいて、ビューの階層を 1 つのコマンドで作成したり突き止めたりすることができます。単一値のフリー・フォーム相関の場合と同様に、複数パーツのストリングにおけるそれぞれのストリング値ごとに、相関関係のある集合オブジェクトが 1 つ検出または作成されます。複数パーツ・ストリングによって識別される、相関関係がある異なる集合オブジェクト間で親関連が存在していない場合には、親関連が作成されます。

コンマまたはブランクを使用して複数パーツ・ストリングを区切ることができます。例えば、`Jane Doe` というストリング値を入力した場合、相関によって `Jane` と `Doe` の 2 つのオブジェクトが検出または作成されます。

RODM CharVar データ・タイプによってサポートされるすべての文字がサポートされます。したがって、相関関係がある 1 つの集合オブジェクト (例えば Margaret_Thatcher) として扱われるようにしたいストリング値の間を、下線文字 (_) でつなぐことができます。

フリー・フォーム相関では、相関関係のある集約オブジェクトがクラス GMFHS_Aggregate_Objects_Class に作成されます。これにより、BLDVIEWS スクリプトによって作成された集合オブジェクトを、相関によって突き止めたりリンクしたりできるようになりました。 BLDVIEWS は、一般に、そのようなオブジェクト

が整合性のある命名体系 (例えば CPNRTR2 および CPNHST14) を備えている場合には、オブジェクトをビューに組み込みます。ビューは、トップダウンで構築されます。 複数フリー・フォーム相関では、複数のオブジェクトの命名に類似性がなくてもかまいません。ビューはボトムアップで構築されます。 BLDVIEWS とトポロジー相関を併用することにより、ユーザーの企業に合ったカスタム・ビューを作成できるようになります。

相関関係がある集合オブジェクトのクラスおよび名前

相関関係がある集合オブジェクトは、相関が検出された最初のオブジェクトの Correlater フィールド値を使用して名前が付けられます。有効な値は以下のとおりです。

- LAN MAC アドレス (例えば、40000A17D006)
- TCP/IP アドレス (例えば、9.37.65.43)
- フリー・フォーム相関 (例えば、Accounting)

ネットワーク・アドレス相関を介して識別された、相関関係のある集合オブジェクトは、クラス aggregateSystem で作成されます。これらのオブジェクトには、最後のエレメントとして MAC アドレスまたは TCP/IP アドレスを含む、複数パーツの OSI 識別名が付けられます。例えば、
`1.3.18.0.0.3519=MultiSys,1.3.18.0.0.6467=40000A17D006` のようになります。フリー・フォーム相関を介して識別された、相関関係のある集合オブジェクトは、クラス GMFHS_Aggregate_Objects_Class で作成されます。これらのオブジェクトの名前は、接頭部も接尾部も付かない、フリー・フォームの相関係数値になります (例えば、`Accounting`)。

オブジェクト名についての詳細は、「*IBM Tivoli NetView for z/OS データ・モデル・リファレンス*」の aggregateSystem クラスの説明を参照してください。

オブジェクト名は、オブジェクト MyName フィールドの値によって定義されます。NetView 管理コンソールでこれらのオブジェクトにラベルを付けるために使用される名前は、 MyName フィールド値またはユーザー定義値のいずれかです。表示ラベルの詳細については、『相関関係がある集合オブジェクトの表示ラベル』を参照してください。

相関関係があるオブジェクト関係

同一の Correlater フィールド値をもつリソースは、1 つの相関関係がある集合オブジェクトによって表されます。これには、異なるトポロジー・エージェントによって管理されているリソースが含まれます。

関係は、リンクを使用して、相関関係があるリソースと相関関係がある集合オブジェクトの間に作成されます。リンクにより、オブジェクトと状況集約の間で、詳細、構成親、および構成子のナビゲーションが使用可能になります。

相関関係がある集合オブジェクトの表示ラベル

相関関係がある集合オブジェクトは、以下のシンボルを使用して表示されます。

図 77. 集合リソースのシンボル

相関関係がある集合オブジェクトのラベルは、相関が検出された最初の値により判別されます。

表 40. 相関関係がある集合オブジェクトのラベル

最初の相関値	リソース・ラベル
MAC アドレス	LAN ワークステーション集合体
IP アドレス	IP システム集合体
Correlator フィールドの値	オープン・システム集合体

相関関係がある集合オブジェクトのフィールド値

相関機能は、メソッド FLCMCON がインストールされているフィールドの値が変更されると、起動されます。メソッド FLCMCON は、メソッド FLCMCOR を起動します。メソッド FLCMCOR は、実オブジェクトの以下のフィールドの値を照会します。

- aIndMACAddress
- segmentNumber
- aUniversallyAdministeredAddress
- adapters
- iPAddress
- netAddress
- sysLocation
- adjacentLinkStationAddress2
- linkName
- ipHostName
- Correlater

これらのフィールドの値は、相関関係がある集合オブジェクトの対応するフィールドの値と比較されます。値が実オブジェクトにはあるが、相関関係がある集合オブジェクトには無い場合、値は、実オブジェクトから、相関関係がある集合オブジェクトの対応するフィールドと DisplayResourceOtherData フィールドの両方にコピーされます。

注:

1. 値が相関関係がある集合オブジェクトのフィールドに割り当てられると、それ以降の相関は、フィールドの値を変更することはできません。
2. これらのフィールドの値を使用するアプリケーションを書く場合、DisplayResourceOtherData フィールドを解析するよりも、個々のフィールドを照会してください。これらのフィールドに関する詳細は、「*IBM Tivoli NetView for z/OS データ・モデル・リファレンス*」を参照してください。

NetView 管理コンソール を使用して、DisplayResourcOtherData フィールドのデータを表示します。この情報は、NetView 管理コンソール の **Data1** フィールドに表示されます。

DisplayResourceOtherData フィールドの値は、常に相関機能によって提供されるわけではありません。 TMR エージェントも aggregateSystem クラス・オブジェクトを作成し、 DisplayResourceOtherData フィールドに値を設定します。相関機能により提供される情報は、 *address* というワードの英小文字 *a* により識別されます。

Visual BLDVIEWS の RODMVIEW または RODMVIEW 機能で相関係数値を設定すると、その結果として生成される相関は、その RODM が次に再始動されるまでの間だけ表示されます。 その期間は、会社の業務内容により、数日の場合も、数カ月の場合もあります。相関係数値を CLIST または BLDVIEWS スクリプトで設定した場合、 RODM が再始動された後でその CLIST または BLDVIEWS スクリプトを再実行することにより、カスタマイズされた相関を復元することができます。カスタマイズにフリー・フォーム相関が含まれている場合には、より簡単な方法で相関係数値を設定することができます。 Visual BLDVIEWS (VBV) が提供するポップアップ・メニューを使用すると、 1 つまたは複数の相関関係があるオブジェクトを選択し、それらのオブジェクトの Correlater フィールドで値を設定し、それらの設定値を 1 つの BLDVIEWS スクリプトとしてホストに対して実行することができます。 このようにすると、RODM が再始動した後で、メインフレームまたは VBV ワークステーションから BLDVIEWS スクリプトを再実行して、カスタム相関を復元することができます。 Visual BLDVIEWS または BLDVIEWS をトポロジー相関と併用する際の具体的な情報については、「*IBM Tivoli NetView for z/OS マルチシステム・マネージャー ユーザーズ・ガイド*」を参照してください。

ユーザー作成オブジェクトに相関を使用する

マルチシステム・マネージャー・エージェントにより検出されたオブジェクトと SNA トポロジー・マネージャー logicalLink クラス (PU) オブジェクトは、自動的に相関付けられます。マルチシステム・マネージャーのオープン・データ・モデル、GMFHS、および追加の SNA トポロジー・マネージャーのオブジェクトを組み込むために相関を拡張することができます。 SNA トポロジー・マネージャーに関する詳細は、386 ページの『SNA トポロジー・マネージャーのオブジェクトを相関付ける』を参照してください。

ユーザーが作成したオブジェクトを相関に組み込むには、以下のタスクを実行します。

- 使用するクラスを選択する。ファイル FLCSDM8 内の、相関で使用可能な任意のクラスを選択できます。オープン・データ・モデルのオブジェクトを使用可能にするためのセットアップはより容易で、サンプル・ファイル FLCSOX01 が提供されています。アプリケーションによって GMFHS 管理のリソース・オブジェクトが作成されている場合は、 GMFHS オブジェクトを使用して作業を続けるほうが容易です。
- 相関に組み込みたい各オブジェクトの以下のデータ・フィールド (1 つまたは複数) に値を設定する。
  - aIndMACAddress (例えば、1.3.18.0.0.5263)
  - iPAddress
  - Correlater

  aIndMACAddress および iPAddress フィールドは、ネットワーク・アドレスに基づく相関をサポートし、 Correlater フィールドは、フリー・フォーム相関をサポートします。

RODMVIEW、CLIST、または BLDVIEWS スクリプトを使用して、オブジェクトにフィールド値を設定することができます。 サンプル・ファイル FLCSOX01 に、REXX CLIST の例があります。 この CLIST は、アプリケーションですでに RODM オブジェクトが作成されている場合に、コードを 1 行追加するだけで、これらのオブジェクトを相関に組み込むことができることを示しています。

マルチシステム・マネージャーおよび SNA トポロジー・マネージャーにより作成されたオブジェクトの相関を拡張する

マルチシステム・マネージャーのオブジェクトと SNA トポロジー・マネージャーの logicalLink クラス (PU) オブジェクトは、自動的に相関付けられます。オブジェクトに関して、マルチシステム・マネージャーまたは SNA トポロジー・マネージャー・エージェントにより検出されていない、相関が可能な情報がある場合、これらのオブジェクトに対して相関を拡張することができます。これらのオブジェクトの相関を拡張するには、以下のタスクを実行します。

- オブジェクトの名前を判別する
- オブジェクトの aIndMACAddress、iPAddress、または Correlater フィールドに値を設定する
- オブジェクトを拡張するのに必要な、データ・モデル独自のタスクを実行する。詳細については、『マルチシステム・マネージャーのオブジェクトを相関付ける』、および 386 ページの『SNA トポロジー・マネージャーのオブジェクトを相関付ける』を参照してください。

SNA トポロジー・マネージャーとマルチシステム・マネージャーは、オブジェクトを動的に作成、削除、更新することに注意してください。フィールド値を追加した後、トポロジーを再獲得する (例えば、TOPOSNA または GETTOPO コマンドを出す) か、または RODM をコールド・スタートすると、追加した値は喪失します。したがって、トポロジーが再獲得されるたびに、CLIST または BLDVIEWS スクリプトを使用して、相関関係があるフィールド値をリセットしてください。

オブジェクト名の判別方法

オブジェクト名は、RODM 内のオブジェクトの MyName フィールドの値により定義されます。ビューで表示されるオブジェクトの名前は、通常、 RODM におけるそのオブジェクト名を簡略にしたものですので、注意してください。ビューで表示される名前は、通常は RODM におけるオブジェクト名として使用するには不適切です。既存のオブジェクトの MyName フィールド値を判別するには、RODMVIEW または Visual BLDVIEWS を使用します。

MyName フィールドの説明および構文については、「*IBM Tivoli NetView for z/OS データ・モデル・リファレンス*」を参照してください。

マルチシステム・マネージャーのオブジェクトを相関付ける

メソッド FLCMCON が、相関付けたいオブジェクトのフィールドに直接ロードされている場合は、フィールドに値を設定します。メソッド FLCMCON がロードされているフィールドを判別するには、 RODM ロード・ファイル FLCSDM8 をブラウズしてください。ほとんどのマルチシステム・マネージャー・オブジェクトの場合、必要な作業はこれだけです。

memberOf フィールドにメソッド FLCMCON がロードされているマルチシステム・マネージャーにより作成されたオブジェクトに対して、追加のネットワーク・アドレス相関を拡張したい場合は、memberOf フィールドにリンクを作成します。

例えば、すでに IP アドレスで相関付けられている Monitor クラス・オブジェクトに MAC アドレス相関を追加したい場合は、そのオブジェクトの memberOf フィールドにリンクを作成します。リンクは、他のどのオブジェクトに対するものでもかまいません。リンクを作成するプロセスは、RODM における他のリンクの作成と同じです。

注: Correlater フィールドを使用するフリー・フォーム相関では、 RODM におけるリンクの作成は不要です。

SNA トポロジー・マネージャーのオブジェクトを相関付ける

SNA トポロジー・マネージャーの logicalLink クラス・オブジェクトは、自動的に相関に組み込まれます。これは、adjacentLinkStationAddress フィールドの値に PU の MAC アドレスが含まれていることがあるためです。相関機能は、このフィールドに MAC アドレスが入っているかどうか判別します。 MAC アドレスが入っている場合、相関機能は、このフィールドを aIndMACAddress フィールドのように扱います。

SNA トポロジー・マネージャーは TCP/IP アドレスを検出しないので、マルチシステム・マネージャー IP エージェントも、 IP アドレスが検出されたリソースにおいて IP アドレスと MAC アドレスの両方を検出しない限り、SNA PU は、そのリソースに関連付けられません。 MAC アドレスと IP アドレスをもつリソースの例として、SNA PU と IP サポートの LAN アダプターを備えた OS/2 ワークステーションがあります。 SNA トポロジー・マネージャーは、OS/2 ワークステーションでのみ MAC アドレスを検出します。

SNA リソースに関して IP アドレス相関を使用可能にするには、ファイル FLCSDM8 で使用可能なオブジェクトの iPAddress フィールドに手作業でアドレスを設定します。これにより、相関は、 SNA オブジェクトを IP アドレスをもつ他のリソースに自動的に関連付けることができます。

相関機能のカスタマイズ

相関機能のカスタマイズは、すべて、RODM ロード・ファイル FLCSDM8 を使用して行います。カスタマイズ後、RODM ロード・ファイル FLCSDM8 を RODM にロードする必要があります。 RODM ロード・ファイル FLCSDM8 が事前にロードされている場合は、RODM をコールド・スタートします。 FLCSDM8 が事前にロードされておらず、他の SNA トポロジー・マネージャーおよびマルチシステム・マネージャーのロード・ファイルがロードされている場合は、 RODM をコールド・スタートせずに FLCSDM8 をロードします。ファイル FLCSDM8 をロードするには、サンプル・ファイル EKGLLOAD を使用する必要があります。 EKGIN3 のステップでデータ・セットとファイル (FLCSDM8) を必ず指定します。

相関機能のカスタマイズには、次の 2 つの方法があります。
- 表示名優先順位を変更する
- 特定のクラスに関する相関を使用不可にする

表示名優先順位を変更する

相関関係のある集合オブジェクトがネットワーク・アドレスによって相関付けられるときに、そのオブジェクトの表示名のタイプを変更することができます。そのオブジェクトがフリー・フォーム相関係数によって相関付けられる場合には、表示名は Correlater フィールドから取られます。この場合、表示名のタイプは変更できません。

図 78 に示されているフィールドが、相関関係がある集合オブジェクト表示名の判別に使用されます。相関機能は、オブジェクトのラベル付けに使用される相関関係がある集合オブジェクト・フィールドを判別するために、ファイル FLCSDM8 にあるこれらのフィールドの優先順位リストを使用します。相関機能は、ヌルでない値が見つかるまで、リストされた順序で集合オブジェクトの各フィールドを照会します。ヌルでない値が見つかると、その値がオブジェクトのラベル付けに使用されます。 表 41 に、使用されるデフォルト優先順位と、その優先順位を使用するエージェントをリストします。

表 41. 相関関係がある集合オブジェクトのデフォルト表示名優先順位

優先順位	名前のタイプ	検出エージェント
1	コンピューター名	TMR
2	IP ホスト名	インターネット、および TMR
3	TCP/IP アドレス	インターネット、および TMR
5	SNA ノード名	SNATM
6	LAN MAC アドレス	LNM、SNATM、およびインターネット

リストされているフィールドの順序をカスタマイズして、表示されるラベルを決定することができます。

例えば、図 78に示されているデフォルト優先順位を使用する場合、インターネットは管理リソースに対してコンピューター名を定義しないため、 IP エージェントが含まれているワークステーションは、コンピューター名を使用して命名されません。このような場合、ワークステーション・オブジェクトは、インターネット・プロトコル・ホスト名を使用してラベル付けされます。

```
(
 (FIELDID) '1.3.18.0.0.6464'.
           '1.3.18.0.0.3315.2.7.202'       -- computerName
 (FIELDID) '1.3.18.0.0.6464'.
           'ipHostName'                    -- ipHostName
 (FIELDID) '1.3.18.0.0.6464'.
           'iPAddress'                     -- iPAddress
 (FIELDID) '1.3.18.0.0.6464'.
           '1.3.18.0.0.2032'               -- snaNodeName
 (FIELDID) '1.3.18.0.0.6464'.
           '1.3.18.0.0.5263'               -- aIndMACAddress
```

図 78. デフォルトの表示名優先順位

388 ページの図 79 に示されているように、 TCP/IP アドレス (優先順位 3) が IP ホスト名 (優先順位 2) の前になるように、ファイル FLCSDM8 をカスタマイズしたとします。この場合、IP エージェントが IP ホスト名と IP アドレスの両方を提

供しており、 IP アドレス名が先にリストされているので、 TCP/IP アドレスがワークステーション・オブジェクトのラベル付けに使用されます。

```
(
 (FIELDID) '1.3.18.0.0.6464'.
           '1.3.18.0.0.3315.2.7.202'      --computerName
 (FIELDID) '1.3.18.0.0.6464'.
           'iPAddress'                    -- iPAddress
 (FIELDID) '1.3.18.0.0.6464'.
           'ipHostName'                   -- ipHostName
 (FIELDID) '1.3.18.0.0.6464'.
           '1.3.18.0.0.2032'              -- snaNodeName
 (FIELDID) '1.3.18.0.0.6464'.
           '1.3.18.0.0.5263'              -- aIndMACAddress
 );
```

図 79. カスタマイズされた表示名優先順位

特定のリソースに関する相関を使用不可にする

相関は、メソッド FLCMCON が明示的にロードされている、ファイル FLCSDM8 のクラスのオブジェクトに関して使用可能です。管理リソース・オブジェクトのクラスに対してトポロジー相関を実行したくない場合は、メソッド FLCMCON をクラスにロードするメソッド・ロード・ステートメントをコメント化してください。

メソッド・ロード・ステートメントは、トポロジー・エージェントによって、ファイル FLCSDM8 内でグループ化されています。コメント化するメソッド・ロード・ステートメントを判別する場合は、以下を行います。

1. 相関関係があるオブジェクトのオブジェクト表示ラベルを判別する。
2. ラベルが表している RODM クラスを判別する。RODMVIEW を使用してクラスを判別するか、または「*IBM Tivoli NetView for z/OS データ・モデル・リファレンス*」に記載されているクラスのリストを参照して、リストされている DisplayResourceType の値とラベルを突き合わせてください。

注: 添付されているファイル FLCSDM8 をそのまま使用すると、メソッド FLCMCON は、マルチシステム・マネージャーおよび SNA トポロジー・マネージャーが自動的に相関付けることのできる、すべてのクラスにロードされます。これにより、相関を拡張したい追加クラスに関する相関も可能になります。未使用のクラスにメソッドをロードする際のメモリーおよび CPU 使用量はわずかなものなので、未使用のクラスに対するメソッド・ロード・ステートメントをコメント化する必要はありません。

第 13 章 RODM メソッドの作成

この章では、RODM メソッドについて説明します。メソッドを使用すると、RODM 内のデータを保守したり、 RODM 内のオブジェクトによって表されるリソースに関連した機能を自動化したりできます。メソッドは、RODM アドレス・スペースに常駐する小さな実行可能プログラムです。メソッドの実行は、ユーザー・アプリケーション、RODM 内のフィールドの変更、または他のメソッドによって行うことも、また RODM の初期化時に行うこともできます。メソッドは実行された方法によって分類されます。

NetView プログラムにより、ユーザーの要求の一部またはすべてに適合するようないくつかの汎用メソッドが提供されています。独自のメソッドの作成に時間を費やす前に、 548 ページの『NetView 提供のメソッド』で説明されている NetView 提供のメソッドが適用できるかどうか検討してください。

どのメソッドも (NetView 提供のメソッドも含めて)、使用する前にインストールしておく必要があります。各メソッドは、RODM 内で EKG_Method クラスのオブジェクトによって表されます。これらのオブジェクトの作成は、メソッドのインストールの一環として行われます。メソッドのインストール、削除、および更新は動的に行うことができます。

メソッドによって最良のパフォーマンスが得られるタスク

このセクションでは、どのようなタスクがメソッドで最良のパフォーマンスを得ることができるのかを説明します。

以下のことを行うときには、メソッドを使用してください。

- RODM データ・キャッシュ内の複数のオブジェクトまたはクラスで複数のアクションを実行する。

  1 つまたは複数の一連のオブジェクトあるいはクラスに対して多数の API 機能を実行するために、 オブジェクト独立メソッドを作成することができます。オブジェクト独立メソッドの詳細については、 391 ページの『オブジェクト独立メソッド』を参照してください。
- RODM の初期化時に構造体およびオブジェクトをロードする。

  RODM プログラムは、初期化メソッドと呼ばれる特殊な形式のオブジェクト独立メソッドをサポートします。 RODM 開始時に初期化メソッドを指定することにより、初期化機能を用意することができます。 初期化メソッドでは、クラス階層構造体をロードしたうえで、それらのクラスのオブジェクトを作成できます。この機能を使用して RODM データ・キャッシュを設定し、RODM 開始後の作業の準備を整えることができます。

  RODM ロード機能は、初期化メソッドとして使用することができます。このメソッドの詳細については、 392 ページの『初期化メソッド』を参照してください。
- RODM データ・キャッシュ内で変更されるデータをフィルター操作する。

アプリケーション変更 API 機能要求と RODM データ・キャッシュ内で変更されるフィールドとの間でフィルター操作を行うために、変更メソッドを作成できます。変更メソッドは、方針、セキュリティー、または妥当性検査の要件に応じて変更 API 機能要求を変更または拒否することができます。このメソッドの詳細については、393 ページの『変更メソッド』を参照してください。

- RODM データ・キャッシュ内で照会されるデータをフィルター操作する。

  アプリケーション照会 API 機能要求と RODM データ・キャッシュ内で照会されるフィールドとの間でフィルター操作を行うために、照会メソッドを作成できます。照会メソッドは、方針、セキュリティー、または妥当性検査の要件に応じて照会 API 機能要求から戻されるデータを変更することができます。このメソッドの詳細については、396 ページの『照会メソッド』を参照してください。
- RODM データ・キャッシュ内のデータの値が変更されたときにアプリケーションに通知する。

  オブジェクトまたはクラスに属するフィールド値が変更されたときに、そのオブジェクトまたはクラスに加入しているアプリケーションに通知するために、 通知メソッドを作成することができます。このメソッドの詳細については、398 ページの『通知メソッド』を参照してください。
- あるオブジェクトまたはクラス内の複数のフィールドで複数のアクションを実行する。

  1 つのオブジェクトまたはクラスに対して多数の API 機能を処理するために、名前付きメソッドを作成できます。このメソッドの詳細については、401 ページの『名前付きメソッド』を参照してください。

メソッドのタイプ

メソッドは、RODM アドレス・スペースにロードされて特定の状況で実行される、実行可能プログラム形式の論理です。メソッドは、実行される状況に基づいて分類されます。特定の種類の機能を提供するために、複数の種類のメソッドを体系化して RODM 製品が作られます。すべてのメソッドはオプションであり、メソッドによって提供される機能は、クラス、オブジェクト、およびメソッドが RODM でどのように定義、編成、および適用されているのかによって、使用できる場合と使用できない場合があります。広い意味では、メソッドにはオブジェクト*独立* メソッドとオブジェクト*特有* メソッドの 2 種類があります。

- オブジェクト独立メソッドは、RODM 内部で実行される呼び出し可能サブルーチンに似ています。このメソッドは RODM 内の異なる多数のオブジェクトに使用されます。オブジェクト独立メソッドは EKG_TriggerOIMethod 機能によって起動されます。この機能は、ユーザー・アプリケーションまたは他のオブジェクト独立メソッドによって発行することも、オブジェクト特有メソッドによって非同期的に発行することもできます。
- オブジェクト特有メソッドは特定のオブジェクトの文脈でのみ実行されます。例えば、特定のオブジェクトを参照するトランザクションによって呼び出されます。実行中のオブジェクト特有メソッドは、そのオブジェクトのフィールドおよびサブフィールド内のデータにのみアクセスできます。 RODM 内のオブジェクト特有メソッドは、トランザクション (すでに説明した照会、変更、および通知

メソッド) の副次作用として起動することも、明示的な参照 (明示要求によって実行された名前付きメソッド) によって起動することもできます。

メソッドは、データを参照して、RODM オブジェクト内のデータを操作することができます。メソッドは、メソッド API 内のルーチンを介して、アクセスできる RODM オブジェクトのフィールドおよびサブフィールドを照会したり変更したりできます。メソッドが RODM データ・キャッシュ内のデータにアクセスするには、メソッド API を使用しなければなりません。

以下のページでは、各種のメソッドとその使用法を説明します。それぞれの説明には、メソッド・インターフェースの疑似コード記述が含まれています。これらの記述はパラメーターだけを記述するものであって、正確なインターフェースを記述してはいません。パラメーターは、アドレスによってメソッドに渡されることが想定されています。 (PL/I スタイルの) 疑似コード例では、構造体の記述子の使用法などの PL/I パラメーター引き渡し規則を示すことは意図していません。メソッド・インターフェースは、パラメーターが引き渡しデータを直接指し示すユーザー API スタイルのインターフェースと整合性をもつようになっています。

オブジェクト独立メソッド

オブジェクト独立メソッドは、RODM 内部で実行される呼び出し可能サブルーチンに似ています。このメソッドは、特定の RODM オブジェクトまたはクラスとは関連していません。このメソッドは RODM 内の異なる多数のオブジェクトに使用されます。オブジェクト独立メソッドは EKG_TriggerOIMethod 機能によって起動されます。この機能は、ユーザー・アプリケーションまたは他のオブジェクト独立メソッドによって発行することも、オブジェクト特有メソッドによって非同期的に発行することもできます。

オブジェクト独立メソッドは、以下のような特性を備えています。
- ユーザー API からもメソッド API からも実行できる。
- 非同期実行のためにメソッドから実行できる。
- 複数のオブジェクトのフィールドにアクセスできる。
- 複数メソッド API 要求を、他のトランザクションによってターゲット・オブジェクトが影響されることなく RODM に出せる。

オブジェクト独立メソッドのパラメーターは、短命パラメーターです。これらのパラメーターは SelfDefining データ・タイプを使用して定義されていて、その値はアプリケーションで定義されています。これらのパラメーターは EKG_TriggerOIMethod 機能から動的に設定されます。

ユーザーが RODM に対して出すことのできる標準の照会および変更トランザクションは、 1 つのオブジェクトとしか対話できないように制限されていますが、オブジェクト独立メソッドは、複数の異なるオブジェクトそれぞれと、順次または同時に対話することができます。オブジェクト独立メソッドは、メソッド API を介して RODM 内のすべてのオブジェクトにアクセスできます。

RODM はトランザクション間の対話を管理し、ターゲット・エンティティーに対するアクションがすべて完了しなければ他のトランザクションがそのエンティティーにアクセスできないようにします。

オブジェクト独立メソッドには、メソッドに関連する長命パラメーターがありません。 1 つの SelfDefining データ・ストリングは可変長 (最長 32767 バイト) で、オブジェクト独立メソッドが実行されるときにそのメソッドに渡される唯一のパラメーターです。 RODM では、そのストリングの内容は制限されません。ユーザーは、メソッドの実行時に渡されるパラメーターと、渡されたバイト・ストリングにメッセージが付加する構文解析および意味とを調整する必要があります。

図 80 は、PL/I でのオブジェクト独立の定義方法を示し、 図 81 は、C でのオブジェクト独立メソッドの定義方法を示しています。

```
ObjIndpMeth: Procedure ( ChStrParm );

 Declare
   ChStrParm     SelfDefiningDataPtr;   /* Pointer to Short-lived, byte string */
   . . . .
   /* code */
   . . . .
 End;
```

図 80. PL/I の場合のオブジェクト独立メソッドのプロシージャー・インターフェース

```
VOID ObjIndpMeth(SelfDefiningDataPtr       **in_ChStrParm);
....
/* code */
....
```

図 81. C の場合のオブジェクト独立メソッドのプロシージャー・インターフェース

初期化メソッド

初期化メソッドは、特殊な種類のオブジェクト独立メソッドです。これは初期化時に RODM によって実行されます。 RODM が初期化メソッドによって開始すると、RODM は自動的にメソッドをインストールして実行し、さらに解放します。初期化メソッドの主要な目的は、RODM データ・キャッシュの初期階層を設定することです。機能の中には、初期化メソッドでしか使用できないものもあります。RODM ロード機能は、RODM 初期化メソッドとして使用することができます。

オブジェクト特有メソッド

オブジェクト特有メソッドは以下のとおりです。

- トランザクションの副次作用として暗黙で実行されるもの
  - 照会メソッド (データの照会時)
  - 変更メソッド (データの変更時)
  - 通知メソッド (データの変更後)
- RODM ユーザー API またはメソッド API を介して要求によって明示的に実行されるもの
  - 名前付きメソッド (フィールド名の指定により呼び出されます)

変更メソッド

変更メソッドは、フィールドの値を変更する EKG_ChangeField または EKG_ChangeMultipleFields 機能要求をトランザクションが出し、かつ、そのフィールドに変更メソッドが定義されている場合に RODM によって起動されます。ただし、トランザクションがフィールドの value サブフィールドの値を変更するために EKG_ChangeSubfield 機能要求を出したときには、変更メソッドは起動されません。変更メソッドは次のように作動します。

- ObjectLink および ObjectLinkList タイプのフィールドを除き、変更されるフィールドの最終値を判別する。これらのフィールドで定義された変更メソッドは、フィールドの値を変更しません。その代わりに、リンクまたはリンク解除アクションを進めることができるかどうかを判別します。
- ローカルでオーバーライドされないかぎり、継承する。
- 変更対象のクラスまたはオブジェクトの文脈で実行する。

変更メソッドのパラメーターは次のとおりです。

field_id
変更されるフィールドのフィールド ID です。

long_lived_parms
アプリケーションで定義されたパラメーターを含む SelfDefining ストリングです。これらのパラメーターは、変更メソッドがインストールされたときに提供されています。

short_lived_parms
アプリケーションで定義されたパラメーターを含む SelfDefining ストリングです。これらのパラメーターは、その変更メソッドを起動する API 機能要求が出されたときにメソッドに対して動的に提供されます。

data_type
変更されるフィールドの RODM データ・タイプです。

CharDataLen
data_type が CharVar または GraphicVar である場合、new_data の整数長です。この長さには、これらのデータ・タイプを表すヌル終了文字は含まれません。

New_data
API 呼び出しによって得られるフィールドの新しい値です。

変更メソッドは、オブジェクトのフィールドと関連付けて、そのフィールドのサブフィールドにすることができます。変更メソッドは、そのフィールドの内容を変更するトランザクション (ユーザー API またはメソッド API のトランザクション) が実行されるたびに実行されます。単純なフィールドをターゲットとする変更トランザクションは、ターゲット・フィールドの change サブフィールドに割り当てられている変更メソッドを起動します。変更メソッドは、ユーザー API またはメソッド API を介してこれらのトランザクションによって起動されます。

また、トランザクションが 2 つのオブジェクト内の 2 つのフィールドをリンクさせるために EKG_LinkTrigger 機能要求または EKG_UnlinkTrigger 機能要求を出したときにも、変更メソッドが起動されます。これらの変更メソッドは、フィールドの値を変更できません。これらの変更メソッドは、リンクまたはリンク解除を進める

ことができるかどうかを示すために戻りコードを設定しなければなりません。変更メソッドが存在しない場合、または変更メソッドが戻りコードを明示的に設定していない場合には、 RODM は戻りコードがゼロであるものと見なし、リンクまたはリンク解除が進められます。 ObjectLink および ObjectLinkList 以外のフィールドでの変更メソッドは、それが定義されているフィールドが直接変更された場合にのみ実行されます。親クラス上の同一フィールドが変更されて、変更された値が継承されたときには、変更メソッドは実行されません。変更メソッドは、子オブジェクトまたはクラスの変更によって実行されることはありません。変更メソッドは、サブフィールドの変更によって実行されることはありません。フィールドの value サブフィールドを操作するトランザクションを使用することにより、変更メソッドの起動を回避できます。

あるフィールドで変更メソッドが定義されている場合、そのフィールドの値に対する変更はその変更メソッドによって行われます。 RODM はそのフィールドの値を変更しません。変更メソッドは、フィールドの value サブフィールドを更新するために EKG_ChangeSubfield 機能を使用しなければなりません。変更メソッドが value サブフィールドを更新するために EKG_ChangeField または EKG_ChangeMultipleFields 機能を使用する場合、その変更メソッドが再帰的に実行されます。 RODM は再帰的なメソッド実行を検出して防止しますが、value サブフィールドの変更は行いません。

変更メソッドは、RODM 外のリソースと対話する必要がある場合、そのリソースに対して非同期的に要求を送り、要求が送られたことを示すための適切なフラグを設定します。変更メソッドは、実リソースからの応答を待たずに処理を続行します。

変更メソッドは特定オブジェクトの特定フィールドと関連しています。実行すべき変更メソッドを起動するのは、そのオブジェクトの特定フィールドに対する変更だけです。オブジェクトのフィールドに関する変更メソッドは、そのオブジェクトが作成されたときの継承により、オブジェクトに自動的に存在することがあります。オブジェクトのフィールドに関する変更メソッドは、そのオブジェクトの作成または削除によっては起動されません。

change サブフィールドのデータ・タイプは MethodSpec です。 MethodSpec データ・タイプは、実行されるメソッドを示します。このデータ・タイプは、メソッドが実行された時にそのメソッドに渡される、長命パラメーターを含んでいます。長命パラメーターは、特定の状態に対して汎用メソッドを適用するために使用できます。

長命パラメーターは、フィールド ID のリストにすることができます。これらは、メソッドが change サブフィールドに対して割り当てられるときに定義されます。フィールド ID のリストは静的です。しかし、フィールド内のデータは動的であり、いつでも変更できます。

メソッドは、メソッド API を介してフィールドの内容を読み取ることができます。したがって変更メソッドは、どのフィールドにパラメーターが入っているのかを指定するフィールド ID のリストを使用して、独自の実行時パラメーターを検出し、意図されたアクションを実行できます。大部分のメソッドは IBM により汎用メソッドとして作成されていて、フィールドの変更を管理するために実行される特定の機能に汎用メソッドを適用するには、複数のパラメーターが必要になることがあり

ます。このように設計されていることにより、デバッグのためにメソッドのパラメーターをユーザー API を介して見ることができるようになっています。

(長命パラメーター以外に) 変更メソッドの実行時に変更メソッドに渡されるパラメーターがもう 1 つあります。フィールド変更のためのユーザー API およびメソッド API 内の機能ブロックは、すべて、短命パラメーターを含んでいます。このパラメーターは、最大長が 254 バイトの SelfDefining データです。機能ブロックにデータが入っているときには、要求側は、トランザクションによって起動されるメソッドに対して呼び出し時に渡す必要のあるデータとして、これらの 254 バイトを使用することができます。

変更メソッドは、フィールドの value サブフィールドを変更するために、 API を介して提供されたデータを獲得します。それらの情報は、4 番目および 5 番目のパラメーターとして渡されます。

図 82 は、PL/I の場合の変更メソッド・パラメーターの例を示しています。 図 83 は、C の場合の変更メソッド・パラメーターの例を示しています。

```
ChngMeth: Procedure ( Field_ID, LLParms, SLParms, DataType, CharDataLen, DataPtr )
 Dcl Field_ID     FieldID;            /* target field of transaction */
 Dcl LLParms     SelfDefiningDataPtr;     /* Pointer to Long-lived field parameters */
 Dcl SLParms     SelfDefiningDataPtr;     /* Pointer to Short-lived Parameter       */
 Dcl DataType    Smallint;                /* Data type of field                     */
 Dcl CharDataLen Integer;                 /* Valid for data type CharVar and GraphicVar */
 Dcl DataPtr     pointer;                 /* Pointer to new data from API call      */
   . . . .
   /* code */
   . . . .
 End;
```

図 82. PL/I の場合の変更インターフェースのプロシージャー・インターフェース

```
 VOID ChngMeth(FieldID                    *in_FieldID,
               SelfDefiningDataPtr        **in_LLParms,
               SelfDefiningDataPtr        **in_SLParms,
               Smallint                   *in_DataType,
               Integer                    *in_CharDataLen,
               Pointer                    **in_DataPtr);
 ....
 /* code */
 ....
```

図 83. C の場合の変更インターフェースのプロシージャー・インターフェース

注: データ・タイプが CharVar または GraphicVar の場合、入力データ・ストリングはヌルで終了しています。 CharVar ストリングは X'00' で終了し、GraphicVar は X'0000' で終了しています。

トランザクション全体の戻りコードと理由コードは、メソッドで利用できるメソッド API 内の呼び出しを介して、変更メソッドから制御できます。

変更メソッドは、メソッド API を介して以下のものにアクセスできます。

- 操作対象のオブジェクトのフィールドおよびサブフィールドに入っているデータ

- このメソッドを起動した機能ブロックのコピー
- フィールドのデータ・タイプを含むオブジェクトの編成

変更メソッドによって行える作業の一部を以下に示します。

- エラー条件の発生時にトランザクションを停止し、 EKG_SetReturnCode 機能を使用して戻りコードと理由コードを設定する。
- EKG_ChangeSubfield 機能を使用してターゲット・オブジェクトのフィールドおよびサブフィールドを変更する。
- EKG_AddNotifySubscription 機能を使用して通知を追加する。
- EKG_MessageTriggeredAction 機能を使用して他のオブジェクトに対するアクションを実行する。
- EKG_OutputToLog 機能を使用して RODM ログへの書き込みを行う。

照会メソッド

照会メソッドは、トランザクションがフィールドの値を照会するときに RODM によって実行されます。しかし、value サブフィールドが明示的に照会されるときには実行されません。照会メソッドは次のように作動します。

- 照会されるフィールドの最終戻りデータ値を判別できる。
- ローカルでオーバーライドされないかぎり、継承される。
- 照会対象のクラスまたはオブジェクトの文脈で実行される。

照会メソッドのパラメーターは以下のとおりです。

field_id
照会されるフィールドのフィールド ID です。

long_lived_parms
アプリケーションで定義されたパラメーターを含む SelfDefining ストリングです。これらのパラメーターは、照会メソッドがインストールされたときに提供されています。

short_lived_parms
アプリケーションで定義されたパラメーターを含む SelfDefining ストリングです。これらのパラメーターは、その照会メソッドを起動する実際の API 機能要求が出されたときに、メソッドに対して動的に提供されます。

照会メソッドは、オブジェクトのフィールドに関連付けさせることができます。あるフィールドに関して照会メソッドが定義されている場合、ユーザー API またはメソッド API を介して EKG_QueryField 機能によってそのフィールドが照会されるたびに、照会メソッドが実行されます。照会メソッドが定義されている場合、フィールドを照会した機能にそのフィールドの値を戻すのは、そのメソッドです。照会メソッドはフィールドの現行値を戻すことも、他のなんらかの値を戻すこともあります。例えば、照会メソッドは、実リソースの現行状況を調べるために、いくつかの実リソースに対してコマンドを出すことができます。

この照会は、EKG_ResponseBlock 機能を使用して、呼び出し側提供の応答ブロックに対する応答を作成することができます。照会メソッドが EKG_ResponseBlock 機能を使用しない場合、RODM は照会されたフィールド内のデータを照会機能に戻します。照会メソッドは、戻される実際の値を作成することができます。このメソッド

は、タイム・スタンプを調べて、フィールドの値が現行値であることを確認することができます。照会メソッドを起動したくない場合、フィールド自体ではなくフィールドの value サブフィールドを照会するために、 EKG_QuerySubfield 機能を使用してください。

照会メソッドは、情報を得るために実リソースに対してコマンドを実行依頼した場合、新しいデータの要求が実行依頼されたことを示す理由コードをただちに呼び出し元に戻す必要があります。メソッドは WAIT 状態になりません。

照会メソッドは特定オブジェクトの特定フィールドに関連付けられます。実行すべき照会メソッドを起動するのは、そのオブジェクトの該当フィールドの照会だけです。

query サブフィールドのデータ・タイプは MethodSpec です。 query サブフィールドは、実行される照会メソッドの名前と、その照会メソッドが実行される特定のオブジェクト、フィールド、および環境に合わせてメソッドの動作をカスタマイズする際に照会メソッドによって使用される (長命) フィールド・パラメーターを指定するフィールド ID のリストを保存することができます。照会メソッドは、メソッド API を介して利用できるルーチンにより、それらのフィールド・パラメーターの内容を読み取ることができます。

要求側アプリケーションによって出された機能ブロックからは、短命パラメーターも抽出されて、呼び出し時に照会メソッドに渡されます。 図 84 は、PL/I の場合の照会メソッド・パラメーターの例を示しています。 図 85 は、C の場合の照会メソッド・パラメーターの例を示しています。

```
QueryMeth: Procedure ( Field_ID, LLParms, SLParms );
 Dcl Field_ID     FieldID;              /* target field of transaction */
 Dcl LLParms     SelfDefiningDataPtr;  /* Pointer to Long-lived field parameters */
 Dcl SLParms     SelfDefiningDataPtr;  /* Pointer to Short-lived Parameter       */
   . . . .
   /* code */
   . . . .
 End;
```

図 84. PL/I の場合の照会メソッドのプロシージャー・インターフェース

```
 VOID QueryMeth(FieldID                    *in_FieldID,
                SelfDefiningDataPtr        **in_LLParms,
                SelfDefiningDataPtr        **in_SLParms);
 ....
 /* code */
 ....
```

図 85. C の場合の照会メソッドのプロシージャー・インターフェース

通知メソッド

通知メソッドは、特定の機能が作成された後で RODM によって実行されます。通知メソッドを実行する機能を判別するには、 423 ページの『第 14 章 アプリケーション・プログラミングの解説』にある機能の説明を参照してください。

通知メソッドは次のように作動します。

- 申請しているユーザーに対する通知を生成する。
- クラスからオブジェクトにだけ継承される。
- 変更対象のクラスまたはオブジェクトの文脈で実行する
- フィールド変更に関する認識を以下のものに伝えることができます。
  - 他のオブジェクト
  - 申請したユーザー

通知メソッドのパラメーターは次のとおりです。

field_id
変更されたフィールドのフィールド ID です。

long_lived_parms
アプリケーションで定義されたパラメーターを含む SelfDefining ストリングです。これらのパラメーターは、通知メソッドがインストールされたときに提供されています。

short_lived_parms
アプリケーションで定義されたパラメーターを含む SelfDefining ストリングです。これらのパラメーターは、その通知メソッドを起動する実際の API 機能要求が出されたときにメソッドに対して動的に提供されます。

change_status
変更後のフィールド値が古いフィールド値に等しいかどうかを指定します。

user_appl_id
通知を受け取るユーザーのユーザー ID です。

notif_queue_id
通知を受け取る通知キューの名前です。

user_word
ユーザーが提供した情報です。

通知メソッドのリストは、notify サブフィールドが存在しているクラスまたはオブジェクトの各フィールドと関連付けられています。このリストは、フィールドの申請リストと呼ばれます。フィールドが変更されるたびに、そのフィールドと関連付けられた通知メソッドの申請リストが処理され、リストに含まれる各メソッドが実行されます。これらのメソッドは、他のオブジェクトと、変更に関する通知を必要とする RODM 外部のアプリケーションの両方に対して、変更の認識を伝えることを目的としています。通知メソッドには、しきい値を超えたときにのみ通知を行うなどの、選択的通知のための論理を組み込むことができます。

フィールドに対する変更トランザクションが指定された場合、そのフィールドで定義されているすべての通知メソッドが起動されます。これらの通知メソッドは、そのフィールドで変更メソッドが定義されているかどうか、およびそのフィールドの値が実際に変化するのかどうかにかかわらず起動されます。各通知メソッドには

RODM によって Change_status パラメーターが渡されます。このパラメーターは、そのフィールドの値が変更トランザクションによって変更されたかどうかをメソッドに通知します。

通知メソッドが起動されないようにするためには、メソッドを起動しない機能を使用してください。以下の機能は、通知メソッドを起動しません。
- EKG_LinkNoTrigger
- EKG_UnlinkNoTrigger
- EKG_ChangeSubfield
- EKG_SwapSubfield

子エンティティーの申請リストは処理されず、通知メソッドは実行されません。通知メソッドは、フィールド内の値がローカルで存在する場合にだけアクティブになります。この動作は、関連するフィールドの値が継承されて親フィールドに対して変更が行われる場合の、変更メソッドの起動を回避する動作と類似しています。

通知メソッドの中には、最初の実行後にそれ自体を削除できるものがあります。例えば、アプリケーションが RODM トランザクションを出した結果、装置をオフラインに変更しようとするコマンドがターゲット・システムに対して出されることがあります。この要求の完了には時間がかかります。

トランザクションは応答が戻るまで待つことができないため、アプリケーションはコマンドの完了時に通知を受ける必要があります。コード (元のトランザクションを実行している変更メソッドなど) は、そのフィールドの申請 (通知) キューに通知メソッドを入れます。装置がオフラインに変更されると、通知メソッドが申請キューからそれ自体を取り出し、元のアプリケーションに対して、要求された変更コマンドが正常に実行されたことを通知します。

メソッドが EKG_AddNotifySubscription 機能を呼び出す場合、そのメソッドは、この機能を実際に実行するために、データ・タイプ SubscriptSpec で示された必要な情報を獲得しなければなりません。これらの情報は、長命パラメーターと短命パラメーターから得られます。

通知メソッドは、アプリケーションがユーザー API およびメソッド API 内で EKG_AddNotifySubscription 機能を使用して明示的に行った要求に基づいて、フィールドの申請リストに入れられます。通知メソッドは、EKG_DeleteNotifySubscription 機能を使用して申請リストから削除できます。

フィールドの申請リストは常に、通知メソッドが申請キューに入ったときの順序どおりに処理されます。メソッドは、キュー内に配置された最初のものから順に、一度に 1 つずつ処理されます。

継承が通知メソッドとどのように対話するのかについて、もう少し説明する必要があります。通知申請は親クラスから子クラスへは継承されません。しかし、クラスからオブジェクトへの継承は、クラスがオブジェクトの 1 次親であっても完全に行われます。通知申請は、任意のクラスまたはオブジェクトと関連付けることができます。クラスと関連付けられた場合にそのクラスが変更されると、そのクラス・フィールドの通知リストが実行されます。オブジェクト・フィールドが変更されると、そのオブジェクト内の該当フィールドに割り当てられている通知申請が実行されます。 1 次親で同一フィールドに割り当てられた通知申請も実行されるため、ク

ラス・レベルで単一の通知申請を使用することにより、そのクラスのすべてのオブジェクトについてそれを実行できるようになります。オブジェクトの親クラスに割り当てられたメソッドで “WhereAmI” メソッド API を使用して、メソッドの実行が起動されたときの状況を判別できます。

NetView プログラムでは、次の 4 つのサンプル通知メソッドがソース形式で提供されています。ユーザー独自の通知メソッドの作成方法について理解を深めるために、これらのメソッドを学習してください。サンプル・メソッドは、CNMSAMP データ・セットの以下のメンバーになっています。

- EKGNEQL
- EKGNLST
- EKGNOTF
- EKGNTHD

これらのメソッドについては、548 ページの『RODM の通知メソッド』で説明します。

図 86 は、PL/I の場合の通知パラメーターの例を示しています。 図 87 は、C の場合の通知パラメーターの例を示しています。

```
NotifMeth: Procedure ( FieldID, LLParms, SLParms, Change_status,
                       User_Appl_ID, Notif_queue_ID, User_word );

 Declare
   FieldID          Field-identifier,     /* Field-identifier of named field  *
   LLParms          SelfDefiningDataPtr,  /* Pointer to Long-lived field parameters */
   SLParms          SelfDefiningDataPtr,  /* Pointer to Short-lived Parameter       */
   Change_status    Smallint,             /* 0 specifies new data was equal to data*/
                                          /* 1 specifies new data was not equalold data*/
   User_Appl_ID     ApplicationID,        /* unique User identifier       */
   Notif_queue_ID   SubscribeID,          /* Notification queue reference */
   User_word        Anonymous(8);         /* remote user spec             */
   . . . .
   /* code */
   . . . .
 End;
```

図 86. PL/I の場合の通知メソッドのプロシージャー・インターフェース

```
VOID NotiMeth(FieldID                    *in_FieldID,
              SelfDefiningDataPtr        **in_LLParms,
              SelfDefiningDataPtr        **in_SLParms,
              Smallint                   *in_Change_status,
              ApplicationID              **in_User_Appl_ID,
              SubscribeID                **in_Notif_queue_ID,
              Anonymous                  **in_User_word);
....
/* code */
....
```

図 87. C の場合の通知メソッドのプロシージャー・インターフェース

名前付きメソッド

名前付きメソッドは、MethodSpec として定義された以下の情報を含むフィールドによって示されます。

- メソッド・オブジェクト ID
- 長命メソッド・パラメーター

名前付きメソッドは、以下のように作動します。

- 1 つのオブジェクトに対して複数のアクションを協調して実行できます。
- 名前付きメソッドのフィールドには query、change、notify、prev_val、および timestamp サブフィールドも含めることができます。

名前付きメソッドのパラメーターは以下のとおりです。

field_id
実行されるフィールドのフィールド ID です。

long_lived_parms
アプリケーションで定義されたパラメーターを含む SelfDefining ストリングです。これらのパラメーターは、名前付きメソッドがインストールされている場合に指定されます。

short_lived_parms
アプリケーションで定義されたパラメーターを含む SelfDefining ストリングです。これらのパラメーターは、その名前付きメソッドを起動する実際の API 機能要求時にメソッドに対して動的に指定されます。

このメソッドが*名前付き* と見なされるのは、フィールド名を使用して参照 (照会、変更、および起動) できるためです。フィールド名は、オブジェクト内にあるデータ・タイプが MethodSpec のフィールドを表します。このタイプのフィールドには、メソッド名と、そのメソッドが実行されたときにメソッドが利用できる長命パラメーターが含まれています。名前付きメソッドを起動するには、ユーザー API およびメソッド API で利用できる明示アクションが使用されます。

名前付きメソッドを使用すると、あるクラスまたはオブジェクトの複数のフィールドを変更できます。 RODM は、名前付きメソッドが実行されるとターゲット・オブジェクトのすべてのフィールドをロックします。その名前付きメソッドが完了するまでは、他のメソッドまたはユーザー・アプリケーションがこれらのフィールドにアクセスすることはできません。これにより、ターゲット・クラスまたはオブジェクトの複数のフィールドに対する更新を調整できるようになります。

多くの名前付きメソッドは、そのすべてがクラスのすべてのオブジェクトと関連付けられることがあるため、名前付きメソッドは、一般的にはクラスから継承されます。オブジェクトに対する多くの標準トランザクションは、NetView 提供のメソッドによってもユーザー作成のメソッドによっても実行することができます。

データ・タイプが MethodSpec になっているフィールド、つまり名前付きメソッドのフィールドには、独自の照会、変更、通知、およびその他の標準サブフィールドを含めることができます。このようなフィールドの value サブフィールド内のデータには、メソッド名とフィールド・パラメーターのリストが含まれます。フィールド・パラメーターには、名前付きメソッドによって実行されるアクションのターゲットを指定することも、その名前付きメソッドの実行に必要な引数を指定することもできます。照会メソッドおよび変更メソッドの場合と同じように、長命フィール

ド・パラメーターのリストは、名前付きメソッドのフィールドに値が割り当てられるときに判別されます。長命パラメーターで参照されるフィールドの値は、いつでも設定できます。

フィールド・パラメーターのほかに、名前付きメソッドを起動するアプリケーションによって、もう 1 つのパラメーターを実行時に名前付きメソッドに渡すことができます。これは、短命パラメーターと呼ばれます。このパラメーターは、長命パラメーターと異なり、名前付きメソッドの実行後は、どのような方法でも保管されることがありません。名前付きメソッドで指定される短命パラメーターは、すべて、最大長が 254 でデータ・タイプが SelfDefining になっています。このような短命パラメーターは可変長のバイト・ストリングであり、要求元アプリケーションと名前付きメソッドが作成された方法と同じ方法で構造化して、認識できるようにすることができます。

NetView プログラムでは、名前付きメソッドのサンプルがソース形式で提供されています。ユーザー独自の名前付きメソッドの作成方法について理解を深めるために、このメソッドを学習してください。サンプルとして提供されているメソッドは、CNMSAMP データ・セットのメンバー EKGMIMV になっています。 このメソッドについては、 552 ページの『RODM の名前付きメソッド』で説明します。

図 88 は、PL/I の場合の名前付きメソッド・パラメーターの例を示しています。図 89 は、C の場合の名前付きメソッド・パラメーターの例を示しています。

```
NamedMeth: Procedure ( Field_ID, LLParms, SLParms );
 Dcl Field_ID     FieldID;          /* Field-identifier of named field  */
 Dcl LLParms     SelfDefiningDataPtr;  /* Pointer to Long-lived field parameters */
 Dcl SLParms     SelfDefiningDataPtr;  /* Pointer to Short-lived Parameter       */
   . . . .
   /* code */
   . . . .
 End;
```

図 88. PL/I の場合の名前付きメソッドのプロシージャー・インターフェース

```
VOID NamedMeth(FieldID                    *in_FieldID,
               SelfDefiningDataPtr        **in_LLParms,
               SelfDefiningDataPtr        **in_SLParms);
....
/* code */
....
```

図 89. C の場合の名前付きメソッドのプロシージャー・インターフェース

名前付きメソッドは、照会メソッドおよび変更メソッドと同じデータにアクセスでき、これらのメソッドと同じ機能をもちます。ただし、名前付きメソッドの明示的な呼び出しは、アプリケーションが RODM を使用して自由に行うことができ、名前付きメソッドが提供する機能の形式は、利用可能なデータおよびサービスによってその機能が実行できる場合には自由です。

オブジェクト特有メソッドの継承

照会、変更、通知、および名前付きの各メソッドは、すべてオブジェクト特有メソッドです。これらのメソッドの中で、RODM オブジェクトのフィールドに値が入っているのは名前付きメソッドだけです。照会、変更、および通知メソッドは、すべてオブジェクトのサブフィールドに保管されます。あるオブジェクトの名前付きメソッド・フィールドとそのフィールドのサブフィールドは、そのオブジェクトの共用クラスのサブフィールドから継承されます。

同様に、名前付きメソッド・フィールドの値と、照会および change サブフィールドの値は、 RODM での継承をサポートする標準原則を使用して、1 次継承によって継承できます。通知メソッドは 1 次親からそのオブジェクト子に継承されます。クラス継承ツリーには継承されません。しかし、オブジェクト・フィールドには、クラス・レベルの通知申請をオーバーライドしないローカル値を追加することができます。 (したがって、値の標準継承は notification サブフィールドには適用されません。)

名前付きメソッド、照会メソッド、変更メソッド、および通知メソッドのすべてをクラスに存在させることもできます。 (オブジェクトの場合と同じように) クラスでの変更メソッドは、変更を行う前にその変更を妥当性検査するために使用したり、ユーザーがそれらの変更を行う権限をもっているのかを検査するために使用したりできます。照会メソッドでは、要求側が要求データを見る権限をもっているかどうかを検査したり、データが戻される前にそのデータを妥当性検査したりできます。同じように、クラスに対する名前付きメソッドは、オブジェクトに対して使用する場合と類似した方法で使用できます。クラスに対する複雑な変更を名前付きメソッドで実行することも、多くの個別クラスに適用できる汎用機能を名前付きメソッドで実行することもできます。また、通知メソッドはクラスでも役に立ちます。

親から値を継承している子に対する変更メソッドおよび通知メソッドは、継承された値が親で変更されても起動されません。したがって、通知メソッドを親 (例えばクラスなど) に適用して、親でパラメーターおよび値が変更されたときにユーザー・アプリケーションが通知を受けられるようにする必要があります。

クラスの 1 次階層の主要な目的は、RODM オブジェクトの編成およびデフォルト値を指定しやすくすることです。オブジェクト・レベルで 1 次階層から継承される値のうちで最も一般的なものとして、以下の値があります。

- 実世界のオブジェクトを反映させるように RODM データの管理を制御するメソッドおよびパラメーター。
- 標準的な限界およびしきい値を表す方針パラメーター。
- 実世界のオブジェクトを管理するために必要な場合には、RODM オブジェクトの長命特性 (容量など)。

これらのメソッドと値はクラスのフィールドに表示されるため、一度指定することにより、 1 次階層によって多くのオブジェクトによって継承されます。

メソッドとしての値が子によって継承されると、そのメソッドが子に関して起動および実行される場合には、実行は子の文脈で行われます。このメソッドは親に存在していますが、継承プロセスによって取り出されるのはその名前と長命パラメーターだけです。このようなメソッドが実行されてフィールドの内容を要求すると、子エンティティーにあるそのフィールドの内容が得られます。

クラスにインストールされた照会、変更、または名前付きメソッドは、2 つの役割を果たすことができます。このメソッドは、子によって継承されるデフォルトの変更メソッドとして使用して、それらの子 (クラスではなくオブジェクトである子も含みます) の文脈で適用することも、親の文脈で標準的な方法 (照会、フィールドの変更、直接呼び出し) によって起動することもできます。

ユーザーが作成するオブジェクト特有メソッドは、オブジェクトで実行される場合とクラスで実行される場合がありますので、注意してください。同じメソッド API 呼び出しを使用して、オブジェクトとクラスの両方で同じ種類の機能を利用することができます。多くのオブジェクト特有メソッドは、そのメソッドが実行される文脈を見つけるために現行エンティティーから WhatIAm フィールドを探すため、文脈が異なると、適切なアクションも異なったものになる可能性があります。

クラスのフィールドに対する照会、変更、名前付き、および通知の各メソッドのフィールドは、オブジェクトに対して起動される場合とまったく同じように、それらのクラスに対するトランザクションの一部として起動されます。また、オブジェクトによる照会、変更、および名前付きメソッドの継承をサポートするためにそれらのメソッドがクラスのフィールドに存在していますが、notification サブフィールドの値の継承は RODM ではサポートされません。

継承によって通知リストが存在している場合、そのリストの初期値はヌル値になります。通知リスト・フィールドがヌル値になっていることは、機能上はリストが存在しないことと同じです。通知リストへの項目の追加は、EKG_AddNotifySubscription 機能を使用して行えます。

つまり、名前付きメソッドと query、change、および notify サブフィールドはすべて、クラスの専用フィールドでも共用フィールドでも標準的な方法で機能します。継承に専用フィールドが関係することはありませんが、照会、変更、および通知メソッドに対応するフィールドが照会または変更されると、それらのメソッドが実行されます。あるフィールドがクラスに存在している場合 (オブジェクトにフィールドがある場合と同じように)、そのフィールドに関する変更トランザクションを実行すると、変更メソッドおよび通知メソッドが起動されますが、フィールドの value サブフィールドに関して変更トランザクションを実行しても変更および通知メソッドは起動されません。この機能は、オブジェクトの場合にサポートされる機能と同じです。

ヌル・メソッド

RODM では、NullMeth と呼ばれる特別なメソッドが提供されています。 NullMeth というオブジェクト ID は、オブジェクト特有メソッドの代わりに使用することができます。 NullMeth は、なにも処理を行わずに呼び出し元に制御権を戻します。NullMeth という値は親クラスからフィールドまたはサブフィールドに継承されます。 MethodSpec タイプのフィールドの値がヌル・メソッドかどうか照会されると、 NullMeth のオブジェクト ID が応答ブロックに戻されます。

NullMeth メソッド名を使用すると、継承された照会または change サブフィールドがなにも行わないように設定できます。この方法の効果は、ローカル・サブフィールドが存在しない場合と同じです。これは、フィールドまたはサブフィールドに関

する標準機能でなんらかのアクションが行われるようになっているときに、若干の例外としてその機能がなにもアクションを行わないようにローカルでオーバーライドされている場合に便利です。

同様に、空の通知リストは、リストが存在していない場合と同じ機能を果たします。対応するフィールドが変更されても、通知メソッドが起動されることはなく、そのイベントはどこにも通知されません。

使用するメソッド・タイプの決定

メソッドを使用する前に、使用する必要のあるメソッドのタイプを判別しなければなりません。使用すべきメソッドのタイプは、どのようなタスクをメソッドで実行したいのかによって異なります。

オブジェクト独立メソッドを使用する場合

RODM データ・キャッシュ内の複数のエンティティーを効率的に操作したい場合は、オブジェクト独立メソッドを使用してください。オブジェクト独立メソッドは、RODM データ・キャッシュ内の任意のクラスまたはオブジェクトに含まれる任意のフィールドを変更または照会できます。

オブジェクト特有メソッドを使用する場合

オブジェクト特有メソッドは、特定のエンティティーに関連付けられたメソッドです。 RODM データ・キャッシュ内の 1 つのエンティティーだけを操作したい場合には、オブジェクト特有メソッドを使用してください。操作対象の特定エンティティーは、実行時に判別され、メソッドが起動されるたびに異なる可能性があります。オブジェクト特有メソッドは、その他のオブジェクトまたはクラスに対するアクションを実行する場合、 EKG_MessageTriggeredAction 機能を使用することができます。また、オブジェクト特有メソッドは、イベントをユーザー・アプリケーションに通知するために通知メソッドを起動することもできます。

オブジェクト特有メソッドには、次の 4 つの種類があります。
- 照会メソッド
- 変更メソッド
- 通知メソッド
- 名前付きメソッド

これらの各メソッドは、特定のタスクを実行するために設計されていて、しかも関連付けられたエンティティーだけでそのタスクを実行することができます。したがって、それ以外のエンティティーのフィールドにはアクセスできません。また、オブジェクト特有メソッドは、それらのメソッドから呼び出せるように設計された API 機能しか呼び出すことができません。オブジェクト特有メソッドで利用できる API 機能のリストについては、 419 ページの『オブジェクト特有メソッドで利用可能なその他のサービス』を参照してください。

照会メソッド

このオブジェクト特有メソッドは、EKG_QueryField API 機能に対する応答としてヌルでない query サブフィールドが照会されたときに起動されます。照会メソッドを使用すると、EKG_QueryField API 機能の呼び出し元に正しい現行データが戻されるようになります。

データが古くなっている可能性のあるエンティティー・フィールドを更新したり、そのフィールドのデータに方針プロシージャー、妥当性検査、またはセキュリティーを適用したりする場合には、このメソッドを使用してください。

変更メソッド

このオブジェクト特有メソッドは、ヌルでない change サブフィールドをもつフィールドが EKG_ChangeField 機能、EKG_ChangeMultipleFields 機能、EKG_LinkTrigger 機能、または EKG_UnlinkTrigger 機能に対する応答として変更されたときに起動されます。変更メソッドを使用すると、機能がデータ・セキュリティー、データ妥当性検査、および方針の要件を適用して、フィールドを正しく変更、リンク、またはリンク解除するようになります。

エンティティー・フィールドのデータに方針プロシージャー、妥当性検査、またはセキュリティーを適用したい場合には、このメソッドを使用してください。

通知メソッド

このオブジェクト特有メソッドは、ヌルでない notify サブフィールドを含むフィールド内の値が変更されたときに起動されます。通知メソッドは、そのフィールドに申請しているアプリケーションに対して、フィールドの値が変更されたことを通知します。

あるアプリケーション・プログラムの操作にとって重要な、エンティティー・フィールド内のフィールド値が変更されたときに、その変更をアプリケーションに通知したい場合には、このメソッドを使用してください。

名前付きメソッド

このオブジェクト特有メソッドは、EKG_TriggerNamedMethod API 機能を呼び出すことによって明示的に起動されます。名前付きメソッドは、特定のエンティティーにあるすべてのフィールドに対して複数の API 機能を実行することができます。RODM は、このメソッドの実行中にエンティティーをロックします。名前付きメソッドが RODM に制御権を戻すまで、他のメソッドまたはアプリケーションは、ターゲット・エンティティーのどのフィールドに対しても照会または変更を行うことができません。

単一エンティティーにおいて複数の API 機能を実行するときに、他のメソッドまたはアプリケーションがそのエンティティーのフィールドを照会または変更できないようにする必要がある場合には、このメソッドを使用してください。

メソッド API の使用

RODM のメソッドを作成するには、RODM のデータおよびサービスへのアクセスが必要です。メソッド API では、メソッドによって呼び出すことができる RODM へのエントリー・ポイントのセットが提供されます。

各種のサービスをメソッドで利用することができます。サービスの中には、オブジェクト独立メソッドだけで利用できるものも、オブジェクト特有メソッドだけで利用できるものもあります。

RODM に対するメソッド API 呼び出しでは、以下のパラメーターが渡されます。

- トランザクション情報ブロック

- 機能ブロック
- 応答ブロック

機能ブロックは、追加パラメーター (例えば、機能のターゲットを識別するエンティティー・アクセス情報ブロックやフィールド・アクセス情報ブロック) を指すことができます。応答ブロックは、いくつかの機能でのみ必要になります。

transaction_info_block、function_block、および response_block の形式は、ユーザー API によって使用されるブロックの形式と同じです。 表 42 には、これらのブロックの詳しい説明が記載されているページが示されています。

表 42. ブロックに関する追加情報

ブロックの種類	参照ページ
Transaction_info_block	350
Response_block	359
Function_block	352

PL/I または C 言語プログラムによる CALL ステートメントは、コード・セグメント EKGMAPI に制御権を移動します。このメソッドは、リンク・エディット・ステップで EKGMAPI モジュールとリンク・エディットしておく必要があります。 図 90 は、PL/I による CALL ステートメントの例を示しています。

```
Declare EKGMAPI Entry( structure, structure, structure );

Call EKGMAPI(  transaction_info_block,
               function_block,
               response_block           /* Null pointer => omitted */
                           );
```

図 90. メソッド API インターフェースの宣言および呼び出し例

レジスター規定

メソッドのコードは、以下のレジスター規定に従って書く必要があります。

レジスター 1

transaction_info_block、function_block、および response_block のアドレスが入っている 3 つの連続したメモリー位置 (パラメーター・リスト) のうちの最初のメモリー位置を指します。

レジスター 12

RODM 実行時環境用に予約されています。このレジスターは、メソッドによって保存する必要があります。 PL/I および C で書かれたコードの場合、このレジスターの要件は生成されたコードと一致します。

レジスター 13

呼び出し元プログラムの 72 バイト保存域のアドレスが入ります。

レジスター 14

呼び出し元プログラムの戻りアドレスが入ります。

レジスター 15
EKGMAPI モジュールの入り口アドレスが入ります。

使用上の注意

すべての RODM 機能に関する詳細は、機能ブロックで指定されます。メソッドは機能ブロックを作成し、必要なトランザクションを要求するためにその機能ブロックを RODM に渡します。メソッド API 機能については、 423 ページの『第 14 章 アプリケーション・プログラミングの解説』で説明されています。

機能ブロックによって指し示される entity_access_information データは、オブジェクト独立メソッドからのメソッド API 呼び出しの場合にも、ユーザー API 呼び出しの場合と同じように解釈されます。しかし、その呼び出しがオブジェクト特有メソッドから行われた場合には、クラスとオブジェクトの情報は無視されます。

オブジェクト特有の変更、照会、通知、および名前付きメソッドは、メソッド API 呼び出しが行われたオブジェクトまたはクラス内のフィールドにしかアクセスできません。

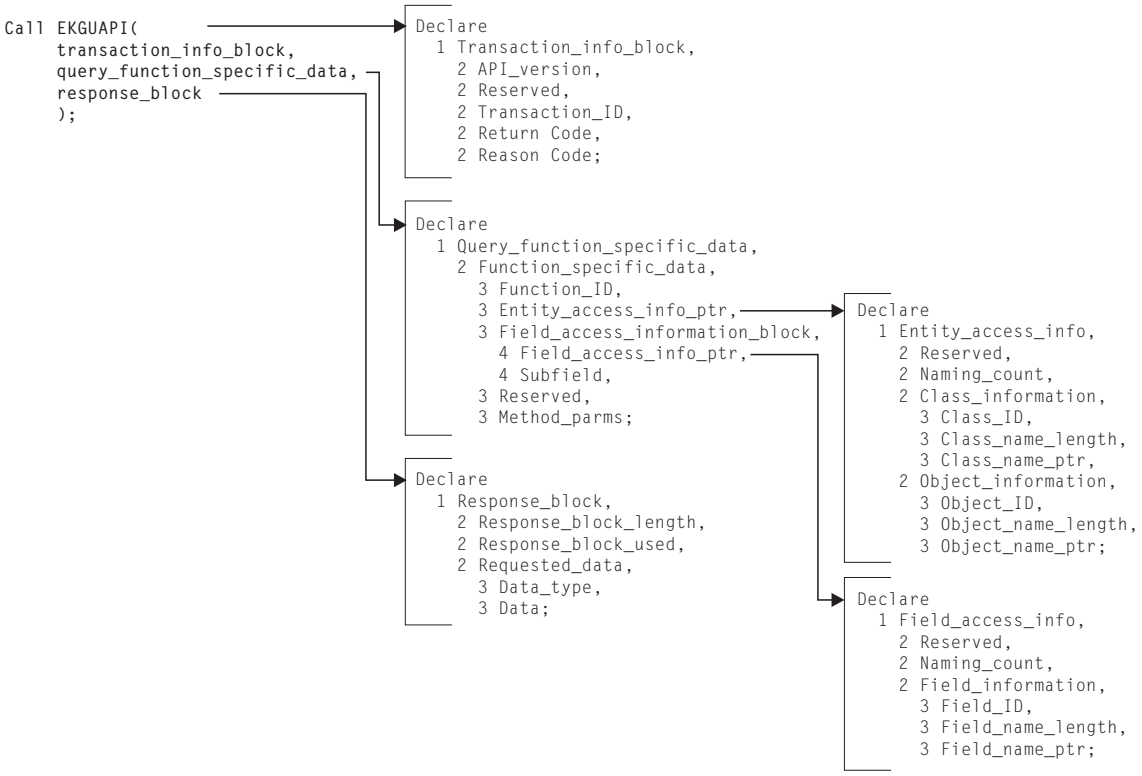


図 91. メソッド API の照会フィールド制御ブロックの例

API 照会機能制御ブロックの例

メソッド・パラメーター

多くのトランザクションでは、メソッドによって渡されたりインストールされたりする任意指定パラメーターが使用されます。メソッド・パラメーターには、次の 2 種類があります。

- 長命パラメーター

- 短命パラメーター

長命パラメーター

長命パラメーターは、静的に定義されるパラメーターです。長命パラメーターは、
- オブジェクト特有メソッドの場合にのみ有効です。
- 可変長の SelfDefining データ・ストリングです。
- 最長 254 バイトまでに制限されています。
- 内部における意味は、ユーザーによって定義され、ユーザーによって解釈されます。

長命パラメーターは、メソッドがサブフィールドに割り当てられるとき (例えば、EKG_AddNotifySubscription 機能によって通知メソッドがインストールされたとき、あるいは、名前付き、照会、または変更メソッドがフィールドまたはサブフィールドに割り当てられたとき) に、そのメソッドによって RODM に保存されます。

これらの長命パラメーターは、ただちに使用されるのではなく、対応するメソッドが (該当の起動機構によって) 実行されるまで保存され、メソッドが実行されるときにそのメソッドによって利用できるようになります。このようにすることにより、汎用メソッドを作成しておいて、そのメソッドがフィールドまたはサブフィールドに割り当てられたときに、指定された必要な機能を提供することができます。

長命パラメーターの形式は可変長の SelfDefining データ・ストリングであり、最大長は 254 バイトです。 254 バイトのデータの内容は RODM によっては指定されません。この内容は、特定のメソッドのインターフェースの仕様によって決まります。実際の SelfDefining データ・ストリングの内容は、フィールドへのメソッド割り当ての際に指定された後では変更できません。ただし、長命パラメーターにオブジェクト内のフィールドに対する参照が含まれている場合、そのフィールドはいつでも変更できます。

短命パラメーター

短命パラメーターは、動的に定義されるパラメーターです。短命パラメーターの特性は、次のようになっています。
- 内部における意味は、オブジェクト特有メソッドの場合にもオブジェクト独立メソッドの場合にも、 API 要求によってそのメソッドが実行されるときにユーザーによって定義され、ユーザーによって解釈される。
- 可変長の SelfDefining データ・ストリング。
- オブジェクト特有メソッドの場合には、254 バイトまでに制限されている。
- オブジェクト独立メソッドの場合には、32767 バイトまでに制限されている。

短命パラメーターは、事前に保管されることはありません。これらのパラメーターは、特定のトランザクション要求 API を介して提供され、そのトランザクションによって起動されるメソッドによってただちに使用できます。これらのパラメーターの形式は、常に可変長の SelfDefining データ・ストリングです。

ユーザー API を介してオブジェクト独立メソッドに渡される短命パラメーターの最大長は 32767 バイトですが、オブジェクト特有メソッドに渡される短命パラメーターは 254 バイトまでに制限されています。これらのストリングの意味は、RODM によって定義されたり制限されたりすることはありません。 RODM はバイト・ス

トリングだけしか調べません。要求元のユーザー・アプリケーションおよび起動されるメソッドは、このバイト・ストリングの内容と矛盾しないように作成されていなければなりません。

メソッドのインストールおよび解放

オブジェクト特有メソッドをオブジェクトのフィールドまたはサブフィールドに割り当てたり、オブジェクト独立メソッドを実行したりする前に、そのメソッドを RODM にインストールする必要があります。メソッドをインストールするためには、EKG_Method クラスのオブジェクトを作成してください。

名前付きメソッドをインストールするには、以下のステップに従ってください。

1. そのメソッドをどこにインストールするのかを決定します。

   名前付きメソッドの場合、クラスまたはオブジェクトで MethodSpec タイプのフィールドを使用しなければなりません。
2. EKG_Method クラスのオブジェクトを作成します。

   このオブジェクトが作成されると、新しく作成されたオブジェクトのオブジェクト ID が戻されます。
3. EKG_ChangeField、EKG_ChangeSubfield、または EKG_ChangeMultipleFields 機能を使用して、MethodSpec フィールドの値をオブジェクト ID に設定し、使用するメソッドに必要な長命パラメーターを設定してください。

RODM のロード機能を使用してメソッドをインストールすることもできます。EKG_Method クラスでオブジェクトを作成すると、RODM がそのメソッドをアドレス・スペースにロードします。このメソッドがインストールされていないうちにフィールドまたはサブフィールドにメソッド名を割り当てようとすると、変更トランザクションからエラー戻りコードが戻されます。

インストール済みのメソッドを変更する必要がある場合には、 EKG_Method クラスの EKG_Refresh フィールドを使用して、そのメソッドの新しいコピーを RODM にロードできます。そのメソッドの新しいコピーをライブラリーからロードするためには、再ロードしたいメソッド・オブジェクトの EKG_Refresh フィールドで指定された名前付きメソッドを起動してください。

あるメソッドが不要になったときには、そのメソッド・オブジェクトに対してオブジェクト削除トランザクションを実行することにより、そのメソッドが占めていたストレージを解放し、内部 RODM 表からそのメソッドの名前とアドレスを除去できます。メソッドを解放できるのは、RODM のどのフィールドまたはサブフィールドにもそのメソッドが値として割り当てられていない場合に限られます。解放されたメソッドは、再インストールされるまでは、フィールドまたはサブフィールドに割り当てることができず、またオブジェクト独立メソッドとして実行することもできません。

他のメソッドは使用する前にインストールしなければなりませんが、ヌル・メソッド NullMeth は常にインストールされていて、解放することができません。NullMeth をインストールまたは解放しようとすると、RODM からエラー戻りコードが戻されます。したがって、メソッド名 NullMeth は RODM で予約されてい

て、ユーザー作成メソッドのために使用することはできません。それ以外の NetView 提供のメソッドは、ユーザー作成メソッドと同様に、使用する前にインストールしなければなりません。

機能の同期実行と非同期実行

あるメソッドが機能または別のメソッドを起動した場合、起動された機能またはメソッドは、起動元のメソッドと同期して実行されます。起動元のメソッドは実行を停止し、起動された機能またはメソッドが終了して戻されるまで、処理を再開しません。メソッド API は EKG_MessageTriggeredAction 機能を提供します。この機能を使用すると、メソッドは、それと非同期的に実行される機能または別のメソッドを起動できるようになります。起動元のメソッドの実行は、起動された機能またはメソッドが開始され、処理を実行して、終了するまで続けられます。

EKG_MessageTriggeredAction 機能は、オブジェクト特有メソッドが、それと関連していない RODM データ・キャッシュ内のエンティティーにアクセスできるようにするための機能ですが、オブジェクト独立メソッドから呼び出すこともできます。また、EKG_MessageTriggeredAction 機能を使用すると、起動元のメソッドと非同期的に以下の機能を実行することができます。

- フィールドまたはサブフィールドの内容の変更またはスワップ
- 2 つのオブジェクトのリンクまたはリンク解除
- フィールド継承の復活
- オブジェクトの作成と削除

メソッド・アンカー・サービス

RODM は、8 バイト作業域を指すポインターを戻す、呼び出し可能なメソッド・アンカー・サービスを備えています。この作業域は、メソッドを呼び出す前にクリアされてゼロになり、そのメソッドによって他のメソッドが起動されたときに作業域の内容が保存されます。

この作業域は、大きくて複雑なメソッドのインストール・モジュール間で通信を行うために使用されることを目的としています。メソッドが実行されるたびに RODM によってクリアされるため、メソッド間の通信には使用できません。

EKGMANC サービス・ルーチンは、PL/I の場合には次のコードを使用して実行してください。

```
DCL WORK_AREA  CHAR(8) BASED(WORK_AREA_PTR);
DCL WORK_AREA_PTR POINTER;
CALL EKGMANC(WORK_AREA_PTR);
```

C の場合には、次のコードを使用してください。

```
char *work_area_ptr;
EKGMANC(&work_area_ptr);
```

EKGMANC 呼び出しからは、戻りコードも理由コードも戻されません。この作業域のアドレスは常に戻されます。

RODM メソッドのコーディング

以下のセクションでは、ユーザー独自のメソッドを作成するための詳細について説明します。これらのセクションには、コンパイラー・オプション、リンク・エディット、および制約事項に関する情報が含まれています。一般的な制約事項と、使用しているプログラム言語の制約事項の両方を必ず検討してください。

インストール先作成メソッド

インストール先作成メソッドは、PL/I または C で書くことができます。これらのメソッドは、PL/I 言語の各国語サポートを使用できます。 DBCS 文字ストリングは、図形定数として操作することができます。

インストール先提供のメソッドは、SBCS または DBCS のいずれの形式で保管された RODM データも参照することができます。

メソッドをコーディングした後で、テスト・データとデバッグ・エイドを使用してそのメソッドを実行し、構文または論理のエラーを検出することができます。詳細については、「*IBM Tivoli NetView for z/OS Programming: PL/I and C*」を参照してください。メソッドのインストールは、RODM の開始 JCL 内の STEPLIB DD ステートメントを指している該当のユーザー・ライブラリーにリンク・エディットすることによって行います。

NetView 提供のメソッド

NetView プログラムには RODM メソッドの基本セットが含まれています。ユーザー独自のメソッドを、PL/I または C のどちらでも作成できます。独自のメソッドによって NetView 提供のメソッドを補完することも、置き換えることもできます。NetView 提供のすべてのメソッドは、 NetView プログラム・プロダクトの CNMLINK ターゲット・ライブラリーに入っています。

現在のところ、RODM で提供されるメソッドには以下のものがあります。

EKGNOTF
変更を通知する通知メソッド。

EKGNLST
変更された値が値のリストのうちのいずれかと同じ場合に通知する通知メソッド。

EKGNEQL
変更された値が特定の値と同じ場合に通知する通知メソッド。

EKGNTHD
変更された値が特定のしきい値に収まっている場合に通知する通知メソッド。

EKGCTIM
オブジェクト独立メソッドを起動してアクションを非同期的に完了させるための変更メソッド。

EKGMIMV
値を増分させるための名前付きメソッド。

EKGSPPI
RODM 自動化プラットフォーム。

すべての通知メソッドは、通知メッセージの原因となったフィールドから得られた現行値、直前の値、およびタイム・スタンプを (これらのサブフィールドが定義されている場合) 通知ブロックに入れて戻します。

RODM のための NetView 提供メソッドについて、以下で機能別に説明します。メソッドに渡されるすべてのパラメーターは、SelfDefining データ・ストリングとして指定されます。

プログラム言語に特有のプリプロセッサー・ステートメント

プログラムをコンパイルしたりソース・コードをリンクしたりするときには、以下のプリプロセッサー・ステートメントを追加してください。

IBM C メソッドのコンパイル

IBM C 言語を使用してメソッドをコンパイルする場合には、次の指針に従ってください。

- 次のプラグマ・ステートメントをコーディングしてください。

  ```
  #pragma linkage(csect,PLI)
  ```

  ただし、csect は外部入り口点の csect の名前です。
- このモジュール内でなんらかの RODM 制御ブロックが参照される場合には、ソース・ファイルにファイル EKG3CINC.H を組み込んでください。このファイルには、RODM のすべての機能と応答ブロック、および RODM 入り口点 EKGMANC、EKGUAPI、EKGMAPI、および EKGWAIT の機能プロトタイプ・ステートメントが含まれています。
- モジュール内で RODM 制御ブロックが参照されていないにもかかわらず、そのモジュールが EKGMANC、EKGUAPI、EKGMAPI、または EKGWAIT を呼び出す場合には、ソース・ファイルにファイル EKG3CEEP.H を組み込んでください。
- コンパイル時には RENT オプションを指定しないでください。

次に示すのは、メソッドをコーディングするための IBM C ソース・コードの例です。

```
#pragma linkage(thisproc,PLI)

#include "EKG3CINC.H"
  /* or */
#include "EKG3CEEP.H"

void thismethod(void arg)
{
   /* code */
}
```

IBM PL/I メソッドのコンパイル

IBM PL/I 言語を使用してメソッドをコンパイルする場合には、次の指針に従ってください。

- このモジュール内でなんらかの RODM 制御ブロックが参照される場合には、ソース・ファイルにファイル EKG1IINC を組み込んでください。このファイルには、RODM のすべての機能と応答ブロック、および RODM 入り口点 EKGMANC、EKGUAPI、EKGMAPI、および EKGWAIT の機能プロトタイプ・ステートメントが含まれています。
- モジュール内で RODM 制御ブロックが参照されていないにもかかわらず、モジュールが EKGMANC または EKGMAPI を呼び出す場合には、ソース・ファイルにファイル EKG1IEEP を組み込んでください。
- コンパイル時には REENTRANT オプションを指定してください。
- メソッドに RODM マクロを組み込む場合は、 MACRO プリプロセッサー・コンパイラー・オプションを指定してください。

次に示すのは、メソッドをコーディングするための IBM PL/I ソース・コードの例です。

```
*PROCESS MACRO;
 thismethod: proc;

%include ekglib(EKG1IINC);
    or
%include ekglib(EKG1IEEP);

/* code */

end thismethod;
```

EKGMAPI を直接呼び出すメソッドのリンク

EKGMAPI を直接呼び出すメソッドをリンクするときには、以下のリンク・エディット制御ステートメントを指定してください。

```
<method object code>

INCLUDE SYSLIB(EKGMAPI)
ENTRY method_name
NAME method_name(R)
```

以下のリンク・エディット・オプションを指定してください。

- AMODE=31
- RMODE=ANY または RMODE=24
- RENT

メソッドの制約事項

すべての RODM メソッドは、デフォルトである PSW キー 8 で実行する必要があります。どのメソッドの PSW キーも変更しないでください。

PL/I 言語の制約事項

PL/I でインストール先定義のメソッドを作成するには、RODM によってサポートされる PL/I コンパイラーが必要です。これらの PL/I プログラムは、特定の呼び出しに関する実行が完了した後でクリーンアップを行うことが予期されています。つまり、動的に割り振られたストレージは、すべて解放されます。さらに、RODM アドレス・スペースで実行される PL/I プログラムは、以下の制約事項に従う必要があります。

- PLITEST の使用

  PLITEST 機能は、RODM アドレス・スペースで実行されるプログラムでは利用できません。
- FETCH および RELEASE の使用

  PL/I プロシージャーは、他の PL/I プロシージャーによってフェッチしたり解放したりすることはできません。ユーザー API によってメソッドの追加および削除がサポートされます。 FETCH および RELEASE の代わりに、これらのサービスを使用することができます。
- DATE 組み込み関数の使用

  PL/I の DATE 組み込み関数は、RODM アドレス・スペースで実行されるプログラムから呼び出すことができません。
- TIME 組み込み関数の使用

  PL/I の TIME 組み込み関数は、RODM アドレス・スペースで実行されるプログラムから呼び出すことができません。
- ファイル入出力の使用

  PL/I のファイル入出力は、RODM アドレス・スペースで実行されるプログラムで使用することはできません。 RODM メソッドは SYSPRINT にはアクセスしません。ただし、ファイル入出力の代わりにログ機能への RODM 出力を使用することができます。
- 言語間通信

  COBOL および FORTRAN ルーチンに対する言語間呼び出しを使用することはできません。 C およびアセンブラーに対する言語間呼び出しだけしか許されていません。
- 遅延

  メソッドの実行を延期することはできません。メソッドはできるだけ早く完了させます。
- 待機

  メソッドの実行を延期することはできません。
- PL/I の DISPLAY ステートメントの使用

  PL/I の DISPLAY ステートメントは、RODM タイプ 1 ログ・レコードに出力を書き込みます。 パフォーマンスおよびログ記録に影響するので、通常は PL/I の DISPLAY ステートメントを使用しないようにしてください。このステートメントの代わりに、EKG_OutputToLog API 機能を使用してください。
- PL/I マルチタスキングの使用

  PL/I マルチタスキング機能は使用できません。タスク管理は、PL/I の機能ではなく RODM の機能によって処理されます。 CALL ステートメントのタスク、イベント、および優先順位オプションは使用することができません。また、COMPLETION、STATUS、および PRIORITY 組み込み関数は使用しないでください。

- MAIN オプションの使用

  PROCEDURE ステートメントの PL/I MAIN オプションを使用してユーザー・メソッドをコーディングすることはできません。
- リンケージ・フィールド

  すべてのメソッドは再入可能でなければなりません。再入可能コードを書くだけでなく、PROCEDURE ステートメントの REENTRANT オプションも使用する必要があります。
- 制御された記憶変数、またはファイル入出力やフェッチ/解放などの疑似レジスター・ベクトルを使用するものは使用できません。
- プログラムで CHECKPOINT、SORT、または PLIDUMP を要求しないでください。
- CHECK および FLOW の PL/I オプションを使用してはなりません。
- ON ユニットおよびシグナルの使用
  - PL/I プログラムでアテンション処理を実行することはできません。その ON ユニットは制御権を受け取ることができません。
  - PL/I プログラムで ERROR または FINISH シグナルを送ってはなりません。
  - PL/I プログラムにエラー時または終了時ステートメントを含めてはなりません。

C 言語における制限事項

メソッドは NORENT オプションを使用してコンパイルしなければなりません。 C プリリンク機能を使用してメソッドをプリリンクしてはなりません。

以下の C 機能は、RODM メソッドでは使用できません。
- Atexit()
- Exit()
- Main()
- すべてのファイルおよびストリーム入出力ステートメントとライブラリー機能

メソッド内のデータには静的ストレージ・クラス指定子を指定しないでください。

ファイル入出力の代わりにログ機能への RODM 出力を使用することができます。

一般的な制約事項

オプション特有メソッドは、そのメソッドが関連付けられているオブジェクトまたはクラスだけしか照会および操作できません。

メソッドに関する制約事項は次のとおりです。
- 名前付きメソッド

  名前付きメソッドは、ユーザー API から直接実行するか、メソッド API を介してオブジェクト独立メソッドから実行するか、またはメソッド API を介して名前付きメソッドから実行すると、呼び出し元と同期的に実行することができます。また、名前付きメソッドは、メソッド API で提供されているメッセージ・インターフェースを介して起動し、呼び出し元と非同期的に実行することもできます。

ユーザー API を介して名前付きメソッドを起動し、非同期的に実行することはできません。

- オブジェクト独立メソッド

  オブジェクト独立メソッドは、ユーザー API またはメソッド API から実行すると、呼び出し元と同期的に実行することができます。また、メソッド API で提供されているメッセージ・インターフェースを介して任意のメソッドから起動して、そのメソッドと非同期的に実行することもできます。

  ユーザー API を介してオブジェクト独立メソッドを起動し、非同期的に実行することはできません。

- 変更メソッド

  変更メソッドは、システム定義フィールドでは使用できません。これらのフィールドの完全なリストについては、243 ページの『システム定義のフィールド』を参照してください。

  LINK フィールド (つまり、データ・タイプが ObjectLink または ObjectLinkList のフィールド) で使用される変更メソッドは、EKG_LinkTrigger および EKG_UnlinkTrigger 機能によって起動されます。これらの変更メソッドの制約事項は以下のとおりです。
  - フィールドを変更することはできません。
  - リンクまたはリンク解除機能を実行することはできません。
  - 戻りコードが非ゼロの場合には、戻りコードを設定しなければなりません。
    - ゼロの戻りコードの場合には、リンクまたはリンク解除を継続することができます。
    - 非ゼロの戻りコードの場合には、リンクまたはリンク解除が禁止されます。
    - 変更メソッドが存在している場合にリンクまたはリンク解除を継続するためには、両方のオブジェクトに対して定義されている変更メソッドからの戻りコードをゼロに設定しなければなりません。

- 通知メソッド

  User_appl_ID、通知メソッド、SubscribeID、および長命パラメーターの特定の組み合わせは、固有の通知メソッドを指定するものであり、特定の notification サブフィールドには 1 回しか割り当てることができません。

- すべてのメソッド
  - すべてのメソッドは、再入可能になるように作成しなければなりません。
  - メソッドは、通知キューを照会したり、それ自体の実行を延期したりすることはできません。
  - RODM が z/OS システム上で作動している場合には、メソッドは、仮想記憶間モードで実行しているアプリケーションに対してオペレーティング・システムが加える制約に従う必要があります。例えば、メソッドは IBM z/Architecture® の SVC 命令を実行する必要のあるサービスを使用してはなりません。

- メソッドが ESTAE、ESTAX、SPIE、または STAE などのリカバリー・ルーチンを使用する場合、そのリカバリー・ルーチンがリカバリー機能を委任するように設定して、異常終了が発生したときに RODM が制御権を受け取るようにしなければなりません。
- 別のメソッドを同期的に実行するためにメソッド API を使用する場合、すでに実行されたメソッドが再帰的に実行されないようにしなければなりません。
- 応答ブロックのオーバーフロー・バッファーをメソッドで利用することはできません。メソッドで提供された応答ブロックが小さすぎて、機能によって戻されたデータを収めきれない場合、応答ブロックに収まらなかったデータは廃棄されます。

RODM メソッドのサービス

RODM 機能の中には、すべてのタイプのメソッドで使用できるものと、特定のタイプのメソッドだけで使用できるものがあります。以下のセクションでは、メソッドのタイプと、それぞれのメソッドで使用できる RODM 機能のリストを示します。

オブジェクト特有メソッドとオブジェクト独立メソッドの両方で利用可能なサービス

プログラムを設計する際に、オブジェクト特有メソッドとオブジェクト独立メソッドの両方で利用可能な以下の機能を使用することができます。

- RODM データの照会
  - EKG_QueryField (470 ページの『EKG_QueryField − フィールドを照会する』を参照)
  - EKG_QueryMultipleSubfields (478 ページの『EKG_QueryMultipleSubfields − 複数の Value サブフィールドを照会する』を参照)
  - EKG_QuerySubfield (486 ページの『EKG_QuerySubfield − サブフィールドを照会する』を参照)
  - EKG_QueryEntityStructure (468 ページの『EKG_QueryEntityStructure − エンティティーの構造を照会する』を参照)
  - EKG_QueryFieldStructure (474 ページの『EKG_QueryFieldStructure − フィールドの構造を照会する』を参照)
  - EKG_QueryFieldID (471 ページの『EKG_QueryFieldID − フィールド ID を照会する』を参照)
  - EKG_QueryFieldName (473 ページの『EKG_QueryFieldName − フィールド名を照会する』を参照)
- RODM データに対するアクション
  - EKG_ChangeField (433 ページの『EKG_ChangeField − フィールドを変更する』を参照)
  - EKG_ChangeMultipleFields (434 ページの『EKG_ChangeMultipleFields − 複数のフィールドを変更する』を参照)
  - EKG_ChangeSubfield (436 ページの『EKG_ChangeSubfield − サブフィールドを変更する』を参照)
  - EKG_RevertToInherited (490 ページの『EKG_RevertToInherited − 継承値を復活させる』を参照)
  - EKG_AddNotifySubscription (430 ページの『EKG_AddNotifySubscription − 通知申請を追加する』を参照)

  - EKG_DeleteNotifySubscription (451 ページの『EKG_DeleteNotifySubscription － 通知申請を削除する』を参照)
  - EKG_TriggerNamedMethod (500 ページの『EKG_TriggerNamedMethod － 名前付きメソッドを起動する』を参照)
- 追加のメソッド・サポート
  - EKG_SendNotification
  - EKG_MessageTriggeredAction
  - EKG_SetReturnCode
  - EKG_OutputToLog
  - EKG_ResponseBlock (名前付きオブジェクト特有メソッド、照会オブジェクト特有メソッド、およびオブジェクト独立メソッドで使用可能)
  - EKG_QueryFunctionBlockContents

この照会機能とアクション機能のリストは、RODM ユーザーがユーザー API を介して利用できるトランザクションのサブセットです。

ユーザー API とメソッド API はともに同じ機能ブロックを使用して、応答ブロックに戻される応答を生成する照会、およびそのような照会を伴うアクションを行うように要求された機能を指定します。また名前付きメソッドは、応答ブロックに戻されるデータを生成することができます。

これらすべての機能ブロックおよび応答ブロックの形式については、343 ページの『第 11 章 RODM を使用するアプリケーションを作成する』を参照してください。ユーザー API の場合と同じように、機能ブロックと応答ブロックが入っているストレージの割り振りおよび解放は、メソッド API のユーザーが行う必要があります。追加のメソッド・サポート機能用のメソッド API の機能ブロックについては、このセクションで説明します。

オブジェクト独立メソッドで利用可能なその他のサービス

以下の追加サービスは、メソッド API およびユーザー API を介してオブジェクト独立メソッドで使用することができます。

- EKG_LinkNoTrigger、EKG_LinkTrigger (460 ページの『EKG_LinkNoTrigger、EKG_LinkTrigger － 2 つのオブジェクトをリンクする』を参照)
- EKG_UnlinkNoTrigger、EKG_UnlinkTrigger (503 ページの『EKG_UnlinkNoTrigger、EKG_UnlinkTrigger － 2 つのオブジェクトをリンク解除する』を参照)
- EKG_CreateObject (445 ページの『EKG_CreateObject － オブジェクトを作成する』を参照)
- EKG_DeleteObject (453 ページの『EKG_DeleteObject － オブジェクトを削除する』を参照)
- EKG_TriggerOIMethod (502 ページの『EKG_TriggerOIMethod － オブジェクト独立メソッドを起動する』を参照)

オブジェクト特有メソッドで利用可能なその他のサービス

以下の追加サービスは、オブジェクト特有メソッドでのみ 使用可能です。

- EKG_WhereAmI

- EKG_QueryObjectName

初期化メソッドで利用可能なサービス

初期化メソッドは、以下の機能を使用することができる唯一のメソッドです。このメソッドは、RODM 初期化時にこれらの機能を実行して、 RODM データ構造を作成し、データを RODM データ・キャッシュにロードすることができます。

- 管理機能
  - EKG_CreateClass (443 ページの『EKG_CreateClass － クラスを作成する』を参照)
  - EKG_CreateField (444 ページの『EKG_CreateField － フィールドを作成する』を参照)
  - EKG_CreateSubfield (447 ページの『EKG_CreateSubfield － サブフィールドを作成する』を参照)
- 制御機能
  - EKG_Checkpoint (437 ページの『EKG_Checkpoint － DASD に RODM チェックポイントを指定する』を参照)

上記のメソッドの機能に対するアクセスは、ユーザー API を介して利用可能なアクセスの場合と類似しています。これらの機能の呼び出しは、メソッド API を使用して RODM を実行することによって行います。これらの機能を使用するためには、標準の機能ブロック定義が必要です。

初期化メソッドで利用可能なメソッド API 機能およびインターフェースには、オブジェクト独立メソッドで使用できるものもあります。ただし、以下の例外があります。初期化メソッドでは、これらの例外については使用しないでください。

- EKG_SendNotification
  - 初期化メソッドの実行時には Notification_queues を登録できないため、この機能は失敗します。
- EKG_ResponseBlock
  - 応答ブロックが初期化メソッドに渡されないため、データが失われます。
- EKG_QueryFunctionBlockContents
  - 初期化メソッドを開始するために機能ブロックが使用されることはありませんので、利用可能なデータがありません。
- EKG_NotificationQueue オブジェクトを作成するための EKG_CreateObject
  - 通知キューは、User_appl_ID をキュー名に連結することによって指名されます。初期化中には User_appl_ID が利用できないため、この機能を初期化に使用すると必ず失敗します。

初期化メソッドが非同期的タスクを開始するためにメッセージ・インターフェースを使用する場合、 RODM の初期化はその非同期的タスクの完了を待たずに続行されます。

RODM メソッド・ライブラリー

メソッド API のサービスにアクセスできるようにするために、RODM には、メソッド API サービスの入り口点を含むライブラリーが備わっています。このライブラリーは RODM メソッド・ライブラリーと呼ばれ、デフォルトの名前は CNMLINK になっています。

このライブラリーは、特に C プログラムと PL/I プログラムでの使用が想定されています。 EKGMAPI などのサービスにアクセスするには、プログラム内で外部入り口として EKGMAPI を宣言してください。外部名を解決するには、CNMLINK ライブラリーを使用してください。

サンプル・ライブラリーに入っているデータ・セット CNMSAMP のメンバー EKGMIMV には、名前付きメソッドから EKGMAPI を呼び出して、指定されたフィールド値を、そのフィールド値を単位として増分させる方法を示す例が含まれています。

第 14 章 アプリケーション・プログラミングの解説

RODM データに対して行われるすべてのトランザクションの詳細は、機能ブロックに指定されています。機能ブロックはユーザーによって作成され、希望するトランザクションを要求するために RODM に渡されます。すべての機能ブロックには、RODM から要求される機能を指定する Function_ID が含まれています。

RODM 機能の要約

この章では、それぞれの RODM 機能について説明します。機能の主要なカテゴリーは、以下のとおりです。

- アクセス機能
- 制御機能
- 管理機能
- アクション機能
- 照会機能
- RODM ユーザー API サービス
- RODM メソッド API サービス

アプリケーション・プログラムで機能ブロックがどのように使用されるのかについては、 343 ページの『第 11 章 RODM を使用するアプリケーションを作成する』を参照してください。 メソッドで機能ブロックがどのように使用されるのかについては、 389 ページの『第 13 章 RODM メソッドの作成』を参照してください。

アクセス機能

アクセス機能を使用すると、ユーザー・アプリケーションと RODM の接続または切断を行うことができます。

EKG_Connect: RODM に接続する
接続機能は、ユーザーを RODM に接続するために呼び出されます。

EKG_Disconnect: RODM から切断する
切断機能は、ユーザーと RODM との接続を終了するために呼び出されます。

制御機能

制御機能を使用すると、該当のアクセス・レベルのユーザー・アプリケーション・プログラムは、 RODM データを DASD に対してチェックポイント設定したり、(データをチェックポイント設定して、あるいはデータをチェックポイント設定しないで) RODM を停止したりできます。

EKG_Checkpoint: RODM チェックポイントを指定する
DASD に RODM データのチェックポイントを指定します。

EKG_Stop: RODM を停止する
RODM サブシステムを停止します。

管理機能

クラス、フィールド、およびサブフィールドを削除または作成するためには、パラメーターとして渡される該当の機能ブロックとともに RODM 管理機能を使用してください。管理呼び出しでは応答ブロックが不要なため、応答ブロック・ポインターはヌルに設定してください。

RODM クラスが最初に作成されたときには、この中には 1 次親のシステム定義フィールドと共用フィールドが含まれています。これらのフィールドの値は、1 次親から継承されます。クラスは、追加フィールドが存在すること、または存在するフィールド内で異なる値が設定されていることにより、親クラスと区別されます。最も一般的には、子クラスは親クラスよりも多くのフィールドを必要とします。これらの追加フィールドは、明示的にクラスに追加しなければなりません。 RODM では、クラスに含めることのできるフィールドの数に制限はありません。

クラスにフィールドを追加することができます。サブフィールドは、すでに存在しているフィールドにしか追加できません。オブジェクトにフィールドを直接追加することはできません。

EKG_CreateClass: クラスを作成する
新しいクラスを RODM データ・キャッシュに作成します。

EKG_CreateField: フィールドを作成する
新しいフィールドをクラスに追加します。

EKG_CreateSubfield: サブフィールドを作成する
新しいサブフィールドをクラスのフィールドに追加します。

EKG_DeleteClass: クラスを削除する
クラスを RODM データ・キャッシュから除去します。

EKG_DeleteField: フィールドを削除する
フィールドをクラスから削除します。

EKG_DeleteSubfield: サブフィールドを削除する
サブフィールドをクラスのフィールドから削除します。

アクション機能

アクション機能は、値を変更し、オブジェクトおよびオブジェクト間のリンクを作成および削除し、通知申請を追加および削除し、また名前付きメソッドとオブジェクト独立メソッドを起動します。 EKG_ExecuteFunctionList 機能を使用してリスト形式でアクション機能を受け渡し、 1 つのインターフェース呼び出しで複数のアクションを行うことができます。

EKG_AddNotifySubscription: 通知申請を追加する
フィールドに申請します。

EKG_AddObjDelSubs: オブジェクト削除申請を追加する
削除の通知を行うようにオブジェクトに申請します。

EKG_ChangeField: フィールドを変更する
フィールドの値を変更します。

EKG_ChangeMultipleFields: 複数のフィールドを変更する
あるオブジェクトの複数のフィールドの値を変更します。

EKG_ChangeSubfield: サブフィールドを変更する
サブフィールドの値を変更します。

EKG_CreateObject: オブジェクトを作成する
RODM データ・キャッシュにオブジェクトを作成します。

EKG_DeleteNotifySubscription: 通知申請を削除する
フィールドへの申請を削除します。

EKG_DeleteObject: オブジェクトを削除する
RODM データ・キャッシュのオブジェクトを削除します。

EKG_DelObjDelSubs: オブジェクト削除申請を削除する
オブジェクトへの申請を削除します。

EKG_LinkNoTrigger: 2 つのオブジェクトをリンクする
2 つのオブジェクトをリンクします。通知メソッドは実行しません。

EKG_LinkTrigger: 2 つのオブジェクトをリンクする
2 つのオブジェクトをリンクします。通知メソッドを実行します。

EKG_RevertToInherited: 継承値を復活させる
データ値のローカル・コピーをフィールドから除去し、継承値によって置き換えます。

EKG_SwapField: フィールドをスワップする
フィールド・データを新規データと比較してスワップします。

EKG_SwapSubfield: サブフィールドをスワップする
サブフィールド・データを新規データと比較してスワップします。

EKG_TriggerNamedMethod: 名前付きメソッドを起動する
名前付きメソッドを実行します。

EKG_TriggerOIMethod: オブジェクト独立メソッドを起動する
オブジェクト独立メソッドを実行します。

EKG_UnlinkNoTrigger: 2 つのオブジェクトをリンク解除する
2 つのオブジェクトをリンク解除します。通知メソッドの実行は行いません。

EKG_UnlinkTrigger: 2 つのオブジェクトをリンク解除する
2 つのオブジェクトをリンク解除します。通知メソッドを実行します。

照会機能

ユーザー・アプリケーション・プログラムは、照会機能を使用することにより、フィールド、サブフィールド、通知キュー、およびアクセス・ブロックに入っている値を照会できます。 EKG_ExecuteFunctionList 機能を使用してリスト形式で照会機能を受け渡し、 1 つのインターフェース呼び出しで複数のアクションを行うことができます。

フィールドの内容または照会対象となる情報は、応答ブロックに入れて戻されます。

オブジェクトまたはクラスのフィールドを照会するときに、そのフィールドと関連付けられた照会メソッドがある場合、フィールドの内容が取り出される前にその照会メソッドが実行されます。そのメソッドは、フィールド内のデータが読み取られ

て呼び出し元に戻される前に、フィールドの内容を変更することが可能です。照会メソッドは、EKG_ResponseBlock 機能を使用することにより、照会操作の戻り値を明示的に設定することができます。照会メソッドが EKG_ResponseBlock 機能を使用する場合、RODM が応答ブロックにデータを入れることはありません。

EKG_Locate: 共用索引付きフィールドを使用してオブジェクトを見つける
指定された探索基準に一致する RODM のすべてのオブジェクトのリストを提供します。

EKG_QueryEntityStructure: エンティティーの構造を照会する
クラスまたはオブジェクト内のすべてのフィールドについて、それぞれのフィールド名、データ・タイプ、および継承状態が指定されたリストを提供します。

EKG_QueryField: フィールドを照会する
フィールドの値を入手します。

EKG_QueryFieldID: フィールド ID を照会する
フィールド名をフィールド ID に変換します。

EKG_QueryFieldName: フィールド名を照会する
フィールド ID をフィールド名に変換します。

EKG_QueryFieldStructure: フィールドの構造を照会する
フィールドの編成 (データ・タイプ、ローカル・コピー・インディケーター、およびサブフィールド・マップ) を提供します。

EKG_QueryMultipleSubfields: 複数の Value サブフィールドを照会する
あるオブジェクトの複数のサブフィールドの値を入手します。

EKG_QueryNotifyQueue: 通知キューを照会する
次のキュー・エレメントが利用可能であれば、そのキュー・エレメントを入手します。

EKG_QueryResponseBlockOverflow : 応答ブロック・オーバーフローを照会する
オーバーフロー応答ブロック・データを入手します。

EKG_QuerySubfield: サブフィールドを照会する
サブフィールドの値を入手します。

RODM ユーザー API サービス

EKG_ExecuteFunctionList: 機能のリストを実行する
ユーザー・アプリケーション・プログラムが、単一の機能呼び出しで RODM 機能のリストを渡せるようにします。

RODM メソッド API サービス

EKG_LockObjectList: オブジェクトのリストをロックする
この API は、オブジェクト独立メソッドが明示的にオブジェクトをロックできるようにするために使用されていました。現在は必要なくなりましたが、互換性のために保持されています。

EKG_MessageTriggeredAction: メッセージを使用してアクションを起動する
オブジェクト特有メソッドが他のオブジェクトまたはクラスに関する非同期 API 機能を起動できるようにします。

EKG_QueryFunctionBlockContents: 機能ブロックの内容を照会する
メソッドに、そのメソッドを起動した機能要求の機能ブロック内容を提供します。

EKG_QueryObjectName: オブジェクト名を照会する
オブジェクト特有メソッドが、ObjectID を対応するオブジェクト名に変換できるようにします。

EKG_OutputToLog: ログに出力する
情報を RODM ログに出力する機能を提供します。

EKG_ResponseBlock: 応答ブロックに出力する
メソッド定義の情報を呼び出し側の応答ブロックに追加します。ただし、照会メソッドの場合には応答ブロックを上書きします。

EKG_SendNotification: 通知を送信する
フィールドが変更されたときに通知情報ブロックを通知キューに送信するための機能を、通知メソッドに提供します。

EKG_SetReturnCode: 戻りコードと理由コードを設定する
メソッドが、メソッドの呼び出し元に戻す戻りコードと理由コードを設定できるようにします。

EKG_UnlockAll: 保持されたエンティティーのロックをすべて解除する
このメソッドは、保持されているすべてのロックを解除するために使用されていました。現在は必要なくなりましたが、互換性のために保持されています。

EKG_WhereAmI: 位置を特定する
オブジェクト特有メソッドが、そのメソッドの起動対象となっているクラス、オブジェクト、およびフィールドを判別できるようにします。

機能の解説

このセクションでは、RODM ユーザー・アプリケーション・プログラム・インターフェースおよび RODM メソッド・アプリケーション・プログラム・インターフェースから利用できる各機能について説明します。このセクションで使用されている形式は、『機能の解説の形式』で説明されています。機能は、機能名のアルファベット順に列挙されています。

機能の解説の形式

このセクションでは、この章で RODM 機能の説明に使用されている形式を説明します。機能は、機能名のアルファベット順に列挙されています。各機能名に続いて、その機能の説明が記載されています。各機能の説明には、以下の解説セクションが含まれています。

- 目的
- 機能ブロックの形式
- 例
- 要約
- 使用法

これらの参照セクションについては、以下のセクションで説明します。

目的

各機能の説明の目的セクションでは、その機能がどのようなことを行うのかを説明します。

機能ブロックの形式

機能ブロックの形式では、その機能に渡す必要のある機能ブロックについて説明します。機能が応答ブロックを戻す場合には、その応答ブロックもこのセクションで説明されています。

機能ブロックの形式の表には、次の 5 つの欄があります。

オフセット
そのパラメーターの先頭までのオフセットを 10 進数バイトで示します。

長さ パラメーターの長さを 10 進数バイトで示します。パラメーターの長さが可変長である場合、長さの欄にはダッシュ (−) が入ります。

タイプ パラメーターの RODM 抽象データ・タイプです。パラメーターの中には、定義された RODM 抽象データ・タイプを使用しないものがいくつかあります。 RODM 抽象データ・タイプを使用しないパラメーターについては、PL/I またはデータ・タイプが示されています。

用途 この用途は、機能へのデータ入力を表す「入力」、または機能によるデータ出力を表す「出力」のいずれかです。予約フィールドおよび特定の機能で使用されないフィールドの場合、用途の欄にはダッシュ (−) が入っています。

パラメーター名
パラメーターの名前です。これらのパラメーターは 507 ページの『機能パラメーターの説明』で説明されています。この名前は、RODM で提供されている機能ブロックまたは応答ブロックの例で使用されている、実際の名前です。

例

例のセクションには、RODM によって機能ごとに提供されているコード例の名前がリストされています。これらの例は、NetView プロダクトとともに出荷されたサンプル・テープに、PL/I と C の両方で提供されています。使用したい機能ごとに、機能ブロックおよび応答ブロックの例をユーザー・アプリケーションまたはメソッドに組み込んでください。機能にアクセスするためには、提供されているパラメーター名を使用してください。これにより、RODM に適用される可能性のあるサービスによるユーザーのプログラムへの影響が限定されます。

PL/I 用の機能ブロックの例と応答ブロックの例には、プリプロセッサー・マクロ置換変数 *EKG_Boundary* が含まれています。 この変数は、PL/I プログラムのために必要な `UNALIGNED BASED(*)` に変換されます。

使用法を示すコーディング例は、各機能の設定方法と呼び出し方法を示すための、実際のコードです。機能の呼び出しについて学習するために、使用法を示すコーディング例を利用してください。ただし、これらの例は、ユーザーのプログラムに組み込むには不適切な可能性があります。

例の表で使用されている名前は、各例のメンバー名です。機能ブロックの例と応答ブロックの例のデフォルト・データ・セット名は、NETVIEW.V5R3M0.SCNMMAC1 です。 使用法を示すコーディング例のデフォルト・データ・セット名は、NETVIEW.V5R3M0.CNMSAMP です。 例えば、EKG_Connect 機能を表す PL/I の機能ブロック例の完全な名前は、NETVIEW.V5R3M0.SCNMMAC1(EKG11101) です。 この機能を表す PL/I の使用法コーディング例の完全な名前は、NETVIEW.V5R3M0.CNMSAMP(EKG51101) です。

要約

要約の表には、各機能について次のトピックがリストされています。

機能 ID
要求されている機能を判別するために RODM で使用される機能 ID。

タイプ アクセスまたは照会などの、機能のタイプ。

ユーザー API
この機能をユーザー・アプリケーションで使用できるかどうかを指定します。

オブジェクト特有メソッド
この機能をオブジェクト特有メソッドで使用できるかどうかを指定します。

オブジェクト独立メソッド
この機能をオブジェクト独立メソッドで使用できるかどうかを指定します。

初期化メソッド
この機能を初期化メソッドで使用できるかどうかを指定します。

起動されるメソッド
この機能が照会、変更、または通知メソッドを起動するかどうか、またどのメソッドを起動するのかを指定します。

EKG_MessageTriggeredAction による起動
この機能を EKG_MessageTriggeredAction 機能によって非同期的に実行できるかどうかを指定します。

許可レベル
この機能を使用するためにユーザー・アプリケーションに割り当てなければならない最低の許可レベルを指定します。

ユーザー・アプリケーションは、特定の RODM 機能を使用するための許可を得ていなければなりません。各機能で、必要な許可レベルが指定されます。アプリケーションは、そのアプリケーションの許可レベル以下の許可レベルを必要とするすべての機能を使用することができます。各アプリケーションの許可レベルは、セキュリティー・システム・プロファイルにアプリケーションの User_appl_ID が作成されるときに指定されます。許可レベルの定義については、「*IBM Tivoli NetView for z/OS* セキュリティー・リファレンス」を参照してください。

使用上の注意

このトピックでは、追加の機能情報および制限が記載されています。

各機能で使用されるパラメーターは、 507 ページの『機能パラメーターの説明』で説明されています。このセクションでは、各パラメーターの一般的な働きを説明し

ます。最大データ長などのような、機能ごとのパラメーターの相違点は、特定機能の使用法のセクションに列挙されています。

RODM 機能によって出される戻りコードおよびそれに関連する理由コードは、 515 ページの『RODM の戻りコードと理由コード』に列挙されています。このセクションには、各機能が使用するすべての理由コード、および特定の理由コードを使用するすべての機能の相互参照表も示されています。この情報は、ユーザー・アプリケーションおよびメソッドで使用するエラー処理ルーチンを設計するために使用できます。

本章の最後のセクションでは、NetView 提供のメソッドについて説明します。これらのメソッドには、RODM で使用可能な通知メソッドと変更メソッドが含まれています。 548 ページの『NetView 提供のメソッド』には、各メソッド、およびそれらのメソッドに渡すパラメーターについて説明されています。

EKG_AddNotifySubscription － 通知申請を追加する

目的

この機能は、オブジェクトまたはクラスのフィールドに通知メソッドを追加します。 RODM は、そのフィールドに関連する申請リストに通知メソッドを含めます。指定された通知キューが存在しない場合、RODM は、指定された User_appl_ID を使用してその通知キューを作成します。

機能ブロックの形式

表 43. EKG_AddNotifySubscription 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Pointer	入力	Entity_access_info_ptr
008	4	Pointer	入力	Field_access_info_ptr
012	8	ApplicationID	入力	User_appl_ID
020	8	SubscribeID	入力	Notification_queue
028	8	Anonymous(8)	入力	User_word
036	8	ObjectID	入力	Notify_method
044	4	SelfDefiningDataPtr	入力	Long_lived_parm

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 44. EKG_AddNotifySubscription 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11412
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG51412

表 44. EKG_AddNotifySubscription 機能用の例の名前 (続き)

例	名前
C 機能ブロック	EKG31412
C 応答ブロック	不要
C 使用法コーディング	EKG61412

要約

表 45. EKG_AddNotifySubscription 機能の要約

機能 ID	1412
タイプ	アクション
ユーザー API	可
オブジェクト特有メソッド	可
オブジェクト独立メソッド	可
初期化メソッド	可
起動されるメソッド	通知キュー・オブジェクトが作成されている場合、 EKG_NotificationQueue クラスの MyObjectChildren フィールドの通知メソッドが起動される。
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	3

使用法

通知申請の詳細については、 364 ページの『RODM 通知プロセス』を参照してください。

通知申請は User_appl_ID、Notification_queue、メソッド・オブジェクト ID、および Long_lived_parm からなり、フィールドに 1 回追加されます。同じ情報が指定された 2 回目の要求が送信されると、その要求はリジェクトされます。

機能ブロックから得られたクラス、オブジェクト、およびフィールドのアクセス情報には、その申請をインストールすべき場所が指定されています。指定されたフィールドの value サブフィールドが EKG_ChangeField 機能または EKG_ChangeMultipleFields 機能によって変更されると、要求された通知メソッドが実行されます。

通知メソッドが実行されると、機能ブロックから得られた Long_lived_parm フィールドの値が通知メソッドに提供されます。このメソッドでは、Long_lived_parm を修正することはできません。

ユーザーは、オブジェクト・フィールドが変更されたときにオブジェクトとその親クラスの両方が実行される場合、通知申請をその両方に割り当てることができます。これらの通知が追加されるときには、RODM は、クラスとオブジェクトに重複する申請が追加されていないかどうかを検査しません。 重複した申請は、個々のクラスまたはオブジェクト・レベルでなければリジェクトされません。

EKG_AddObjDelSubs － オブジェクト削除申請を追加する

目的

この機能は、オブジェクトに削除申請を追加します。そのオブジェクトが削除されると、 RODM は通知ブロックをユーザーに送信します。

機能ブロックの形式

表 46. EKG_AddObjDelSubs 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Pointer	入力	Entity_access_info_ptr
008	8	ApplicationID	入力	User_appl_ID
016	8	SubscribeID	入力	Notification_queue
024	8	Anonymous(8)	入力	User_word
032	4	SelfDefiningDataPtr	入力	Long_lived_parm

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 47. EKG_AddObjDelSubs 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11417
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG51417
C 機能ブロック	EKG31417
C 応答ブロック	不要
C 使用法コーディング	EKG61417

要約

表 48. EKG_AddObjDelSubs 機能の要約

機能 ID	1417
タイプ	アクション
ユーザー API	可
オブジェクト特有メソッド	可
オブジェクト独立メソッド	可
初期化メソッド	不可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	3

使用法

削除通知申請は User_appl_ID、Notification_queue、および Long_lived_parm からなり、オブジェクトに 1 回追加されます。同じ情報が指定された 2 回目の要求が送信されると、その要求はリジェクトされます。

この機能ブロックから得られたオブジェクト・アクセス情報には、申請をインストールすべき場所が指定されています。指定したオブジェクトが EKG_DeleteObject 機能によって削除されると、通知ブロックがユーザー・アプリケーションに送信されます。通知ブロックの内容は、EKG_QueryNotifyQueue 機能からの出力です。詳細については、481 ページの『EKG_QueryNotifyQueue － 通知キューを照会する』を参照してください。

EKG_ChangeField － フィールドを変更する

目的

この機能は、オブジェクトまたはクラスのフィールドの値を変更します。この機能は、そのフィールドで定義されている変更または通知メソッドを起動します。

機能ブロックの形式

表 49. EKG_ChangeField 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Pointer	入力	Entity_access_info_ptr
008	4	Pointer	入力	Field_access_info_ptr
012	2	Smallint	入力	Subfield
014	2	Smallint	入力	Data_type
016	4	Integer	入力	New_char_data_length
020	4	Pointer	入力	New_data_ptr
024	4	SelfDefiningDataPtr	入力	Method_parms

オフセット 012 の Subfield パラメーターは、現在使用されていません。

上記のパラメーターの詳細については、507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、257 ページの『要約データ・タイプ参照』を参照してください。

例

表 50. EKG_ChangeField 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11401
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG51401
C 機能ブロック	EKG31401
C 応答ブロック	不要

表 50. EKG_ChangeField 機能用の例の名前 (続き)

例	名前
C 使用法コーディング	EKG61401

要約

表 51. EKG_ChangeField 機能の要約

機能 ID	1401
タイプ	アクション
ユーザー API	可
オブジェクト特有メソッド	可
オブジェクト独立メソッド	可
初期化メソッド	可
起動されるメソッド	変更および通知メソッドが起動される
EKG_MessageTriggeredAction 機能による起動	可
許可レベル	3

使用法

New_data_ptr によって指し示される新しい値は、変更対象のターゲット・フィールドと同じデータ・タイプでなければなりません。新しい値は、そのデータ・タイプに合わせて正しく形式設定しなければなりません。 Data_type フィールドには、ターゲット・フィールドと同じデータ・タイプを指定しなければなりません。

データ・タイプが ObjectID、ObjectIDList、ObjectLink、ObjectLinkList、ClassID、ClassIDList、または ClassLinkList になっているフィールドをこの機能で変更することはできません。これらのフィールドは、RODM によって、または LINK および UNLINK トランザクションによって設定されます。

MyName や MyID のような読取専用アクセスの RODM システム定義フィールドは、この機能では変更できません。

複数のフィールド値は、EKG_ChangeMultipleFields 機能を使用して変更することができます。

EKG_ChangeMultipleFields − 複数のフィールドを変更する

目的

この機能を使用すると、あるオブジェクトの複数のフィールドの値を変更することができます。この機能は、そのフィールドで定義されている変更および通知メソッドを起動します。

機能ブロックの形式

表 52. EKG_ChangeMultipleFields 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID

表 52. EKG_ChangeMultipleFields 機能用の機能ブロック (続き)

オフセット	長さ	タイプ	用途	パラメーター名
004	4	Pointer	入力	Entity_access_info_ptr
008	4	Integer	入力	Number_of_fields
注: 構造体配列の第 1 エレメント				
012	4	Pointer	入力	Field_access_info_ptr
016	2	Anonymous(2)	-	予約済み
018	2	Smallint	入力	Data_type
020	4	Integer	入力	New_char_data_length
024	4	Pointer	入力	New_data_ptr
028	4	SelfDefiningDataPtr	入力	Method_parms
032	4	Integer	入力	Return_code
036	4	Integer	入力	Reason_code

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 53. EKG_ChangeMultipleFields 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11419
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG51419
C 機能ブロック	EKG31419
C 応答ブロック	不要
C 使用法コーディング	EKG61419

要約

表 54. EKG_ChangeMultipleFields 機能の要約

機能 ID	1419
タイプ	アクション
ユーザー API	可
オブジェクト特有メソッド	可
オブジェクト独立メソッド	可
初期化メソッド	可
起動されるメソッド	変更および通知メソッドが起動される
EKG_MessageTriggeredAction 機能による起動	可
許可レベル	3

使用法

New_data_ptr によって指し示される新しい値は、変更対象のターゲット・フィールドと同じデータ・タイプでなければなりません。新しい値は、そのデータ・タイプに合わせて正しく形式設定しなければなりません。 Data_type フィールドには、ターゲット・フィールドと同じデータ・タイプを指定しなければなりません。

データ・タイプが ObjectID、ObjectIDList、ObjectLink、ObjectLinkList、ClassID、ClassIDList、または ClassLinkList になっているフィールドをこの機能で変更することはできません。これらのフィールドは、RODM によって、または LINK および UNLINK トランザクションによって設定されます。

MyName や MyID のような読取専用アクセスの RODM システム定義フィールドは、この機能では変更できません。

EKG_ChangeSubfield − サブフィールドを変更する

目的

この機能を使用すると、変更および通知メソッドを起動せずにサブフィールドの値を変更することができます。

機能ブロックの形式

表 55. EKG_ChangeSubfield 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Pointer	入力	Entity_access_info_ptr
008	4	Pointer	入力	Field_access_info_ptr
012	2	Smallint	入力	Subfield
014	2	Smallint	入力	Data_type
016	4	Integer	入力	New_char_data_length
020	4	Pointer	入力	New_data_ptr
024	4	—	—	使用されません

上記のパラメーターの詳細については、507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、257 ページの『要約データ・タイプ参照』を参照してください。

例

表 56. EKG_ChangeSubfield 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11403
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG51403
C 機能ブロック	EKG31403
C 応答ブロック	不要

表 56. EKG_ChangeSubfield 機能用の例の名前 (続き)

例	名前
C 使用法コーディング	EKG61403

要約

表 57. EKG_ChangeSubfield 機能の要約

機能 ID	1403
タイプ	アクション
ユーザー API	可
オブジェクト特有メソッド	可
オブジェクト独立メソッド	可
初期化メソッド	可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	可
許可レベル	3

使用法

value サブフィールドを変更する場合、新しいデータのデータ・タイプはフィールドのデータ・タイプと同じでなければなりません。それ以外のサブフィールドを変更する場合、サブフィールドのデータ・タイプはサブフィールド・タイプによって決定され、RODM は、機能ブロック内の data_type フィールドが指定のサブフィールドと互換性があるかどうかを検査します。

value サブフィールドを変更しても、prev_val および timestamp サブフィールドは変更されず、また変更または通知メソッドを実行することもありません。

EKG_Checkpoint − DASD に RODM チェックポイントを指定する

目的

この機能を使用すると、RODM がストレージ内のデータのコピーをチェックポイント・データ・セットに書き込みます。このチェックポイント・データ・セットは、システム障害の後で RODM データをリカバリーするために使用してください。

機能ブロックの形式

表 58. EKG_Checkpoint 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 59. EKG_Checkpoint 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11201
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG51201
C 機能ブロック	EKG31201
C 応答ブロック	不要
C 使用法コーディング	EKG61201

要約

表 60. EKG_Checkpoint 機能の要約

機能 ID	1201
タイプ	制御
ユーザー API	可
オブジェクト特有メソッド	不可
オブジェクト独立メソッド	不可
初期化メソッド	可
起動されるメソッド	通知
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	4

使用法

EKG_Checkpoint 機能は、事前に定義され割り振られている VSAM 線形データ・セットに RODM を書き込みます。このデータ・セットは、RODM チェックポイント・データ・セットと呼ばれます。

チェックポイント機能は、メンバー EKGCUST 内の CHECKPOINT_FUNCTION ステートメントを使用して制御されます。このステートメントは、チェックポイント機能を使用不可にするか、またはチェックポイントが失敗した際のチェックポイント機能の対応を制御する場合に使用してください。詳細については、「*IBM Tivoli NetView for z/OS アドミニストレーション・リファレンス*」を参照してください。

EKG_Checkpoint 機能がチェックポイント・データ・セットに書き込むデータには、以下のものが含まれます。

- RODM マスター・ウィンドウ − RODM アドレス・スペースに存在する RODM データ域であり、 RODM システム情報が入っています。 RODM マスター・ウィンドウのデータは、マスター・ウィンドウ・チェックポイント・ファイルに書き込まれます。
- RODM 変換ウィンドウ − RODM アドレス・スペースに存在する RODM データ域であり、 RODM データ・キャッシュにデータを正しくマッピングおよびア

ドレッシングできるようにするアドレス情報が入っています。 RODM 変換ウィンドウのデータは、変換ウィンドウ・チェックポイント・ファイルに書き込まれます。

- RODM データ・ウィンドウ − データ・スペースに存在する RODM データ域であり、データ・キャッシュ内の実際のデータが入っています。 RODM データ・ウィンドウのデータは、データ・ウィンドウ・チェックポイント・ファイルに書き込まれます。

チェックポイント処理は、次のステップで行われます。

1. チェックポイントの開始 − RODM はメッセージをコンソールに送り、 RODM が静止していることをオペレーターに通知します。
2. 静止 − チェックポイント静止期間中、RODM はメソッド API 要求を許可しますが、新しいユーザー API 要求は拒否します。静止期間が終了したときに、ユーザー API、メソッド API、または非同期トランザクションが実行されていない場合には、RODM はチェックポイント処理における次のステップ (第 1 段階のチェックポイント) に進みます。これらの実行がまだ行われている場合には、RODM は要応答オペレーター向け書き込み (WTOR) メッセージを出し、オペレーターからの指示を要求します。その場合、オペレーターは、次の 3 つのオプションのうちの 1 つを選択する必要があります。

   オプション
   意味

   1 再び静止を行います。このオプションは、チェックポイントがどうしても必要な場合に選択してください。しかし、静止を正常に行うために RODM に別の静止期間を指定してください。

   2 第 1 段階のチェックポイントを無条件に開始します。このオプションは、チェックポイントがただちに必要な場合、またはオプション 1 を試みた後で選択してください。

   3 チェックポイント要求を停止します。このオプションは、オプション 1 を試行した後、または、RODM の重要なタスクを停止してはならない場合に選択します。
3. 第 1 段階のチェックポイント − 静止期間が終了して、すべてのトランザクションの処理が終了したか、またはオペレーターが無条件のチェックポイントを要求した場合、 RODM はマスター・ウィンドウおよび変換ウィンドウをそれぞれのチェックポイント・ファイルに書き込みます。
4. 第 2 段階のチェックポイント − 第 1 段階のチェックポイントが終了した後で、 RODM は、トランザクションを再開できるようになったことを通知するメッセージをコンソールに送ります。 RODM はその後で、データ・ウィンドウ・チェックポイント・ファイルにデータ・ウィンドウを一度に 1 つずつ書き込み始めます。このチェックポイント段階では、ユーザー・アプリケーションによってトランザクション要求を出すことができます。ただし、アクセスする必要のある特定のデータ・ウィンドウがデータ・ウィンドウ・チェックポイント・ファイルに書き込まれている最中の場合、またはまだデータ・ウィンドウ・チェックポイント・ファイルに書き込まれていない場合には、トランザクションは失敗します。
5. チェックポイントの終了 − すべてのデータ・ウィンドウがデータ・ウィンドウ・チェックポイント・ファイルに書き込まれると、 RODM はメッセージをコ

ンソールに送り、チェックポイント処理が終了したこと、およびチェックポイント処理が成功したかどうかに応じて 2 つの EKG_System オブジェクト・フィールドが更新されることを通知します。

チェックポイント処理が成功した場合には、 EKG_System オブジェクトの EKG_LastCheckpointID フィールドが RODM によって更新され、最後のチェックポイント・トランザクションのトランザクション ID を反映するようになります。それ以外の場合、EKG_LastCheckpointID フィールドは変更されません。

EKG_System オブジェクトの EKG_LastCheckpointResult フィールドは、MODIFY コマンドから出されたチェックポイント処理の現行トランザクション ID、またはチェックポイント処理を要求したユーザー API のトランザクション ID によって更新されます。また、EKG_LastCheckpointResult フィールドは、チェックポイント処理の結果を、戻りコードおよび理由コードを使用して反映させます。このフィールドに申請したアプリケーション・プログラムは、チェックポイント処理が完了したことを示す通知を受け取ります。

チェックポイント処理を除き、RODM ユーザー API で出されるすべてのトランザクションは、それが完了するまでユーザーが実行の制御権を取り戻さない点で、同期的なトランザクションです。チェックポイント処理の場合には、チェックポイント要求が記録されるとアプリケーションが制御権を取り戻します。実際には、チェックポイント操作はアプリケーション内の他の処理と非同期的に処理されます。オペレーターが要求したチェックポイント処理にも、これと同じチェックポイント処理に関する非同期処理が MODIFY コマンドを介して適用されます。

チェックポイント制御のコーディング: RODM は、チェックポイント操作を完了するたびに EKG_System クラスの EKG_LastCheckpointResult フィールドを更新します。 EKG_LastCheckpointResult フィールドには、チェックポイント操作を要求したトランザクションのトランザクション ID と、そのチェックポイント操作の結果を表す戻りコードおよび理由コードが入ります。アプリケーションはこのフィールドに申請して、各チェックポイント操作の完了について通知を受けるようにすることができます。

EKG_LastCheckpointResult フィールドに申請して、チェックポイントの結果についての通知を受けるようにします。これにより、ユーザーはそのフィールドを照会して、チェックポイント操作の結果を判別できるようになります。チェックポイント操作が正常に行われなかった場合、ユーザーはチェックポイント処理が失敗した原因を判別することができます。

ユーザー・アプリケーションは、RODM に関するトランザクションのレコードまたはジャーナルを保持することができます。チェックポイント操作とチェックポイント操作の間で RODM に障害が起こった場合、アプリケーションは、RODM によるチェックポイント操作の対象となっていたトランザクション、および再送信する必要のあるトランザクションを判別することができます。そのジャーナル内のトランザクションのうちで EKG_LastCheckPointID フィールド以下の数値のトランザクションは、すべて正常に完了したチェックポイント操作のチェックポイント・データ・セットに反映され、ジャーナルから消去することができます。
EKG_LastCheckPointID フィールドよりも大きな数値のトランザクションはすべて再送信して、RODM を障害以前の状況に復元する必要があります。

チェックポイント操作が開始されてから第 1 段階が完了するまでの間、RODM は追加のトランザクション要求を拒否し、キーワード TRANSPARENT_CHECKPOINT=NO がカスタマイズ・ファイルで指定されている場合は、この状況を表す戻りコードおよび理由コードを戻します。

ユーザー・アプリケーションは、EKG_AddNotifySubscription 機能を使用して、EKG_LastCheckpointID フィールド、EKG_LastCheckpointResult フィールド、またはその両方に申請することができます。 430 ページの『EKG_AddNotifySubscription − 通知申請を追加する』を参照してください。この申請を行うために、NetView 提供の通知メソッド EKGNOTF を使用することができます。 EKGNOTF の説明については、 548 ページの『RODM の通知メソッド』を参照してください。

EKG_Connect − RODM に接続する

目的

接続機能を使用すると、アプリケーション・プログラムで RODM を使用できるようになります。これは、アプリケーションが RODM に対して出すことのできる最初の機能です。

機能ブロックの形式

表 61. EKG_Connect 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	8	Char(8)	入力	User_password
012	4	Pointer	入力	Stop_ECB
016	8	TransID	出力	Last_checkpoint_ID
024	4	Anonymous(4)	—	予約済み

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 62. EKG_Connect 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11101
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG51101
C 機能ブロック	EKG31101
C 応答ブロック	不要
C 使用法コーディング	EKG61101

要約

表 63. EKG_Connect 機能の要約

機能 ID	1101
タイプ	アクセス
ユーザー API	可
オブジェクト特有メソッド	不可
オブジェクト独立メソッド	不可
初期化メソッド	不可
起動されるメソッド	通知
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	1

使用法

User_appl_ID は、ユーザー・アクセス権限を判別したり、登録済みの ECB を適切なユーザーに関連付けたりするために使用します。

RODM のインストールが行われているシステムがシステム許可機能によって保護されている場合、ユーザーはブランクのユーザー ID を使用して RODM に接続することができます。 RODM はシステム許可機能からユーザー ID を取得し、それを使用してユーザーの RODM におけるアクセス権限を判別します。そのシステムがシステム許可機能によって保護されていない場合、ユーザーはブランクのユーザー ID によって RODM に接続することはできません。

ユーザー・アプリケーションが EKG_Connect 機能要求を出すと、 RODM は EKG_User システム定義クラスからユーザー・オブジェクトを作成します。

348 ページの『アクセス・ブロック』に説明してあるように、アクセス・ブロックが渡されなければなりません。アクセス・ブロック内のユーザーの sign_on_token パラメーターは、RODM によって設定されます。このパラメーターは、RODM に対するその後の呼び出しのため、ユーザー・アプリケーションで変更してはなりません。

ユーザーは、申請通知キューを除去せずに RODM から切断することができます。このユーザー・アプリケーション ID によって所有されている通知キューに再び通知する前に、このユーザーのすべての通知キューおよび申請通知に関連付けられているすべての ECB アドレスを、新しいアドレス・スペース用にリセットする必要があります。

EKG_Connect 機能が出されたアドレス・スペース内のすべてのタスクは、固有の RODM 許可ユーザー ID を指定して RODM に接続することによって、あるいは sign_on_token を使用することによって、RODM にアクセスできます。
sign_on_token は、接続元の TCB が終了しているとき、または EKG_Disconnect 機能が実行されるときには無効です。

EKG_CreateClass − クラスを作成する

目的

この機能は、RODM データ・キャッシュ内で、指定された親クラスの子として新しいクラスを作成します。 RODM は、新しいクラスの親の MyClassChildren リンク・リスト・フィールドに新しいクラス ID 項目を追加します。

機能ブロックの形式

表 64. EKG_CreateClass 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Pointer	入力	Class_access_info_ptr
008	4	Pointer	入力	Parent_access_info_ptr
012	4	SelfDefiningDataPtr	入力	Method_parms

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 65. EKG_CreateClass 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11302
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG51302
C 機能ブロック	EKG31302
C 応答ブロック	不要
C 使用法コーディング	EKG61302

要約

表 66. EKG_CreateClass 機能の要約

機能 ID	1302
タイプ	管理
ユーザー API	可
オブジェクト特有メソッド	不可
オブジェクト独立メソッド	不可
初期化メソッド	可
起動されるメソッド	親クラスの MyClassChildren および WhatIAm フィールドで通知メソッドが起動される。
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	5

使用法

クラス名を提供すると、RODM は関連する ID を戻します。

クラスとともに作成されるのは、システム定義フィールド、および 1 次階層を介して継承されたフィールドだけです。これらの追加フィールドは、RODM への呼び出しによって明示的に追加しなければなりません。

クラスを作成すると、その親に子クラスがなかった場合には、親の WhatIAm フィールドの値が変更されます。

EKG_CreateField − フィールドを作成する

目的

この機能は、RODM データ・キャッシュ内のクラスに新しいフィールドを作成します。

機能ブロックの形式

表 67. EKG_CreateField 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Pointer	入力	Class_access_info_ptr
008	4	Pointer	入力	Field_access_info_ptr
012	2	Smallint	入力	Field_type_flag
014	2	Smallint	入力	Data_type
016	4	Bit(32)	入力	Subfield_map

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 68. EKG_CreateField 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11304
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG51304
C 機能ブロック	EKG31304
C 応答ブロック	不要
C 使用法コーディング	EKG61304

要約

表 69. EKG_CreateField 機能の要約

機能 ID	1304

表 69. EKG_CreateField 機能の要約 (続き)

タイプ	管理
ユーザー API	可
オブジェクト特有メソッド	不可
オブジェクト独立メソッド	不可
初期化メソッド	可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	5

使用法

フィールドの初期値は、フィールドのデータ・タイプのヌル値です。

フィールドが作成されるとき、RODM は次の規則を適用します。

- クラスに追加されるフィールドが共用フィールドであって、それと同じ名前およびフィールド (つまり、データ・タイプおよびサブフィールドの定義) がすでにサブクラスに定義されている場合、追加フィールドは指定されたクラスに定義され、サブクラスで定義されていたフィールドはそのフィールドのローカル値として取り扱われます (これにより、そのサブクラスのもとに継承される値が影響を受けます)。サブクラス内のフィールドのデータ・タイプが新しいデータ・タイプと異なっている場合、新しい定義は拒否されます。
- 追加される新規フィールドが専用フィールドである場合、サブクラス定義に関する検査は行われません。
- 新規のフィールド定義が共用フィールドの定義であって、指定されたクラスのサブクラス内にすでに専用定義が存在する場合、新しいフィールド定義は拒否されます。

このフィールドがすでに存在していて、既存フィールドのデータ・タイプおよびサブフィールド定義が要求されたフィールドとまったく同じである場合、警告用の戻りコードが生成され、さらにその条件を示す理由コードが戻されます。元のフィールドは、すでに定義されているまま残されます。

無効なサブフィールドが指定された場合、RODM はそのサブフィールドを作成しません。ただし、そのフィールドおよび要求されたすべての有効なサブフィールドの作成は行います。 RODM は、警告用の戻りコード 4 と理由コード 100 を戻します。

EKG_CreateObject − オブジェクトを作成する

目的

この機能は、RODM データ・キャッシュ内に新しいオブジェクトを作成します。RODM は、新しいオブジェクトの親の MyObjectChildren リンク・リスト・フィールドに新しいオブジェクト ID 項目を追加します。

機能ブロックの形式

表 70. EKG_CreateObject 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Pointer	入力	Entity_access_info_ptr
008	4	SelfDefiningDataPtr	入力	Method_parms

上記のパラメーターの詳細については、507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、257 ページの『要約データ・タイプ参照』を参照してください。

例

表 71. EKG_CreateObject 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11409
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG51409
C 機能ブロック	EKG31409
C 応答ブロック	不要
C 使用法コーディング	EKG61409

要約

表 72. EKG_CreateObject 機能の要約

機能 ID	1409
タイプ	アクション
ユーザー API	可
オブジェクト特有メソッド	不可
オブジェクト独立メソッド	可
初期化メソッド	可 [1]
起動されるメソッド	親クラスの MyClassChildren および WhatIAm フィールドで通知メソッドが起動される。
EKG_MessageTriggeredAction 機能による起動	可
許可レベル	5 (メソッド・オブジェクトの作成) 3 (その他のオブジェクトの作成)

注: [1] 初期化メソッドでは、EKG_NotificationQueue クラスのオブジェクトを作成することはできません。

使用法

Entity_access_info_ptr は、作成されるオブジェクトの親クラスを指定するエンティティー・アクセス・ブロックを指し示す必要があります。エンティティー・アクセス・ブロックの Object_name_ptr は任意指定です。Object_name_ptr を指定する場合

には、要求された新しいオブジェクトの名前が入っている ObjectName タイプのフィールドを指す必要があります。指定しなかった場合には、RODM が新しいオブジェクトに値を割り当てます。

EKG_Method クラスまたは EKG_NotificationQueue クラスのオブジェクトを作成する場合、オブジェクト名は必ず指定してください。これらのクラスのオブジェクト名は 8 文字までに制限されています。

このインターフェースによってオブジェクト名が呼び出し元に戻されることはありませんが、オブジェクトの MyName フィールドを照会することにより、オブジェクト名にアクセスできます。 RODM は EKG*ddddddd* 形式で名前を割り当てます。この *ddddddd* は、0000000 から 9999999 までの 10 進数です。オブジェクト名を指定する場合、EKG で始まるオブジェクト名は指定しないでください。

オブジェクトが正常に作成されると、エンティティー・アクセス・ブロックの Object_ID フィールドが RODM によって設定されます。 Method_Parms の short_lived_parameters がクラスの MyObjectChildren フィールドにある通知メソッドに渡され、通知メソッドが存在する場合にはそれが起動されます。

新しく作成されたオブジェクトには、その新規オブジェクトのクラスで示されるすべての共用のローカル定義フィールドおよび継承フィールドが含まれます。 RODM によって設定されたシステム定義フィールド、および空のフィールドである ObjectLink タイプのフィールドを除き、これらのフィールドに最初に入っている値はクラスから継承したデフォルト値です。

notify サブフィールドを除くすべてのサブフィールドは、最初は、継承された値が入った状態で新しいオブジェクトに存在します。 notify サブフィールドは、最初はヌル値になっています。

このオブジェクトが作成されたときに、親クラスにオブジェクト子がない場合、RODM はクラスの WhatIAm フィールドを更新して、そのクラスがオブジェクト子をもつようになったことを表します。

EKG_CreateSubfield − サブフィールドを作成する

目的

この機能は、既存クラス内の既存フィールドに関する 1 つまたは複数のサブフィールドを RODM データ・キャッシュ内に作成します。

機能ブロックの形式

表 73. EKG_CreateSubfield 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Pointer	入力	Class_access_info_ptr
008	4	Pointer	入力	Field_access_info_ptr
012	4	Bit(32)	入力	Subfield_map

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 74. EKG_CreateSubfield 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11306
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG51306
C 機能ブロック	EKG31306
C 応答ブロック	不要
C 使用法コーディング	EKG61306

要約

表 75. EKG_CreateSubfield 機能の要約

機能 ID	1306
タイプ	管理
ユーザー API	可
オブジェクト特有メソッド	不可
オブジェクト独立メソッド	不可
初期化メソッド	可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	5

使用法

サブフィールドは、クラスの既存フィールドにだけ作成することができます。サブフィールドは、そのフィールドが作成されたクラスで作成しなければなりません。

指定されたサブフィールドのうちの 1 つがすでに存在していて、その他の指定されたサブフィールドが存在していない場合、存在していないサブフィールドが作成され、警告用の戻りコードが生成されます。

EKG_DeleteClass − クラスを削除する

目的

この機能は、RODM データ・キャッシュから既存のクラスを削除します。 RODM は、削除されたクラスの MyID フィールドの値を、削除されたクラスの親の MyClassChildren リンク・リスト・フィールドから除去します。

機能ブロックの形式

表 76. EKG_DeleteClass 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Pointer	入力	Class_access_info_ptr
008	4	SelfDefiningDataPtr	入力	Method_parms

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 77. EKG_DeleteClass 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11303
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG51303
C 機能ブロック	EKG31303
C 応答ブロック	不要
C 使用法コーディング	EKG61303

要約

表 78. EKG_DeleteClass 機能の要約

機能 ID	1303
タイプ	管理
ユーザー API	可
オブジェクト特有メソッド	不可
オブジェクト独立メソッド	不可
初期化メソッド	不可
起動されるメソッド	親クラスの MyClassChildren および WhatIAm フィールドで通知メソッドが起動される。
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	5

使用法

RODM システム定義クラス、または子をもつクラスを削除することはできません。

クラスを削除することによってその親クラスにクラス子がなくなると、親クラスの WhatIAm フィールドの値が変更されます。

EKG_DeleteField − フィールドを削除する

目的

この機能は、RODM データ・キャッシュ内のクラスからフィールドを削除します。このフィールドは、このクラスからそのフィールドを継承したクラスおよびオブジェクトからも削除されます。

機能ブロックの形式

表 79. EKG_DeleteField 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Pointer	入力	Class_access_info_ptr
008	4	Pointer	入力	Field_access_info_ptr

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 80. EKG_DeleteField 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11305
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG51305
C 機能ブロック	EKG31305
C 応答ブロック	不要
C 使用法コーディング	EKG61305

要約

表 81. EKG_DeleteField 機能の要約

機能 ID	1305
タイプ	管理
ユーザー API	可
オブジェクト特有メソッド	不可
オブジェクト独立メソッド	不可
初期化メソッド	不可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	5

使用法

フィールドは、クラスからのみ削除することができます。オブジェクトから削除することはできません。

クラス上の共用フィールドを削除すると、すべての下層クラスからそのフィールドの存在が除去されます。

クラスから共用フィールドを削除できるようにするには、そのクラスおよびそのクラスの下層のクラスから作成されたすべてのオブジェクトを削除しなければなりません。

フィールドに割り当てられたローカル値は、そのフィールドが削除されると廃棄されます。

専用フィールドはいつでも削除することができます。

EKG_DeleteNotifySubscription － 通知申請を削除する

目的

この機能は、フィールドから 1 つまたは複数の通知申請を削除します。

機能ブロックの形式

表 82. EKG_DeleteNotifySubscription 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Pointer	入力	Entity_access_info_ptr
008	4	Pointer	入力	Field_access_info_ptr
012	24	RecipientSpec	入力	Subscription_info
036	8	ObjectID	入力	Notify_method
044	4	SelfDefiningDataPtr	入力	Long_lived_parm

上記のパラメーターの詳細については、507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、257 ページの『要約データ・タイプ参照』を参照してください。

例

表 83. EKG_DeleteNotifySubscription 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11413
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG51413
C 機能ブロック	EKG31413
C 応答ブロック	不要
C 使用法コーディング	EKG61413

要約

表 84. EKG_DeleteNotifySubscription 機能の要約

機能 ID	1413
タイプ	アクション
ユーザー API	可
オブジェクト特有メソッド	可
オブジェクト独立メソッド	可
初期化メソッド	可
起動されるメソッド	通知メソッドが起動される。
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	3

使用法

通知申請を削除しても、削除機能を出した時点で通知キューで待機していた通知ブロックは削除されません。通知キュー・オブジェクトは削除されません。

削除対象の通知申請は、User_appl_ID、Notification_queue、Notify_method field、および Long_lived_parm の 4 つのフィールドによって固有に識別されます。EKG_DeleteNotifySubscription 機能は、これらの 4 つのフィールドを使用して、次のうちの該当する最初の規則に基づいて 1 つまたは複数の通知申請を削除します。

1. Notification_queue フィールドがアスタリスクとそれに続く 7 桁のブランク ("* ") に設定されていて、 Notify_method フィールドと Long_lived_parm フィールドがヌル値に設定されている場合、指定された User_appl_ID フィールドに関連付けられたすべての申請が削除されます。
2. Notification_queue フィールドがアスタリスクとそれに続く 7 桁のブランク ("* ") に設定されている場合、指定された User_appl_ID、Notify_method、および Long_lived_parm の各フィールドに関連付けられたすべての申請が削除されます。
3. Notify_method フィールドがヌル値に設定されている場合、 RODM は Notify_method フィールド内の値にかかわらず、他の基準を満たす通知申請を削除します。
4. Long_lived_parm フィールドにヌル値が設定されている場合、 RODM は Long_lived_parm フィールド内の値にかかわらず、他の基準を満たす通知申請を削除します。

User_appl_ID をヌル値に指定した場合の効果は、その他のパラメーターにヌル値を指定した場合とは異なります。ヌルの User_appl_ID 値は、EKG_AddNotifySubscription 機能の場合と同じように解釈されます。 RODM でデフォルト値を提供する必要があります。このデフォルトは、EKG_AddNotifySubscription 機能の場合とまったく同じように解釈されます (430 ページの『EKG_AddNotifySubscription － 通知申請を追加する』を参照してください)。

ヌルの Long_lived_parm を指定するには、値がゼロの Long_lived_parm データ・タイプを指すポインターを宣言してください。

EKG_DeleteObject − オブジェクトを削除する

目的

この機能は、指定されたクラスから既存のオブジェクトを削除します。 RODM は、削除されたオブジェクトの親クラスの MyObjectChildren フィールドから、削除されたオブジェクトのオブジェクト ID を削除します。

機能ブロックの形式

表 85. EKG_DeleteObject 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Pointer	入力	Entity_access_info_ptr
008	4	SelfDefiningDataPtr	入力	Method_parms

上記のパラメーターの詳細については、507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、257 ページの『要約データ・タイプ参照』を参照してください。

例

表 86. EKG_DeleteObject 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11410
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG51410
C 機能ブロック	EKG31410
C 応答ブロック	不要
C 使用法コーディング	EKG61410

要約

表 87. EKG_DeleteObject 機能の要約

機能 ID	1410
タイプ	アクション
ユーザー API	可
オブジェクト特有メソッド	不可
オブジェクト独立メソッド	可
初期化メソッド	可
起動されるメソッド	オブジェクト・クラスの MyClassChildren および WhatIAm フィールドで通知メソッドが起動される。
EKG_MessageTriggeredAction 機能による起動	可
許可レベル	5 (メソッド・オブジェクトの削除) 3 (その他のオブジェクトの削除)

使用法

Method_parms データは、オブジェクト・クラス上の MyObjectChildren および WhatIAm フィールドに割り当てられた任意の通知メソッドに渡されます。

このオブジェクトを削除する前に、ターゲット・オブジェクトのすべてのフィールドから他のオブジェクトへの、すべての ObjectLink タイプのリンクを削除しなければなりません。まだ ObjectLink タイプのリンクが存在している場合、RODM はエラーを戻します。

このオブジェクトが削除された後で、このオブジェクトの親クラスにオブジェクト子が存在しなくなった場合、RODM はクラスの WhatIAm フィールドを更新して、そのクラスに子がなくなったことを表します。

EKG_DeleteSubfield － サブフィールドの削除

目的

この機能は、RODM データ・キャッシュ内のクラスの指定されたフィールドから、1 つまたは複数のサブフィールドを削除します。これらのサブフィールドは、フィールドが作成されたクラス内のフィールドから削除しなければなりません。 RODM は、指定されたフィールドを継承したクラスまたはオブジェクトからもそれらのサブフィールドを削除します。

機能ブロックの形式

表 88. EKG_DeleteSubfield 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Pointer	入力	Class_access_info_ptr
008	4	Pointer	入力	Field_access_info_ptr
012	4	Bit(32)	入力	Subfield_map

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 89. EKG_DeleteSubfield 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11307
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG51307
C 機能ブロック	EKG31307
C 応答ブロック	不要
C 使用法コーディング	EKG61307

要約

表 90. EKG_DeleteSubfield 機能の要約

機能 ID	1307
タイプ	管理
ユーザー API	可
オブジェクト特有メソッド	不可
オブジェクト独立メソッド	不可
初期化メソッド	不可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	5

使用法

サブフィールドの削除は、それが作成されたクラスからしか行えません。親クラスでサブフィールドが定義されている場合、そのサブフィールドを継承している子からではなく、その親クラスからサブフィールドを削除しなければなりません。

value サブフィールドを削除することはできません。 Subfield_map ビット 1 の値は、この機能の場合には常に 0 (ゼロ) でなければなりません。

RODM に対して複数のサブフィールドの削除を指示したときに、存在していないサブフィールドが指定されている場合、RODM は警告を戻します。しかし、それ以外の存在するサブフィールドは削除されます。

共用フィールドのサブフィールドをクラスから削除するためには、そのクラスおよびそのクラスの下位クラスから、すべてのオブジェクトを削除しておかなければなりません。

EKG_DelObjDelSubs － オブジェクト削除申請を削除する

目的

この機能は、あるオブジェクトに関する削除申請を削除します。

機能ブロックの形式

表 91. EKG_DelObjDelSubs 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Pointer	入力	Entity_access_info_ptr
008	8	ApplicationID	入力	User_appl_ID
016	8	SubscribeID	入力	Notification_queue
024	8	Anonymous(8)	入力	User_word
032	4	SelfDefiningDataPtr	入力	Long_lived_parm

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 92. EKG_DelObjDelSubs 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11418
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG51418
C 機能ブロック	EKG31418
C 応答ブロック	不要
C 使用法コーディング	EKG61418

要約

表 93. EKG_DelObjDelSubs 機能の要約

機能 ID	1418
タイプ	アクション
ユーザー API	可
オブジェクト特有メソッド	可
オブジェクト独立メソッド	可
初期化メソッド	不可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	3

使用法

削除申請を削除しても、その削除機能を発行した時点で通知キューに入っている通知ブロックは削除されません。通知キュー・オブジェクトは削除されません。

削除対象の申請は、User_appl_ID、Notification_queue、および Long_lived_parm の 3 つのフィールドによって固有に識別されます。 EKG_DelObjDelSubs 機能は、これらの 3 つのフィールドを使用して、以下のうちで該当する最初の規則に基づき、1 つまたは複数の削除申請を削除します。

1. Notification_queue フィールドがアスタリスクとそれに続く 7 桁のブランク ("* ") に設定されていて、 Long_lived_parm フィールドがヌル値に設定されている場合、指定された User_appl_ID フィールドに関連付けられたすべての申請が削除されます。
2. Notification_queue フィールドがアスタリスクとそれに続く 7 桁のブランク ("* ") に設定されている場合、指定された User_appl_ID フィールドおよび Long_lived_parm フィールドに関連付けられたすべての申請が削除されます。

3. Long_lived_parm フィールドにヌル値が設定されている場合、 RODM は Long_lived_parm フィールド内の値にかかわらず、他の基準を満たす通知申請を削除します。

User_appl_ID をヌル値に指定した場合の効果は、その他のパラメーターにヌル値を指定した場合とは異なります。ヌルの User_appl_ID 値は、EKG_AddObjDelSubs 機能の場合と同じように解釈されます。 RODM でデフォルト値を提供する必要があります。このデフォルトは、EKG_AddObjDelSubs 機能の場合とまったく同じように解釈されます (507 ページの『機能パラメーターの説明』を参照してください)。

ヌルの Long_lived_parm を指定するには、値がゼロの Long_lived_parm データ・タイプを指すポインターを宣言してください。

EKG_Disconnect − RODM から切断する

目的

この機能は RODM からユーザー・アプリケーションを切断します。

機能ブロックの形式

表 94. EKG_Disconnect 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 95. EKG_Disconnect 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11102
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG51102
C 機能ブロック	EKG31102
C 応答ブロック	不要
C 使用法コーディング	EKG61102

要約

表 96. EKG_Disconnect 機能の要約

機能 ID	1102
タイプ	アクセス
ユーザー API	可
オブジェクト特有メソッド	不可
オブジェクト独立メソッド	不可

表 96. EKG_Disconnect 機能の要約 (続き)

初期化メソッド	不可
起動されるメソッド	通知
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	1

使用法

RODM からの切断後は、EKG_Connect 機能要求を出すまで、RODM は、切断されたアクセス・ブロックを使用した他の機能要求を受け入れません。

RODM から切断するときの通知キューおよび通知申請の処理は、ユーザー・オブジェクトの EKG_StopMode フィールドを設定することによって制御します。後で再接続する予定がない場合には、ユーザー・オブジェクトの EKG_StopMode を 1 に設定して、すべての通知申請を削除してください。 231 ページの『EKG_User クラス』の EKG_StopMode フィールドを参照してください。

切断を行うと、ユーザー・アプリケーション ID に関する通知キューのうち、アクティブになっているすべてのもの (EKG_NotificationQueue クラスの対応するオブジェクトの EKG_Status が 1 になっているもの) は、引き続き通知ブロックを蓄えます。後で再接続する場合、既存のすべての通知キュー・オブジェクト内で通知 ECB (EKG_ECBAddress フィールド) を再設定してからでなければ、通知を受け取ることはできません。

RODM からの切断を行ったときに、すべての申請が削除されていて (あるいは設定されている申請がなくて) 通知キューが除去されている場合には、ユーザー・オブジェクトは削除されます。

EKG_ExecuteFunctionList — 機能のリストを実行する

目的

この機能は、単一のインターフェース呼び出しによって RODM 機能のリストを実行します。 RODM は、機能リストを管理して、呼び出し中に他のトランザクションによってターゲット・エンティティーが影響を受けないようにします。

機能ブロックの形式

表 97. EKG_ExecuteFunctionList 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Integer	入力	Number_of_Functions
注: 構造体配列の第 1 エレメント				
008	0	Structure	—	Function_info_array
008	4	Pointer	入力	Function_block_ptr
012	4	Pointer	出力	Response_block_reference
016	4	Integer	出力	Response_block_used
020	4	Integer	出力	Return_code

表 97. EKG_ExecuteFunctionList 機能用の機能ブロック (続き)

オフセット	長さ	タイプ	用途	パラメーター名
024	4	Integer	出力	Reason_code
注: 構造体配列の第 2 エレメント (使用する場合)				
028	0	Structure	—	Function_info_array
028	4	Pointer	入力	Function_block_ptr
032	4	Pointer	出力	Response_block_reference
036	4	Integer	出力	Response_block_used
040	4	Integer	出力	Return_code
044	4	Integer	出力	Reason_code

注: 機能ブロックには Number_of_functions 配列エレメントが含まれます。

表 98. EKG_FunctionList 機能用の応答ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Response_block_length
004	4	Integer	出力	Response_block_used
008	—	—	出力	Response_data

注: 機能がデータを戻さない場合は、応答ブロックは不要です。
Response_block_used は、全機能について合計したものです。機能ブロックには、個々の機能によって使用される量が含まれます。

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 99. EKG_ExecuteFunctionList 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11600
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG51600
C 機能ブロック	EKG31600
C 応答ブロック	不要
C 使用法コーディング	EKG61600

要約

表 100. EKG_ExecuteFunctionList 機能の要約

機能 ID	1600
タイプ	ユーザー API サービス
ユーザー API	可
オブジェクト特有メソッド	不可

表 100. EKG_ExecuteFunctionList 機能の要約 (続き)

オブジェクト独立メソッド	不可
初期化メソッド	不可
起動されるメソッド	可
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	2 (照会のリストのみ) 3 (アクションを含むリスト)

許可レベル: EKG_ExecuteFunctionList はアクション機能と照会機能だけを実行することができます。これらのアクション機能および照会機能には、3 を超える許可レベルを割り当てることはできません。

使用法

EKG_ExecuteFunctionList 機能用のトランザクション情報ブロックに戻される戻りコードおよび理由コードは、個々の機能に関する最高の戻りコードおよびそれに対応する理由コードです。

RODM は、機能リストを管理して、呼び出し中に他のトランザクションによってターゲット・エンティティーが影響を受けないようにします。

応答ブロックのオーバーフロー状態が生じると、出力長の値 (response_block_used パラメーター) はすべて RODM によって設定されますが、オーバーフロー・バッファーに完全に収まったトランザクション結果のポインター値 (例えば、response_block_reference パラメーター) はヌル値に設定されます。オーバーフロー・ブロックを検索する際には、元の呼び出しで戻された長さ情報を使用してそのデータを解析してください。

ユーザーが使用を許可されていない機能がリストに含まれている場合には、それらの機能はスキップされ (アクションは試みられません)、それらの機能に関してエラー戻りコードおよび理由コードが設定されます。

EKG_LinkNoTrigger、EKG_LinkTrigger － 2 つのオブジェクトをリンクする

目的

これらの機能は、2 つのオブジェクト上にある 2 つのフィールド間でリンクを確立するために使用されます。 EKG_LinkTrigger 機能は変更メソッドおよび通知メソッドを起動し、 EKG_LinkNoTrigger 機能は起動しません。

機能ブロックの形式

表 101. EKG_LinkNoTrigger 機能および EKG_LinkTrigger 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Pointer	入力	Entity_access_info_ptr_1
008	4	Pointer	入力	Field_access_info_ptr_1
012	4	Pointer	入力	Entity_access_info_ptr_2

表 101. EKG_LinkNoTrigger 機能および EKG_LinkTrigger 機能用の機能ブロック (続き)

オフセット	長さ	タイプ	用途	パラメーター名
016	4	Pointer	入力	Field_access_info_ptr_2
020	4	SelfDefiningDataPtr	入力	Method_parms[1]

注:

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 102. EKG_LinkNoTrigger 機能および EKG_LinkTrigger 機能用の例の名前

例	名前
PL/I 機能ブロック (EKG_LinkTrigger)	EKG11405
PL/I 機能ブロック (EKG_LinkNoTrigger)	EKG11406
PL/I 応答ブロック	不要
PL/I 使用法コーディング (EKG_LinkTrigger)	EKG51405
PL/I 使用法コーディング (EKG_LinkNoTrigger)	EKG51406
C 機能ブロック (EKG_LinkTrigger)	EKG31405
C 機能ブロック (EKG_LinkNoTrigger)	EKG31406
C 応答ブロック	不要
C 使用法コーディング (EKG_LinkTrigger)	EKG61405
C 使用法コーディング (EKG_LinkNoTrigger)	EKG61406

要約

表 103. EKG_LinkNoTrigger 機能および EKG_LinkTrigger 機能の要約

機能 ID	
EKG_LinkNoTrigger	1406
EKG_LinkTrigger	1405
タイプ	アクション
ユーザー API	可
オブジェクト特有メソッド	不可
オブジェクト独立メソッド	可
初期化メソッド	可
起動されるメソッド EKG_LinkTrigger	変更メソッドおよび通知メソッド
EKG_LinkNoTrigger	不可
EKG_MessageTriggeredAction 機能による起動	可
許可レベル	3

使用法

リンクは、オブジェクト内のフィールドでのみ行うことができます。クラスのフィールドはリンクできません。リンクされるフィールドは、異なるオブジェクト上になければなりません。

リンクされる 2 つのフィールドのタイプは、ともに ObjectLink または ObjectLinkList になっていなければなりません。 1 つのリンクだけが必要な場合には、ObjectLink フィールドを使用してください。 1 つのフィールドについて複数のリンクが必要な場合には、ObjectLinkList フィールドを使用してください。

ObjectLinkList タイプのフィールド内のリンクの順序は予想できません。

すでに他のフィールドにリンクされている、タイプが ObjectLink のフィールドでリンクを実行すると、リンク機能は失敗します。

すでに他のフィールドにリンクされている、タイプが ObjectLinkList のフィールドでリンクを実行すると、リンク機能は成功します。リンク先のフィールドのタイプも ObjectLinkList である場合、そのリンクは追加され、それ以前のリンクは保存されます。

GMFHS リソースで EKG_LinkNoTrigger を使用することはできません。

EKG_LinkTrigger 機能が出されると、通知メソッドが起動される前にリンク操作が実行されます。リンクされるフィールドの一方または両方で変更メソッドが定義されている場合には、以下のうちの 1 つが当てはまる場合にかぎり、変更メソッドの後でリンクが行われます。

- 両方の変更メソッドが、EKG_SetReturnCode でゼロの戻りコードを明示的に設定している。
- どちらの変更メソッドでも戻りコードを設定していない。この場合、RODM によってゼロの戻りコードが想定され、リンクが続行されます。

リンクが続行されない場合には、通知メソッドは起動されません。オブジェクトが正常にリンクされると、次の順序で通知メソッドが起動されます。

1. Field_access_info_ptr_1 によって指定されたフィールドに関する通知メソッド
2. Field_access_info_ptr_2 によって指定されたフィールドに関する通知メソッド
3. 第 1 フィールドの親クラスに関する通知メソッド
4. 第 2 フィールドの親クラスに関する通知メソッド

EKG_Locate − 共用索引付きフィールドを使用してオブジェクトを見つける

目的

この機能は、RODM 内にあるオブジェクトのうちで探索基準に一致するすべてのオブジェクトのオブジェクト ID のリストを戻します。探索基準は、public_indexed として定義されている文字フィールドの値として指定されます。共用索引フィールドおよび EKG_Locate 機能の説明については、 253 ページの『索引付きフィールド』を参照してください。

機能ブロックの形式

表 104. EKG_Locate 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Anonymous(4)	—	予約済み。X'00000000' でなければなりません。
008	4	Pointer	入力	Field_access_info_ptr
012	2	Smallint	入力	Data_type。4 または 32 でなければなりません。
014	2	Anonymous(2)	—	予約済み。X'0000' でなければなりません。
016	4	Integer	入力	Indexed_data_length
020	4	Pointer	入力	Indexed_data_ptr

表 105. EKG_Locate 機能用の応答ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Response_block_length
004	4	Integer	出力	Response_block_used
008	—	ObjectIDList	出力	Requested_data

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 106. EKG_Locate 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11509
PL/I 応答ブロック	EKG21509
PL/I 使用法コーディング	EKG51509
C 機能ブロック	EKG31509
C 応答ブロック	EKG41509
C 使用法コーディング	EKG61509

要約

表 107. EKG_Locate 機能の要約

機能 ID	1509
タイプ	Query
ユーザー API	可
オブジェクト特有メソッド	可
オブジェクト独立メソッド	可
初期化メソッド	可

表 107. EKG_Locate 機能の要約 (続き)

起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	2

使用法

EKG_Locate 機能は、オブジェクトがどのクラスにあるのかに関係なく、指定されたフィールドのある RODM 内のすべてのオブジェクト上で処理します。

EKG_Locate 機能は、public_indexed 専用に作成された、データ・タイプが CharVar および IndexList のフィールドで作動します。 DisplayResourceName というフィールドで EKG_Locate 機能を使用すると、 RODM は、フィールドまたは探索基準の case とは無関係に、探索基準を満たすすべてのオブジェクトのオブジェクト ID を戻します。 DBCS 値の場合、予期しない一致が生じる可能性があります。

EKG_LockObjectList − オブジェクトのリストをロックする

目的

この機能は、これまで、オブジェクトのリストを明示的にロックするために使用されていました。現在では RODM は自動的にロックを制御するようになったので、この機能は不要になりました。この機能は、既存のアプリケーションとの互換性を維持するために残されています。この機能を必要とする既存のアプリケーションを変更する必要はありません。

機能ブロックの形式

表 108. EKG_LockObjectList 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Integer	入力	Object_list_length
注: 構造体配列の第 1 エレメント				
008	0	Structure	—	Object_array
008	8	ObjectID	入力	Object_ID
016	4	Integer	出力	Reason_code[1]
注: 構造体配列の第 2 エレメント (使用する場合)				
020	0	Structure	—	Object_array
020	8	ObjectID	入力	Object_ID
028	4	Integer	出力	Reason_code[1]

注: 機能ブロックには Object_list_length 配列エレメントが入っています。

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 109. EKG_LockObjectList 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG12002
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG52002
C 機能ブロック	EKG32002
C 応答ブロック	不要
C 使用法コーディング	EKG62002

要約

表 110. EKG_LockObjectList 機能の要約

機能 ID	2002
タイプ	メソッド API サービス
ユーザー API	不可
オブジェクト特有メソッド	不可
オブジェクト独立メソッド	可
初期化メソッド	可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	不要

使用法

既存のアプリケーションとの互換性のために、 Reason_code フィールドには常に値 0 が戻されます。

EKG_MessageTriggeredAction － メッセージを使用してアクションを起動する

目的

この機能は、RODM 機能を非同期的に実行します。この機能を使用すると、オブジェクト特有メソッドがデータ・キャッシュ内の他のオブジェクトに対して作用できるようになります。

機能ブロックの形式

表 111. EKG_MessageTriggeredAction 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Pointer	入力	Function_block_ptr

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 112. EKG_MessageTriggeredAction 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG12009
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG52009
C 機能ブロック	EKG32009
C 応答ブロック	不要
C 使用法コーディング	EKG62009

要約

表 113. EKG_MessageTriggeredAction 機能の例

機能 ID	2009
タイプ	メソッド API サービス
ユーザー API	不可
オブジェクト特有メソッド	可
オブジェクト独立メソッド	可
初期化メソッド	可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	不要

使用法

EKG_MessageTriggeredAction 機能は、必ずしもすべての機能を実行できるわけではありません。各機能の要約の表の 『EKG_MessageTriggeredAction 機能による起動』の項目に、その機能が EKG_MessageTriggeredAction 機能で実行できるかどうかが示されています。

EKG_MessageTriggeredAction 機能を使用するメソッドは、その機能要求が RODM によって受け入れられたかどうかを示す戻りコードと理由コードを受け取ります。ただし、いつアクションがとられるのかは、そのメソッドには分かりません。EKG_MessageTriggeredAction 機能によって起動されるメソッドおよび実行される機能に関する問題を検出するには、EKG_System クラスと EKG_User クラスの EKG_LastAsyncError フィールドに申請してください。詳細については、 372 ページの『非同期エラー通知』を参照してください。

EKG_MessageTriggeredAction 機能によって実行される機能は、呼び出し元のメソッドに応答ブロックを戻すことができません。

この機能は、オブジェクト特有メソッドで使用することを目的としたものであり、オブジェクト特有メソッドが、そのメソッドと関連付けられているオブジェクト以外のオブジェクトに対して作用できるようにします。しかし、オブジェクト独立メソッドでもこの機能を使用することができます。

EKG_OutputToLog − ログに出力する

目的

この機能は、現行の RODM ログ・データ・セットにログ・レコードを書き込みます。これにより、メソッドでエラーまた診断情報を記録できるようになります。

機能ブロックの形式

表 114. EKG_OutputToLog 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Pointer	入力	Log_message
008	2	Smallint	入力	Message_CCSID

上記のパラメーターの詳細については、507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、257 ページの『要約データ・タイプ参照』を参照してください。

例

表 115. EKG_OutputToLog 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG12008
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG52008
C 機能ブロック	EKG32008
C 応答ブロック	不要
C 使用法コーディング	EKG62008

要約

表 116. EKG_OutputToLog 機能の要約

機能 ID	2008
タイプ	メソッド API サービス
ユーザー API	不可
オブジェクト特有メソッド	可
オブジェクト独立メソッド	可
初期化メソッド	可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	不可

表 116. EKG_OutputToLog 機能の要約 (続き)

許可レベル	不要

使用法

RODM は、メソッドが文字ストリング (タイプ 1 のログ・レコード) を書き込むことができるログ (VSAM 入力順データ・セット) を保守します。このログは、RODM が RODM のエラー条件に関するエラー・レコードを書き込むログと同じものです。

RODM は、RODM ログのレコードの先頭にメソッド名、タイム・スタンプ、固有トランザクション ID、およびログ・レコード・タイプを書き込みます。

EKG_QueryEntityStructure − エンティティーの構造を照会する

目的

この機能は、オブジェクトまたはクラスの構造体を照会し、そのフィールドのリストを戻します。このフィールド・リストには、フィールド名、フィールド ID、データ・タイプ、および継承状況が含まれます。

機能ブロックの形式

表 117. EKG_QueryEntityStructure 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Pointer	入力	Entity_access_info_ptr
008	4	—	—	使用されません
012	2	—	—	使用されません
014	2	Anonymous(2)	—	予約済み
016	4	—	—	使用されません

表 118. EKG_QueryEntityStructure 機能用の応答ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Response_block_length
004	4	Integer	出力	Response_block_used
008	2	Smallint	出力	Field_info_element_size
010	2	Smallint	出力	Field_info_count
注: 構造体配列の第 1 エレメント				
012	0	Structure	—	Field_info_array
012	4	FieldID	出力	Field_ID
016	2	Bit(16) ビット 0 ビット 1 ビット 2	− 出力 出力 出力	Bit_map • Private_public_flag • Local_inherited_flag • Indexed_flag
018	2	Smallint	出力	Data_type

表 118. EKG_QueryEntityStructure 機能用の応答ブロック (続き)

オフセット	長さ	タイプ	用途	パラメーター名
020	67	ShortName	出力	Field_name
087	1	—	—	予約済み
注: 構造体配列の第 2 エレメント (使用する場合)				
088	0	Structure	—	Field_info_array
088	4	FieldID	出力	Field_ID
092	2	Bit(16)	－	Bit_map
		ビット 0	出力	• Private_public_flag
		ビット 1	出力	• Local_inherited_flag
		ビット 2	出力	• Indexed_flag
094	2	Smallint	出力	Data_type
096	67	ShortName	出力	Field_name
161	1	—	—	予約済み

注: 機能ブロックには、Field_info_count 配列エレメントが入っています。

上記のパラメーターの詳細については、507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、257 ページの『要約データ・タイプ参照』を参照してください。

例

表 119. EKG_QueryEntityStructure 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11503
PL/I 応答ブロック	EKG21503
PL/I 使用法コーディング	EKG51503
C 機能ブロック	EKG31503
C 応答ブロック	EKG41503
C 使用法コーディング	EKG61503

要約

表 120. EKG_QueryEntityStructure 機能の要約

機能 ID	1503
タイプ	Query
ユーザー API	可
オブジェクト特有メソッド	可
オブジェクト独立メソッド	可
初期化メソッド	可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	2

使用法

応答データには、オブジェクトまたはクラスの各フィールドについて 1 つずつの配列エレメントから構成される配列が入っています。応答ブロックには Field_info_count エレメントが入っています。各エレメントのサイズは Field_info_element_size になっています。

EKG_QueryField － フィールドを照会する

目的

この機能は、オブジェクトまたはクラスのフィールドの値を照会します。

機能ブロックの形式

表 121. EKG_QueryField 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Pointer	入力	Entity_access_info_ptr
008	4	Pointer	入力	Field_access_info_ptr
012	2	—	—	使用されません
014	2	Anonymous(2)	—	予約済み
016	4	SelfDefiningDataPtr	入力	Method_parms

表 122. EKG_QueryField 機能用の応答ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Response_block_length
004	4	Integer	出力	Response_block_used
008	2	Smallint	出力	Data_type
010	—	Anonymous	出力	Data

上記のパラメーターの詳細については、507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、257 ページの『要約データ・タイプ参照』を参照してください。

例

表 123. EKG_QueryField 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11501
PL/I 応答ブロック	EKG21501
PL/I 使用法コーディング	EKG51501
C 機能ブロック	EKG31501
C 応答ブロック	EKG41501
C 使用法コーディング	EKG61501

要約

表 124. EKG_QueryField 機能の要約

機能 ID	1501
タイプ	Query
ユーザー API	可
オブジェクト特有メソッド	可
オブジェクト独立メソッド	可
初期化メソッド	可
起動されるメソッド	ターゲット・フィールドに関する照会メソッドが起動されます。
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	2

使用法

このフィールドに照会メソッドがある場合には、そのメソッドが実行されたときに Method_parm フィールドが照会メソッドに渡されます。フィールドに照会メソッドがない場合には、Method_parm フィールドは無視されます。

value サブフィールドを照会していて、戻されたデータ・タイプが CharVar である場合、データ・ストリングの直後にヌルの終了バイト X'00' が続きます。 value サブフィールドを照会していて、戻されたデータ・タイプが GraphicVar である場合、データ・ストリングの直後にヌルの終了ダブルバイト X'0000' が続きます。

照会が正常に行われた場合には、RODM は、戻された値がローカル値であるのか、継承値であるのかを示す理由コードを戻します。

複数のフィールド値を照会したい場合には、EKG_QueryMultipleSubfields 機能を使用してください。

EKG_QueryFieldID − フィールド ID を照会する

目的

この機能は、指定されたフィールド名からフィールド ID を戻します。

機能ブロックの形式

表 125. EKG_QueryFieldID 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	—	—	使用されません
008	4	Pointer	入力	Field_access_info_ptr
012	2	—	—	使用されません
014	2	Anonymous(2)	—	予約済み
016	4	—	—	使用されません

表 126. EKG_QueryFieldID 機能用の応答ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Response_block_length
004	4	Integer	出力	Response_block_used
008	4	FieldID	出力	Field_ID

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 127. EKG_QueryFieldID 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11505
PL/I 応答ブロック	EKG21505
PL/I 使用法コーディング	EKG51505
C 機能ブロック	EKG31505
C 応答ブロック	EKG41505
C 使用法コーディング	EKG61505

要約

表 128. EKG_QueryFieldID 機能の要約

機能 ID	1505
タイプ	Query
ユーザー API	可
オブジェクト特有メソッド	可
オブジェクト独立メソッド	可
初期化メソッド	可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	2

使用法

Field_access_info_ptr の Field_ID は、この機能では無視されます。

この機能は、指定されたフィールド名からフィールド ID を入手します。このフィールド名がどのクラスにも定義されていない場合には、 RODM は戻りコード 4 と理由コード 56 を戻します。

RODM データ・キャッシュ内にあるすべてのクラスに定義された同一のフィールド名は、すべて同じフィールド ID を共用するので、この機能では、異なるクラスにある同一のフィールド名を区別するためのクラス情報が必要になることはありません。

注: オブジェクト名と関連付けられたオブジェクト ID を入手するには、指定されたクラスにあるそのオブジェクトの MyID フィールドを照会してください。クラス名と関連付けられたクラス ID を入手するには、そのクラスの MyID フィールドを照会してください。

EKG_QueryFieldName − フィールド名を照会する

目的

この機能は、指定されたフィールド ID からフィールド名を戻します。

機能ブロックの形式

表 129. EKG_QueryFieldName 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Pointer	入力	Entity_access_info_ptr
008	4	Pointer	入力	Field_access_info_ptr
012	2	—	—	使用されません
014	2	Anonymous(2)	—	予約済み
016	4	—	—	使用されません

表 130. EKG_QueryFieldName 機能用の応答ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Response_block_length
004	4	Integer	出力	Response_block_used
008	67	ShortName	出力	Field_name

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 131. EKG_QueryFieldName 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11506
PL/I 応答ブロック	EKG21506
PL/I 使用法コーディング	EKG51506
C 機能ブロック	EKG31506
C 応答ブロック	EKG41506
C 使用法コーディング	EKG61506

要約

表 132. EKG_QueryFieldName 機能の要約

機能 ID	1506
タイプ	Query
ユーザー API	可
オブジェクト特有メソッド	可
オブジェクト独立メソッド	可
初期化メソッド	可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	2

使用法

この機能は、オブジェクトまたはクラス内の指定されたフィールド ID からフィールド名を入手します。そのオブジェクトまたはクラスについて該当のフィールド ID が定義されていない場合、理由コードとともに警告メッセージが戻されます。

RODM データ・キャッシュ内にあるすべてのクラスに定義された同一のフィールド名は、すべて同じフィールド ID を共用しますが、必ずしもすべての同一のフィールド ID が同じフィールド名を共用するわけではありません。しかし、特定のオブジェクトまたはクラス内のすべてのフィールド ID は、そのオブジェクトまたはクラス内では固有の ID です。したがって、指定されたフィールド ID からフィールド名を固有に識別するためには、オブジェクトまたはクラス情報が必要です。

オブジェクト ID と関連付けられたオブジェクト名を入手するには、そのオブジェクトの MyName フィールドを照会してください。クラス ID と関連付けられたクラス名を入手するには、そのクラスの MyName フィールドを照会してください。

この機能では、フィールド・アクセス情報ブロック内の Field_ID パラメーターを設定しなければなりません。フィールド・アクセス情報ブロックの Field_name パラメーターは、この機能の場合には無視されます。

EKG_QueryFieldStructure — フィールドの構造を照会する

目的

この機能は指定されたフィールドの定義を照会し、そのフィールドのデータ・タイプ、継承状態、およびサブフィールド・マップを戻します。

機能ブロックの形式

表 133. EKG_QueryFieldStructure 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Pointer	入力	Entity_access_info_ptr
008	4	Pointer	入力	Field_access_info_ptr
012	2	—	—	使用されません

表 133. EKG_QueryFieldStructure 機能用の機能ブロック (続き)

オフセット	長さ	タイプ	用途	パラメーター名
014	2	Anonymous(2)	—	予約済み
016	4	—	—	使用されません

表 134. EKG_QueryFieldStructure 機能用の応答ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Response_block_length
004	4	Integer	出力	Response_block_used
008	2	Smallint	出力	Data_type
010	2	Smallint	出力	Inheritance_state
012	4	Bit(32)	出力	Subfield_map
016	4	Bit(32)	出力	Local_copy_map

上記のパラメーターの詳細については、507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、257 ページの『要約データ・タイプ参照』を参照してください。

例

表 135. EKG_QueryFieldStructure 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11504
PL/I 応答ブロック	EKG21504
PL/I 使用法コーディング	EKG51504
C 機能ブロック	EKG31504
C 応答ブロック	EKG41504
C 使用法コーディング	EKG61504

要約

表 136. EKG_QueryFieldStructure 機能の要約

機能 ID	1504
タイプ	Query
ユーザー API	可
オブジェクト特有メソッド	可
オブジェクト独立メソッド	可
初期化メソッド	可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	2

使用法

notify サブフィールドの値が継承されることはありません。 notify サブフィールドがある場合、このサブフィールドには常にローカルで定義された値が入っています。この値は、最初はヌルになっています。

データ・タイプが ClassLinkList、ObjectLink、または ObjectLinkList になっているサブフィールドの値は、継承されることはありません。これらのサブフィールドがある場合、その値は常にローカルで定義された値になっています。これらの値は、最初はヌルになっています。

value サブフィールドは、常にローカルで作成されます。この値は、継承されることも、ローカルで定義されることもあります。この値は、最初は継承されます。

prev_val サブフィールドと timestamp サブフィールドの値は、継承されることはありません。これらのサブフィールドがある場合、その値は常にローカルで定義された値になっています。これらの値は、最初はヌルになっています。

EKG_QueryFunctionBlockContents － 機能ブロックの内容を照会する

目的

このメソッド API 機能は、この機能を起動したユーザー API またはメソッド API 機能要求の機能ブロックのコピーを入手します。この機能を使用すると、起動されたメソッドが、起動側の機能に関する情報を入手できます。

機能ブロックの形式

表 137. EKG_QueryFunctionBlockContents 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID

表 138. EKG_QueryFunctionBlockContents 機能用の応答ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Response_block_length
004	4	Integer	出力	Response_block_used
008	4	Integer	出力	Function_block_origin
012	—	Anonymous	出力	Function_block_copy

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 139. EKG_QueryFunctionBlockContents 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG12001

表 139. EKG_QueryFunctionBlockContents 機能用の例の名前 (続き)

例	名前
PL/I 応答ブロック	EKG22001
PL/I 使用法コーディング	EKG52001
C 機能ブロック	EKG32001
C 応答ブロック	EKG42001
C 使用法コーディング	EKG62001

要約

表 140. EKG_QueryFunctionBlockContents 機能の要約

機能 ID	2001
タイプ	メソッド API サービス
ユーザー API	不可
オブジェクト特有メソッド	可
オブジェクト独立メソッド	可
初期化メソッド	不可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	不要

使用法

この機能が変更、照会、または通知メソッドによって呼び出されると、この機能は、そのメソッドを起動した機能の機能ブロックの内容を戻します。例えば、EKG_ChangeField 機能が通知メソッドを起動した場合、その通知メソッドによって呼び出された EKG_QueryFunctionBlockContents 機能は、 EKG_ChangeField 機能の機能ブロックを戻します。

この機能が、オブジェクト独立メソッドによって出されると、この機能は EKG_TriggerOIMethod 機能の機能ブロックの内容を戻します。

この機能が名前付きメソッドによって呼び出されると、この機能は EKG_TriggerNamedMethod 機能の機能ブロックの内容を戻します。

この機能によって戻される機能ブロック・データは、Function_block_copy に入ります。この機能ブロック内のポインターは、同じ Function_block_copy 内の対応する情報ブロックを指します。 EKG_QueryFunctionBlockContents 機能を使用するメソッドは、それらのポインターを使用して、 Function_block_copy 内のすべての情報を得ることができます。

戻された機能ブロック内にあるすべてのポインターは、応答ブロック内のデータを指すように調整されているため、メソッドは、これらのポインターを使用して RODM データまたは元の機能ブロックを変更することはできません。

戻された機能ブロック内のポインターによって参照されたデータは、機能ブロックのコピーのすぐ後の応答ブロック内に置かれます。

応答ブロックのサイズが不十分なために戻されたすべての機能ブロック・データを収めきれない場合、Response_block_used フィールドは実際に必要なサイズに設定され、応答ブロック内のデータは切り捨てられます。

新規のデータ値を、変更 API 機能データを入れて戻された機能ブロックの応答ブロックに入れることができない場合、他の機能ブロックのデータが提供されますが、New_data_ptr はヌルに設定されます。

新規のデータ値または旧データ値のいずれかを、スワップ API 機能データを入れて戻された機能ブロックの応答ブロックに入れることができない場合、他の機能ブロックのデータが提供され、RODM は次のことを行います。

- New_data_ptr ポインターによって指定された値を応答ブロックに入れることができない場合、 RODM は New_data_ptr および Old_data_ptr をヌルに設定します。
- それ以外の場合には、新規のデータは応答ブロックに置かれます。
  - Old_data_ptr ポインターによって指定された値を応答ブロックに入れることができない場合、 RODM は Old_data_ptr をヌルに設定します。

応答ブロック・サイズの不足は、応答ブロック・オーバーフロー条件とは見なされません。 RODM は切り詰められたデータと必要なデータ長を戻しますが、切り捨てられた部分のデータが必要な場合には、メソッドは、より大きな応答ブロックを指定して要求を再開しなければなりません。

EKG_QueryMultipleSubfields － 複数の Value サブフィールドを照会する

目的

この機能は、ユーザー API またはメソッド API に対する単一の呼び出しによって、 1 つのオブジェクトに関する複数の value サブフィールドを照会します。この機能はオブジェクト・サブフィールドを照会するものであって、クラス・サブフィールドは照会しません。この機能によって、関連する照会メソッドが起動されることはありません。

機能ブロックの形式

表 141. EKG_QueryMultipleSubfields 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Pointer	入力	Entity_access_info_ptr
008	4	Integer	入力	Number_of_subfields
注: 構造体配列の第 1 エレメント				
012	0	Structure	—	Field_info_array
012	4	Pointer	入力	Field_access_info_ptr
016	4	Anonymous(4)	—	予約済み
020	4	Pointer	出力	Response_block_reference
024	4	Integer	出力	Response_block_used
028	4	Integer	出力	Return_code

表 141. EKG_QueryMultipleSubfields 機能用の機能ブロック (続き)

オフセット	長さ	タイプ	用途	パラメーター名
032	4	Integer	出力	Reason_code
注: 構造体配列の第 2 エレメント (使用する場合)				
036	0	Structure	—	Field_info_array
036	4	Pointer	入力	Field_access_info_ptr
040	4	Anonymous(4)	—	予約済み
044	4	Pointer	出力	Response_block_reference
048	4	Integer	出力	Response_block_used
052	4	Integer	出力	Return_code
056	4	Integer	出力	Reason_code

注: 機能ブロックには、Number_of_subfields 配列のエレメントが入っています。

表 142. EKG_QueryMultipleSubfields 機能用の応答ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Response_block_length
004	4	Integer	出力	Response_block_used
008	0	Anonymous(1)	出力	Requested_info_array
注: 要求された情報配列の第 1 エレメントおよびそれ以降のエレメント				
008	2	Smallint		Data_type
010	—	Anonymous		Data_value

配列に関する注:

- すべてのサブフィールド照会が正常に行われた場合には、応答ブロックには Number_of_subfields 配列エレメントが入ります。失敗した照会は配列に含まれません。
- 機能ブロック内の Response_block_used フィールドは、応答ブロック内の対応するエレメントの長さを定義します。
- 戻りコードおよび理由コードは、照会された個々のサブフィールドに対応しています。

上記のパラメーターの詳細については、507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、257 ページの『要約データ・タイプ参照』を参照してください。

例

表 143. EKG_QueryMultipleSubfields 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11508
PL/I 応答ブロック	EKG21508
PL/I 使用法コーディング	EKG51508
C 機能ブロック	EKG31508
C 応答ブロック	EKG41508
C 使用法コーディング	EKG61508

要約

表 144. EKG_QueryMultipleSubfields 機能の要約

機能 ID	1508
タイプ	メソッド API サービス
ユーザー API	可
オブジェクト特有メソッド	可
オブジェクト独立メソッド	可
初期化メソッド	可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	2

使用法

EKG_QueryMultipleSubfields 機能は照会メソッドを起動しません。

Number_of_subfields フィールドには、100,000 を超える値を指定することはできません。

Entity_access_info_block、Response_block、照会されるフィールドの数、およびフィールド ID またはフィールド名のリスト (これらは、要求されるフィールドごとに 1 ブロックずつある Field_access_info_block で指定します) は、ユーザーが指定する必要があります。

EKG_QueryMultipleSubfields 機能用のトランザクション情報ブロックに戻された戻りコードおよび理由コードは、すべての個々の照会用の最高の戻りコードと、それに対応する最初の理由コードです。

応答ブロックのオーバーフロー状態が生じると、出力長の値 (response_block_used パラメーター) はすべて RODM によって設定されますが、オーバーフロー・バッファーに完全に収まったトランザクション結果のポインター値 (例えば、response_block_reference パラメーター) はヌル値に設定されます。EKG_QueryResponseBlockOverflow を使用してオーバーフロー・ブロックを検索する際には、元の呼び出しで戻された長さ情報を使用して、ユーザーがそのデータを解析してください。オーバーフロー処理はユーザー API だけで利用することができます。この機能に関するメソッド API は、オーバーフロー・データを廃棄します。

照会されたサブフィールドが CharVar タイプのデータを戻した場合、データ・ストリングの直後にヌルの終了バイト X'00' が続きます。照会されたサブフィールドが GraphicVar タイプのデータを戻した場合、データ・ストリングの直後にヌルの終了ダブルバイト X'0000' が続きます。

照会が正常に行われた場合には、RODM は、戻された値がローカル値であるのか、継承値であるのかを示す理由コードを戻します。

EKG_QueryNotifyQueue － 通知キューを照会する

目的

この機能は、指定された通知キューから次の通知ブロックを戻します。

機能ブロックの形式

表 145. EKG_QueryNotifyQueue 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	8	SubscribeID	入力	Notification_queue

表 146. EKG_QueryNotifyQueue 機能用の応答ブロック (通知ブロック)

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Response_block_length
004	4	Integer	出力	Response_block_used
008	2	Smallint	出力	Notification_queue_count
010	2	Smallint	出力	Response_block_type
012	4	ClassID	出力	Class_ID
016	8	ObjectID	出力	Object_ID
024	4	FieldID	出力	Field_ID
028	2	Smallint	出力	Subfield
030	8	ApplicationID	出力	User_appl_ID
038	8	SubscribeID	出力	Notification_queue
046	8	MethodName	出力	Method_name
054	8	Anonymous(8)	出力	User_word
062	—	SelfDefining	出力	User_area

上記のパラメーターの詳細については、507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、257 ページの『要約データ・タイプ参照』を参照してください。

例

表 147. EKG_QueryNotifyQueue 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11507
PL/I 応答ブロック	EKG21507
PL/I 使用法コーディング	EKG51507
C 機能ブロック	EKG31507
C 応答ブロック	EKG41507
C 使用法コーディング	EKG61507

要約

表 148. EKG_QueryNotifyQueue 機能の要約

機能 ID	1507
タイプ	Query
ユーザー API	可
オブジェクト特有メソッド	不可
オブジェクト独立メソッド	不可
初期化メソッド	不可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	2

使用法

照会された通知キューが空でない場合には、通知キュー内の最初の (最も古い) 通知ブロックが応答ブロックに戻され、その通知ブロックは通知キューから削除されます。応答ブロック内の Notification_queue_count フィールドは、この機能が呼び出される前に通知キューに入っていた通知ブロックの数を示しています。Notification_queue_count の値がゼロよりも大きい場合、この値は、通知ブロックが応答ブロックに入れられたことを示しています。

応答ブロックの Class_ID、Object_ID、Field_ID、および Subfield の各フィールドは、通知を生成したメソッドが置かれているクラスまたはオブジェクト、フィールド、およびサブフィールドを示しています。

- Class_ID と Object_ID がともにヌルになっている場合、その通知はオブジェクト独立メソッドによって起動されています。その場合、Field_ID および Subfield はゼロに設定されます。
- Object_ID がヌルで、Class_ID はヌルでない場合、そのフィールドはクラス内にあります。
- Object_ID フィールドがヌルでない場合、Class_ID フィールドはオブジェクト・クラスを示し、そのフィールドはオブジェクト内にあります。
- 通知機能を呼び出した実行中のメソッドが照会、変更、または通知メソッドである場合、 Subfield フィールドは、そのタイプのサブフィールドの ID を表します。この場合、Field_ID フィールドで示されているフィールドが変更されたために、この通知が生成された可能性があります。
- Subfield フィールドが notify サブフィールドを示している場合、そのフィールドが変更されたために通知メソッドが起動されています。
- 実行中のメソッドが名前付きメソッドである場合、Subfield フィールドは value サブフィールドを表す 1 に設定されます。
- 実行中のメソッドがオブジェクト独立メソッドである場合、Subfield フィールドはゼロに設定されます。

戻された User_appl_ID は、通知を起動したユーザーを示しています。

通知キュー・フィールドには、元の申請で指定されたものと同じ通知キュー名が入っています。

User_word フィールドには元の申請で指定されたものと同じユーザー情報が入っている場合がありますが、実際には、このフィールドに戻される値は通知メソッドによって決められます。

Method_name フィールドは、通知元のメソッドの名前を示しています。

User_area ストリングには、通知元のメソッドによって提供された、最長 32767 バイトまでのデータが入っています。

EKG_QueryObjectName − オブジェクト名を照会する

目的

この機能は、オブジェクト ID を指定した場合には、オブジェクトのオブジェクト名を戻します。この機能はオブジェクト特有メソッドでのみ使用できます。オブジェクト特有メソッドは、この機能を使用することにより、そのメソッドと関連付けられているオブジェクトだけでなく任意のオブジェクトのオブジェクト名も入手できます。

機能ブロックの形式

表 149. EKG_QueryObjectName 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	8	ObjectID	入力	Object_ID

表 150. EKG_QueryObjectName 機能用の応答ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Response_block_length
004	4	Integer	出力	Response_block_used
008	67	ShortName	出力	Class_name
075	1	—	—	予約済み
076	—	ObjectName	出力	Object_name

上記のパラメーターの詳細については、507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、257 ページの『要約データ・タイプ参照』を参照してください。

例

表 151. EKG_QueryObjectName 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG12011
PL/I 応答ブロック	EKG22011
PL/I 使用法コーディング	EKG52011
C 機能ブロック	EKG32011
C 応答ブロック	EKG42011

表 151. EKG_QueryObjectName 機能用の例の名前 (続き)

例	名前
C 使用法コーディング	EKG62011

要約

表 152. EKG_QueryObjectName 機能の要約

機能 ID	2011
タイプ	メソッド API サービス
ユーザー API	不可
オブジェクト特有メソッド	可
オブジェクト独立メソッド	不可
初期化メソッド	不可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	不要

使用法

オブジェクト特有メソッドは、リンク・フィールドを介して他のオブジェクトのオブジェクト ID にアクセスできます。この機能を使用すると、オブジェクト特有メソッドはオブジェクト名とオブジェクト ID を関連付けることができます。そして、その後で EKG_MessageTriggeredAction 機能を使用して、他のオブジェクトに対してなんらかのアクションを実行することができます。

MyName フィールド上に照会メソッドがある場合、この機能はその照会メソッドを起動しません。

EKG_QueryResponseBlockOverflow — 応答ブロック・オーバーフローを照会する

目的

この機能は、応答ブロック・オーバーフロー・バッファーを照会します。このオーバーフロー・バッファーには、すでに生じている応答ブロック・オーバーフローの原因となったユーザー・アプリケーションからの出力の超過部分が含まれています。

機能ブロックの形式

表 153. EKG_QueryResponseBlockOverflow 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Anonymous(4)	—	予約済み
008	8	TransID	入力	Correlation_ID

表 154. EKG_QueryResponseBlockOverflow 機能用の応答ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Response_block_length
004	4	Integer	出力	Response_block_used
008	—	Anonymous	出力	Data

上記のパラメーターの詳細については、507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、257 ページの『要約データ・タイプ参照』を参照してください。

例

表 155. EKG_QueryResponseBlockOverflow 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11510
PL/I 応答ブロック	EKG21510
PL/I 使用法コーディング	EKG51510
C 機能ブロック	EKG31510
C 応答ブロック	EKG41510
C 使用法コーディング	EKG61510

要約

表 156. EKG_QueryResponseBlockOverflow 機能の要約

機能 ID	1510
タイプ	Query
ユーザー API	可
オブジェクト特有メソッド	不可
オブジェクト独立メソッド	不可
初期化メソッド	不可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	2

使用法

応答ブロック内の Data フィールドには、元の機能によって戻された応答ブロック内のデータの続きが含まれています。長さのフィールドまたはヘッダーのあるデータ・タイプの場合、長さのフィールドまたはヘッダーは通常、元の応答ブロックに保管されます。

RODM は、ユーザー・アプリケーション・プログラムから呼び出された機能についてだけ、オーバーフロー・バッファーを提供します。ユーザー API 照会要求に対して値を戻す照会メソッドの場合、そのメソッドによって応答ブロックに出力された

データは、すべて呼び出し元に戻されます。データの量がユーザー提供の応答ブロックのサイズを超えている場合、 RODM はデータの超過部分を応答ブロック・オーバーフロー・バッファーに入れます。

その他のすべてのメソッド、およびメソッド API 照会要求によって起動された照会メソッドの場合、そのメソッドによって応答ブロックに出力されたデータは、必ずしもすべて呼び出し元に戻されないことがあります。データの量がメソッド提供の応答ブロックのサイズを超えている場合、 RODM はデータを応答ブロックのサイズに合わせて切り詰め、超過部分を廃棄します。

RODM がデータをオーバーフロー・バッファー内に置いた場合、RODM が指定されたアクセス・ブロックを使用して他の機能要求を受け入れる前に、ユーザーは EKG_QueryResponseBlockOverflow 機能を使用してバッファーの内容を検索しなければなりません。

オーバーフロー・データを検索するためのオーバーフロー・バッファーに対する呼び出しは 1 回だけしか行えません。指定された Response_block_length がバッファー内のデータの量よりも小さい場合、 RODM は、指定されたサイズに基づいて応答ブロックにデータを入れ、残りのデータを廃棄します。

RODM によって維持される Response block overflow バッファーは、Transaction_ID によって識別されます。オーバーフローの原因となった機能のトランザクション情報ブロックに戻された Transaction_ID 値を、この機能要求の Correlation_ID パラメーターとして指定してください。

オーバーフロー・バッファー内のデータを使用せずに廃棄する場合は、EKG_QueryResponseBlockOverflow 機能を呼び出すときに Response_block_length を 0 に設定してください。

応答ブロックのオーバーフローに関する追加情報については、 359 ページの『応答ブロック』を参照してください。

EKG_QuerySubfield − サブフィールドを照会する

目的

この機能は、オブジェクトまたはクラスのフィールドのサブフィールドの値を照会します。

機能ブロックの形式

表 157. EKG_QuerySubfield 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Pointer	入力	Entity_access_info_ptr
008	4	Pointer	入力	Field_access_info_ptr
012	2	Smallint	入力	Subfield
014	2	Anonymous(2)	—	予約済み
016	4	—	—	使用されません

表 158. EKG_QuerySubfield 機能用の応答ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Response_block_length
004	4	Integer	出力	Response_block_used
008	2	Smallint	出力	Data_type
010	—	Anonymous	出力	Data

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 159. EKG_QuerySubfield 機能用の名前の例

例	名前
PL/I 機能ブロック	EKG11502
PL/I 応答ブロック	EKG21502
PL/I 使用法コーディング	EKG51502
C 機能ブロック	EKG31502
C 応答ブロック	EKG41502
C 使用法コーディング	EKG61502

要約

表 160. EKG_QuerySubfield 機能の要約

機能 ID	1502
タイプ	Query
ユーザー API	可
オブジェクト特有メソッド	可
オブジェクト独立メソッド	可
初期化メソッド	可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	2

使用法

prev_val または timestamp のような、RODM で管理されるサブフィールドに対する照会は、その他のサブフィールドに対する照会とは異なります。 RODM で管理されるサブフィールドの値は、常に該当の value サブフィールドと対応しています。オブジェクトに value サブフィールドのローカル値があって、管理サブフィールドが存在する場合、その管理サブフィールドの値は次の 2 つのうちのいずれかになります。

- フィールド値の設定時点ですでに prev_val または timestamp が存在していた場合、 prev_val または timestamp サブフィールドには、適切な値を反映するローカル値が入ります。
- これらのサブフィールドが最後のローカル・フィールド値の設定後に作成された場合、これらのサブフィールドはヌル値になります。

RODM 管理サブフィールドが照会される際:

- フィールドにローカル値が入っていて、管理サブフィールドが存在する場合、ローカル値が戻されます。
- フィールドにローカル値が入っていない場合、管理サブフィールドの値は、継承されたフィールドをもとに判別されます。

照会されたサブフィールドが CharVar タイプのデータを戻した場合、データ・ストリングの直後にヌルの終了バイト X'00' が続きます。照会されたサブフィールドが GraphicVar タイプのデータを戻した場合、データ・ストリングの直後にヌルの終了ダブルバイト X'0000' が続きます。

Notification サブフィールド値が継承されることはありません。 notification サブフィールドに対して起動された EKG_QuerySubfield 機能は、そのサブフィールドがローカルで定義されている場合にだけ値を戻します。データ・タイプが ClassLinkList、ObjectLink、および ObjectLinkList のサブフィールドは、継承されることがありません。 value、prev_val、または timestamp サブフィールドに対して起動された EKG_QuerySubfield 機能は、それらのサブフィールドがローカルで定義されている場合にだけ値を戻します。それ以外の場合、この照会ではヌル値が戻されます。

照会が正常に行われた場合には、RODM は、戻された値がローカル値であるのか、継承値であるのかを示す理由コードを戻します。

EKG_ResponseBlock － 応答ブロックに出力する

目的

この機能は、現行の応答ブロックにデータを書き込みます。データのタイプは SelfDefining です。

機能ブロックの形式

表 161. EKG_ResponseBlock 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	SelfDefiningDataPtr	入力	Data_to_be_returned

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 162. EKG_ResponseBlock 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG12004
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG52004
C 機能ブロック	EKG32004
C 応答ブロック	不要
C 使用法コーディング	EKG62004

要約

表 163. EKG_ResponseBlock 機能の要約

機能 ID	2004
タイプ	メソッド API サービス
ユーザー API	不可
オブジェクト特有メソッド	照会および名前付きのみ
オブジェクト独立メソッド	可
初期化メソッド	不可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	不要

使用法

オブジェクト独立メソッドまたは名前付きメソッドがこの機能を実行するたびに、新しい SelfDefining データ・ストリングが現行の応答ブロックに追加されます。照会メソッドがこの機能を実行するたびに、新しい SelfDefining データ・ストリングによって現行の応答ブロックが上書きされます。

Data_to_be_returned によって指し示されたデータのサイズが現行の応答ブロックのサイズより大きい場合、RODM はデータを応答ブロックのサイズに合わせて切り詰め、警告戻りコードを出します。この機能は、応答ブロック・オーバーフロー・バッファーには書き込みを行いません。

EKG_ResponseBlock 機能は現行の応答ブロックにデータを書き込みます。この機能の場合の現行応答ブロックは、この機能を出したメソッドの応答ブロックです。メソッドが別のメソッドを呼び出すことがあるため、このブロックは、最初に実行されたメソッドの機能ブロックとは異なる可能性があります。

この機能が照会メソッドによって使用される場合、RODM は次のアクションを行います。

- RODM は、自己定義ストリングから得られた長さフィールドを使用して応答ブロックのストレージ要件を判別し、そのフィールドをデータから除去します。つまり、応答ブロック内のデータが RODM によって直接提供されたのか、あるいは

この機能を使用してメソッドによって提供されたのかに関係なく、アプリケーションは、応答ブロック内のデータとまったく同じ形式でデータを入手します。

- この自己定義ストリングを通してユーザーに戻された値はヌル・ストリングであってはなりません (つまり、自己定義ストリングの長さは 2 よりも長くなければなりません)。自己定義ストリングが正しく形式化されていない場合、RODM は応答ブロックを修正しません。

EKG_RevertToInherited － 継承値を復活させる

目的

この機能は、オブジェクトまたはクラスのフィールドまたはサブフィールドのローカル定義された値を削除します。これにより、そのフィールドまたはサブフィールドは、親クラスで定義された値を継承するようになります。

機能ブロックの形式

表 164. EKG_RevertToInherited 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Pointer	入力	Entity_access_info_ptr
008	4	Pointer	入力	Field_access_info_ptr
012	2	Smallint	入力	Subfield

上記のパラメーターの詳細については、507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、257 ページの『要約データ・タイプ参照』を参照してください。

例

表 165. EKG_RevertToInherited 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11411
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG51411
C 機能ブロック	EKG31411
C 応答ブロック	不要
C 使用法コーディング	EKG61411

要約

表 166. EKG_RevertToInherited 機能の要約

機能 ID	1411
タイプ	アクション
ユーザー API	可
オブジェクト特有メソッド	可
オブジェクト独立メソッド	可

表 166. EKG_RevertToInherited 機能の要約 (続き)

初期化メソッド	可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	可
許可レベル	3

使用法

クラスでローカルに作成されたフィールドおよびサブフィールドは、親クラスから継承されません。これらのフィールドおよびサブフィールドは継承されないため、これらのフィールドおよびサブフィールドには復活すべき継承値がありません。この機能のターゲットがローカルで作成されている場合、RODM は警告戻りコードを戻します。

次のいずれのフィールドあるいはサブフィールドに対しても、EKG_RevertToInherited 機能を使用することはできません。

- システム定義フィールド
- データ・タイプが ObjectLink または ObjectLinkList のフィールド
- notify サブフィールド
- prev_val サブフィールド
- timestamp サブフィールド
- 次のシステム・クラスのもとで読み取り専用として定義されたシステム・フィールド
  - EKG_System クラス
  - EKG_User クラス
  - EKG_Method クラス
  - EKG_NotificationQueue クラス

prev_val または timestamp サブフィールドが定義されていて、 value サブフィールドが EKG_RevertToInherited 機能のターゲットになっている場合、 prev_val および timestamp サブフィールドも継承値に復活されます。 prev_val および timestamp サブフィールドの継承の詳細については、 245 ページの『RODM サブフィールド』を参照してください。

Subfield パラメーターを 0 に設定すると、notify サブフィールド以外のそのフィールドのすべてのサブフィールドで継承値が復活するようになります。 Subfield パラメーターを 4 (notify)、5 (prev_val)、または 6 (timestamp) に設定することはできません。

継承値を復活させるときには、同じフィールドのサブフィールドを異なるレベルの親クラスから継承させることができます。例えば、value サブフィールドの値を直接の親クラスから継承し、query サブフィールドの値をその親クラスの親クラスから継承することができます。

EKG_SendNotification － 通知を送信する

目的

この機能は、オブジェクトまたはクラス内のフィールドの値が変更されたときに、指定された通知キューに通知ブロックを送ります。

機能ブロックの形式

表 167. EKG_SendNotification 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	8	ApplicationID	入力	User_appl_ID
012	8	SubscribeID	入力	Notification_queue
020	8	Anonymous(8)	入力	User_word
028	4	SelfDefiningDataPtr	入力	Method_output_message

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 168. EKG_SendNotification 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG12005
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG52005
C 機能ブロック	EKG32005
C 応答ブロック	不要
C 使用法コーディング	EKG62005

要約

表 169. EKG_SendNotification 機能の要約

機能 ID	2005
タイプ	メソッド API サービス
ユーザー API	不可
オブジェクト特有メソッド	可
オブジェクト独立メソッド	可
初期化メソッド	不可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	不要

使用法

この機能は、通知ブロックを作成し、指定された User_appl_ID に関する指定された Notification_queue にその通知ブロックを入れます。指定された Notification_queue が空の場合、RODM はこのキューに関連付けられたユーザーの ECB を通知します。

通知に関する詳細については、430 ページの『EKG_AddNotifySubscription − 通知申請を追加する』、451 ページの『EKG_DeleteNotifySubscription − 通知申請を削除する』、および 481 ページの『EKG_QueryNotifyQueue − 通知キューを照会する』を参照してください。通知キューに関するユーザーの ECB の通知が失敗した場合、 RODM は EKG_User_Class オブジェクトの EKG_StopMode フィールドの値に基づいて、すべての通知キューおよび申請を除去します。 EKG_StopMode がとる可能性のある値については、231 ページの『EKG_User クラス』を参照してください。

EKG_SetReturnCode − 戻りコードと理由コードを設定する

目的

この機能は、メソッドがその呼び出し元に戻す戻りコードと理由コードを設定します。

機能ブロックの形式

表 170. EKG_SetReturnCode 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Integer	入力	Value_for_return_code
008	4	Integer	入力	Value_for_reason_code

上記のパラメーターの詳細については、507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、257 ページの『要約データ・タイプ参照』を参照してください。

例

表 171. EKG_SetReturnCode 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG12006
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG52006
C 機能ブロック	EKG32006
C 応答ブロック	不要
C 使用法コーディング	EKG62006

要約

表 172. EKG_SetReturnCode 機能の要約

機能 ID	2006
タイプ	メソッド API サービス
ユーザー API	不可
オブジェクト特有メソッド	可
オブジェクト独立メソッド	可
初期化メソッド	可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	不要

使用法

EKG_SetReturnCode 機能は、Value_for_return_code の値が呼び出し元の戻りコードのそれまでの値よりも大きい場合には、その戻りコードを Value_for_return_code パラメーターの値に合わせて変更します。この機能は、戻りコードが変更された場合には、呼び出し元の理由コードの値を Value_for_reason_code パラメーターの値に合わせて変更します。

Value_for_return_code の値は、0、4、8、または 12 のいずれかです。
Value_for_reason_code の値は、0 から 65535 までのいずれかです。理由コードを出すメソッドを作成する場合には、49152 から 65535 までの範囲の理由コードを使用してください。

ユーザー作成のメソッドにより出される戻りコードについては、次の指針に従ってください。

戻りコード
意味

0 理由コードも 0 である場合には、操作が正常に行われていて、問題はありません。 理由コードが 0 以外の場合には、操作は正常に行われていますが、メッセージがログに記録される場合があります。

4 問題が発生しています。後でその要求または機能をやり直してください。理由コードで詳しい情報が提供されている可能性があります。

8 論理エラーのために要求または機能が失敗しました。要求または機能をやり直さないでください。理由コードで詳しい情報が提供されている可能性があります。

12 RODM が利用できないために要求または機能が失敗しました。要求または機能をやり直さないでください。理由コードで詳しい情報が提供されている可能性があります。

EKG_SetReturnCode を呼び出すメソッドが、EKG_ExecuteFunctionList ユーザー API 呼び出しのリストに含まれる機能によって開始されたトランザクションから起動された場合、戻りコードと理由コードは、リスト内のその機能に関する個々の戻りコードおよび理由コードに伝搬されます。さらに、この戻りコードがリスト内のすべ

ての機能の戻りコードのうちで最高の戻りコードである場合には、この戻りコードと理由コードは、トランザクション情報ブロックで設定される EKG_ExecuteFunctionList ユーザー API トランザクションの戻りコードおよび理由コードになります。

EKG_SetReturnCode 機能が呼び出され、指定された戻りコードが EKG_User クラス・オブジェクトの EKG_MLogLevel 以上である場合、RODM は、オブジェクト特有メソッドについてはタイプ 3 のログ・レコードを書き込み、オブジェクト独立メソッドについてはタイプ 4 のログ・レコードを書き込みます。非同期的に実行されているメソッドによってこの機能が要求された場合には、RODM は戻りコードと MLOG_LEVEL カスタマイズ・パラメーターを比較してから、上記のようにログ・レコードを書き込みます。非同期的に実行されているメソッドからログ・レコードが書き込まれた場合には、 RODM は EKG_LastAsyncError フィールドをその戻りコードに設定し、このフィールドに申請されているすべてのアプリケーションに関する通知メソッドを起動します。

RODM が戻りコードおよび理由コードを判別する方法の詳細については、 362 ページの『トランザクションのエラー条件』を参照してください。

メソッドを作成する際には、メソッドから戻りコードおよび理由コードを出すことの意味に注意する必要があります。メソッドによって戻された理由コードと戻りコードをアプリケーションがどのように解釈するのかについては、 362 ページの『トランザクションのエラー条件』を参照してください。

EKG_Stop － RODM を停止する

目的

この機能は、ユーザーが接続されている RODM プログラムを停止します。オプションで、RODM が停止前にチェックポイント操作を実行するように指定できます。

機能ブロックの形式

表 173. EKG_Stop 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	2	Smallint	入力	Stop_type

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 174. EKG_Stop 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11202
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG51202

表 174. EKG_Stop 機能用の例の名前 (続き)

例	名前
C 機能ブロック	EKG31202
C 応答ブロック	不要
C 使用法コーディング	EKG61202

要約

表 175. EKG_Stop 機能の要約

機能 ID	1202
タイプ	制御
ユーザー API	可
オブジェクト特有メソッド	不可
オブジェクト独立メソッド	不可
初期化メソッド	不可
起動されるメソッド	EKG_LastCheckpointID フィールドにインストールされた通知メソッドは、チェックポイント操作が正常に行われた場合にのみ起動されます。 EKG_LastCheckpointResult フィールドにインストールされた通知メソッドは、チェックポイント操作が要求されたときに起動されます。通知メソッドは、これら以外のフィールドにインストールすることはできません。
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	6

使用法

この機能によって停止された RODM は、オペレーター・コマンドでしか再始動できません。

EKG_SwapField − フィールドをスワップする

目的

この機能は、ターゲット・フィールドの値と指定されたテスト値とを比較します。両方の値が同じである場合、この機能はターゲット・フィールドの値を、指定された新しい値に変更します。

機能ブロックの形式

表 176. EKG_SwapField 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Pointer	入力	Entity_access_info_ptr
008	4	Pointer	入力	Field_access_info_ptr

表 176. EKG_SwapField 機能用の機能ブロック (続き)

オフセット	長さ	タイプ	用途	パラメーター名
012	2	—	—	使用されません
014	2	Smallint	入力	Data_type
016	4	Integer	入力	New_char_data_length
020	4	Pointer	入力	New_data_ptr
024	4	Integer	入力	Old_char_data_length
028	4	Pointer	入力	Old_data_ptr
032	4	SelfDefiningDataPtr	入力	Method_parms

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 177. EKG_SwapField 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11402
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG51402
C 機能ブロック	EKG31402
C 応答ブロック	不要
C 使用法コーディング	EKG61402

要約

表 178. EKG_SwapField 機能の要約

機能 ID	1402
タイプ	アクション
ユーザー API	可
オブジェクト特有メソッド	不可
オブジェクト独立メソッド	不可
初期化メソッド	不可
起動されるメソッド	通知メソッドと変更メソッドが起動されます。
EKG_MessageTriggeredAction 機能による起動	可
許可レベル	3

使用法

RODM は、この機能のターゲットとなっているフィールドの値と、Old_data_ptr によって指し示されたテスト値を比較します。両方の値が同じである場合、RODM はターゲット・フィールドの値を、New_data_ptr によって指し示された新しい値に変

更します。両方の値が異なっている場合には、RODM はフィールドの値を変更せず、戻りコード 8 と理由コード 39 を戻します。

新しいデータのデータ・タイプは、ターゲット・フィールドのデータ・タイプと同じでなければなりません。 EKG_SwapField 機能は、データ・タイプが ObjectID、ObjectIDList、ObjectLink、 ObjectLinkList、ClassID、ClassIDList、または ClassLinkList になっているフィールドに使用することはできません。

New_data_ptr がヌルの場合、RODM はフィールドをそのデータ・タイプのヌル値に設定します。

ターゲット・フィールドに関して変更メソッドが定義されている場合、 Old_data_ptr によって指し示された値がターゲット・フィールドの値と等しいときには、 RODM は変更メソッドを起動します。 RODM は、変更メソッドを起動する際に、ターゲット・フィールドの値を変更する代わりに New_data_ptr の値を変更メソッドに渡します。

ターゲット・フィールドに関して通知メソッドが定義されている場合、この機能またはターゲット・フィールドに関する変更メソッドによってターゲット・フィールドが正常に変更されると、 RODM は通知メソッドを起動します。ターゲット・フィールドがオブジェクト上にある場合、RODM は、そのオブジェクトの親クラス内の同じフィールドについて定義された通知メソッドも起動します。

EKG_SwapField 機能は、ターゲット・フィールドの値が正常に更新されると、戻りコード 0 を戻します。変更の詳細については、理由コードで示されます。

理由コード
説明

0 ローカル値が存在していて、その値が変更されました。

26 既存の値が新規の値と同じです。

142 継承値が存在していて、その値がローカル値によって置き換えられました。

0 (ゼロ) と 26 の両方、または 26 と 142 の両方に該当する状況では、RODM は常に 26 を戻します。

EKG_SwapSubfield − サブフィールドをスワップする

目的

この機能は、ターゲット・サブフィールドの値と指定されたテスト値とを比較します。両方の値が同じである場合、この機能はターゲット・サブフィールドの値を、指定された新しい値に変更します。

機能ブロックの形式

表 179. EKG_SwapSubfield 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Pointer	入力	Entity_access_info_ptr
008	4	Pointer	入力	Field_access_info_ptr

表 179. EKG_SwapSubfield 機能用の機能ブロック (続き)

オフセット	長さ	タイプ	用途	パラメーター名
012	2	Smallint	入力	Subfield
014	2	Smallint	入力	Data_type
016	4	Integer	入力	New_char_data_length
020	4	Pointer	入力	New_data_ptr
024	4	Integer	入力	Old_char_data_length
028	4	Pointer	入力	Old_data_ptr
032	4	—	—	使用されません

上記のパラメーターの詳細については、507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、257 ページの『要約データ・タイプ参照』を参照してください。

例

表 180. EKG_SwapSubfield 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11404
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG51404
C 機能ブロック	EKG31404
C 応答ブロック	不要
C 使用法コーディング	EKG61404

要約

表 181. EKG_SwapSubfield 機能の要約

機能 ID	1404
タイプ	アクション
ユーザー API	可
オブジェクト特有メソッド	不可
オブジェクト独立メソッド	不可
初期化メソッド	不可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	可
許可レベル	3

使用法

RODM は、この機能のターゲットとなっているサブフィールドの値と、Old_data_ptr によって指し示されたテスト値を比較します。両方の値が同じである場合、RODM はターゲット・サブフィールドの値を、New_data_ptr によって指し示された新しい値に変更します。両方の値が異なっている場合には、RODM はサブフィールドの値を変更せず、戻りコード 8 と理由コード 39 を戻します。

新しいデータのデータ・タイプは、既存のサブフィールドのデータ・タイプと同じでなければなりません。 EKG_SwapSubfield 機能は、データ・タイプが ObjectID、ObjectIDList、ObjectLink、 ObjectLinkList、ClassID、ClassIDList、または ClassLinkList になっているサブフィールドには使用できません。

New_data_ptr がヌルの場合、RODM はサブフィールドをそのデータ・タイプのヌル値に設定します。

サブフィールドの値がこの機能によって変更される際に、RODM がメソッドを起動したり、 prev_val サブフィールドと timestamp サブフィールドの更新を行うことはありません。

EKG_SwapSubfield 機能は、ターゲット・サブフィールドの値が正常に更新されると、戻りコード 0 (ゼロ) を出します。変更の詳細については、理由コードで示されます。

理由コード
説明

0 ローカル値が存在していて、その値が変更されました。

26 既存の値が新規の値と同じです。

142 継承値が存在していて、その値がローカル値によって置き換えられました。

0 (ゼロ) と 26 の両方、または 26 と 142 の両方に該当する状況では、RODM は常に 26 を戻します。

EKG_TriggerNamedMethod − 名前付きメソッドを起動する

目的

この機能は、指定されたオブジェクトまたはクラス内で名前付きメソッドを起動します。

機能ブロックの形式

表 182. EKG_TriggerNamedMethod 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Pointer	入力	Entity_access_info_ptr
008	4	Pointer	入力	Field_access_info_ptr
012	4	SelfDefiningDataPtr	入力	Method_parms

表 183. EKG_TriggerNamedMethod 機能用の応答ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Response_block_length
004	4	Integer	出力	Response_block_used
008	—	Anonymous	出力	Concat_of_strings

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 184. EKG_TriggerNamedMethod 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11415
PL/I 応答ブロック	EKG21415
PL/I 使用法コーディング	EKG51415
C 機能ブロック	EKG31415
C 応答ブロック	EKG41415
C 使用法コーディング	EKG61415

要約

表 185. EKG_TriggerNamedMethod 機能の要約

機能 ID	1415
タイプ	アクション
ユーザー API	可
オブジェクト特有メソッド	可
オブジェクト独立メソッド	可
初期化メソッド	可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	可
許可レベル	5 (EKG_Refresh 名前付きメソッドを起動) 3 (その他の名前付きメソッドを起動)

使用法

Field_access_info_ptr は、MethodSpec タイプのフィールドを指していなければなりません。この MethodSpec フィールドの method_parameter_list は、名前付きメソッドの長命パラメーターになります。 Method_Parms パラメーターが指す SelfDefining ストリングは、名前付きメソッドの短命パラメーターになります。 SelfDefining ストリングの最大長は 254 バイトです。

名前メソッドは、その名前メソッドの定義が行われているオブジェクトまたはクラス内のフィールドにのみ作動します。

名前付きメソッドによって応答ブロックのオーバーフローが発生した場合、その名前付きメソッドはオーバーフローに関する戻りコードと理由コードを受け取ります。しかし、これらのメソッドは、メソッドを起動したプログラムへはこの戻りコードと理由コードを渡さないことがあります。名前付きメソッドが起動された場合には、応答ブロックに入れて戻された Response_block_used パラメーターと

Response_block_length パラメーターを常に比較してください。 Response_block_used パラメーターの値が Response_block_length パラメーターの値よりも大きい場合には、オーバーフローが発生しています。

EKG_TriggerOIMethod － オブジェクト独立メソッドを起動する

目的

この機能はオブジェクト独立メソッドを起動します。

機能ブロックの形式

表 186. EKG_TriggerOIMethod 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	8	MethodName	入力	Method_name
012	4	SelfDefiningDataPtr	入力	Method_parms

表 187. EKG_TriggerOIMethod 機能用の応答ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Response_block_length
004	4	Integer	出力	Response_block_used
008	—	Anonymous	出力	Concat_of_strings

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 188. EKG_TriggerOIMethod 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG11416
PL/I 応答ブロック	EKG21416
PL/I 使用法コーディング	EKG51416
C 機能ブロック	EKG31416
C 応答ブロック	EKG41416
C 使用法コーディング	EKG61416

要約

表 189. EKG_TriggerOIMethod 機能の要約

機能 ID	1416
タイプ	アクション
ユーザー API	可
オブジェクト特有メソッド	不可

表 189. EKG_TriggerOIMethod 機能の要約 (続き)

オブジェクト独立メソッド	可
初期化メソッド	可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	可
許可レベル	3

使用法

Method_parms で指し示されるフィールドの最大長は 32767 バイトです。

オブジェクト独立メソッドは、EKG_Method クラスのもとでメソッド・オブジェクトを作成することによってインストールしてからでなければ、この機能で起動できません。

オブジェクト独立メソッドによって応答ブロック内でオーバーフローが発生した場合には、そのオブジェクト独立メソッドはオーバーフローに関する戻りコードと理由コードを受け取ります。しかし、これらのメソッドは、メソッドを起動したプログラムへはこの戻りコードと理由コードを渡さないことがあります。オブジェクト独立メソッドがトリガーされた場合には、応答ブロックに入れて戻された Response_block_used パラメーターと Response_block_length パラメーターを常に比較してください。 Response_block_used パラメーターの値が Response_block_length パラメーターの値よりも大きい場合には、オーバーフローが発生しています。

EKG_UnlinkNoTrigger、EKG_UnlinkTrigger － 2 つのオブジェクトをリンク解除する

目的

これらの機能は、2 つのオブジェクト間のリンクを削除します。EKG_UnlinkTrigger 機能は変更メソッドと通知メソッドを起動しますが、EKG_UnlinkNoTrigger 機能は起動しません。

機能ブロックの形式

表 190. EKG_UnlinkNoTrigger 機能および EKG_UnlinkTrigger 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID
004	4	Pointer	入力	Entity_access_info_ptr_1
008	4	Pointer	入力	Field_access_info_ptr_1
012	4	Pointer	入力	Entity_access_info_ptr_2
016	4	Pointer	入力	Field_access_info_ptr_2
000	4	Pointer	入力	Method_parms[1]

注:

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 191. EKG_UnlinkNoTrigger 機能および EKG_Unlink Trigger 機能用の例の名前

例	名前
PL/I 機能ブロック (EKG_UnlinkTrigger)	EKG11407
PL/I 機能ブロック (EKG_UnlinkNoTrigger)	EKG11408
PL/I 応答ブロック	不要
PL/I 使用法コーディング (EKG_UnlinkTrigger)	EKG51407
PL/I 使用法コーディング (EKG_UnlinkTrigger)	EKG51407
PL/I 使用法コーディング (EKG_UnlinkNoTrigger)	EKG51408
C 機能ブロック (EKG_UnlinkTrigger)	EKG31407
C 機能ブロック (EKG_UnlinkNoTrigger)	EKG31408
C 応答ブロック	不要
C 使用法コーディング (EKG_UnlinkTrigger)	EKG61407
C 使用法コーディング (EKG_UnlinkNoTrigger)	EKG61408

要約

表 192. EKG_UnlinkNoTrigger 機能と EKG_UnlinkTrigger 機能の要約

機能 ID EKG_UnlinkNoTrigger EKG_UnlinkTrigger	1408 1407
タイプ	アクション
ユーザー API	可
オブジェクト特有メソッド	不可
オブジェクト独立メソッド	可
初期化メソッド	可
起動されるメソッド EKG_UnlinkTrigger EKG_UnlinkNoTrigger	変更メソッドおよび通知メソッド 不可
EKG_MessageTriggeredAction 機能による起動	可
許可レベル	3

使用法

ObjectLinkList タイプのフィールド内のリンクの順序は予想できません。

リンク解除されるフィールドのタイプは、ObjectLink または ObjectLinkList でなければなりません。これらのフィールドは、EKG_LinkNoTrigger 機能または EKG_LinkTrigger 機能を使用してリンクされたものでなければなりません。ObjectLink フィールドには、1 つのリンクしかありません。 ObjectLinkList フィールドには、1 つのフィールドについて複数のリンクが存在することがあります。

GMFHS リソースでは EKG_UnlinkNoTrigger を使用しないでください。

EKG_UnlinkTrigger 機能が出されると、通知メソッドが起動される前にリンク解除操作が実行されます。リンク解除されるフィールドの一方または両方で変更メソッドが定義されている場合には、以下のうちの 1 つが当てはまる場合にかぎり、変更メソッドの後でリンク解除が行われます。

- 両方の変更メソッドが、EKG_SetReturnCode でゼロの戻りコードを明示的に設定している。
- どちらの変更メソッドでも戻りコードを設定していない。この場合、RODM によってゼロの戻りコードが想定され、リンク解除が続行されます。

リンク解除操作が続行されない場合、通知メソッドは起動されません。フィールドのリンク解除が正常に行われた場合、次の順序で通知メソッドが起動されます。

1. Field_access_info_ptr_1 によって指定されたフィールドに関する通知メソッド
2. Field_access_info_ptr_2 によって指定されたフィールドに関する通知メソッド
3. 第 1 フィールドの親クラスに関する通知メソッド
4. 第 2 フィールドの親クラスに関する通知メソッド

EKG_UnlockAll − 保持されたエンティティーのロックをすべて解除する

目的

この機能は、これまで、レベル 1 のオブジェクト独立メソッドによって保持されたすべてのロックを解放するために使用されていました。現在では RODM は自動的にロックを制御するようになったので、この機能は不要になりました。この機能は、既存のアプリケーションとの互換性を維持するために残されています。この機能を必要とする既存のアプリケーションを変更する必要はありません。

機能ブロックの形式

表 193. EKG_UnlockAll 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID

上記のパラメーターの詳細については、507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、257 ページの『要約データ・タイプ参照』を参照してください。

例

表 194. EKG_UnlockAll 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG12003
PL/I 応答ブロック	不要
PL/I 使用法コーディング	EKG52003
C 機能ブロック	EKG32003
C 応答ブロック	不要
C 使用法コーディング	EKG62003

要約

表 195. EKG_UnlockAll 機能の要約

機能 ID	2003
タイプ	メソッド API サービス
ユーザー API	不可
オブジェクト特有メソッド	不可
オブジェクト独立メソッド	可
初期化メソッド	可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	不要

EKG_WhereAmI ― 位置を特定する

目的

この機能は、メソッド名が割り当てられているクラス、オブジェクト、フィールド、およびサブフィールドと、そのオブジェクト特有メソッドが実行されているコンテキストを戻します。

機能ブロックの形式

表 196. EKG_WhereAmI 機能用の機能ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Function_ID

表 197. EKG_WhereAmI 機能用の応答ブロック

オフセット	長さ	タイプ	用途	パラメーター名
000	4	Integer	入力	Response_block_length
004	4	Integer	出力	Response_block_used
008	4	ClassID	出力	Class_ID
012	8	ObjectID	出力	Object_ID
020	4	FieldID	出力	Field_ID
024	2	Smallint	出力	Subfield
026	2	Anonymous(2)	—	予約済み
028	8	ObjectID	出力	Requesting_method_ID

上記のパラメーターの詳細については、 507 ページの『機能パラメーターの説明』を参照してください。上記の抽象データ・タイプの詳細については、 257 ページの『要約データ・タイプ参照』を参照してください。

例

表 198. EKG_WhereAmI 機能用の例の名前

例	名前
PL/I 機能ブロック	EKG12007
PL/I 応答ブロック	EKG22007
PL/I 使用法コーディング	EKG52007
C 機能ブロック	EKG32007
C 応答ブロック	EKG42007
C 使用法コーディング	EKG62007

要約

表 199. EKG_WhereAmI 機能の要約

機能 ID	2007
タイプ	メソッド API サービス
ユーザー API	不可
オブジェクト特有メソッド	可
オブジェクト独立メソッド	不可
初期化メソッド	不可
起動されるメソッド	不可
EKG_MessageTriggeredAction 機能による起動	不可
許可レベル	不要

使用法

Subfield パラメーターはメソッドのタイプを示しています。 Subfield パラメーターは、名前付きメソッドの場合には 1 に設定されます。

メソッドがクラス上で定義されている場合には、Object_ID パラメーターはヌルに設定されます。

機能パラメーターの説明

Bit_map
フィールドを記述するフラグのビットマップを表します。 Bit_map は、Private_public_flag と Local_inherited_flag からなります。

Change_status
Change_status パラメーターは、フィールドの値が変更されているかどうかをメソッドに通知するために使用されます。

Class_access_info_ptr
Class_access_info_ptr は、機能呼び出しによってクラス情報のみが使用されている場合の、エンティティー・アクセス情報ブロックを指すポインターです。 Naming_count 情報がゼロに設定されている場合には、このアクセス・ブロックに入るオブジェクト情報はヌル値に設定されている必要があります。

Class_ID
クラス ID を表します。

Class_name
この機能の対象となるクラスの名前を表します。

Concat_of_strings
Anonymous タイプの応答データ・ストリングです。このストリングは、0 個以上の SelfDefining データ・ストリングが連結されたものです。

Correlation_ID
RODM によって割り当てられたトランザクションの固有な ID を表します。

Data
RODM 機能によって戻されるデータを表します。このデータのタイプは Data_type です。オーバーフロー・ブロックの Data パラメーターの場合のデータ・タイプは、オーバーフローの原因となった機能の元の応答ブロックで指定されています。

Data_to_be_returned
Data_to_be_returned パラメーターは、応答ブロックのデータ域に連結する対象を指すように、呼び出し元で設定しなければなりません。

Data_type
指定されたパラメーターの RODM 抽象データ・タイプを表します。

Entity_access_info_ptr
この機能の対象となるエンティティーを指定しているエンティティー・アクセス情報ブロックを指すポインターです。

Entity_access_info_ptr_1
この機能の対象となる最初のエンティティーを指定しているエンティティー・アクセス情報ブロックを指すポインターです。

Entity_access_info_ptr_2
この機能の対象となる 2 番目のエンティティーを指定しているエンティティー・アクセス情報ブロックを指すポインターです。

Field_access_info_ptr
この機能の対象となるオブジェクトのフィールドを指定しているフィールド・アクセス情報ブロックを指すポインターです。

Field_access_info_ptr_1
この機能の対象となる最初のオブジェクトのフィールドを指定しているフィールド・アクセス情報ブロックを指すポインターです。

Field_access_info_ptr_2
この機能の対象となる 2 番目のオブジェクトのフィールドを指定しているフィールド・アクセス情報ブロックを指すポインターです。

Field_ID
フィールド ID を表します。

Field_info_array
EKG_QueryEntityStructure の場合には、オブジェクトまたはクラスを構成するフ

ィールドを記述するパラメーターの配列を表します。
EKG_QueryMultipleSubfields の場合には、value サブフィールドが照会されるフィールドの配列を表します。

Field_info_count
Field_info_array 内のフィールドの数を表します。

Field_info_element_size
Field_info_array の各エレメントのサイズを表します。

Field_name
フィールドの名前を表します。 最大長が 67 バイトの可変長フィールドです。

Field_type_flag
Field_type_flag は、新しいフィールドが共用、専用、または共用索引付きのいずれであるのかを指定します。有効な値は、以下のとおりです。

値	意味
1	共用
2	専用
3	共用索引付き

Function_block_copy
照会される機能ブロックのコピー。Function_block_copy パラメーターには、実行中のメソッドを起動した機能の機能ブロックのコピーが指定されます。

Function_block_origin
Function_block_origin パラメーターは、元の機能がユーザー・アプリケーションによって呼び出されたのか、メソッドによって呼び出されたのかを指定します。有効な値は、以下のとおりです。

値	意味
1	ユーザー・アプリケーション
2	メソッド

Function_block_ptr
実行される機能の機能ブロックを指すポインターです。機能ブロックの形式については、特定の機能の説明を参照してください。

Function_ID
RODM がこの機能を識別できるようにするための機能 ID を表します。

Function_info_array
実行される機能の配列を表します。

Indexed_data_length
RODM が探し出そうとしている索引付きデータの長さを表します。

Indexed_data_ptr
RODM が探し出そうとしている索引付きデータを指すポインターです。この索引付きデータのデータ・タイプは、CharVar または IndexList でなければなりません。 Indexed_data_ptr は、CharVar データ値または個々の IndexList データ項目の文字データの最初のバイトを指していなければなりません。文字ストリングの長さは Indexed_data_length で指定されていなければなりません。

Inheritance_state
このフィールドの値は常に 1 です。

Last_checkpoint_ID
最後のチェックポイント要求のトランザクション ID を表します。 Last_checkpoint_ID は、RODM がコールド・スタートしたときにはゼロに設定されます。

Local_copy_map
Local_copy_map は、次のように定義されているビットマップです (ビットには、左から順に 1 から 32 までの番号が付いています)。 RODM は、対応するサブフィールドにローカル定義データが入っていることを示すために、出力ブロック内の Local_copy_map ビットを 1 に設定します。

ビット	サブフィールド
1	値
2	Query
3	Change
4	Notify
5	Prev_val
6	Timestamp
7 から 32	予約済み

Local_inherited_flag
あるフィールドがローカルに定義されているのか、または親クラスから継承されているのかを指定するフラグです。有効な値は、以下のとおりです。

値	意味
0	ローカル定義
1	継承

Log_message
Log_message パラメーターは、RODM ログに書き込まれる文字ストリングを指します。これは、最大長が 32709 バイトの AnonymousVar ストリングです。

Long_lived_parm
通知メソッドに渡される長命パラメーターを指すポインターです。このポインターで識別されるパラメーターの最大長は 254 バイトです。

Message_CCSID
Message_CCSID 値は、Log_message によって指し示されるストリングに使用されるコード・ページおよび文字セット定義を識別します。この値は、RODM ログ・データ・セットを処理するアプリケーションで使用することができます。

Method_name
この機能が起動するメソッドの名前、またはこの通知ブロックを通知キューに入れるメソッドの名前です。

Method_output_message
呼び出し元のメソッドによって通知キューに入れられてユーザー・アプリケーションに渡されるデータを指し示すポインターです。 メッセージの最大長は 32767 バイトです。

Method_parms
メソッドに渡される短命パラメーターを指すポインターです。この短命パラメーターは、機能の対象となるオブジェクトに関連付けられた通知メソッドに渡され

ます。 EKG_SwapField 機能の場合には、この短命パラメーターは変更メソッドにも渡されます。 EKG_QueryField 機能の場合には、この短命パラメーターは、通知メソッドの代わりに照会メソッドに渡されます。 EKG_TriggerNamedMethod 機能および EKG_TriggerOIMethod 機能の場合には、この短命パラメーターは、起動されたメソッドに渡されます。

New_char_data_length
データ・タイプが CharVar および GraphicVar の新規データの長さです。このパラメーターは、その他のデータ・タイプの場合には無視されます。指し示されるデータは、文字データの最初のバイトでなければならず、その長さは New_Char_data_length パラメーターで指定されていなければなりません。

New_data_ptr
ターゲット・フィールドの値に置き換わって入る新しいデータを指すポインターです。

Notification_queue
この機能によって指定される Notification_queue です。 364 ページの『RODM 通知プロセス』を参照してください。

Notification_queue_count
この機能がキューに作用する前に notification_queue に入っている、通知ブロックの数を表します。

Notify_method
この通知申請と関連付けられた通知メソッドのオブジェクト ID です。

Number_of_fields
変更されるフィールドの数を表す値です。

Number_of_functions
実行される機能の数を表す値です。機能ごとに Function_information_array のエレメントを 1 つずつ指定してください。

Number_of_subfields
照会されるサブフィールドの数を表す値です。照会ごとに Field_info_array のエレメントを 1 つずつ指定してください。

Object_array
この機能の対象となるオブジェクトの配列です。

Object_ID
この機能の対象となるオブジェクトのオブジェクト ID、またはこの機能の対象となるオブジェクトの Object_array の 1 つのエレメントです。

Object_list_length
配列内のオブジェクトの数を表します。

Object_name
この機能の対象となるオブジェクトの名前です。

Old_char_data_length
古いデータのデータ・タイプが CharVar または GraphicVar である場合は、古いデータの長さを表します。このパラメーターは、その他のデータ・タイプの場合には無視されます。

Old_data_ptr
古いデータを指すポインターです。

Private_public_flag
Private_public_flag は、フィールドが (子に継承されない) 専用であるかまたは (子によって継承される) 共用であるのかを表します。有効な値は、以下のとおりです。

値	意味
0	共用
1	専用

Parent_access_info_ptr
Parent_access_info_ptr は、この機能呼び出しによってクラス情報だけが使用される場合のエンティティー・アクセス情報ブロックを指すポインターです。Naming_count 情報がゼロに設定されている場合には、このアクセス・ブロックに入るオブジェクト情報はヌル値に設定されている必要があります。

Reason_code
RODM から戻される理由コードです。

Requesting_method_ID
現行メソッド・オブジェクトのメソッド Object_ID です。

Response_block_length
この機能を使用しているメソッドまたはアプリケーションによって提供された応答ブロックの長さをバイト単位で表しています。この値には、Response_block_length パラメーターと Response_block_used パラメーターのための 8 バイトが含まれていなければなりません。

Response_block_reference
応答内の、この機能で戻されたデータの最初のバイトのアドレスを指す、RODM によって設定されたポインターです。このパラメーターは、データが戻されないときにはゼロに設定されます。 1 つのユーザー API 呼び出しから行われるすべての操作で、1 つの共通の応答ブロックが共用されます。これらの対話には、EKG_ExecuteFunctionList または EKG_QueryMultipleSubfields 機能呼び出しで指定された事項がすべて含まれます。

Response_block_type
response_type_block は、通知ブロックが通知メソッドによって生成されたのか、オブジェクト削除申請によって生成されたのかを表します。有効な値は、以下のとおりです。

値	意味
1	通知メソッドによって生成された
2	オブジェクトが削除されていたときに、そのオブジェクトに関するオブジェクト削除申請が存在していて生成された

Response_block_used
RODM によって戻されたデータの長さをバイト単位で表します。メソッドまたはアプリケーションによって提供された応答ブロックが小さすぎて、戻されるデータを収めきれない場合、Response_block_used の値は、そのデータを収めるために応答ブロックのサイズに設定されます。この値は、Response_block_length の値よりも大きく、Response_block_length パラメーターおよび

Response_block_used パラメーター用の 8 バイトを含んでいます。このパラメーターは、データが戻されないときにはゼロに設定されます。

トランザクションが応答ブロック・データを提供して、それによって応答ブロック・オーバーフローが発生しなかった場合、Response_block_used パラメーターは、 Response_block_length パラメーター以下の値に設定されます。トランザクションが応答ブロック・オーバーフローを発生させた場合には、Response_block_used パラメーターは Response_block_length パラメーターよりも大きな値になります。

Response_data
照会機能によって戻されたデータを含む EKG_ExecuteFunctionList 応答ブロック内の領域です。機能ブロックの内の Response_block_reference ポインター (上記参照) を使用して、機能ごとのデータを検索します。形式は、機能の通常応答ブロック内で 8 バイトのヘッダーに続く形式と同じです。

Return_code
Return_code と Reason_code の値は、この特定の機能要求の状況を表します。最高の数値は、EKGUAPI 呼び出しの Transaction_info_block パラメーターにも重複して入ります。最も重大なエラーが重なっている場合、それらのうちで最初のものが報告されます。

Stop_ECB
オペレーター要求または API 要求に対する対応として現行バージョンの RODM が停止することをユーザーに通知するために使用するパラメーターです。ユーザー・アプリケーションが EKGWAIT を呼び出す場合は、この ECB が常にリストに組み込まれている必要があります。

Stop_type
RODM が静止してチェックポイント操作を実行した後で、その RODM を停止させる場合には、Stop_type として 1 を指定してください。 RODM が最終的なチェックポイント操作を実行せずに静止した後で、その RODM を停止させる場合には、Stop_type として 2 を指定してください。

Subfield
この機能の特定のサブフィールドを識別します。 EKG_WhereAmI 以外のすべての機能で有効な値は、次のとおりです。

値	サブフィールド
0	Notify (EKG_RevertToInherited 機能に対してのみ有効) を除くすべてのサブフィールド
1	値
2	Query
3	Change
4	Notify
5	Prev_val
6	Timestamp

EKG_WhereAmI 機能で有効な値は、次のとおりです。

値	サブフィールド
1	Value (メソッドは、名前付きメソッドでなければなりません)
2	Query
3	Change

4 Notify (通知メソッド)

Subfield_map
Subfield_map は、次のように定義されたビットマップです (ビットには、左から順に 0 から 31 までの番号が付いています)。ビットを 1 に設定すると、機能がそのサブフィールドを対象としていることが指定されます。 RODM は、対応するサブフィールドが存在することを示すために、出力ブロック内の Subfield_map ビットを 1 に設定します。

ビット サブフィールド
0 値
1 Query
2 Change
3 Notify
4 Prev_val
5 Timestamp
6 から 31
予約済み (ゼロに設定しなければなりません)

Subscription_info
この機能の対象となる通知申請を指定します。 Subscription_info は RecipientSpec データ・タイプとして定義されていて、指定された申請に関する User_appl_ID と Notification_queue を含んでいます。

User_appl_ID
このフィールドに関する機能ブロックの User_appl_ID パラメーターがヌル値 (ブランク) に設定されている場合、RODM は、このトランザクションを開始した Access_block で定義された User_appl_ID 値をデフォルトとして使用します。申請通知の場合、User_appl_ID パラメーターには通知を受けるアプリケーションを指定します。メッセージ・インターフェースを介して開始されたメソッドでヌルの User_appl_ID が指定されている場合、RODM によって提供された名前は、メッセージ・トランザクションを出した Access_block で指定された名前です。

APF (許可プログラム機能) で許可されたプログラムの場合、 User_password を指定する必要はありません。 User_password が指定されていない User_appl_ID は、RODM に対してユーザーを識別し、そのユーザーの権限レベルを判別します。 APF 許可されていないアプリケーション・プログラムには、User_password が必要です。 EKG_Connect 機能によって User_appl_ID と User_password が組み合わされ、 RODM に対してユーザーを識別し、ユーザーの権限レベルを判別します。

User_password
APF 許可されていないアプリケーション・プログラムの場合、ユーザー権限レベルの検査と RODM への接続を検査するために、RODM アクセス・ブロックで User_appl_ID と User_password の両方を指定する必要があります。検査された User_appl_ID は、ユーザーに与えられたアクセス権限の特定レベルを判別するために RODM によって使用されます。このパラメーターは最大長が 8 バイトであり、それよりも短い値の場合にはパラメーターの左側に左寄せされ、右側がブランクで埋め込まれます。

APF 許可されていないプログラムについて User_appl_ID および User_password を検査する際に、 RODM は z/OS システムの RACROUTE インターフェース

を使用します。 ユーザー ID、パスワード、およびアクセス許可レベルは、これらのインターフェースをサポートするセキュリティー管理プログラムに登録されていることが前提となります。

User_appl_ID が指定されている場合、User_password 値は、APF 許可されていないプログラムで有効なものでなければなりません。 Access_block 内の User_appl_ID パラメーターがすべてブランクになっている場合には、 APF 許可されているプログラムの場合にも、APF 許可されていないプログラムの場合にも、 User_password フィールドは無視されます。リソース・アクセス制御機能 (RACF) などのシステム許可機能 (SAF) 製品は、許可されたユーザー ID をこの機能呼び出しと関連付けようとします。そのユーザー ID が見つからない場合には、接続要求は拒否されます。検査済みのユーザー ID が見つかった場合には、そのユーザー ID が Access_block の User_appl_ID パラメーターに入ります。

User_area
通知ブロックを通知キューに入れるメソッドによって提供されたデータが入っているデータ域です。

User_word
User_word パラメーターは、呼び出しパラメーターを介して通知メソッドに渡される情報として使用されます。このパラメーターは、EKG_AddNotifySubscription 機能で使用される機能ブロック内で呼び出し元によって設定され、RODM 内で申請要求とともに保存され、通知メソッドによって引き渡しパラメーターとして利用することができ、通知が行われたときに未修正のまま通知機能に渡されるようになります。 User_word の最終的な値は、通知メソッドによって決められます。

Value_for_reason_code
メソッドの呼び出し元に渡される理由コードです。

Value_for_return_code
メソッドの呼び出し元に渡される戻りコードです。

RODM の戻りコードと理由コード

RODM に対してユーザーが行ったそれぞれの機能呼び出しについて、RODM プログラムは戻りコードと理由コードを出します。理由コードには、問題の原因となっている可能性のある事柄に関する特定情報が示されます。

以下の 4 つのセクションでは、4 つの戻りコードに関連して出される可能性のある理由コードについて説明します。表には、説明と推奨される訂正処理が示されています。 536 ページの『各機能に関する理由コードのリスト』および 539 ページの『各理由コードに関連する機能のリスト』には、ユーザーが使用する特定の機能呼び出しに対して出されるコードを判別できるように、相互参照が示されています。546 ページの『NetView 提供のメソッドに関連する理由コードのリスト』 には、NetView 提供のメソッドで戻される理由コードのリストが示されています。

理由コードは、それを出したプログラムまたはメソッドに応じて、次の 3 つの範囲に分類されます。

範囲	発行元
0 から 32767	RODM アプリケーション・プログラム・インターフェース

32768 から 49151
NetView 提供のメソッド
49152 から 65535
ユーザー作成のメソッド

理由コードを出すメソッドを作成する場合には、49152 から 65535 までの範囲の理由コードを使用してください。

32781 から 32996 の範囲の理由コードは、NetView 提供のメソッド EKGCTIM、EKGMIMV、EKGNEQL、EKGNLST、EKGNOTF、EKGNTHD、および FLBTRNMM によって出されます。これらの理由コードは、メソッドが RODM トランザクションからエラーまたは警告を受け取ったときに出されます。メソッドによって出された理由コードから 32780 を引くと、そのトランザクションに関して RODM が出した元の理由コードの値が得られます。元の値は、以下の表から探すことができます。メソッドは、変更を行わずにトランザクションの戻りコードを出します。

32810 から 32904 の範囲の理由コードは、EKGSPPI メソッドがプログラム間インターフェース・モジュール CNMNETV からエラーを受け取ったときにこのメソッドによって出されます。出される理由コードは、CNMNETV からの戻りコードに 32809 を加えたものです。 EKGSPPI メソッドによって出される理由コードから 32809 を引いてください。これによって得られた結果が、CNMNETV からの戻りコードです。 この戻りコードの意味については、「*IBM Tivoli NetView for z/OS アプリケーション・プログラマーズ・ガイド*」を参照してください。

メソッドを作成する際には、メソッドから戻りコードと理由コードを出す意味を認識している必要があります。メソッドによって戻された理由コードと戻りコードをアプリケーションがどのように解釈するのかについては、362 ページの『トランザクションのエラー条件』を参照してください。

「*IBM Tivoli NetView for z/OS Troubleshooting Guide*」には、RODM の問題、特に異常終了の障害追及に関する詳細が記載されています。

戻りコード 0 の場合の理由コード

表 200 には、戻りコード 0 とともに戻される理由コードが示されています。

表 200. 戻りコード 0 の場合の理由コード

理由コード	説明	訂正アクション
0	システムが、要求された機能を正常に実行しました。	不要
26	新しいデータ値は、古いデータ値と同じです。このフィールドのローカル・コピーが存在していなかった場合には、ローカル・コピーが作成されます。	不要
48	使用されません。	不要
142	システムが要求を正常に実行し、ローカル・コピーが作成されました。	不要
143	システムが要求を正常に実行し、戻り値は継承値になっています。	不要
167	使用されません。	不要

表 200. 戻りコード 0 の場合の理由コード (続き)

理由コード	説明	訂正アクション
180	ユーザーが RODM から切断されたときに、ユーザー・オブジェクトは削除されません。原因としては、StopMode でキュー・オブジェクトの維持が指定されているために、ユーザー・オブジェクトからキュー・オブジェクトへのリンクが除去されなかったことが考えられます。	不要
185	切断が成功しました。ユーザー・オブジェクトは、通知キュー・オブジェクトへのリンクがまだ存在しているため、RODM から削除されません。	再度接続して切断してください。
32769	比較されるデータ値が一致していません。	比較対象のデータの value サブフィールドを指定してください。

戻りコード 4 の場合の理由コード

表 201 には、戻りコード 4 とともに戻される理由コードが示されています。

表 201. 戻りコード 4 の場合の理由コード

理由コード	説明	訂正アクション
1	RODM が次のいずれかを行っているために、システムが要求を拒否しました。 • 静止 − チェックポイント要求の後で現行のすべてのトランザクションの完了を待っている • マスター・ウィンドウと変換ウィンドウをチェックポイント・データ・セットに書き込んでいる RODM はすべての新しいユーザー API 要求を拒否し、この理由コードを戻します。	チェックポイント処理の完了後に要求を再試行してください。
2	RODM の開始中であるため、システムが要求を拒否しました。	RODM が完全に初期化されてから要求を再試行してください。
3	RODM の停止中であるため、システムが要求を拒否しました。	指定された RODM を再始動させるか、あるいは RODM アクセス・ブロック内の RODM_name フィールドを更新して別の既存 RODM に接続してください。要求を再試行してください。
5	RODM がチェックポイント要求付きで停止しているため、システムが要求を拒否しました。指定された Sign_on_token は無効になりました。	この理由コードが EKG_Connect 機能の結果として戻された場合には、指定された RODM の再始動後に要求を再試行してください。この理由コードが EKG_Connect 機能の結果として戻されたのではない場合には、アクセス・ブロック内の RODM_name フィールドを訂正して別の RODM に接続し、新しい Sign_on_token を入手してください。新しい Sign_on_token を指定して要求を再試行してください。

表 201. 戻りコード 4 の場合の理由コード (続き)

理由コード	説明	訂正アクション
6	RODM がチェックポイント要求なしで停止しているため、システムが要求を拒否しました。指定された Sign_on_token は無効になりました。	この理由コードが EKG_Connect 機能の結果として戻された場合には、指定された RODM の再始動後に要求を再試行してください。この理由コードが EKG_Connect 機能の結果として戻されたのではない場合には、アクセス・ブロック内の RODM_name フィールドを訂正して別の RODM に接続し、新しい Sign_on_token を入手してください。新しい Sign_on_token を指定して要求を再試行してください。
24	システムが、通知リスト内の 1 つまたは複数のメソッドを起動できません。元のトランザクションは正常に完了しています。原因として、通知メソッドが再帰的であること、または既存のメソッドでエラーが発生していることが考えられます。	通知リスト内のすべてのメソッドが有効であるようにしてください。
27	応答ブロックの大きさが不十分です。オーバーフロー・ブロックが作成されます。メソッドによって出された照会機能については、オーバーフロー・ブロックは作成されません。	応答ブロック・オーバーフロー照会機能を使用して、オーバーフロー・ブロックからデータを取り出してください。
28	RODM ログ・ファイルが利用できません。 1 次および 2 次ログ・ファイルの両方のオープンまたは書き込みが正常に行えないか、あるいは LOGT コマンドが出されました。トランザクションが失敗しました。	システム管理者に連絡してください。
29	ログ・レコード・サイズがデフォルトの最大値 32761 バイトよりも大きくなっています。レコードは 32761 バイトに切り詰められます。	機能ブロック内の Method_parms のサイズを検査するか、あるいはログへの出力 (2008) 機能で指定されたログ・メッセージのサイズを検査してください。
30	機能ブロック内の Stop_ECB がヌルです。指定された RODM が停止するときには、ユーザーに通知が出されません。	不要
34	指定されたキュー・オブジェクトが作成されますが、ユーザー・オブジェクトとのリンクは作成できません。必要なストレージが使用できない可能性があります。	不要
38	RODM によって出された WTOR に対する直接応答により、オペレーターがチェックポイント要求を停止しました。この理由コードは、EKG_System オブジェクトの EKG_LastCheckpointResult フィールドに入っているもので、メソッド API またはユーザー API を介して戻されることはありません。	オペレーターに連絡してください。
40	このフィールドにすでに 1 次継承値が入っているため、システムはこのフィールド値を変更しません。	不要
41	フィールドがローカルで作成されているために、システムが要求を拒否しました。	不要

表 201. 戻りコード 4 の場合の理由コード (続き)

理由コード	説明	訂正アクション
42	指定されたメソッドは、正常に実行されなかったモジュール最新表示によって削除されているため、ヌル・モジュールになっています。 トランザクションが失敗しました。	メソッドを最新表示して要求を再試行してください。それが正常に行われない場合には、メソッドを削除してから再インストールしてください。
44	このユーザーに関しては、指定された通知キューにメッセージが入っていません。	不要
46	ユーザーによって提供された応答ブロックがヌルであるため、オーバーフロー・ブロックは検索されずに消去されます。	不要
47	ユーザーによって提供された応答ブロックの大きさが不十分なために、オーバーフロー・データの一部が廃棄されました。	不要
48	使用されません。	不要
49	使用されません。	不要
50	ユーザーが指定された ECB アドレスに WAIT を要求していないため、あるいは指定された ECB アドレスが無効であるために通知が失敗しました。ユーザー・オブジェクトの StopMode に応じて、キュー・オブジェクトまたは申請が削除されます。	不要
52	指定されたクラスが存在していないか、指定されたオブジェクト ID が存在していないため、システムが要求を拒否しました。 RODM は、照会機能の場合には戻りコード 4 を設定し、その他の機能の場合には戻りコード 8 を設定します。	このクラスまたは親クラスを訂正してください。要求を再試行してください。
54	指定されたオブジェクトが存在していないために、システムが要求を拒否しました。 RODM は、照会機能の場合には戻りコード 4 を設定し、その他の機能の場合には戻りコード 8 を設定します。	オブジェクト・データを訂正してください。要求を再試行してください。
56	指定されたフィールドが存在していないために、システムが要求を拒否しました。 RODM は、照会機能の場合には戻りコード 4 を設定し、その他の機能の場合には戻りコード 8 を設定します。	フィールド・データを訂正してください。要求を再試行してください。
57	オブジェクトの指定された 1 次親がオブジェクト子をもつクラスではないため、システムが要求を拒否しました。 RODM は、照会機能の場合には戻りコード 4 を設定し、その他の機能の場合には戻りコード 8 を設定します。	1 次親クラス・データを訂正してください。 1 次親クラス・データが正しい場合には、オブジェクト ID のクラス ID 部分を検査してください。要求を再試行してください。
62	指定されたサブフィールドが存在していないために、システムが要求を拒否しました。 RODM は、照会機能の場合には戻りコード 4 を設定し、その他の機能の場合には戻りコード 8 を設定します。	サブフィールド・データを訂正してください。要求を再試行してください。
72	ターゲット・フィールドはすでにリンクされています。システムは何のアクションも行っていません。	エンティティーおよびフィールド情報を訂正してください。要求を再試行してください。
75	ターゲット・フィールドはリンクされていません。システムは何のアクションも行っていません。	フィールド情報を訂正してください。要求を再試行してください。

表 201. 戻りコード 4 の場合の理由コード (続き)

理由コード	説明	訂正アクション
81	指定されたメソッドがインストールされていないために、システムが要求を拒否しました。 RODM は、メソッド・オブジェクトを削除する要求の場合には戻りコード 4 を設定し、その他の機能の場合には戻りコード 8 を設定します。	指定されたメソッドをインストールしてください。要求を再試行してください。
92	作成対象のフィールドが指定されたクラスのもとにすでに存在しているために、システムが要求を拒否しました。	フィールド・データを訂正してください。要求を再試行してください。
97	指定されたフィールド名のフィールドが子クラスにすでに存在しています。新しいフィールドが親クラスに作成され、子クラス上の既存のフィールドは、ローカルで定義されたデータを含むものとしてマーク付けされます。	不要
100	要求された 1 つまたは複数のサブフィールドが無効です。有効なサブフィールドは作成されています。 RODM は、フィールド作成機能の場合には戻りコード 4 を設定し、サブフィールド作成機能の場合には戻りコード 8 を設定します。	サブフィールド・マップを訂正してください。要求を再試行してください。
104	指定された 1 つまたは複数のサブフィールドがすでに存在しています。	サブフィールド・マップを訂正してください。要求を再試行してください。
110	指定されたオブジェクト名が指定された 1 次親クラスのもとで別のオブジェクトによって使用されているため、システムが要求を拒否しました。	オブジェクト名を訂正してください。要求を再試行してください。
112	指定されたフィールドに同じパラメーターを使用する通知申請がすでに入っているため、システムが要求を拒否しました。	要求データを訂正してください。 要求を再試行してください。
133	システムが timestamp サブフィールドの値を更新できません。ストレージが不足している可能性があります。	同じリソースに対して別のトランザクションを出して、そのトランザクションからの戻りコードおよび理由コードを検査してください。戻りコードが 12 で理由コードが 211 の場合には、ストレージが不足しています。 問題の原因がストレージ不足の場合には、ストレージを解放して要求を再試行してください。
146	指定された 1 つまたは複数のサブフィールドが、指定されたフィールドに存在していません。	サブフィールド情報を訂正してください。要求を再試行してください。
158	キューが上限に達したため、通知を通知キューに入れることができません。	通知キューの内容を照会するか、あるいは EKG_Maximum_Q_Entries の値を増やしてください。
173	システムによる要求の実行が正常に行われ、RODM によって通知キューが 1 つ作成されました。	この通知キュー・オブジェクトの EKG_ECBAddress を有効な値に変更してください。
174	通知情報ブロックが通知キューに入っています。 ECB アドレスがヌルになっているため、システムが指定されたユーザーに通知できません。	不要

表 201. 戻りコード 4 の場合の理由コード (続き)

理由コード	説明	訂正アクション
175	ユーザー・メッセージの長さが 32767 バイトを超えているため、メッセージの一部が切り捨てられました。	不要
181	指定されたキューがアクティブになっていないため、通知をこのキューに接続することはできません。	指定されたキュー・オブジェクトの EKG_Status 値を変更してください。
182	通知が通知キューに入りました。 システムが指定されたユーザーに通知できません。	不要
183	通知ブロックからの情報が応答ブロックに入りました。システムが、通知ブロックによって使用されるストレージを解放できません。	不要
191	指定されたメソッド・オブジェクトが NullMeth オブジェクトであるため、システムが要求を拒否しました。	メソッド・オブジェクト情報を訂正してください。要求を再試行してください。
204	応答ブロック内の元データが上書きされます。	不要
205	使用されません。	不要
206	使用されません。	不要
208	ユーザーがオーバーフロー・データを保管しないように指定したため、応答ブロック・オーバーフロー・データが廃棄されます。	応答ブロック・オーバーフロー・データが必要な場合には、 EKG_RBOverflowAction フィールドの値を保管するように変更してください。要求を再試行してください。
209	ECB アドレス 0 が検出されたため、ECB のリストでの待機を要求するユーザー要求が完了しませんでした。	ECB アドレスを訂正してください。
221	使用されません。	不要
604	エージェントが正しくない相関値 (ネットワーク・アドレス) を指定したため、相互に関連付けられた集合オブジェクトは作成されませんでした。	エージェント (分散マネージャー) が有効なネットワーク・アドレスを指定するように変更してください。
605	相互に関連付けられた集合オブジェクトがすでに存在するため、相互に関連付けられた集合オブジェクトは作成されませんでした。	不要
32770	長さが 32767 バイトを超えているため、NetView 提供の通知メソッドから出されたメソッド出力メッセージの一部が廃棄されます。要求が正常に完了しました。	メソッド出力メッセージを訂正してください。
45081	メソッドでエラーが発生しましたが、機能を完了させることができました。正しくないフィールド値が提供され、RODM がデフォルト値を使用したか、あるいは RODM 内のフィールドの値が正常に変更された後で、このメソッドが通知メソッドの障害を検出しました。	このエラーの原因となった条件を訂正して、今後障害が発生しないようにしてください。メソッドは、タイプ 1 の RODM ログ・エントリーとして書かれたメッセージに、エラーに関する情報を記録します。エラーの原因が通知メソッドの障害である場合、このメッセージには、通知メソッドによって設定された理由コードが含まれます。エラーの原因がフィールド値の誤りである場合、RODM ログには、フィールド、誤った値、およびその代わりに使用されたデフォルト値が指定されます。誤った値を訂正してください。

戻りコード 8 の場合の理由コード

表 202 には、戻りコード 8 とともに戻される理由コードが示されています。

表 202. 戻りコード 8 の場合の理由コード

理由コード	説明	訂正アクション
8	API バージョンが無効なために、システムが要求を拒否しました。	トランザクション情報ブロック内の API バージョン情報を訂正してください。要求を再試行してください。
9	要求された機能を使用する許可を呼び出し元が得ていないために、システムが要求を拒否しました。	User_appl_ID を正しく指定するか、あるいはシステム管理者に連絡して権限レベルの変更を依頼してください。
10	機能 ID が無効なために、システムが要求を拒否しました。	機能 ID を訂正してください。要求を再試行してください。
11	短命パラメーターを RODM にコピーするのに十分なストレージがシステムにないために、要求された機能が完了していません。	未使用のエンティティーおよびフィールドを除去するか、あるいはシステム管理者に連絡してください。要求を再試行してください。
13	指定された RODM が見つからないために、システムが要求を拒否しました。	指定された名前を使用して RODM を開始するか、あるいは RODM アクセス・ブロック内の RODM_name フィールドを訂正してください。要求を再試行してください。
14	誤った Sign_on_token が検出されたために、システムが要求を拒否しました。ユーザー・アプリケーションが EKG_Connect 機能によって指定の RODM に接続されていないか、あるいは Sign_on_token が変更されています。 複数のアプリケーションが同じユーザー ID を使用して RODM に接続されていて、そのうちの 1 つが RODM から切断されると、残りのアプリケーションに関する Sign_on_token は RODM によって取り消されます。この理由コードは、残りのアプリケーションが RODM に機能要求を送ったときに出されます。	ユーザー・アプリケーションで Sign_on_token を修正しないようにしてください。 EKG_Connect 機能を使用して指定の RODM に接続し、有効な Sign_on_token を入手してください。新しい Sign_on_token を指定して要求を再試行してください。
15	並行して実行される API 機能呼び出しの数がカスタマイズ・ファイルで指定された上限に達したため、システムが要求を拒否しました。	後で要求を再試行するか、あるいは RODM カスタマイズ・ファイル内の CONCURRENT_USERS 値を大きくしてください。 RODM をウォーム・スタートさせてください。
16	現在 RODM がシステム内に存在していないために、システムが要求を拒否しました。	指定された名前を使用して RODM を開始させてください。要求を再試行してください。
17	CSA 内で RODM サービス・モジュールが見つからないために、システムが要求を拒否しました。	システム管理者に連絡してください。
18	指定された機能がこのメソッドで許容されていないために、システムが要求を拒否しました。	機能ブロック内の機能 ID を訂正してください。要求を再試行してください。
21	ユーザーが指定したリスト要求の数が 0 または負であるため、システムは機能の要求されたリストを実行することができません。	Number_of_functions フィールドを訂正してください。要求を再試行してください。

表 202. 戻りコード 8 の場合の理由コード (続き)

理由コード	説明	訂正アクション
22	通知キュー名がヌルになっているために、システムが要求を拒否しました。	通知キュー名を訂正してください。 要求を再試行してください。
23	RODM に渡される (タイプが CharVar、GraphicVar、MethodSpec、SelfDefining、または BERVar の) データが無効なために、システムが要求を拒否しました。	データを訂正してください。要求を再試行してください。
33	ログ・レコード情報を保管するためのストレージが利用できないために、システムが要求を拒否しました。	未使用のエンティティーを削除してください。要求を再試行してください。
35	チェックポイント・マスター・ウィンドウのエラーです。 RODM 始動 JCL の EKGMAST DD ステートメントで指定された VSAM データ・セットが利用可能または使用可能になっていません。この理由コードは、EKG_System オブジェクトの EKG_LastCheckpointResult フィールドに入っているもので、メソッド API またはユーザー API を介して戻されることはありません。	システム管理者に連絡してください。
36	チェックポイント変換ウィンドウのエラーです。 RODM 始動 JCL の EKGTRAN DD ステートメントで指定された VSAM データ・セットが利用可能または使用可能になっていません。この理由コードは、EKG_System オブジェクトの EKG_LastCheckpointResult フィールドに入っているもので、メソッド API またはユーザー API を介して戻されることはありません。	システム管理者に連絡してください。
37	チェックポイント・データ・ウィンドウのエラーです。 RODM 始動 JCL の DD ステートメントで指定された、名前の接頭部が EKGD になっている 1 つ以上の VSAM データ・セットが利用可能または使用可能になっていません。この理由コードは、EKG_System オブジェクトの EKG_LastCheckpointResult フィールドに入っているもので、メソッド API またはユーザー API を介して戻されることはありません。	システム管理者に連絡してください。
39	Old_data_ptr によって指し示されたデータがターゲット・フィールドと異なっているために、システムが要求を拒否しました。	Old_data_ptr を訂正してください。要求を再試行してください。
52	指定されたクラスが存在していないか、指定されたオブジェクト ID が存在していないため、システムが要求を拒否しました。 RODM は、照会機能の場合には戻りコード 4 を設定し、その他の機能の場合には戻りコード 8 を設定します。	このクラスまたは親クラスを訂正してください。要求を再試行してください。
54	指定されたオブジェクトが存在していないために、システムが要求を拒否しました。 RODM は、照会機能の場合には戻りコード 4 を設定し、その他の機能の場合には戻りコード 8 を設定します。	オブジェクト・データを訂正してください。要求を再試行してください。

戻りコード 8 の場合の理由コード

表 202. 戻りコード 8 の場合の理由コード (続き)

理由コード	説明	訂正アクション
56	指定されたフィールドが存在していないために、システムが要求を拒否しました。 RODM は、照会機能の場合には戻りコード 4 を設定し、その他の機能の場合には戻りコード 8 を設定します。	フィールド・データを訂正してください。要求を再試行してください。
57	オブジェクトの指定された 1 次親がオブジェクト子をもつクラスではないため、システムが要求を拒否しました。 RODM は、照会機能の場合には戻りコード 4 を設定し、その他の機能の場合には戻りコード 8 を設定します。	1 次親クラス・データを訂正してください。 1 次親クラス・データが正しい場合には、オブジェクト ID のクラス ID 部分を検査してください。要求を再試行してください。
60	フィールド・タイプが共用であって、このクラスまたは下層クラスにオブジェクトがまだ存在しているために、システムが要求を拒否しました。	共用フィールドまたはそのサブフィールドを削除する前に、このクラスに属するオブジェクトを削除してください。
61	サブフィールド番号が無効なために、システムが要求を拒否しました。	サブフィールド番号を訂正してください。要求を再試行してください。
62	指定されたサブフィールドが存在していないために、システムが要求を拒否しました。 RODM は、照会機能の場合には戻りコード 4 を設定し、その他の機能の場合には戻りコード 8 を設定します。	サブフィールド・データを訂正してください。要求を再試行してください。
65	データ・タイプが ObjectLink または ObjectLinkList のフィールドにはこの機能が提供されないため、システムが要求を拒否しました。	機能 ID またはフィールド ID を訂正してください。要求を再試行してください。
66	新しいデータのデータ・タイプが指定されたフィールドのデータ・タイプと異なっているために、システムが要求を拒否しました。	データ・タイプまたはフィールドを訂正してください。要求を再試行してください。
67	システムで定義されたフィールドにこの機能が適用されないために、システムが要求を拒否しました。	機能 ID またはフィールドを訂正してください。要求を再試行してください。
70	この機能が notify サブフィールドに適用されないために、システムが要求を拒否しました。	サブフィールドまたは機能 ID を訂正してください。要求を再試行してください。
71	この機能が prev_val または timestamp サブフィールドに適用されないために、システムが要求を拒否しました。	サブフィールドまたは機能 ID を訂正してください。要求を再試行してください。
73	2 つのターゲット・オブジェクトが同一であるために、システムが要求を拒否しました。	エンティティー情報を訂正してください。要求を再試行してください。
74	フィールドのデータ・タイプがリンクまたはリンク解除機能では許されないために、システムが要求を拒否しました。	フィールド情報を訂正してください。要求を再試行してください。
76	notify サブフィールドが存在していないために、システムが要求を拒否しました。	指定されたフィールドに関する notify サブフィールドを作成してください。
77	システムで定義されたフィールドの一部にこの機能が適用されないために、システムが要求を拒否しました。	フィールドを訂正してください。要求を再試行してください。
79	リスト内の指定された機能ブロック・ポインターがヌルになっているために、システムが要求を拒否しました。	機能ブロック・ポインターを訂正してください。要求を再試行してください。

表 202. 戻りコード 8 の場合の理由コード (続き)

理由コード	説明	訂正アクション
80	このモジュールが再帰的にそれ自体を呼び出すために、システムが要求を拒否しました。	関連するメソッドを更新して、再帰呼び出しを除去してください。要求を再試行してください。
81	指定されたメソッド、または指定されたメソッドによって呼び出されたメソッドがインストールされていないために、システムが要求を拒否しました。RODM は、メソッド・オブジェクトを削除する要求の場合には戻りコード 4 を設定し、その他の機能の場合には戻りコード 8 を設定します。	指定されたメソッドをインストールしてください。指定されたメソッドが正しくインストールされている場合には、指定されたメソッドによって呼び出されるすべてのメソッドを正しくインストールしてください。要求を再試行してください。
83	応答ブロックの長さが 8 バイトに満たないために、システムが要求を拒否しました。	応答ブロック長を訂正してください。要求を再試行してください。
84	このユーザーはすでに RODM に接続されています。	不要
85	指定された Stop_type が無効であるために、システムが要求を拒否しました。	Stop_type を訂正してください。 Stop_type の値として 1 または 2 を指定することができます。要求を再試行してください。
86	指定されたクラスが無効であるか、あるいは RODM 予約クラス名であるために、システムが要求を拒否しました。	クラス名を訂正してください。要求を再試行してください。
87	指定されたクラス名が別のクラスで使用されているために、システムが要求を拒否しました。	クラス名を訂正してください。要求を再試行してください。
89	汎用クラスまたはシステム作成クラスを削除できないために、システムが要求を拒否しました。	クラス情報を訂正してください。 要求を再試行してください。
90	指定されたクラスに属するなんらかのエンティティーが存在しているために、システムが要求を拒否しました。	指定されたクラスに属するすべてのエンティティーを削除してください。要求を再試行してください。
91	指定されたフィールド名が無効であるか、あるいは RODM 予約フィールド名であるために、システムが要求を拒否しました。	フィールド名を訂正してください。要求を再試行してください。
93	作成対象のフィールドがすでにサブクラス内に存在していて、そのデータ・タイプまたはサブフィールドが異なっているために、システムが要求を拒否しました。	フィールド・データを訂正してください。要求を再試行してください。
94	作成対象のフィールドがすでに子クラス内に別のフィールド・タイプで存在しているために、システムが要求を拒否しました。	フィールド・データを訂正してください。要求を再試行してください。
95	フィールド・タイプ・フラグが無効なために、システムが要求を拒否しました。	フィールド・タイプ・フラグを訂正してください。フィールド・タイプ・フラグの値として 1、2、または 3 を指定することができます。要求を再試行してください。
96	データ・タイプが無効であるか予約データ・タイプであるために、システムが要求を拒否しました。	データ・タイプを訂正してください。予約データ・タイプを使用してフィールドを作成することはできません。要求を再試行してください。
98	ユーザー・アプリケーションにはシステムで定義されたフィールドを削除することが許されないため、システムが要求を拒否しました。	フィールド情報を訂正してください。要求を再試行してください。

表 202. 戻りコード 8 の場合の理由コード (続き)

理由コード	説明	訂正アクション
100	要求された 1 つまたは複数のサブフィールドが無効です。有効なサブフィールドは作成されています。 RODM は、フィールド作成機能の場合には戻りコード 4 を設定し、サブフィールド作成機能の場合には戻りコード 8 を設定します。	サブフィールド・マップを訂正してください。要求を再試行してください。
103	指定されたクラスに属するフィールドまたはサブフィールドが存在していないために、システムが要求を拒否しました。	クラス情報を訂正して、フィールドまたはサブフィールドが存在するクラスを指定してください。要求を再試行してください。
106	value サブフィールドを削除できないために、システムが要求を拒否しました。	サブフィールド名を訂正してください。要求を再試行してください。
107	メソッド・オブジェクト名が無効なために、システムが要求を拒否しました。	メソッド名を訂正してください。要求を再試行してください。
108	メソッドが使用中であるために、メソッドを削除する要求をシステムが拒否しました。	メソッド・オブジェクトの EKG_UsageCount フィールドの値を検査してください。値が 0 よりも大きい場合には、メソッドは使用中です。後で要求を再試行してください。
109	ユーザーが提供したオブジェクト名が無効であるか、あるいは RODM 予約オブジェクト名であるために、システムが要求を拒否しました。	要求が non_connect 要求であった場合には、オブジェクト名を訂正してください。要求が接続要求であった場合には、User_appl_ID を訂正して RODM オブジェクト名の規則に従うようにしてください。要求を再試行してください。
111	指定されたオブジェクトが他のオブジェクトとリンクしているために、システムが要求を拒否しました。	指定されたオブジェクトから他のすべてのオブジェクトをリンク解除してください。要求を再試行してください。
113	指定された申請が存在していないために、システムが要求を拒否しました。	要求データを訂正してください。 要求を再試行してください。
115	このフィールドのデータ・タイプがこの機能では無効なために、システムが要求を拒否しました。	フィールド・データ・タイプを訂正してください。要求を再試行してください。
117	機能情報配列内の機能 ID が無効なために、リスト内の機能が拒否されました。有効な機能 ID が指定された機能は処理されます。	機能情報配列内の機能 ID を訂正してください。
120	指定相関 ID のオーバーフロー・ブロックが存在していないために、システムが要求を拒否しました。	相関 ID を訂正してください。 要求を再試行してください。

表 202. 戻りコード 8 の場合の理由コード (続き)

理由コード	説明	訂正アクション
127	ユーザー ID が RODM に許可されていないために、システムが要求を拒否しました。	RODM のセキュリティーをアクティブにして実行**している**場合は、 RODM に接続しようとしているタスクが、使用しているセキュリティー・プロダクトに定義されており、RODMMGR クラスに定義されている該当する RODM リソースに対して読み取りアクセス権限を持っているようにしてください。例えば、RODM に接続するためには、ユーザーは少なくとも RODMMGR クラスの RODM1 リソースに対するアクセス権限を持っている必要があります。 (RODM1 の RODM 部分は、EKCUST の SEC_RNAME キーワードによって判別されます。) RODM に接続しようとしているタスクが開始済みプロシージャーである場合は、タスクを SAF プロダクトの STARTED クラスに定義してください。RACF の場合、これは、開始済みプロシージャー・テーブルにタスクを定義することによっても行えますが、STARTED クラスを使用する方法をお勧めします。 RODM のセキュリティーをアクティブにして実行**していない**場合は、ユーザー ID をブランクにして RODM に接続することが可能ですが、これは、許可されません。セキュリティーがアクティブでない場合は、接続要求にユーザー ID を指定する必要があります。 セキュリティーがアクティブでない場合に RODM ローダーを実行すると、ローダーが最初にブランクのユーザー ID で接続しようとするので、この理由コードを受け取ります。 その後、ローダーは、ブランクでないユーザー ID で自動的に接続しようとします。この場合、理由コードは無視してもかまいません。 **注:** ユーザー ID は SAF プロダクトから入手できるので、RODM がセキュリティーをアクティブにして実行している場合、ブランクのユーザー ID で実行することが許可されます。 セキュリティーをアクティブにして実行する場合、以下を行う必要があります。 • SAF プロダクトをインストールする • RODM に対するセキュリティー・クラスをアクティブにする (RODMMGR またはユーザー定義) • セキュリティー・クラスを EKGCUST の SEC_CLASS キーワードで識別する

戻りコード 8 の場合の理由コード

表 202. 戻りコード 8 の場合の理由コード (続き)

理由コード	説明	訂正アクション
128	次のいずれかの理由により、システムが要求を拒否しました。 • パスワードが満了している • パスワードが許可されていない • ユーザー ID が SAF プロダクトで取り消された	パスワードが満了しているか、あるいは許可されていない場合には、問題を訂正して要求を再試行してください。 パスワードの問題ではない場合には、セキュリティー管理者に SAF プロダクトのユーザー ID の状況を検査するように依頼してください。
130	仮想記憶間モードで接続が要求されたために、システムが要求を拒否しました。	非仮想記憶間モードで接続要求を出してください。
131	オーバーフロー・ブロックが消去されていないために、システムが要求を拒否しました。	応答ブロック・オーバーフロー照会要求を出してオーバーフロー・データを取り出してください。要求を再試行してください。
134	システムが prev_val サブフィールドの値を更新できません。ストレージが不足している可能性があります。	同じリソースに対して別のトランザクションを出して、そのトランザクションからの戻りコードおよび理由コードを検査してください。戻りコードが 12 で理由コードが 211 の場合には、ストレージが不足しています。 問題の原因がストレージ不足の場合には、ストレージを解放して要求を再試行してください。
135	長命パラメーターの長さが無効なために、システムが要求を拒否しました。	パラメーター長を訂正してください。要求を再試行してください。
136	Method_parms の長さが無効なために、システムが要求を拒否しました。	パラメーター長を訂正してください。最大長は 254 バイトです。要求を再試行してください。
138	Old_data_ptr がヌルになっているために、システムが要求を拒否しました。	Old_data_ptr を訂正してください。 要求を再試行してください。
139	フィールド ID が指定されていなくて、フィールド名ポインターまたはフィールド名長が無効であるために、システムが要求を拒否しました。	有効なフィールド ID、または有効なフィールド名ポインターおよびフィールド名長を指定してください。要求を再試行してください。
140	クラス ID が指定されていなくて、クラス名が無効であるために、システムが要求を拒否しました。	有効なクラス ID または有効なクラス名を指定してください。要求を再試行してください。
141	データ・タイプが ObjectLink の指定されたフィールドがすでに別のフィールドにリンクされているために、システムが要求を拒否しました。	フィールド情報を訂正してください。要求を再試行してください。
144	システムで作成されたフィールドまたはサブフィールドをユーザー・アプリケーションによって削除できないために、システムが要求を拒否しました。	フィールドまたはサブフィールド情報を訂正してください。要求を再試行してください。
145	指定されたフィールドまたはサブフィールドが読取専用であるために、システムが要求を拒否しました。	フィールドまたはサブフィールド情報を訂正してください。要求を再試行してください。
147	新しいデータの長さが無効なために、システムが要求を拒否しました。	データ長を訂正してください。 要求を再試行してください。
148	サブフィールド作成機能がシステム定義フィールドでは無効なために、システムが要求を拒否しました。	フィールド情報を訂正してください。要求を再試行してください。

表 202. 戻りコード 8 の場合の理由コード (続き)

理由コード	説明	訂正アクション
150	オブジェクト ID が指定されていなくて、オブジェクト名情報が無効であるために、システムが要求を拒否しました。	有効なオブジェクト ID または有効なオブジェクト名を指定してください。要求を再試行してください。
159	オブジェクト・ディレクトリーまたはフィールド名表が上限サイズに達したために、システムが要求を拒否しました。	別のオブジェクトまたはフィールドを選択してください。要求を再試行してください。
160	フィールド名が指定されていないために、システムが要求を拒否しました。	フィールド名を指定してください。 要求を再試行してください。
163	エンティティー・アクセス情報ブロックを指すポインターが無効であるために、システムが要求を拒否しました。	エンティティー・アクセス情報ブロックのポインターを訂正してください。要求を再試行してください。
164	フィールド・アクセス情報ブロックを指すポインターが無効であるために、システムが要求を拒否しました。	フィールド・アクセス情報ブロックのポインターを訂正してください。要求を再試行してください。
165	エンティティー・アクセス情報ブロックの Naming_count が無効であるために、システムが要求を拒否しました。	Naming_count 値を訂正してください。有効な Naming_count 値は 0、1、および 2 です。要求を再試行してください。
166	フィールド・アクセス情報ブロックの Naming_count が無効であるために、システムが要求を拒否しました。	Naming_count 値を訂正してください。有効な Naming_count 値は 0 および 1 です。要求を再試行してください。
169	オブジェクト ID が指定されていないために、システムが要求を拒否しました。	オブジェクト ID を指定してください。要求を再試行してください。
170	ユーザー・アプリケーションがシステム・オブジェクトを作成または削除できないために、システムが要求を拒否しました。	親クラス情報を訂正してください。要求を再試行してください。
176	新しいデータが無効なために、システムが要求を拒否しました。	新しいデータを訂正してください。 EKG_StopMode の有効な値は 1、2、および 3 です。 EKG_Status および EKG_MTraceFlag の有効な値は 0 および 1 です。 EKG_RBOverflowAction および EKG_ExternalLogState の有効な値は 1 および 2 です。 EKG_LogLevel および EKG_MLogLevel の有効な値は 0 から 999 です。要求を再試行してください。
186	ユーザー・アプリケーションがシステム・クラスに属するクラスを作成できないために、システムが要求を拒否しました。	親クラス情報を訂正してください。要求を再試行してください。
187	指定されたサブフィールド・マップがヌルであるために、システムが要求を拒否しました。	サブフィールド・マップを訂正してください。要求を再試行してください。
192	指定された機能 ID が有効な非同期実行ではないために、システムが要求を拒否しました。	機能 ID を訂正してください。要求を再試行してください。
193	メソッドによって設定された戻りコードまたは理由コードが無効です。	メソッドの戻りコードまたは理由コードを訂正してください。有効な戻りコードは 0、4、8、および 12 です。有効な理由コードは 0 から 65535 までです。
195	システム・フィールドをクラス・レベルで変更できないために、システムが要求を拒否しました。	データを訂正してください。要求を再試行してください。

表 202. 戻りコード 8 の場合の理由コード (続き)

理由コード	説明	訂正アクション
201	戻されるデータがヌル・ストリングであるために、システムが要求を拒否しました。	データを訂正してください。要求を再試行してください。
202	チェックポイント機能が使用不能になっているために、システムが要求を拒否しました。	RODM の開始時にすべてのチェックポイント・データ・セットが定義されているようにしてください。
203	応答ブロックがないために、システムが要求を拒否しました。	応答ブロックを指定してください。要求を再試行してください。
207	EKG_Connect 機能を完了させることができません。原因としては、RACF がアクティブであってもカスタマイズ・ファイルで指定されたクラスが RACF 内でアクティブになっていないこと、またはそのクラスが RACF で定義されていないことが考えられます。	システム管理者に連絡してください。
210	ECB の数がゼロであるために、ECB のリストでの待機を要求するユーザー要求が完了しませんでした。	リスト内の ECB の数を訂正してください。
214	エンティティー・アクセス情報ブロックの Naming_count が無効であるために、システムが要求を拒否しました。 この機能には有効なオブジェクト・アクセス情報が必要であるため、エンティティー・アクセス情報ブロックの Naming_count は 0 または 2 でなければなりません。	Naming_count を訂正してください。有効な値は 0 および 2 です。要求を再試行してください。
215	ユーザーが NullMeth の EKG_MTraceFlag を更新することは許されないため、システムが要求を拒否しました。	メソッド・オブジェクト情報を訂正してください。
220	リンクまたはリンク解除されるフィールドに対して定義されている変更メソッドの一方または両方が非ゼロの戻りコードを戻したため、システムがリンクまたはリンク解除要求を拒否しました。	変更メソッドを検査して、リンクまたはリンク解除を許容するためにどのような基準が使用されているのかを調べ、それらの基準に従ってください。
223	機能ブロックの Number_of_subfields フィールドに指定された値がゼロ、ゼロ未満、または 100,000 よりも大きな値になっていたため、システムが複数サブフィールド照会要求を拒否しました。 機能ブロックの Number_of_Fields フィールドに指定された値がゼロ、ゼロ未満、または 256 よりも大きな値になっていたため、システムが複数フィールド変更要求を拒否しました。	Number_of_subfields フィールドに正しい値を指定してください。 Number_of_Fields フィールドに正しい値を指定してください。
224	索引付きフィールドではこの入力データ・タイプが使用できないために、システムが要求を拒否しました。	入力データ・タイプまたは Field_type_flag を訂正してください。要求を再試行してください。
225	対応する Field_type_flag を指定してフィールドが作成されていないため、システムが要求を拒否しました。	フィールド名、またはフィールド ID およびフィールド・タイプの情報を訂正してください。要求を再試行してください。
226	APF 許可されていないプログラムに、ブランクのパスワードを指定して接続しようとしました。	機能ブロックの User_password フィールドで正しいパスワードを指定するか、あるいはプログラムを APF 許可プログラムにしてください。

表 202. 戻りコード 8 の場合の理由コード (続き)

理由コード	説明	訂正アクション
227	機能ブロック内の予約フィールドがゼロになっていないために、システムが要求を拒否しました。	機能ブロック内のすべての予約フィールドをゼロに設定するようにしてください。要求を再試行してください。
228	位置付け機能の索引付きデータ長フィールドが無効であるために、システムが要求を拒否しました。	索引付きデータ長フィールドが、データ・タイプ CharVar の場合には 0 から 32767 までに、またデータ・タイプ IndexList の場合には 0 から 254 までになるようにしてください。要求を再試行してください。
229	位置付け機能の索引付きデータ値ポインターが無効であるために、システムが要求を拒否しました。	索引付きデータ値ポインターを訂正してください。要求を再試行してください。
230	IndexList フィールドの長さが、各エレメントの 2 バイト長フィールドを含む各エレメントの合計長と異なっているために、システムが要求を拒否しました。	長さを正しく指定してください。要求を再試行してください。
231	IndexList フィールドに重複値が含まれているために、システムが要求を拒否しました。	フィールド内でそれぞれの値が固有になるようにしてください。要求を再試行してください。
232	IndexList フィールドの値から検出された長さが無効であるために、システムが要求を拒否しました。	それぞれの値の長さを 0 バイトから 254 バイトまでにしてください。要求を再試行してください。
32768	Long_lived_parm で指定されたデータが無効です。要求コード、オプション・コード、または change_status 使用可能化パラメーターが誤っている可能性があります。また、テスト値のデータ・タイプが無効な可能性もあります。要求は失敗しました。	Long_lived_parm を訂正してください。
32771	照会された value サブフィールドのデータ・タイプが無効であるために、システムが要求を拒否しました。	使用されているメソッドのデータ・タイプを正しく指定してください。 NetView 提供のメソッドについては、 548 ページの『NetView 提供のメソッド』を参照してください。メソッドに渡されるパラメーター・リストを訂正してください。
32772	指定されたフィールド内の値のデータ・タイプが無効なために、システムが NetView 提供の通知メソッドを拒否しました。要求は失敗しました。	指定されたフィールドのフィールド・データ・タイプを訂正してください。有効なフィールド・データ・タイプは Smallint、Integer、 Floating、TimeStamp、または CharVar です。
32790	メソッドに渡された短命パラメーターがヌルです。	ポインターを訂正してください。
32791	短命パラメーター内の 1 つまたは複数のデータ項目が CharVar データ・タイプになっていません。	短命パラメーターを訂正してください。
32792	短命パラメーター内の 1 つまたは複数のデータ項目が長すぎます。	短命パラメーターを訂正してください。
32793	短命パラメーター内のデータ項目の数が誤っています。	短命パラメーターを訂正してください。
32794	EKGSPPI メソッドに渡される短命パラメーターの RCVRID_CHARVAR 値がブランクまたはヌルになっています。	RCVRID_CHARVAR に有効な値を指定してください。
32795	EKGSPPI メソッドに渡される短命パラメーターの ASSIST_CHARVAR 値が無効です。	ASSIST_CHARVAR に有効な値を指定してください。

表 202. 戻りコード 8 の場合の理由コード (続き)

理由コード	説明	訂正アクション
32796	EKGSPPI メソッドに渡される短命パラメーターの TASKINFO_CHARVAR 値が無効です。	TASKINFO_CHARVAR に有効な値を指定してください。
32797	EKGSPPI メソッドに渡される短命パラメーターの TASKNAME_CHARVAR 値がブランクまたはヌルになっています。	TASKNAME_CHARVAR に有効な値を指定してください。
32798	EKGSPPI メソッドに渡される短命パラメーターの CMD_CHARVAR 値がブランクまたはヌルになっています。	CMD_CHARVAR に有効な値を指定してください。
45057	DUIFCUAP メソッドのパラメーターにより、2 つのオブジェクト間での AggregationParent と AggregationChild のリンクの削除が指定されています。しかし、指定されたオブジェクト間にはこのリンクが存在していません。	これらのオブジェクトがこれまでにリンクされたことがないか、あるいは DUIFCUAP メソッドによってすでにリンク解除されている場合には、アクションは不要です。オブジェクトが DUIFCUAP メソッドによらずにリンク解除されている場合には、DUIFFAWS メソッドを実行してください。オブジェクトが DUIFCUAP メソッドでリンク解除されていて、そのメソッドが正常に完了しなかった場合には、DUIFFAWS メソッドを実行してください。
45058	DUIFCUAP メソッドのパラメーターにより、2 つのオブジェクト間での AggregationParent と AggregationChild のリンクの作成が指定されています。しかし、指定されたオブジェクト間にはこのリンクがすでに存在しています。	オブジェクトがすでに DUIFCUAP メソッドによってリンクされている場合には、処理は不要です。オブジェクトが DUIFCUAP メソッドによらずにリンクされている場合には、 DUIFFAWS メソッドを実行してください。オブジェクトが DUIFCUAP メソッドによってリンクされていて、そのメソッドが正常に完了しなかった場合には、DUIFFAWS メソッドを実行してください。
45061	使用されません。	不要
45066	DUIFCUAP が、要求されたリンクを作成しません。要求されたリンクを作成すると、集約階層でループが発生するか、あるいは、集約階層におけるリンク要求の対象となったオブジェクトの上層ですでにループが発生しています。ループ・パスに関する情報は RODM ログに書き込まれています。	DUIFCUAP に渡すパラメーターを訂正して、リンク対象の正しいオブジェクトを指定するか、あるいは集約階層からループを除去してください。DUIFFAWS を実行して、集合オブジェクトが正しく初期化されるようにしてください。 DUIFCUAP を再び実行して、希望するリンクを作成してください。
45070	メソッドに提供された短命パラメーターが無効です。これらのパラメーターは、INVOKED_WITH RODM ロード機能のプリミティブ・ステートメントによって提供された可能性があります。 これらのパラメーターは RODM ログに書き込まれています。	RODM ログを調べてメソッドに送られたパラメーターの形式および値がそのメソッドに適しているかどうか確認してください。
45071	DUIFCUAP メソッドまたは DUIFCLRT メソッドに対する入力パラメーターで指定されたオブジェクトが RODM 内に存在していません。欠落しているオブジェクトに関する情報が RODM ログに書き込まれています。	欠落したオブジェクトを作成するか、あるいはメソッドの入力パラメーターを訂正して、要求を再試行してください。
45077	このトランザクション用に起動されたメソッドに関してエラーが発生しました。診断情報が RODM ログに書き込まれています。	RODM ログを調べて、発生した特定のエラーに関する情報を探してください。

表 202. 戻りコード 8 の場合の理由コード (続き)

理由コード	説明	訂正アクション
45078	トランザクションの処理中にエラーが発生しました。 RODM データ・キャッシュに矛盾したフィールド値が含まれている可能性があります。	RODM ログを調べて、発生した特定のエラーに関する情報を探してください。特定エラーを訂正してください。トランザクションを繰り返すか、あるいは DUIFFAWS メソッドを実行してください。
45079	トランザクションの処理中にエラーが発生しました。トランザクションのために必要な変更の一部が完了しましたが、全部は完了していません。	RODM ログを調べて、発生した特定のエラーに関する情報を探してください。特定エラーを訂正してください。トランザクションを繰り返してください。
45080	EKG_ChangeField 機能要求の New_data_ptr パラメーターで指定されたデータの値またはデータ・タイプが無効です。	RODM ログを調べて、エラーが発生した特定のフィールドに関する情報を探してください。エラーを訂正してください。トランザクションを繰り返してください。
45082	トランザクションの処理中にエラーが発生しました。 1 つまたは複数の集合オブジェクトの DisplayStatus フィールドの値が誤っている可能性があります。	RODM ログを調べて、発生した特定のエラーに関する情報を探してください。特定エラーを訂正してください。トランザクションを繰り返すか、あるいは DUIFFRAS メソッドを実行してください。
45083	自己定義メソッドのパラメーターに入れてメソッドに渡されたオブジェクトが、予期されたクラスではありません。	有効なメソッド・パラメーターを指定するようにしてください。 GMFHS メソッド DUIFCLRT の場合には、メソッド・パラメーターで指定された最初のオブジェクトは実オブジェクト、集合オブジェクト、またはシャドー・オブジェクトでなければならず、指定された 2 番目のオブジェクトは表示リソース・タイプのオブジェクトでなければなりません。 GMFHS メソッド DUIFCUAP の場合には、メソッド・パラメーターで指定された最初のオブジェクトは実オブジェクトまたは集合オブジェクトでなければならず、指定された 2 番目のオブジェクトは集合オブジェクトでなければなりません。
45092	GMFHS アプリケーションの RODM への接続が失敗しました。別の GMFHS アプリケーションがすでに RODM に接続されています。	GMFHS 初期化メンバー (DUIGINIT) で指定されている RODM アプリケーションの名前が正しいかどうかを確認してください。一度に 1 つの GMFHS アプリケーションしか RODM に接続することはできません。
45093	RODM にインストールされた GMFHS メソッドのバージョンが、 RODM と接続しようとしている GMFHS アプリケーションのバージョンと互換性がありません。	GMFHS 初期化メンバー (DUIGINIT) で指定されている RODM アプリケーションの名前が正しいかどうかを確認してください。 GMFHS アプリケーションのバージョンは、RODM にインストールされている GMFHS メソッドのバージョンと同じでなければなりません。

戻りコード 12 の場合の理由コード

534 ページの表 203 には、戻りコード 12 とともに戻される理由コードが示されています。

戻りコード 12 の場合の理由コード

表 203. 戻りコード 12 の場合の理由コード

理由コード	説明	訂正アクション
7	使用されません。	不要
19	オーバーフロー・ブロックに十分なストレージがないため、応答ブロック・オーバーフロー・データの一部またはすべてが廃棄されます。	オーバーフロー・ブロックを消去するために、応答ブロック・オーバーフロー照会を出してください。より大きな応答ブロックを使用して要求を再試行してください。応答ブロック内の Response_block_used フィールドには、応答データを収めるために必要なストレージの量が示されています。
20	トランザクションの途中で異常終了が発生したため、要求された機能が完了していない可能性があります。	このトランザクションのために RODM に渡された制御ブロックが有効であるかどうか確認してください。異常終了の問題の診断については、「*IBM Tivoli NetView for z/OS Troubleshooting Guide*」を参照してください。
25	RODM によってチェックポイント処理で現在書き込まれているデータ・ウィンドウ内のデータをトランザクションが更新しようとしたため、システムが要求を拒否しました。	後で要求を再試行してください。
63	指定されたメソッドのモジュールを RODM アドレス・スペースにロードできないため、システムがメソッド・オブジェクトを作成する要求を拒否しました。	メソッド・ライブラリーにこのメソッドが存在していることを確認してください。
68	使用されません。	不要
82	指定されたメソッドのモジュールが、失敗したモジュール最新表示によって削除されています。	メソッドのモジュールを最新表示し直してください。
118	使用されません。	不要
121	ストレージ不足のため、システムが要求を拒否しました。以下の位置のいずれかで、ストレージが使い尽くされました。 • VSAM 変換チェックポイント・データ・セット • 変換ウィンドウ	この戻りコードおよび理由コードの理由のほとんどは、 VSAM データ・セットが小さすぎるためです。この場合、メッセージ EKG1116I もコンソールに書き込まれます。このメッセージを受け取った場合は、 RODM 変換チェックポイント・データ・セットのサイズを増加してください。チェックポイント・データ・セットのサイズは、 RODM 始動 JCL 内の DD 名 EKGTRAN によって指定します。
122	ストレージ不足のため、システムが要求を拒否しました。以下の位置のいずれかで、ストレージが使い尽くされました。 • VSAM チェックポイント・データ・セット • RODM データ・スペース	この戻りコードおよび理由コードの理由のほとんどは、 VSAM チェックポイント・データ・セットが小さすぎるためです。その場合は、システム・コンソールにメッセージ EKG1117I が出力されます。このメッセージを受け取った場合は、 RODM データ・ウィンドウ・チェックポイント・データ・セットのサイズを増加するか、または別のデータ・ウィンドウ・チェックポイント・データ・セットを追加してください。チェックポイント・データ・セットは、RODM 始動 JCL 内の DD 名 EKGDnnn によって指定します。
123	使用されません。	不要

表 203. 戻りコード 12 の場合の理由コード (続き)

理由コード	説明	訂正アクション
124	このクラスで利用できる ID がないために、システムが要求を拒否しました。	未使用のエンティティーを削除してください。要求を再試行してください。
126	このフィールドで利用できる ID がないために、システムが要求を拒否しました。	未使用のフィールドを削除してください。要求を再試行してください。
156	通知キュー・ブロック用のストレージがないため、キュー・オブジェクトを作成する要求をシステムが拒否しました。	未使用のエンティティーを削除してください。後で要求を再試行してください。
157	通知情報ブロック用のストレージがないため、システムが要求を拒否しました。	後で要求を再試行してください。
177	利用可能なシステム生成オブジェクト名がないため、システムが要求を拒否しました。	オブジェクト名を指定してください。要求を再試行してください。
179	ユーザー・オブジェクトを作成できないため、システムが要求を拒否しました。原因としては、利用可能なストレージが不足していることが考えられます。	後で要求を再試行してください。
188	使用されません。	不要
194	メソッドで実行エラーが発生したため、システムが要求を完了できません。	RODM ログ・レコードに追加情報が書き込まれているかどうか調べてください。
198	ユーザー・オブジェクトのフィールドを変更できないため、システムが要求を拒否しました。ストレージが不足している可能性があります。	ストレージを解放して要求を再試行してください。
199	オペレーターがユーザー・トランザクションを取り消しました。	オペレーターに連絡して確認してください。
200	RODM が静止しているため、システムがユーザー・トランザクションを取り消しました。	後で要求またはメソッドを再試行してください。
211	利用可能なストレージがないため、システムがエラーを処理できません。保持されているストレージを解放できません。システムは、再始動しなければ使用できません。	システム管理者に連絡して RODM の再始動を依頼してください。
212	リカバリー不能エラーが発生したため、システムがトランザクションを完了できません。 RODM はタイプ 2 のログ・レコードを RODM ログに書き込みます。	ログ・レコードの内容を調べて、異常終了の原因となったトランザクションに関する情報を探してください。異常終了の問題の診断については、「*IBM Tivoli NetView for z/OS Troubleshooting Guide*」を参照してください。
213	RODM がアプリケーションまたはメソッドのインターフェース・ブロックにアクセスしたときに異常終了が発生したため、要求された機能が完了しませんでした。	アドレス例外の原因となるようなエラーがないか、インターフェース・ブロックを調べてください。異常終了の問題の診断については、「*IBM Tivoli NetView for z/OS Troubleshooting Guide*」を参照してください。
216	使用されません。	不要
240	RODM トランザクションが正常に完了しませんでした。異常終了が起こった可能性があります。	RODM ログを調べて、発生した特定のエラーに関する情報を探してください。エラーを訂正した後でトランザクションを繰り返してください。

表 203. 戻りコード 12 の場合の理由コード (続き)

理由コード	説明	訂正アクション
600	相関機能によって出された EKG_QueryMultipleSubfields 要求は、 1 つの実オブジェクトに対して失敗しました。	相関機能が正しく実行された場合には、このエラーを無視してください。相関機能が正しく実行されなかった場合には、IBM ソフトウェア・サポートに連絡してください。
601	相関機能によって出された EKG_QueryMultipleSubfields 要求は、 1 つの集合オブジェクトに対して失敗しました。	相関機能が正しく実行された場合には、このエラーを無視してください。相関機能が正しく実行されなかった場合には、IBM ソフトウェア・サポートに連絡してください。
602	相関機能によって出された EKG_ChangeMultipleFields 要求は、 1 つの集合オブジェクトに対して失敗しました。	相関機能が正しく実行された場合には、このエラーを無視してください。相関機能が正しく実行されなかった場合には、IBM ソフトウェア・サポートに連絡してください。
603	相関機能によって出された EKG_Locate 要求は、 1 つの実オブジェクトに対して失敗しました。	相関機能が正しく実行された場合には、このエラーを無視してください。相関機能が正しく実行されなかった場合には、IBM ソフトウェア・サポートに連絡してください。
603	相関機能によって出された EKG_Locate 要求は、 1 つの実オブジェクトに対して失敗しました。	相関機能が正しく実行された場合には、このエラーを無視してください。相関機能が正しく実行されなかった場合には、IBM ソフトウェア・サポートに連絡してください。
604	相関可能値の長さが 2 文字未満であったため、相関機能によって aggregateSystem オブジェクトが作成されませんでした。	相関機能が正しく実行された場合には、このエラーを無視してください。相関機能が正しく実行されなかった場合には、IBM ソフトウェア・サポートに連絡してください。
605	相関機能によって出された EKG_CreateObject 要求が、 1 つの集合オブジェクトについて失敗しました。	相関機能が正しく実行された場合には、このエラーを無視してください。相関機能が正しく実行されなかった場合には、IBM ソフトウェア・サポートに連絡してください。
606	相関機能によって出された EKG_TriggerOIMethod 要求が、 1 つの集合オブジェクトに関する DisplayResourceType とのリンクに失敗しました。	相関機能が正しく実行された場合には、このエラーを無視してください。相関機能が正しく実行されなかった場合には、IBM ソフトウェア・サポートに連絡してください。
45085	使用されません。	不要
45086	ビュー内のオブジェクトが変更されたときにエラーが発生しました。	RODM ログを調べて、発生した特定のエラーに関する情報を探してください。エラーを訂正した後でトランザクションを繰り返してください。

各機能に関する理由コードのリスト

表 204 には、RODM API 機能の機能 ID、および各機能により戻される理由コードがリストされています。

表 204. API 機能に関連する理由コード

機能 ID	理由コード
ユーザー API の共通理由コード	0 1 2 3 5 6 8 9 10 13 14 15 16 17 20 23 25 131 199 200 211 212 213 240

表 204. API 機能に関連する理由コード (続き)

機能 ID	理由コード
メソッド API の共通理由コード	0 10 18 20 192
1101	30 84 109 127 128 130 179 198 207
1102	180 198
1201	35 36 37 38 202
1202	85 202
1302	24 52 86 87 121 122 124 136 140 163 165 186 32768 32769 32770 32772
1303	24 52 89 90 136 140 163 165 32768 32769 32770 32772
1304	52 91 92 93 94 95 96 97 100 121 122 126 139 140 159 160 163 164 165 166
1305	52 60 98 103 139 140 144 163 164 165 166
1306	52 100 103 104 122 139 140 148 163 164 165 166 187
1307	52 60 98 103 106 139 140 144 146 163 164 165 166 187
1401	24 26 42 52 54 56 57 65 66 67 80 81 122 133 134 135 136 139 140 142 145 147 150 163 164 165 166 176 194 195 215 230 231 232 32768 32769 32770 32771 32772
1402	24 26 39 52 54 56 57 65 66 67 80 81 122 133 134 135 136 138 139 140 142 145 147 150 163 164 165 166 176 194 195 215 230 231 232 32768 32769 32770 32772
1403	26 52 54 56 57 61 62 65 66 67 70 71 81 122 135 139 140 142 145 147 150 163 164 165 166 176 195 215 230 231 232
1404	26 39 52 54 56 57 61 62 65 66 67 70 71 81 122 135 138 139 140 142 145 147 150 163 164 165 166 176 195 215 230 231 232
1405	24 52 54 56 57 72 73 74 122 133 136 139 140 141 145 150 163 164 166 214 220 32768 32769 32770 32772
1406	52 54 56 57 72 73 74 122 133 139 140 141 145 150 163 164 166 214
1407	24 52 54 56 57 73 74 75 133 136 139 140 145 150 163 164 166 214 220 32768 32769 32770 32772
1408	52 54 56 57 73 74 75 133 139 140 145 150 163 164 166 214
1409	22 24 34 52 63 107 109 110 121 122 136 140 150 156 159 163 165 170 177 214 604 605 32768 32769 32770 32772

表 204. API 機能に関連する理由コード (続き)

機能 ID	理由コード
1410	22 24 52 54 57 81 107 108 111 113 136 140 150 163 170 191 214 32768 32769 32770 32772
1411	40 41 52 54 56 57 61 62 65 67 70 71 122 139 140 145 150 163 164 165 166
1412	22 52 54 56 57 76 77 81 112 122 135 139 140 150 156 163 164 165 166 173
1413	22 52 54 56 57 76 77 113 135 139 140 150 163 164 165 166
1415	42 52 54 56 57 80 81 82 115 136 139 140 150 163 164 165 166 191 194 214 32768 32771
1416	11 42 80 81 191 194
1417	22 52 54 57 77 109 112 122 135 140 150 156 163 173 214
1418	22 52 54 57 77 109 113 135 140 150 163 214
1419	24 26 42 52 54 56 57 65 66 67 80 81 122 133 134 135 136 139 140 142 145 147 150 163 164 165 166 176 194 215 223 227 230 231 232 602 32768 32769 32770 32771 32772
1501	19 27 42 52 54 56 57 80 83 136 139 140 143 150 163 164 165 166 194 208
1502	19 27 52 54 56 57 61 62 83 136 139 140 143 150 163 164 165 166 208
1503	19 27 52 54 57 83 139 140 150 163 164 165 166 208
1504	19 27 52 54 56 57 83 139 140 150 163 164 165 166 208
1505	19 27 56 139 164 166 208
1506	19 27 52 54 56 57 83 139 140 150 163 164 165 166 208
1507	19 22 27 44 54 57 83 183 208
1508	27 52 56 83 139 140 150 163 164 165 166 223 600 601
1509	139 164 166 224 225 227 228 229 603
1510	46 47 120
1600	21 79 83 117
2001	27 83
2002	21 52 54 118
2004	27 83 201 203 204
2005	22 50 157 158 174 175 181 182
2006	28 29 33 193 45086

表 204. API 機能に関連する理由コード (続き)

機能 ID	理由コード
2008	28 29 33
2011	19 27 54 169 208
EKGWAIT	209 210

各理由コードに関連する機能のリスト

表 205 には、RODM の理由コードと、各理由コードを戻す RODM API 機能の機能 ID がリストされています。機能 ID から機能名を調べるには、545 ページの表 206 を参照してください。

表 205. 各理由コードに関連する機能 ID

理由コード	機能 ID
0	ユーザー API およびメソッド API の共通理由コード
1	ユーザー API の共通理由コード
2	ユーザー API の共通理由コード
3	ユーザー API の共通理由コード
5	ユーザー API の共通理由コード
6	ユーザー API の共通理由コード
8	ユーザー API の共通理由コード
9	ユーザー API の共通理由コード
10	ユーザー API およびメソッド API の共通理由コード
11	1416
13	ユーザー API の共通理由コード
14	ユーザー API の共通理由コード
15	ユーザー API の共通理由コード
16	ユーザー API の共通理由コード
17	ユーザー API の共通理由コード
18	メソッド API の共通理由コード
19	1501 1502 1503 1504 1505 1506 1507 2011
20	ユーザー API およびメソッド API の共通理由コード
21	1600 2002
22	1409 1410 1412 1413 1417 1418 1507 2005
23	ユーザー API の共通理由コード
24	1302 1303 1401 1402 1405 1407 1409 1410 1419
25	ユーザー API の共通理由コード
26	1401 1402 1403 1404 1419
27	1501 1502 1503 1504 1505 1506 1507 1508 2001 2004 2011

表 205. 各理由コードに関連する機能 ID (続き)

理由コード	機能 ID
28	2006 2008
29	2006 2008
30	1101
33	2006 2008
34	1409
35	1201
36	1201
37	1202
38	1201
39	1402 1404
40	1411
41	1411
42	1401 1402 1415 1416 1419 1501
44	1507
46	1510
47	1510
50	2005
52	1302 1303 1304 1305 1306 1307 1401 1402 1403 1404 1405 1406 1407 1408 1409 1410 1411 1412 1413 1415 1417 1418 1419 1501 1502 1503 1504 1506 1508 2002
54	1401 1402 1403 1404 1405 1406 1407 1408 1410 1411 1412 1413 1415 1417 1418 1419 1501 1502 1503 1504 1506 1507 2002 2011
56	1401 1402 1403 1404 1405 1406 1407 1408 1411 1412 1413 1415 1419 1501 1502 1504 1505 1506 1508
57	1401 1402 1403 1404 1405 1406 1407 1408 1410 1411 1412 1413 1415 1417 1418 1419 1501 1502 1503 1504 1506 1507
60	1305 1307
61	1403 1404 1411 1502
62	1403 1404 1411 1502
63	1409
65	1401 1402 1403 1404 1411 1419
66	1401 1402 1403 1404 1419
67	1401 1402 1403 1404 1411 1419
70	1403 1404 1411
71	1403 1404 1411
72	1405 1406
73	1405 1406 1407 1408

表 205. 各理由コードに関連する機能 ID (続き)

理由コード	機能 ID
74	1405 1406 1407 1408
75	1407 1408
76	1412 1413
77	1412 1413 1417 1418
79	1600
80	1401 1402 1415 1416 1419 1501
81	1401 1402 1403 1404 1410 1412 1415 1416 1419
82	1415
83	1501 1502 1503 1504 1506 1507 1508 1600 2001 2004
84	1101
85	1202
86	1302
87	1302
89	1303
90	1303
91	1304
92	1304
93	1304
94	1304
95	1304
96	1304
97	1304
98	1305 1307
100	1304 1306
103	1305 1306 1307
104	1306
106	1307
107	1409 1410
108	1410
109	1101 1409 1417 1418
110	1409
111	1410
112	1412 1417
113	1410 1413 1418
115	1415
117	1600
118	2002
120	1510

表 205. 各理由コードに関連する機能 ID (続き)

理由コード	機能 ID
121	1302 1304 1409
122	1302 1304 1306 1401 1402 1403 1404 1405 1406 1409 1411 1412 1417 1419
124	1302
126	1304
127	1101
128	1101
130	1101
131	ユーザー API の共通理由コード
133	1401 1402 1405 1406 1407 1408 1419
134	1401 1402 1419
135	1401 1402 1403 1404 1412 1413 1417 1418 1419
136	1302 1303 1401 1402 1405 1407 1409 1410 1415 1419 1501 1502
138	1402 1404
139	1304 1305 1306 1307 1401 1402 1403 1404 1405 1406 1407 1408 1411 1412 1413 1415 1419 1501 1502 1503 1504 1505 1506 1508 1509
140	1302 1303 1304 1305 1306 1307 1401 1402 1403 1404 1405 1406 1407 1408 1409 1410 1411 1412 1413 1415 1417 1418 1419 1501 1502 1503 1504 1506 1508
141	1405 1406
142	1401 1402 1403 1404 1419
143	1501 1502
144	1305 1307
145	1401 1402 1403 1404 1405 1406 1407 1408 1411 1419
146	1307
147	1401 1402 1403 1404 1419
148	1306
150	1401 1402 1403 1404 1405 1406 1407 1408 1409 1410 1411 1412 1413 1415 1417 1418 1419 1501 1502 1503 1504 1506 1508
156	1409 1412 1417
157	2005
158	2005
159	1304 1409
160	1304

表 205. 各理由コードに関連する機能 ID (続き)

理由コード	機能 ID
163	1302 1303 1304 1305 1306 1307 1401 1402 1403 1404 1405 1406 1407 1408 1409 1410 1411 1412 1413 1415 1417 1418 1419 1501 1502 1503 1504 1506 1508
164	1304 1305 1306 1307 1401 1402 1403 1404 1405 1406 1407 1408 1411 1412 1413 1415 1419 1501 1502 1503 1504 1505 1506 1508 1509
165	1302 1303 1304 1305 1306 1307 1401 1402 1403 1404 1409 1411 1412 1413 1415 1419 1501 1502 1503 1504 1506 1508
166	1304 1305 1306 1307 1401 1402 1403 1404 1405 1406 1407 1408 1411 1412 1413 1415 1419 1501 1502 1503 1504 1505 1506 1508 1509
169	2011
170	1409 1410
173	1412 1417
174	2005
175	2005
176	1401 1402 1403 1404 1419
177	1409
179	1101
180	1102
181	2005
182	2005
183	1507
186	1302
187	1306 1307
191	1410 1415 1416
192	メソッド API の共通理由コード
193	2006
194	1401 1402 1415 1416 1419 1501
195	1401 1402 1403 1404
198	1101 1102
199	ユーザー API の共通理由コード
200	ユーザー API の共通理由コード
201	2004
202	1201 1202
203	2004
204	2004
207	1101

表 205. 各理由コードに関連する機能 ID (続き)

理由コード	機能 ID
208	1501 1502 1503 1504 1505 1506 1507 2011
209	EKGWAIT
210	EKGWAIT
211	ユーザー API の共通理由コード
212	ユーザー API の共通理由コード
213	ユーザー API の共通理由コード
214	1405 1406 1407 1408 1409 1410 1415 1417 1418
215	1401 1402 1403 1404 1419
220	1405 1407
223	1419 1508
224	1509
225	1509
226	1101
227	1419 1509
228	1509
229	1509
230	1401 1402 1403 1404 1419
231	1401 1402 1403 1404 1419
232	1401 1402 1403 1404 1419
240	ユーザー API の共通理由コード
600	1508
601	1508
602	1419
603	1509
604	1409
605	1409
32768	1302 1303 1401 1402 1405 1407 1409 1410 1415 1419
32769	1302 1303 1401 1402 1405 1407 1409 1410 1419
32770	1302 1303 1401 1402 1405 1407 1409 1410 1419
32771	1401 1415 1419
32772	1302 1303 1401 1402 1405 1407 1409 1410 1419
45086	2006

機能 ID に対応する機能名のリスト

545 ページの表 206 には、機能 ID に対応する RODM API 機能名がリストされています。

表 206. 機能 ID に対応する機能名

機能 ID	機能名
1101	EKG_Connect
1102	EKG_Disconnect
1201	EKG_Checkpoint
1202	EKG_Stop
1302	EKG_CreateClass
1303	EKG_DeleteClass
1304	EKG_CreateField
1305	EKG_DeleteField
1306	EKG_CreateSubfield
1307	EKG_DeleteSubfield
1401	EKG_ChangeField
1402	EKG_SwapField
1403	EKG_ChangeSubfield
1404	EKG_SwapSubfield
1405	EKG_LinkTrigger
1406	EKG_LinkNoTrigger
1407	EKG_UnLinkTrigger
1408	EKG_UnLinkNoTrigger
1409	EKG_CreateObject
1410	EKG_DeleteObject
1411	EKG_RevertToInherited
1412	EKG_AddNotifySubscription
1413	EKG_DeleteNotifySubscription
1415	EKG_TriggerNamedMethod
1416	EKG_TriggerOIMethod
1417	EKG_AddObjDelSubs
1418	EKG_DelObjDelSubs
1419	EKG_ChangeMultipleFields
1501	EKG_QueryField
1502	EKG_QuerySubfield
1503	EKG_QueryEntityStructure
1504	EKG_QueryFieldStructure
1505	EKG_QueryFieldID
1506	EKG_QueryFieldName
1507	EKG_QueryNotifyQueue
1508	EKG_QueryMultipleSubfields
1509	EKG_Locate
1510	EKG_QueryResponseBlockOverflow
1600	EKG_ExecuteFunctionList
2001	EKG_QueryFunctionBlockContents

表 206. 機能 ID に対応する機能名 (続き)

機能 ID	機能名
2002	EKG_LockObjectList
2003	EKG_UnlockAll
2004	EKG_ResponseBlock
2005	EKG_SendNotification
2006	EKG_SetReturnCode
2007	EKG_WhereAmI
2008	EKG_OutputToLog
2009	EKG_MessageTriggeredAction
2011	EKG_QueryObjectName

NetView 提供のメソッドに関連する理由コードのリスト

表 207 には、NetView 提供のメソッドと、それらの各メソッドによって戻される理由コードがリストされています。

表 207. NetView 提供のメソッドに関連する理由コード

メソッド	理由コード
DUIFCATC	45070 45077 45078 45081 45088
DUIFCCAN	45077 45081 45088
DUIFCLRT	45070 45071 45077 45078 45081 45083 45088
DUIFCUAP	45065 45070 45071 45077 45081 45088
DUIFCUUS	45070 45077 45078 45081 45088
DUIFECDS	45070 45077 45078 45079 45081 45088
DUIFFAWS	45088
DUIFFIRS	45070 45077 45078 45081 45088
DUIFFRAS	45077 45081 45088
DUIFFSUS	45070 45077 45078 45081 45088
DUIFRFDS	45077 45081 45088
DUIFVCFT	45070 45077 45081 45088
EKGCTIM	32768 32771 32780
EKGMIMV	32768 32771 32780
EKGNEQL	32768 32769 32770 32772 32780 32954
EKGNLST	32768 32769 32770 32772 32780 32954
EKGNOTF	32768 32770 32780 32954
EKGNTHD	32768 32769 32770 32772 32780 32954
EKGSPPI	32780 32790 32791 32792 32793 32794 32795 32796 32797 32798 32809+

RODM パフォーマンスを最大にする方法

このセクションでは、RODM 実行時のシステム・パフォーマンスを最大にする方法について説明します。データ・モデルの構造とサイズ、メソッドの設計、およびユーザー・アプリケーションの設計は、すべてパフォーマンスに影響を与えます。

データ・モデルの構造とサイズ

機能によっては、オブジェクトと汎用クラス (ルート) の間のクラスの数が増えると、実行時間が長くなるものがあります。縦方向のクラス数は最小限に抑えてください。最良のパフォーマンスを得るためには、縦方向のクラス数が 100 を超えないようにしてください。

メソッドの設計

ユーザーが作成するメソッドでは、可能な場合にはメソッドを起動しない機能を使用してください。これにより、単一のトランザクションによって実行されるアクションの有効範囲が制限され、システム使用率が減少します。

ユーザー・アプリケーションの設計

あるフィールドに関して照会メソッドを起動する必要がなくて、ユーザーのデータ・モデルに多数の縦方向クラスが含まれている場合には、フィールド照会機能の代わりにサブフィールド照会機能を使用してパフォーマンスを向上させることができます。

RODM 通知処理では、通知申請ごとにリソースが使用されます。不要な通知申請を削除してください。

カスタマイズ・パラメーターとシステム・フィールド

最高のパフォーマンスを得るためには、ロギングが最小限に抑制されるように RODM ロギング・レベルを設定してください。 LOG_LEVEL 、 MLOG_LEVEL カスタマイズ・パラメーター、 EKG_User クラスの対応する EKG_LogLevel および EKG_MLogLevel フィールドの推奨値は 8 です。

注: 8 未満の値を指定すると、GMFHS がメソッド・エラーを報告することがあります。

索引付きフィールド

EKG_Locate 機能を使用すると、アプリケーションがオブジェクト ID のリストを取り出しやすくなります。フィールド照会機能を使用してデータ・モデル全体をスキャンする代わりに、 EKG_Locate 機能を使用して、該当のオブジェクト ID を含むテーブルだけをスキャンしてください。

パフォーマンスを向上させるためには、データ・モデルを移植する前に索引付きフィールドを作成する必要があります。オブジェクト内のそれぞれの索引付きフィールド値の最初の 254 バイトを固有にすることによっても、パフォーマンスが向上します。

NetView 提供のメソッド

このセクションでは、NetView 提供のメソッドについて簡単に紹介します。これらのメソッドは、特定の種類の機能を提供するために RODM とともに配布されているものです。 NetView 提供のメソッドに代えて、ローカルで開発されたメソッドを追加することができます。 NetView 提供のメソッドは、メソッド API を使用します。このセクションでは、これらのメソッドについて機能別に説明します。メソッドに渡されるすべてのパラメーターは、SelfDefining データ・ストリングとして指定されます。

RODM の通知メソッド

すべての RODM 通知メソッドは、データが変更されたことを必要な申請者に通知する以外に、 value、prev_val、および timestamp サブフィールドの値を戻します。これらのデータは、通知キュー・ブロックの User_area に入れて申請者に戻されます。このブロックについては、481 ページの『EKG_QueryNotifyQueue − 通知キューを照会する』を参照してください。 User_area にすべてのデータが入りきらない場合には、ヌルのデータ・ストリングが戻されます。データは次の順序で戻されます。

1. value サブフィールド内の値
2. prev_val サブフィールド内の値
3. timestamp サブフィールドの値

戻されるデータのデータ・タイプは SelfDefining です。各出力値の前には (1、2、3、およびそれ以上の番号に対応する) タグ・コードが付けられ、そのデータがどのサブフィールドから得られたのかが区別されます。特定のサブフィールドが定義されていない場合、対応するタグ・コードは SelfDefining データ・ストリングに組み込まれません。 表 208 は、データ・ストリングに戻されるデータの例です。

表 208. EKGNOTF 通知メソッドで戻される User_data の例

オフセット	長さ	値	説明
000	2	34	SelfDefining ストリングの全長
002	2	21	Smallint データ・タイプ・コード
004	2	01	Value フィールド・インディケーター
006	2	10	Value データ・タイプ・フラグ (Integer)
008	4	*value*	Value データ (Integer)
012	2	21	Smallint データ・タイプ・コード
014	2	02	Prev_val フィールド・インディケーター
016	2	10	Prev_val データ・タイプ・フラグ (Integer)
018	4	*prev_val*	Prev_val データ (Integer)
022	2	21	Smallint データ・タイプ・コード
024	2	03	Timestamp フィールド・インディケーター
026	2	27	Timestamp データ・タイプ・フラグ (TimeStamp)
028	8	*timestamp*	Timestamp データ (TimeStamp)

NetView 提供の通知メソッドは、フィールドのデータ値が変更されて、古い値とは異なる新しい値になった場合にのみ申請者に通知を出します。さらに、それぞれのメソッドには、次のような通知の実行方法を示すパラメーターを渡す必要があります。

Always
メソッドが実行されるたびに、呼び出しパラメーターを通じてそのメソッドに示された申請者に通知が送られます。

Once 単一の通知が生成され、メソッドがフィールドの通知リストからそれ自体を削除します。

オブジェクトのフィールドに通知メソッドがインストールされてから、そのオブジェクト・フィールドに対して変更が行われると、そのフィールドに割り当てられた通知申請が実行されます。オブジェクトの通知が処理された後で、1 次親内の同じフィールドに割り当てられた通知申請が実行されます。

生成された通知を割り当てできるのは次のデータ・タイプのフィールドのみであるかどうかを判別するために、比較演算を実行するメソッド。

- Smallint
- Integer
- Float
- TimeStamp
- CharVar

EKGNOTF: 一般通知

機能 関連するフィールド値に対して行われた変更を申請者に通知します。

長命パラメーター
実行オプションとして Always (常に) または Once (一度のみ) を指定する 2 バイトの整数コードです。

表 209. EKGNOTF の長命パラメーターの説明

オフセット	長さ	値	説明
000	2	8	SelfDefining ストリングの全長
002	2	21	Smallint データ・タイプ・コード
004	2	1 または 2	2 バイトの整数 (1=常に、2=一度のみ)
006	2	21	Smallint データ・タイプ・コード
008	2	1 または 2	2 バイトの整数 (1=データ値が前の値と異なる場合にだけ通知、2=常に通知)

短命パラメーター
必要ありません。

EKGNEQL: 等しい場合に通知

機能 関連するフィールド値に対して行われた変更によって、そのフィールドが長命パラメーターと同じになったときに、申請者に通知します。適切な比較を行う方法を判別するために、この機能はサポートされるすべての RODM データ・タイプを認識できなければなりません。

長命パラメーター

always (常に) または once (一度のみ) のいずれかの実行オプションを指定する 2 バイト整数コードの後に、申請されたフィールドに対してテストされる値が続いたものです。この長命パラメーターは、現行オブジェクト内のテスト値が指定されているフィールドの Field_ID を指定します。

表 210. EKGNEQL の長命パラメーターの説明

オフセット	長さ	値	説明
000	2	14	SelfDefining ストリングの全長
002	2	21	Smallint データ・タイプ・コード
004	2	1 または 2	2 バイトの整数 (1=常に、2=一度のみ)
006	2	21	Smallint データ・タイプ・コード
008	2	1 または 2	2 バイトの整数 (1=データ値が前の値と異なる場合にだけ通知、2=常に通知)
010	2	26	Smallint データ・タイプ・コード (FieldID)
012	4	Field_ID	テスト値の Field_ID

短命パラメーター

必要ありません。

EKGNLST: リストに等しい場合に通知

機能 関連するフィールド値に対して行われた変更によって、そのフィールドが長命パラメーターの値のうちの 1 つと同じになったときに、申請者に通知します。適切な比較を行う方法を判別するために、この機能はサポートされるすべての RODM データ・タイプを認識できなければなりません。

長命パラメーター

always (常に) または once (一度のみ) のいずれかの実行オプションを指定する 2 バイト整数コードの後に、申請されたフィールドに対してテストされる値の数とそれらの値のリストが続いたものです。この長命パラメーターは、現行オブジェクト内の比較リスト・カウントが指定されているフィールドの Field_ID、およびテスト値が指定されているフィールドの Field_ID のリストを指定します。

表 211. EKGNLST の長命パラメーターの説明

オフセット	長さ	値	説明
000	2	14+(N*6)	SelfDefining ストリングの全長
002	2	21	Smallint データ・タイプ・コード
004	2	1 または 2	2 バイトの整数 (1=常に、2=一度のみ)
006	2	21	Smallint データ・タイプ・コード
008	2	1 または 2	2 バイトの整数 (1=データ値が前の値と異なる場合にだけ通知、2=常に通知)
010	2	10	Smallint データ・タイプ・コード (Integer)
012	4	N (範囲は 0..n)	後に続く Field_ID の数
016	2	26	Smallint データ・タイプ・コード (FieldID)

表 211. EKGNLST の長命パラメーターの説明 (続き)

オフセット	長さ	値	説明
018	4	Field_ID	最初のテスト値の Field_ID **注:** 配列のエレメント
010+(N*6)	2	26	Smallint データ・タイプ・コード (FieldID)
012+(N*6)	4	Field_ID	N 番目のテスト値の Field_ID

短命パラメーター

必要ありません。

EKGNTHD: しきい値を超えたときに通知

機能 関連するフィールド値に対して行われた変更によって、そのフィールドが長命パラメーターで指定されたしきい値を超えたときに、申請者に通知します。このメソッドには、以下の 3 つのオプションが用意されています。

- ユーザーが上限を指定します。関連するフィールドの値がパラメーターよりも大きい場合には、申請者に通知が行われます。
- ユーザーが下限を指定します。関連するフィールドの値がパラメーターよりも小さい場合には、申請者に通知が行われます。
- ユーザーがパラメーター値のペアを指定します。関連するフィールドの値が上限パラメーターよりも大きい場合、または下限パラメーターよりも小さい場合、申請者に通知が行われます。

長命パラメーター

always (常に) または once (一度のみ) のいずれかの実行オプションを指定する 2 バイト整数コードの後に、実行される特定機能およびしきい値が続いたものです。この長命パラメーターは、現行オブジェクト内の機能コードが指定されているフィールドの Field_ID、およびしきい値を指定する必要のある Field_ID を指定します。

表 212. EKGNTHD の長命パラメーターの説明

オフセット	長さ	値	説明
000	2	20 または 26	SelfDefining ストリングの全長
002	2	21	Smallint データ・タイプ・コード
004	2	1 または 2	2 バイトの整数 (1=常に、2=一度のみ)
006	2	21	Smallint データ・タイプ・コード
008	2	1 または 2	2 バイトの整数 (1=データ値が前の値と異なる場合にだけ通知、2=常に通知)
010	2	10	Integer データ・タイプ・コード
012	4	1、2、または 3	オプション・コード (1=上限、2=下限、3=範囲)
016	2	26	Smallint データ・タイプ・コード (FieldID)
018	4	Field_ID	1 の場合には上限、2 または 3 の場合には下限
注: 以下のパラメーターはオプション・コード 3 の場合のみ			
022	2	26	Smallint データ・タイプ・コード (FieldID)
024	4	Field_ID	3 の場合には上限

短命パラメーター
必要ありません。

RODM の変更メソッド

EKGCTIM: オブジェクト独立メソッドの起動

機能 指定された機能を呼び出し側のメソッドの実行と非同期的に実行するには、メッセージ機能を使用してオブジェクト独立メソッドを起動してください。実状況の送信側と通信するためにオブジェクト独立メソッドを起動したい場合などには、このメソッドと関連するフィールドから得られた古い値と新しい値の情報を通知するために、 SWAP 機能ブロックを渡すことができます。これにより、オブジェクト独立メソッドが、実デバイス状況を古い状態から新しい状態に変更するように実状況の送信側に指示するようになることがあります。

長命パラメーター
オブジェクト独立メソッドに渡される必須機能ブロックを作成するためのデータが提供されているフィールドの Field_ID のリストです。このパラメーター・ストリングのそれぞれの連続 4 バイトは、現行オブジェクト内のフィールドの FieldID として解釈されます。指定されたフィールドが照会され、その情報が EKG_TriggerOIMethod 機能の機能ブロックに入ります。

表 213. EKGCTIM の長命パラメーターの説明

オフセット	長さ	値	説明
000	2	12	SelfDefining ストリングの全長
002	2	26	Smallint データ・タイプ・コード (FieldID)
004	4	Field_ID	メソッド名が入っているフィールド
008	2	26	Smallint データ・タイプ・コード (FieldID)
010	4	Field_ID	SelfDefining ストリングとして短命パラメーターのリストを含むフィールド

短命パラメーター
必要ありません。

RODM の名前付きメソッド

EKGMIMV: 増分値

機能 現行オブジェクトで定義された、指定されたフィールドの値を、指定された値ずつ増やします。

長命パラメーター
2 つの Field_ID が必要です。ストリングの最初の 4 バイトには、値が増やされるフィールドの Field_ID を指定します。 2 つ目の 4 バイトには、増分値が入っているフィールドの Field_ID を指定します。これらのフィールドは整数データ・タイプでなければなりません。増分値を負の数にして、指定されたフィールド値を減らすこともできます。

表 214. EKGMIMV の長命パラメーターの説明

オフセット	長さ	値	説明
000	2	12	SelfDefining ストリングの全長
002	2	26	Smallint データ・タイプ・コード (FieldID)
004	4	Field_ID	値が増やされるフィールド
008	2	26	Smallint データ・タイプ・コード (FieldID)
010	4	Field_ID	増分値が入っているフィールド

短命パラメーター

必要ありません。

EKGCTIM: オブジェクト独立メソッドの起動

この機能に関して説明された変更メソッドと同じ働きをします。このメソッドは、RODM にインストールした後では、どちらの方法でも使用できます。

RODM のオブジェクト独立メソッド

EKGSPPI: NetView へのコマンドの送信

EKGSPPI メソッドは RODM 自動化プラットフォームのサービスの 1 つです。NetView を使用した自動化タスクの詳細については、 217 ページの『第 8 章 RODM 自動化プラットフォームを使用する』を参照してください。 EKGSPPI メソッドと自動化プラットフォームを使用した RODM の詳細な自動化シナリオが、「*IBM Tivoli NetView for z/OS 自動操作ガイド*」の章に記載されています。

機能: オブジェクト独立メソッドは、NetView の DSIQTSK タスクにコマンドを送信します。 DSIQTSK がそれらのコマンドを自動タスクにディスパッチし、自動タスクがコマンドを出します。 NetView では、 EKGSPPI を呼び出すメソッドの例として、 EKGCPPI という変更メソッドと、EKGOPPI というオブジェクト独立メソッドの、2 つのメソッド例が提供されています。これらのメソッドの例をモデルとして使用し、EKGSPPI を起動するユーザー独自のメソッドを作成することができます。

EKGSPPI メソッドを起動するには、 EKG_MessageTriggeredAction 機能を使用する方法が最適です。このようにすると、EKGSPPI が非同期的に実行できるようになります。 EKG_MessageTriggeredAction 機能は、EKGSPPI に渡されるパラメーターを含んでいる EKG_TriggerOIMethod 機能を指定します。

長命パラメーター: 必要ありません。

短命パラメーター: EKGSPPI は、SelfDefining データ・タイプの短命パラメーターを受け入れます。この短命パラメーターには、7 つのデータ項目が含まれています。各データ項目のデータ・タイプは CharVar または AnonymousVar です。 7 つのデータ項目は、すべて示されたとおりの順序で指定しなければなりませんが、そのうちのいくつかはヌル値にすることができます。 EKGSPPI メソッドは、各データ項目に指定された値から先行ブランク文字を削除します。

データ項目に使用される値は、サンプル・メソッド EKGCPPI および EKGOPPI で使用されている変数です。 この短命パラメーターの中のデータ項目は次の 7 つです。

RCVRID_CHARVAR
このデータ項目は、EKGSPPI が送ったコマンドのコマンド受信側の名前を表します。これは、DSIQTSK タスクのための DSIQTSKI 初期設定ファイルで定義された CMDRCVR の ID フィールドで提供された名前です。 EKGSPPI メソッドはこの名前を大文字に変換します。この名前の最大長は 8 文字です。

ASSIST_CHARVAR
このデータ項目は、コマンドの実行前にそのコマンドを NetView コンソールに送るのかどうかを表します。コマンドはメッセージ (DWO670I) 形式で出されます。 DWO670I の自動化テーブル・トラップで SAVECMD が指定されている場合には、そのコマンドは SAVECMD の送信先オペレーター用に保存することができます。オペレーターは、ASSISCMD を使用してそのコマンドをパネルに表示させることができます。 オペレーターは、NetView 援助パネルからコマンドを発行、変更、または取り消すことができます。有効な値は、以下のとおりです。

値　　意味

ASSIST
オペレーターにコマンドを送信します。

NOASSIST
コマンドをオペレーターに送信せずに発行します。

ヌルまたはブランク
コマンドをオペレーターに送信せずに発行します。

この値の最大長は 8 文字です。

TASKINFO_CHARVAR
このデータ項目は、そのコマンドが特定の NetView 自動タスクによって実行されるのかどうかを表します。有効な値は、以下のとおりです。

値　　意味

ANY DSIQTSK は、DSIQTSK に対して定義されている (最後に使用された自動タスクの) 次の自動タスクにコマンドを送ります。自動タスクは、DSIPARM の DSIQTASKI メンバーで定義されたとおりの順序で使用されます。

ONLY DSIQTSK は、短命パラメーターのデータ項目 TASKNAME_CHARVAR で指定された自動タスクにこのコマンドを送ります。指定された自動タスクが利用不能である場合には、コマンドは発行されません。

ONLYANY
DSIQTSK は、短命パラメーターのデータ項目 TASKNAME_CHARVAR で指定された自動タスクにこのコマンドを送ります。指定された自動タスクが利用不能である場合、DSIQTSK は、DSIQTSK に対して定義されている (最後に使用された自動タスクの) 次の自動タスクにコマンドを送ります。自動タスクは、DSIPARM の DSIQTASKI メンバーで定義されたとおりの順序で使用されます。

ヌルまたはブランク
DSIQTSK は、DSIQTSK に対して定義されている (最後に使用された自動タスクの) 次の自動タスクにコマンドを送ります。自動タスクは、DSIPARM の DSIQTASKI メンバーで定義されたとおりの順序で使用されます。

この値の最大長は 8 文字です。

TASKNAME_CHARVAR
このデータ項目は、DSIQTSK がコマンドを送った送信先自動タスクの名前を表します。これは、DSIQTSK タスクの初期メンバーである DSIQTSKI の TASK ステートメントで指定された名前です。 TASKINFO_CHARVAR が ONLY または ONLYANY の場合には、 TASKNAME_CHARVAR を指定する必要があります。 EKGSPPI メソッドはこの名前を大文字に変換します。この値の最大長は 8 文字です。

SENDER_CHARVAR
このデータ項目は、ASSIST_CHARVAR として ASSIST が指定されたコマンドの送信側を表します。この名前は、オペレーターに送られるメッセージに含まれています。 EKGSPPI メソッドはこの名前を大文字に変換します。この値の最大長は 8 文字です。

CMD_CHARVAR
このデータ項目は、発行されるコマンドを表しています。
COMMAND_CHARVAR 値は必須です。この値の最大長は 240 文字です。

CMD_DESC_CHARVAR
このデータ項目は、発行されるコマンドの説明を表しています。この値には、ブランクまたはヌルを指定することができます。この値の最大長は 780 文字です。短命パラメーターの ASSIST_CHARVAR データ項目で ASSIST が指定されている場合には、この説明は援助パネルに表示されます。

出力: このコマンドは、NetView の DSIQTSK タスクに送られます。

RODM ロード機能を使用して EKGSPPI メソッドを実行することができます。556 ページの図 92 は、 RODM ロード機能プリミティブ・ステートメントを使用して EKGSPPI メソッドを呼び出す例を示しています。

注: RODM ロード機能は、APF (許可プログラム機能) から許可を与えられているプログラムではありません。 DSIQTSK によって管理される NetView プログラム間インターフェース・コマンドの受信側が APF 許可を必要とする場合、ジョブは失敗し、EKGSPPI メソッドによって戻りコード 8 と理由コード 32832 が戻されます。

```
OP EKGSPPI INVOKED_WITH       -- Trigger the EKGSPPI method --
 (SELFDEFINING)
 ( (CHARVAR) 'CNM01'         -- Command receiver name --
   (CHARVAR) 'NOASSIST'      -- Issue without operator intervention --
   (CHARVAR) 'ONLYANY'       -- Use named autotask if available --
   (CHARVAR) 'AUTO1'         -- Autotask name --
   (CHARVAR) 'LOAD FUN'      -- Name of sender of command --
   (CHARVAR) 'some reasonable command goes here'
                             -- Command to be sent --
   (CHARVAR) 'This command is sent using the RODM load function.'
             ' It is an example of triggering the EKGSPPI method '
             ' using a RODM load function primitive statement.'
                             -- Command description --              );
```

図 92. EKGSPPI を呼び出すための RODM ロード機能プリミティブ・ステートメントの例

GMFHS メソッド

このセクションで説明されているメソッドは、GMFHS で使用するために提供されているものです。これらのメソッドは、ユーザーが作成した自動化コードとともに使用することもできます。

これらの GMFHS メソッドは、説明されている用途でのみ使用するようにしてください。例えば、名前付きメソッドをオブジェクト独立メソッドとしては使用しないでください。

このセクションで説明されている GMFHS メソッドのほかに、 GMFHS はユーザーのプログラムで使用できない別のメソッドも使用します。以下のリストで示されているメソッドは、ユーザー作成プログラムでは使用しないでください。

- DUIFCAAP
- DUIFCADT
- DUIFCAPC
- DUIFCASB
- DUIFCATC
- DUIFCCAP
- DUIFCDTC
- DUIFCDUC
- DUIFCGRA
- DUIFCGRT
- DUIFCGR2
- DUIFCGR3
- DUIFCLSR
- DUIFCLS2
- DUIFCLS3
- DUIFCMUU
- DUIFCRDC
- DUIFCRTP
- DUIFCRTU
- DUIFCRUC
- DUIFCSRT
- DUIFCURA

- DUIFCUTC
- DUIFEGSN
- DUIFITKN
- DUIFRAIP
- DUIFRRTC
- DUIFVCVT
- DUIFVDRT
- DUIFVEFC
- DUIFVEVF
- DUIFVEXV
- DUIFVFPV
- DUIFVGET
- DUIFVIEW
- DUIFVLST
- DUIFVLTT
- DUIFVMDR
- DUIFVNGI
- DUIFVNGN
- DUIFVNOI
- DUIFVNOT
- DUIFVPFR
- DUIFVSUB
- DUIFVTKN
- DUIFVUNS
- DUIFVUPD
- DUIFVVLC

DUIFCCAN: すべての注記の消去

このオブジェクト独立メソッドは、RODM におけるすべての実オブジェクトおよび集合オブジェクトに関するすべての UserStatus フラグで、注釈フィールドを消去するために、任意のアプリケーションによって実行することができます。

機能: DUIFCCAN メソッドを使用すると、それぞれの実オブジェクトおよび集合オブジェクトのトポロジー・コンソールを経由しないで、すべての注釈フィールドを消去することができます。 ″DUIFCCAN″ というオペレーター ID は、その注釈が、オペレーターによってではなく、このメソッドによって消去されたことを示すために設定されます。

入力: このメソッドは入力パラメーターを必要とせず、以下の RODM ロード機能プリミティブ・ステートメントによって起動できます。

```
OP DUIFCCAN INVOKED WITH;
```

出力: このメソッドは、エラーが検出されると戻りコードと理由コードを設定し、RODM ログにタイプ 1 のレコードを書き込みます。 546 ページの表 207 には、このメソッドで戻される可能性のある理由コードのリストが示されています。

DUIFCLRT: リソース・タイプ・リンク・メソッド

このメソッドは、以下のリンクまたはリンク解除を行うために実行されるオブジェクト独立メソッドです。

- 実オブジェクト、集合オブジェクト、またはシャドー・オブジェクトの DisplayResourceType フィールドと、 Display_Resource_Type_Class のオブジェクトのリソース・フィールド
- View_Information_Reference_Object の DisplayResourceType フィールドと、Display_Resource_Type_Class のオブジェクトのリソース・フィールド

機能: DUIFCLRT メソッドは、実リソースまたは集合リソースの DisplayResourceType フィールドが変更されたときに、影響を受ける集合リソースの DisplayStatus フィールド値が再計算されるようにするために使用します。これらの変更は、以下の場合に行われます。

- GMFHS_Managed_Real_Objects_Class オブジェクトの DisplayResourceType 値が変更されると、そのオブジェクトの DefaultAggregationPriorityCopy 値の変更が必要になることがあります。この変更によってその実リソースの実質的な集約優先順位が影響を受ける場合、変更の影響を受けた集合リソースを更新して、DisplayStatus 値を再計算する必要があります。この変更を行うために、DUIFCLRT メソッドは DUIFCAPC メソッドを起動します。
- GMFHS_Aggregate_Objects_Class のオブジェクト内で DisplayResourceType リンクが変更されると、GMFHS はその集合体の DisplayStatus フィールドの値を再計算します。

DUIFCLRT メソッドは、(EKGLISLM および EKGLIILM 初期化メソッドを含む) 他のメソッドでは起動できません。 DUIFCUAP メソッドは、別のメソッドで起動しないでください。

図 93 は、RODM ロード機能プリミティブ・ステートメントを使用して DUIFCLRT メソッドを起動する例を示しています。

```
OP DUIFCLRT INVOKED_WITH (SELFDEFINING)
 (
  (SMALLINT) 1
  (CHARVAR) '
  (CHARVAR) 'Display_Resource_Type_Class.DUIXC_RTN_NN_DOMAIN_AGG'
  (OBJECTID) 'View_Information_Reference_Class'.
   '1.3.18.0.0.2150_Reference'
 );
```

図 93. DUIFCLRT を呼び出す RODM ロード機能プリミティブ・ステートメント

入力: DUIFCLRT メソッドへの入力パラメーターは、SELFDEFINING データ・タイプの 4 つの項目のうちの ***3 つ*** を使用して指定してください。 表 215 には、これらの項目の要約が示されています。表の後で、各項目について詳しく説明されています。

表 215. DUIFCLRT 操作の入力値

項目	説明	データ・タイプ	必須 / 任意指定
1	リンクまたはリンク解除	CHARVAR または SMALLINT	必須
2	リソース・オブジェクト	CHARVAR または OBJECTID	任意指定 [1]
3	表示リソース・タイプ	CHARVAR または OBJECTID	必須

表 215. DUIFCLRT 操作の入力値 (続き)

項目	説明	データ・タイプ	必須 / 任意指定
4	ビュー情報参照オブジェクト	CHARVAR または OBJECTID	任意指定 [1]

注: [1] リソース・オブジェクトまたはビュー情報参照オブジェクトのいずれかを必ず指定しなければなりませんが、両方を指定することはできません。

1 最初の項目は操作を表すもので、 CHARVAR データ・タイプでも SMALLINT データ・タイプでも指定できます。有効な値は、以下のとおりです。

表 216. DUIFCLRT 操作の入力値

操作	CHARVAR	SMALLINT
リソースのリンク	LINK	1
リソースのリンク解除	UNLINK	2

2 2 番目の項目はリンクまたはリンク解除される実オブジェクト、集合オブジェクト、またはシャドー・オブジェクトを表すもので、CHARVAR データ・タイプでも OBJECTID データ・タイプでも指定できます。この項目は任意指定ですが、指定しなかった場合には 4 番目の項目を指定しなければなりません。この項目を指定しない場合には、ヌル文字を指定する必要があります。例えば、次のようにコーディングしてください。

```
(CHARVAR) '
```

CHARVAR 項目の場合には、クラス名とオブジェクト名をピリオドで区切って指定してください。 OBJECTID 項目の場合には、単一引用符で囲んだクラス名と単一引用符で囲んだオブジェクト名を、ピリオドで区切って指定してください。例えば、次のようにコーディングしてください。

```
(CHARVAR) 'Display_Resource_Type_Class.DUIXC_RTN_NN_DOMAIN_AGG'
(OBJECTID) 'Display_Resource_Type_Class'.'DUIXC_RTN_NN_DOMAIN_AGG'
```

CHARVAR データ項目で使用されているクラス名またはオブジェクト名にピリオドが含まれている場合には、その名前を 2 つの単一引用符で囲んでください。例えば、クラス名が `Class.name` である場合、次のようにコーディングしてください。

```
(CHARVAR)'Class.name'.Object'
```

CHARVAR または OBJECTID データ項目で使用されているクラス名またはオブジェクト名に単一引用符 (') が含まれている場合には、2 つの単一引用符を使用してその単一引用符を指定してください。例えば、あるオブジェクトの名前が `Greg'sObject` である場合には、次のようにコーディングしてください。

```
(CHARVAR)'Class.Greg'sObject'
```

3 3 番目の項目は、リンクまたはリンク解除される Display_Resource_Type_Class のオブジェクトを表します。この項目は必須です。3 番目の項目の形式は、2 番目の項目の形式と同じです。

4 4 番目の項目は、リンクまたはリンク解除される View_Information_Reference _Object を表します。この項目は任意指定ですが、指

定しなかった場合には 2 番目の項目を指定しなければなりません。この項目を指定しない場合には、ヌル文字を指定する必要があります。例えば、次のようにコーディングしてください。

```
(CHARVAR) '
```

4 番目の項目の形式は、2 番目の項目の形式と同じです。

出力: リンクまたはリンク解除が行われます。

このメソッドは、エラーが検出されると戻りコードと理由コードを設定し、RODM ログにタイプ 1 のレコードを書き込みます。 546 ページの表 207 には、このメソッドで戻される可能性のある理由コードのリストが示されています。

DUIFCUAP: 集約パス更新メソッド

これは、GMFHS_Aggregate_Objects_Class のオブジェクト内の AggregationChild フィールドおよび別の GMFHS_Aggregate_Objects_Class オブジェクトまたは GMFHS_Managed_Real_Objects_Class オブジェクトの AggregationParent フィールドを使用して 2 つのリソース・オブジェクトがリンクまたはリンク解除されたときに実行される、オブジェクト独立メソッドです。

機能: このメソッドは、リンクまたはリンク解除の及ばない集合リソースおよびその集合祖先に属している ″Value.″ (count) フィールドおよび DisplayStatus フィールド値が、実リソースの集約子孫の追加 (リンクの場合) または削除 (リンク解除の場合) を反映して更新されるようにするために使用します。

またこのメソッドは、集約階層でループが発生しないようにするためにも使用してください。集約階層のループは、集合オブジェクトの AggregationParent フィールドに、同じオブジェクトの AggregationChild フィールドに対するリンク、または直接あるいは他の集合オブジェクトを介して最初の集合オブジェクトの AggregationChild フィールドにリンクされる AggregationParent フィールドを含んでいるオブジェクトに対するリンクが含まれているときに発生します。

GMFHS の操作中は、集約階層での集合リソースの追加または削除を行うためには DUIFCUAP メソッドだけを使用してください。この要件は RODM によって強制されていませんので、ご注意ください。

GMFHS は RODM ロード機能を使用しなければ集約階層を変更できないため、RODM ロード機能を介して間接的にしか DUIFCUAP メソッドを使用しません。

DUIFCUAP メソッドは、(EKGLISLM および EKGLIILM 初期化メソッドを含む) 他のメソッドでは起動できません。 DUIFCUAP メソッドは、別のメソッドで起動しないでください。 図 94 は、RODM ロード機能プリミティブ・ステートメントを使用して DUIFCUAP メソッドを起動する例を示しています。

```
OP DUIFCUAP INVOKED_WITH (SELFDEFINING)
   ((CHARVAR)'LINK'
   (CHARVAR)'GMFHS_Aggregate_Objects_Class.ETHERNET'
   (CHARVAR)'GMFHS_Aggregate_Objects_Class.WESTCTR');
```

図 94. DUIFCUAP を呼び出す RODM ロード機能プリミティブ・ステートメント

入力: DUIFCUAP メソッドへの入力パラメーターは、SELFDEFINING データ・タイプの 3 つの項目を使用して指定してください。

- 最初の項目は操作を表すもので、 CHARVAR データ・タイプでも SMALLINT データ・タイプでも指定できます。有効な値は、以下のとおりです。

表 217. DUIFCUAP 操作の入力値

操作	CHARVAR	SMALLINT
リソースのリンク	LINK	1
リソースのリンク解除	UNLINK	2

- 2 番目の項目は、集約階層内のより下層にある、リンクまたはリンク解除の対象となる実オブジェクトまたは集合オブジェクトを表します。このデータ項目は、CHARVAR データ・タイプでも OBJECTID データ・タイプでも指定できます。CHARVAR 項目の場合には、クラス名とオブジェクト名をピリオドで区切って指定してください。 OBJECTID 項目の場合には、単一引用符で囲んだクラス名と単一引用符で囲んだオブジェクト名を、ピリオドで区切って指定してください。例えば、次のような場合:

  ```
  (CHARVAR)'GMFHS_Aggregate_Objects_Class.ETHERNET'
  (OBJECTID)'GMFHS_Aggregate_Objects_Class'.'ETHERNET'
  ```

  CHARVAR データ項目で使用されているクラス名またはオブジェクト名にピリオドが含まれている場合には、その名前を 2 つの単一引用符で囲んでください。例えば、クラス名が `Class.name` である場合、次のようにコーディングしてください。

  ```
  (CHARVAR)'Class.name'.Object'
  ```

  CHARVAR または OBJECTID データ項目で使用されているクラス名またはオブジェクト名に単一引用符 (') が含まれている場合には、2 つの単一引用符を使用してその単一引用符を指定してください。例えば、あるオブジェクトの名前が `Greg'sObject` である場合には、次のようにコーディングしてください。

  ```
  (CHARVAR)'Class.Greg'sObject'
  ```

- 3 番目の項目は、集約階層内のより下層にある、リンクまたはリンク解除の対象となる集合オブジェクトを表します。 3 番目の項目の形式は、2 番目の項目の形式と同じです。

出力: リンクまたはリンク解除が行われます。

このメソッドは、エラーが検出されると戻りコードと理由コードを設定し、RODM ログにタイプ 1 のレコードを書き込みます。 546 ページの表 207 には、このメソッドで戻される可能性のある理由コードのリストが示されています。

DUIFCUUS: ユーザー状況更新メソッド

これは、GMFHS のための初期 RODM 構造ロード中に、GMFHS_Displayable_Objects_Parent クラスに属するすべてのオブジェクトの UpdateUserStatus フィールドにインストールされた、名前付きメソッドです。GMFHS_Monitorable_Objects_Class がこのメソッドを継承します。

機能: このメソッドは、GMFHS_Managed_Real_Objects_Class、GMFHS_Aggregate_Objects_Class、および GMFHS_Shadow_Objects_Class を含む、GMFHS_Displayable_Objects_Parent_Class クラスの下層クラスの UserStatus フィールド値を変更する必要があるアプリケーションで使用してください。

入力: DUIFCUUS_Update_User_Status メソッドで必要な入力は、以下のとおりです。

- UserStatus の変更すべきビットを指定する 4 バイト・マスク
- 新しい値を含む 4 バイトの UserStatus
- UserStatus フィールドを変更するオペレーター ID、メソッド名、またはプロダクトを含む 8 バイトの文字フィールド
- 予約フィールド用の 20 バイト・ブロック

ビット値の説明も含め、UserStatus フィールドについては、「*IBM Tivoli NetView for zOS データ・モデル・リファレンス*」を参照してください。

以下の例では、UserStatus ビットの設定方法を説明します。これらのビットは、異なる値を見やすくするために、複数行に分割されました。

必須ビットは、以下のとおりです。

- 最初の 16 バイトには、マスク、UserStatus、およびオペレーター ID を指定します。
- 次の 20 バイトは予約済みです。

次の RODM ロード機能プリミティブ・ステートメントの例では、OPER1 が WESTCTR オブジェクトのマーク・ビットを設定しています。

```
OP 'GMFHS_Aggregate_Objects_Class'.'WESTCTR'.'UpdateUserStatus'
    INVOKED_WITH (SELFDEFINING)
      ((ANONYMOUSVAR)X'8000000080000000D6D7C5D9F1404040'
                      '00000000000000000000000000000000000000000000');
```

次の RODM ロード機能プリミティブ・ステートメントの例では、OPER1 が WESTCTR オブジェクトのマーク・ビットを消去しています。

```
OP 'GMFHS_Aggregate_Objects_Class'.'WESTCTR'.'UpdateUserStatus'
    INVOKED_WITH (SELFDEFINING)
      ((ANONYMOUSVAR)X'8000000000000000D6D7C5D9F1404040'
                      '00000000000000000000000000000000000000000000');
```

注:

1. DUIFCUUS に対する入力として送信できる最小バイト数は 36 です。マスク、UserStatus、およびオペレーター ID を必要に応じて設定し、残りの 20 バイトはゼロにしてください。
2. オペレーター ID を指定するときには、以下のことに注意してください。
   - オペレーター ID は 8 バイトでなければなりません
   - オペレーター ID はすべてブランクでもかまいません

DUIFCUUS メソッドは、変更可能なビットを、変更対象のオブジェクトのクラスに応じて制限します。

- マーク・ビット (0x80000000) は、どのオブジェクトの場合にも変更できます。

- 中断ビット (0x20000000) および自動消去中断ビット (0x60000000) は、GMFHS_Real_Objects_Class の子であるクラスのオブジェクトの場合にだけ変更できます。

  注: 集合オブジェクトの中断ビットを設定すると、実オブジェクトの中断を簡単に行うことができます。集合オブジェクト自体は中断されません。その代わりに、その集合オブジェクトに関して子中断ビット (0x00800000) が設定され、その集合体の子であるすべての実オブジェクトが、その中断ビットを継承します。中断ビットに加えて、自動再開ビットも設定することができます。このビットも、子である実オブジェクトによって継承されます。
- 子中断ビット (0x00800000) は、集合体単位で消去することができます。その集合体の子であるすべての実オブジェクトの中断ビットおよび自動再開ビットも消去されます。
- 集合体しきい値矛盾ビット (0x08000000) は、GMFHS_Aggregate_Objects_Class クラスのオブジェクトの場合にだけ変更できます。
- 自動化の進行中ビット (0x04000000) は、どのオブジェクトでも変更できます。
- 未モニター・ビットは、GMFHS_Real_Objects_Class の子であるクラスのオブジェクトの場合にだけ変更できます。

出力: このメソッドが EKG_TriggerNamedMethod 機能を使用して起動される場合には、出力の応答ブロックを用意してください。この応答ブロックには、少なくとも 22 バイトの長さが必要です。応答ブロックの Concat_of_strings フィールドは、次の形式の SelfDefining ストリングです。

表 218. DUIFCUUS メソッドの出力

オフセット	長さ	値	説明
000	2	12	SelfDefining ストリングの全長
002	2	30	データ・タイプ AnonymousVar
004	2	8	AnonymousVar データの長さ
006	8		更新後の UserStatus フィールドの timestamp サブフィールドの値

このメソッドは、エラーが検出されると戻りコードと理由コードを設定し、RODM ログにタイプ 1 のレコードを書き込みます。 546 ページの表 207 には、このメソッドで戻される可能性のある理由コードのリストが示されています。

DUIFECDS: 表示状況変更メソッド

このメソッドは、GMFHS_Managed_Real_Objects_Class で定義されたオブジェクトすべての ChangeDisplayStatus フィールドにインストールされている名前つきメソッドです。

機能: このメソッドは、GMFHS_Managed_Real_Objects_Class のオブジェクトの DisplayStatus フィールドを変更し、その変更の効果を呼び出し元に報告します。DisplayStatus フィールドは、以下の条件のいずれかが満たされた場合にのみ変更されます。

- 無条件変更入力パラメーターが非ゼロになっている

- 時刻入力パラメーターが変更対象オブジェクトの SourceStatusUpdateTime フィールドの値以上になっている

次の RODM ロード機能プリミティブ・ステートメントは、SourceStatusUpdateTime フィールドの値が 9304021430000Z0000 以下の場合にだけ、オブジェクト TRMD401 の DisplayStatus を 129 (適格) に設定します。

```
OP 'GMFHS_Managed_Real_Objects_Class'.'TRMD401'.'ChangeDisplayStatus'
  INVOKED_WITH (SELFDEFINING)
  ((ANONYMOUSVAR)X'000000810011F9F3F0F4F0F2F1F4F3F0F0F0E9F0F0F0F00000');
```

入力: この入力は、名前付きメソッドの標準入力です。
DUIFECDS_Change_Display_Status メソッドで必要な short_lived_parm 入力は、以下のとおりです。

- Display_status (Integer) 新規 DisplayStatus
- UTC (世界標準時) 形式による Source_status_time (CharVar(17)) 新規 SourceStatusUpdateTime。 DUIFECDS に提供するタイム・スタンプは UTC に合わせて正規化しなければなりません。つまり、タイム・スタンプの符号およびオフセットの部分は `Z0000` でなければなりません。
- Unconditional_change (Smallint)。 0 の場合、このメソッドは、ターゲット・オブジェクトの SourceStatusUpdateTime フィールドが Source_status_time 入力パラメーターよりも小さいときにだけターゲット・オブジェクトの DisplayStatus を変更します。 0 以外の場合、このメソッドは、Source_status_time 入力パラメーターを検査しないでターゲット・オブジェクトの DisplayStatus を変更します。

出力: このメソッドが EKG_TriggerNamedMethod 機能を使用して起動される場合には、出力の応答ブロックを用意してください。この応答ブロックには、少なくとも 22 バイトの長さが必要です。応答ブロックの Concat_of_strings フィールドは、次の形式の SelfDefining ストリングです。

表 219. DUIFECDS メソッドの出力

オフセット	長さ	値	説明
000	2	12	SelfDefining ストリングの全長
002	2	30	データ・タイプ AnonymousVar
004	2	16	AnonymousVar データの長さ
006	4		DisplayStatus フィールドの新しい Integer 値
010	4		DisplayStatus フィールドの直前の Integer 値
014	8		更新後の DisplayStatus フィールドの timestamp サブフィールドの値

無条件変更パラメーターが 0 に設定されていて時刻パラメーターが SourceStatusUpdateTime フィールドよりも小さいために、このメソッドでターゲット・オブジェクトの DisplayStatus フィールドが変更されなかった場合、このメソッドは出力パラメーターを次のように設定します。

- 新規 DisplayStatus は DisplayStatus の現行値に設定されます。
- 直前の DisplayStatus は DisplayStatus の現行値に設定されます。
- Timestamp は 0 に設定されます。

このメソッドは、エラーが検出されると戻りコードと理由コードを設定し、RODM ログにタイプ 1 のレコードを書き込みます。 546 ページの表 207 には、このメソッドで戻される可能性のある理由コードのリストが示されています。

DUIFFAWS: 集約ウォーム・スタート・メソッド

これは、RODM データ・キャッシュの実オブジェクトおよび集合オブジェクト内の状況集約に関連するフィールドを初期化するために実行されるオブジェクト独立メソッドです。 GMFHS は、以下の場合にこのメソッドを実行します。

- 始動時の構成定義を初期化するとき
- GMFHS が RODM に対する失われた接続をリカバリーしたとき
- CONFIG NETWORK コマンドが処理されるとき

DUIFFAWS メソッドを使用不能にするには、 GMFHS 始動プロシージャーで AGGRST=NO パラメーターをコーディングするか、あるいは GMFHS DUIGINIT ファイルで LCON-AGGRST-REQUIRED=NO をコーディングしてください。

機能: このメソッドは、以下のフィールドを初期化し直します。

- Display_Resource_Type_Class オブジェクトにリンクされている各実オブジェクトの DefaultAggregationPriorityValue フィールド
- Display_Resource_Type_Class オブジェクトにリンクされている各集合オブジェクトの、以下のフィールド
  - NOXCPTCount
  - PriorityXCPTCount
  - SuspendedCount
  - StatusGroupCounts
  - TotalRealResourceCount
  - UnknownCount
  - XCPTCount

これらのフィールドを初期化し直した後で、このメソッドは各集合オブジェクトの状況を計算し直します。

障害またはアプリケーション・エラーによって前のリスト内の集合オブジェクト・フィールドのうちの 1 つまたは複数が正しくなくなった場合には、RODM ロード機能を使用して DUIFFAWS メソッドを起動することができます。

次の RODM ロード機能ステートメントは DUIFFAWS メソッドを起動します。

```
OP DUIFFAWS INVOKED_WITH;
```

入力: このメソッドには、入力パラメーターはありません。

出力: このメソッドは、エラーが検出されると戻りコードと理由コードを設定し、RODM ログにタイプ 1 のレコードを書き込みます。 546 ページの表 207 には、このメソッドで戻される可能性のある理由コードのリストが示されています。

DUIFFIRS: 初期リソース状況設定メソッド

このメソッドは、構成定義の後で GMFHS によって起動されます。リソース状況請求の対象とならず、またドメイン内のリソースに関するアラートを受け取る能力が

NMG の AgentStatus に依存しないことを示す AgentStatusEffect フィールドを含む NMG_Class オブジェクトにリンクされている Non_SNA_Domain_Class オブジェクトごとに起動されます。

このメソッドは、ドメインの InitialResourceStatus フィールドの値が 132 (不明) でない場合にリソース状況請求が行われない非 SNA ドメインに関して、ゲートウェイ通信セッションが確立されたときにも起動されます。

機能: このメソッドは、構成の初期化中に GMFHS によって起動されます。関連付けられている NMG で AgentStatusEffect が 0 に指定されている場合にリソース状況請求の対象とならない非 SNA ドメインごとに起動されます。

このメソッドは、非 SNA ドメインの InitialResourceStatus フィールドの値が 132 (不明) でない場合に、その非 SNA ドメイン内のリソースに関して状況請求が開始されたときにも起動されます。

入力: DUIFFIRS_Set_Initial_Resource_Status メソッドで必要な入力は次のとおりです。

- Non_SNA_Domain_Class オブジェクトの RODM オブジェクト ID
- 変更と関連付けられる UTC タイム・スタンプ形式の時刻
- 無条件変更インディケーター。この 2 バイト・フィールドが 0 になっていない場合、このメソッドは、非 SNA ドメイン内のすべてのリソースを、そのドメインの InitialResourceStatus フィールドの値に設定します。無条件変更インディケーターが 0 になっている場合、このメソッドは、非 SNA ドメイン内のリソースで DisplayStatus が 132 (不明) に指定されているときには、そのリソースを InitialResourceStatus フィールドの値に設定します。

次の 16 進数ストリングは、DUIFFIRS メソッドに対する入力パラメーターの例を示しています。この例は、SNA_Domain_Class に属する、 RODM オブジェクト ID の値が X'00010010F9DC34AA' のターゲット・オブジェクトを指定しています。時刻は、1993 年 5 月 2 日の 1430Z に指定されています。無条件変更インディケーターは 1 に設定され、ドメイン内のすべてのリソースが更新されるようになっています。入力パラメーターは次のとおりです。

```
X'00010010F9DC34AAF9F3F0F5F0F2F1F4F3F0F0F0E9F0F0F0F00001'
```

出力: このメソッドは、エラーが検出されると戻りコードと理由コードを設定し、RODM ログにタイプ 1 のレコードを書き込みます。 546 ページの表 207 には、このメソッドで戻される可能性のある理由コードのリストが示されています。

DUIFFRAS: 集合体状況再計算メソッド

このオブジェクト独立メソッドは、すべての集合オブジェクトの DisplayStatus を計算し直すために起動することができます。

機能: このメソッドは、各集合体の状況カウンターに基づいて各集合オブジェクトの状況を計算し直します。

入力: このメソッドには、入力パラメーターは必要ありません。このメソッドは、以下の RODM ロード機能プリミティブ・ステートメントで起動されます。

```
OP DUIFFRAS INVOKED_WITH;
```

出力: このメソッドは、エラーが検出されると戻りコードと理由コードを設定し、RODM ログにタイプ 1 のレコードを書き込みます。 546 ページの表 207 には、このメソッドで戻される可能性のある理由コードのリストが示されています。

DUIFFSUS: 不明状況設定メソッド

このオブジェクト独立メソッドは、指定された Non_SNA_Domain_Class の Resources フィールドにリンクされたすべての実オブジェクトの DisplayStatus フィールドを 132 (不明) に設定するために起動されます。GMFHS は、以下の場合にこのメソッドを起動します。

- DUIFFIRS メソッドが起動されないそれぞれの非 SNA に関して構成定義が初期化された後
- Non_SNA_Domain_Class オブジェクトの ReportsToAgent フィールドにクラスされた NMG_Class オブジェクトの AgentStatus フィールドが 1 (適合) または 3 (中間) から 0 (不明) または 2 (不良) に変更され、AgentStatusEffect フィールド値が、ドメイン内のリソースに関するアラートを受け取る機能が NMG の AgentStatus の影響を受けたことを示しているとき
- GMFHS が、ドメインと関連付けられているトランザクション・プログラムまたはエレメント・マネージャーがダウンしていることを示すアラートを受け取ったとき

機能: このメソッドは、指定された Non_SNA_Domain_Class オブジェクトの Resources フィールドにリンクされたすべての実リソース・オブジェクトの DisplayStatus フィールドの値を 132 (不明) に設定します。また、これらの各オブジェクトの SourceStatusUpdateTime フィールドの値を、指定された値に設定します。

入力: DUIFFSUS_Set_Unknown_Status メソッドで必要な入力は、次のとおりです。

- ドメインの RODM オブジェクト ID を表す DomainObjectID
- UTC 形式による、SourceStatusUpdateTime フィールドの新しい値を表す StatusUpdateTime

次の 16 進数ストリングは、DUIFFSUS メソッドに対する入力パラメーターの例を示しています。この例は、SNA_Domain_Class に属する、 RODM オブジェクト ID の値が X'00010010F9DC34AA' のターゲット・オブジェクトを指定しています。時刻は、1993 年 5 月 2 日の 1430Z に指定されています。入力パラメーターは次のとおりです。

```
X'00010010F9DC34AAF9F3F0F5F0F2F1F4F3F0F0F0E9F0F0F0F0'
```

出力: このメソッドは、エラーが検出されると戻りコードと理由コードを設定し、RODM ログにタイプ 1 のレコードを書き込みます。 546 ページの表 207 には、このメソッドで戻される可能性のある理由コードのリストが示されています。

DUIFRFDS : DisplayStatus 変更メソッド DUIFCRDC の最新表示

このオブジェクト独立メソッドは、RODM で定義されている各実リソースおよび集合リソースに関して、 DisplayStatus フィールドを現行の DisplayStatus 値に変更するために、任意のアプリケーションによって実行することができます。

機能: このメソッドは、DisplayStatus マッピング・テーブル DUIFSMT が変更された場合に役立ちます。ネットワークからの状況変更によって例外ビューの更新が起

動されるのを待つ代わりに、 DUIFRFDS メソッドを実行して状況を変更させ、それによってそのオブジェクトの例外状態を計算し直すようにできます。 これで、適切な例外ビューが更新されます。

入力: このメソッドは入力パラメーターを必要とせず、以下の RODM ロード機能プリミティブ・ステートメントによって起動できます。

```
OP DUIFRFDS INVOKED_WITH;
```

このメソッドを起動する例については、サンプル CNMSJH13 を参照してください。

出力: このメソッドは、エラーが検出されると戻りコードと理由コードを設定し、RODM ログにタイプ 1 のレコードを書き込みます。 546 ページの表 207 には、このメソッドで戻される可能性のある理由コードのリストが示されています。

DUIFVCFT: 例外状態の変更

このオブジェクト独立メソッドは、ユーザー・メソッドがオブジェクトの例外状態を変更するときに実行することができます。

機能: メソッド DUIFVCFT を実行するユーザー・メソッドは、 DisplayStatus マッピング・テーブル DUIFSMT 内で USRXMETH キーワードによって指定されます。サンプル・ユーザー・メソッド DUIFCUXM および DUIFCUX2 は、ResourceTraits フィールド内の値 XCPT または NOXCPT を実 DisplayStatus 変更の処理と同じ方法に設定するために、メソッド DUIFVCFT を実行します。
DUIFVCFT は、次にメソッドを起動して、例外状態の変更によってオープンの例外ビューとの間でオブジェクトの追加もしくは削除が起こるかどうかを判別します。

入力: 表 220 は、メソッド DUIFVCFT の入力パラメーターのリストを示しています。

表 220. DUIFVCFT の入力値

パラメーター	データ・タイプ	フィールドの長さ
Total_Length	SMALLINT	2
Data_Type	SMALLINT	2
Data_Length	SMALLINT	2
Resource_Object_ID	OBJECTID	8
Requested_exception_status	INTEGER	4

出力: リソースの ResourceTraits フィールドが、要求された例外状態を反映して更新されます。

このメソッドは、エラーが検出されると戻りコードと理由コードを設定し、RODM ログにタイプ 1 のレコードを書き込みます。 546 ページの表 207 には、このメソッドで戻される可能性のある理由コードのリストが示されています。

注:

1. Resource_Object_ID は、DisplayStatus の変更によってユーザー・メソッドが起動される原因となったリソースのオブジェクト ID です。

2. リソースを例外状況にしたくない 場合には、 Requested_exception_status を 0 に設定してください。 DUIFVCFT は、このリソースの ResourceTraits フィールドを値 NOXCPT に設定します。
3. リソースを例外状況にしたい 場合には、 Requested_exception_status を 1 に設定してください。 DUIFVCFT は、このリソースの ResourceTraits フィールドを値 XCPT に設定します。
4. 詳細については、 130 ページの『例外ビューの DisplayStatus メソッドを作成する』を参照してください。

DUIFVINS: ビュー細分性インストール・メソッド (DUIFVNOT)

このオブジェクト独立メソッドは、クラスまたはフィールドにメソッド DUIFVNOT をインストールします。

機能: DUIFVINS は、データ・モデルに追加される新規クラスまたは接続性フィールドごとに実行する必要があります。

DUIFVNOT は、クラスのすべてのオブジェクトによって継承されます。 GMFHS が DUIFVNOT をインストールするフィールドの全リストについては、サンプル FLBTRDME を参照してください。

入力: 表 221 は、メソッド DUIFVINS の入力パラメーターのリストを示しています。

表 221. DUIFVINS の入力値

パラメーター	データ・タイプ	フィールドの長さ
enable_change_status	SMALLINT	2
rule	INTEGER	4
notification_method	OBJECTID	8
class	CLASSID	4
field	FIELDID	4

enable_change_status

このパラメーターは、フィールドが直前の値と同じ値に設定されたときにビュー変更通知 (VCN) が出されないようにするために使用します。

このパラメーターの値は次のとおりです。

0 prev_val サブフィールドがフィールドに存在しない場合、またはフィールドが直前の値と同じ値に変更されて VCN を出す必要がある場合に使用します。

1 prev_val サブフィールドがフィールドに存在していて、フィールドが直前の値と同じ値に変更されたときには VCN を出さない場合に使用します。

rule VCN のフィールドの変更結果を出すかどうかを判別するための基準です。 ANY_FIELD_OBJECT_CHANGE を除き、これらの各規則では、変更に関連するオブジェクト ID またはクラス ID およびフィールド ID が、少なくとも 1 つの現在オープンされているビューの構成に使用されていることが暗黙に想定されています。

このパラメーターの値は次のとおりです。

1 OBJECT_CHANGE: フィールドがオブジェクト・レベルで変更された場合にビュー更新を送信します。

2 VALUE_INCREASE: フィールドがオブジェクト・レベルで変更されて、フィールドの値が大きくなった場合にビュー更新を送信します。

3 VALUE_DECREASE: フィールドがオブジェクト・レベルで変更されて、フィールドの値が小さくなった場合にビュー更新を送信します。

4 CONNECTIVITY: この規則は ObjectLink および ObjectLinkList データ・タイプに適用されます。フィールドがオブジェクト・レベルで変更され、リンクまたはリンク解除によって、ビューに表示される接続性が変更された場合に、ビュー更新を送信します。以下のビュー・タイプの場合には、オブジェクトのうちの少なくとも一方が現在ビューに含まれているときにビュー変更が表示されます。

構成親
構成論理
構成物理
構成バックボーン
構成子
構成子 II
構成子 III

その他のすべてのビュー・タイプの場合には、両方のオブジェクトがビューに含まれていなければビュー変更は表示されません。

5 CLASS_CHANGE: フィールドがクラス・レベルで変更された場合にビュー更新を送信します。

6 OBJECT_OR_CLASS_CHANGE: フィールドがオブジェクト・レベルまたはクラス・レベルで変更された場合にビュー更新を送信します。

7 ANY_FIELD_OBJECT_CHANGE: ビューの作成に使用されていたフィールドであるかどうかにかかわらず、そのフィールドがオブジェクト・レベルで変更された場合にはビュー更新を送信します。これは、例外ビューの場合も含めて、ビューの作成に関係していないフィールドをモニターしたい場合に使用します。その他の規則では、例外ビューについて VCN が出されることはありません。詳細については、118 ページの『例外ビューのオブジェクトおよび基準を定義する』を参照してください。

5000 LU_CHANGE: LU タイプのオブジェクトでフィールドが変更され、その monitoringLuCollection フィールドが、LU コレクションが変更されていないことを示しているときに、ビュー変更を送信します。

notification_method
通知メソッド DUIFVNOT のオブジェクト ID。

class DUIFVNOT をインストールすべきクラスのクラス ID。

field DUIFVNOT をインストールすべきフィールドのフィールド ID。

次に示すのは、DUIFVINS を実行するための RODM ローダー・ステートメントの例です。

```
OP DUIFVINS INVOKED_WITH (SELFDEFINING)
 (
  (SMALLINT) 0
  (INTEGER)  1
  (OBJECTID) EKG_Method.DUIFVNOT
  (CLASSID)  GMFHS_Real_Objects_Class
  (FIELDID)  GMFHS_Real_Objects_Class.DisplayResourceType
 );
```

出力: このメソッドは、エラーが検出されると戻りコードと理由コードを設定し、RODM ログにタイプ 1 のレコードを書き込みます。 546 ページの表 207 には、このメソッドで戻される可能性のある理由コードのリストが示されています。

第 5 部 付録

付録 A. RODM ツール

NetView は、RODM とともに使用するために、以下のツールを提供します。
- RODMView
- RODM アンロード機能
- FLCARODM (RODM アクセス機能)
- BLDVIEWS
- ビジュアル BLDVIEWS (VBV)

RODMView 機能は、RODM データ・キャッシュ内のフィールドの値を表示および更新するための対話式アプリケーション・プログラムです。 RODMView は NetView プログラム内で OST タスクの制御下で実行されます。

RODM アンロード機能は、クラス、オブジェクト、およびフィールドをアンロードするために使用することができます。例えば、RODM アンロード機能を使用して、既存の RODM をアンロードし、 RODM アンロード機能からの出力を指定して新しいバージョンの RODM をロードすることにより、あるバージョンの RODM から別のバージョンの RODM にマイグレーションすることができます。詳細については、613 ページの『RODM アンロード機能』を参照してください。

FLCARODM は、RODM に対する高速で効率の良い REXX インターフェースを提供します。 (FLCARODM は以前、 RODM アクセス機能またはマルチシステム・マネージャー・アクセス機能として知られていました。) FLCARODM により、REXX で作成された NetView CLIST を使用して、オブジェクトの作成、更新、および削除を行うことができます。 FLCARODM は RODM への簡単なインターフェースを提供し、これにより、RODM へのバッチ要求を発行することによる処理上の利点を活用できるようになります。詳細については、618 ページの『FLCARODM』を参照してください。

BLDVIEWS は、ユーザーのネットワーク・レイアウトおよびそれをモニターするためにユーザーが使用したいスタイルに合った、カスタム・ビューを定義するために使用するツールです。このツールは、GMFHS、SNA トポロジー・マネージャー、およびマルチシステム・マネージャーのデータ・モデルのオブジェクトを操作します。マルチシステム・マネージャーで検出されたネットワーク・リソース・オブジェクトに関して、 NetView 管理コンソール (NetView 管理コンソール) から汎用コマンド・サポートを使用可能にすることによって、 BLDVIEWS により、マルチシステム・マネージャーの主要なリソースに関する汎用コマンドにデフォルトのコマンド・セットを簡単にマップすることができます。詳細については、666 ページの『BLDVIEWS』を参照してください。

Visual BLDVIEWS (VBV) は、RODM のビューおよび情報の管理を単純化するアプリケーションです。 VBV は、BLDVIEWS ツールおよび RODMView ツールに対する、グラフィカルなドラッグ・アンド・ドロップ・インターフェースを提供します。詳細については、VBV オンライン・ヘルプを参照してください。

この付録のパネルのいくつかは、GMFHS 情報を示します。

RODMView

このセクションでは、RODMView の使用方法を説明します。以下のトピックについて説明します。

- RODMView 内のナビゲーション
- RODMView の制約事項
- RODMView の開始
- RODMView 機能の使用

RODMView 内のナビゲーション

RODMView 内でのナビゲーションは、以下のようにして行うことができます。

- メイン・メニューを使用する
- アクセラレーター PF キーを使用する
- パネルの下部に表示される PF キーを使用する

パネル・データ入力フィールドは下線の付いた行によって示され、各パネルの下部にはコマンド行があります。

メニューを使用するナビゲーション

RODMView にはメイン・メニューがあります。このパネルは 578 ページの図 96 で説明されています。必要なオプションにナビゲートするには、対応する選択番号を入力するか、あるいはカーソルで該当行を選択して **Enter** を押してください。無効なオプションを入力すると、エラー・メッセージが表示されます。

どの RODMView パネルからも、関連するアクセラレーター PF キーを押すことにより、別の RODMView 機能のパネルに直接ナビゲートすることができます。 表 222 に示すように、 PF13 から PF22 までのアクセラレーター PF キーは、それぞれオプション番号 1 から 10 までに対応しています。

表 222. アクセラレーター PF キーとオプション

PF キー	オプション	パネル
PF13	オプション 1	アクセスおよび制御
PF14	オプション 2	単純照会
PF15	オプション 3	複合照会
PF16	オプション 4	探索アクション
PF17	オプション 5	リンク/リンク解除
PF18	オプション 6	フィールド変更
PF19	オプション 7	サブフィールド・アクション
PF20	オプション 8	作成アクション
PF21	オプション 9	削除アクション
PF22	オプション 10	メソッド・アクション

パーソナル・コンピューター・ベースの多くの端末エミュレーターでは、PF13 から PF22 までの範囲の PF キーは、シフト・キー (またはその他の制御キー) を押しながら、オプション番号に直接対応する 1 から 10 までの範囲の PF キーを押すことによってアクセスできます。

アクティブな PF キーのリストは、RODMView パネルの下部に表示されます。表示される PF キーおよびそれらのキーが果たす機能は、表示されているパネルによって異なります。 表 223 に、PF キーと対応する機能がリストされています。

表 223. PF キーの機能

PF キー	機能
PF1	ヘルプ情報を表示する。
PF2	コマンドを終了して抜け出す。
PF3	直前のパネルに制御権を戻す。
PF4	照会入力フィールドをクリアする。
PF5	• 照会出力を表示しているときに最新の検出要求を繰り返す。 • 照会および検索パネルを表示しているときに最新の照会を再表示するか出力を探索する。
PF6	リング内の次のアプリケーションにロールする。
PF7	直前のパネルに戻る。
PF8	次のパネルに進む。
PF9	照会出力を NetView ログにコピーする。
PF10	カーソルが照会または探索出力の 16 進数のオブジェクト ID に位置付けられているときに、そのオブジェクト ID を他のパネルの入力行にコピーする。
PF11	カーソルが照会または探索出力の SystemView® クラスまたはフィールド名に位置付けられているときに、 SystemViewテキスト名と数値 ID との間の変換を行う。
PF12	RODMView コマンド行に入力されたコマンドを再呼び出しする。

RODMView の制約事項

RODMView の制約事項を以下に列挙します。

- RODMView が実行できるコマンドの最大長は 240 文字です。 RODMView が実行するコマンドは、ロング・ネームの代わりに、クラス、オブジェクト、またはフィールド ID を使用して短くすることができます。
- RODM のオブジェクト名の全長は 254 バイトまでですが、オブジェクト名入力フィールドの最大長はどの RODMView パネルでも 64 バイトまでに制限されています。名前の代わりにオブジェクト ID を使用するか、あるいは名前の中でパターン・マッチング文字 (ワイルドカード) を使用することにより、文字数の制限を回避することができます。
- ワイルドカードがサポートされるのは照会機能のみです。
- 1 つの NetView セッションでは、一度に 1 つの RODMView コピーしか実行できません。 RODMView の別のコピーを実行しようとすると、プログラムが終了し、前のコピーの RODMView が制御権を取り戻します。
- EKGVACTM コマンド・プロセッサーの特定のキーワードを制限することができます。

参照: 保護可能なキーワードのリストは、「*IBM Tivoli NetView for z/OS* アドミニストレーション・リファレンス」を参照してください。それ以外の RODMView コマンド・プロセッサーでは、キーワードを制限することはできません。

RODMView の開始

RODMView を開始するには、図 95 に示すように、 NetView NCCF コマンド行で **RODMVIEW** と入力してください。

```
NCCF                    N E T V I E W    A01NV OPER2    10/18/97 12:34:56
- A01NV    DSI020I OPERATOR OPER2 LOGGED ON FROM TERMINAL A01A703 USING
           PROFILE (A75PROF ), HCL ( )
C A01NV    CNM357I PFKDEF : PF KEY SETTINGS NOW ESTABLISHED.
C A01NV    +              : DISPFK TO SEE YOUR PF KEY SETTINGS
--------------------------------------------------------------------------------
 ???
RODMVIEW
```

図 95. RODMView NetView コマンド行呼び出し

RODMView のメイン・メニューが、 図 96 のように表示されます。

```
EKGVMMNI                R O D M V i e w  A01NV OPER2    10/18/97 12:34:56

Select one of the following, press Enter.

                         _ 1.  Access and Control
                           2.  Simple Query
                           3.  Compound Query
                           4.  Locate Objects
                           5.  Link/Unlink
                           6.  Change Field
                           7.  Subfield Actions
                           8.  Create Actions
                           9.  Delete Actions
                           10. Method Actions

CMD==>
F1= Help   F2= End    F3= Return                        F6= Roll  F12=PrevCmd
```

図 96. RODMView メイン・メニュー — EKGVMMNI

RODMView メニューからは、利用可能な任意の機能を選択することができます。オプションの選択は、次の 3 つの方法で行うことができます。
- 選択項目の隣のプロンプトで対応する番号を入力する。
- 選択項目の行にカーソルを移動し、**Enter** を押す。
- アクセラレーター PF キーを使用する。

他のなんらかの機能を使用する前に、RODM にサインオンする必要があります。

アクセスおよび制御機能

メイン・メニューから「**1. Access and control (1. アクセスおよび制御)**」を選択して、 図 97 に示す「Access and Control (アクセスおよび制御)」パネルを表示させてください。

```
EKGVACTI              Access and Control  A01NV OPER2    10/18/97 12:34:56

RODM name . . rodmname
User ID . . . rodmuser

User password

RODM function connect (COnnect, Disconnect, CHeckpoint, Stop, Update)

Query pattern matching character *
Checkpoint before stop Y (Y, N)  For Stop function only









CMD==>
F1= Help   F2= End    F3= Return                          F6= Roll  F12=PrevCmd
```

図 97. RODMView Access and Control (アクセスおよび制御) パネル － EKGVACTI

RODM 名と次のいずれかの機能を入力してください。
- COnnect (接続)
- Disconnect (切断)
- CHeckpoint (チェックポイント)
- Stop (停止)
- Update (更新)

注:

1. 機能の名前の大文字部分は、その機能を指定するために入力する最小限の文字を表しています。例えば接続機能を指定するには、**CO** と入力してください。
2. RODM を開始してからでなければ、 RODMView を使用して RODM に接続することはできません。

ユーザー ID を指定しなかった場合には、デフォルトとして NetView オペレーター ID が使用されます。ユーザー・パスワードを指定しなかった場合には、ユーザー・パスワードとしてブランクが使用されます。

照会パターン・マッチング文字は、照会を発行するときにワイルドカードとして使用される文字です。アスタリスク (*) はオブジェクト名の一部として有効な文字であり、ワイルドカードとして使用するには不適切なので、注意してください。接続機能では、ワイルドカードに値が割り当てられます。切断と再接続を行わずにこの値を変更するには、更新機能を使用してください。このパネルでパターン・マッチング文字を変更した場合、その変更は接続または更新要求が正常に行われた場合にのみ有効になります。

ユーザーのシステムでシステム許可機能が使用可能になっている場合、RODM はその機能を使用します。 選択した機能を実行するには、ユーザーのユーザー ID が許可されていなければなりません。このユーザー ID は、NetView のオペレーター ID とは異なる可能性があります。不明な場合には、セキュリティー管理者にご確認ください。他の RODM ユーザーおよびアプリケーションとのアクセス競合を避けるには、各 RODM ユーザーに、z/OS システム全体で固有の RODM ユーザー ID を割り当てるのが最善の方法です。

必要なフィールドに情報を入力して **Enter** を押すと、パネルの下部近くにメッセージが表示され、要求の結果が通知されます。

```
EKGVACTI              Access and Control  A01NV OPER2    10/18/97 12:34:56

RODM name . . RODMNAME
User ID . . . RODMUSER

User password

RODM function CONNECT (COnnect, Disconnect, CHeckpoint, Stop, Update)

Query pattern matching character *
Checkpoint before stop Y (Y, N)  For Stop function only









EKGV0000I Request is successful(0/0)
CMD==>
F1= Help   F2= End    F3= Return                          F6= Roll  F12=PrevCmd
```

図 98. 接続が成功した場合の RODMView メッセージ

図 98 の左下隅のメッセージ行は、要求が成功して RODM から戻りコード 0 (ゼロ) と理由コード 0 (ゼロ) が戻されたことを表しています。戻りコードと理由コードは、メッセージの隣に括弧で囲まれて示されます。この例では、戻りコードも理由コードもともに 0 になっています。

RODMView は、これらの戻りコードと理由コードの組み合わせを RODM から受け取ると、その組み合わせを変更して関連の RODMView メッセージを表示しようとします。 RODM の戻りコードと理由コードの組み合わせは数値になっているため、 RODMView は最も一般的な組み合わせのみを変換します。 RODM から戻された戻りコードと理由コードのペアが RODMView によって変換されない場合には、RODM の戻りコードと理由コードが次のメッセージで表示されます。

```
EKGV8037E RODM return code/reason code is (return_code/reason_code)
```

RODM 特有のすべての戻りコードと理由コードは、範囲が 0 から 49151 までになっています。詳細については、515 ページの『RODM の戻りコードと理由コード』を参照してください。

RODM が特に原因となっていない問題が RODMView コマンド・プロセッサーによって検出された場合、理由コードは 67000 よりも大きくなっています。これらの理由コードは RODMView によって変換され、対応するメッセージが表示されます。

RODM に正常にサインオンできたら、 **PF3** を押して RODMView のメイン・メニューに戻ってください。

単純照会機能

各種の照会をさまざまなレベルの詳細さで実行するには、 RODMView メイン・メニューから「**2. Simple Query (単純照会)**」を選択してください。 図 99 に示すような「Simple Query (単純照会)」パネルが表示されます。

```
EKGVQUEI                    Simple Query  A01NV OPER2    10/18/97 12:34:56

RODM name   RODMNAME
User ID . . RODMUSER

SystemView class name _
Class name    _
Class ID      _

Object name   _
Object ID     _ (Hexadecimal value)

SystemView field name _
Field name    _
Field ID      _

Level of field detail . . DATA   (Struct, Data, Hex)
Level of subfield detail  NONE   (Struct, Data, Hex, None)
Maximum lines returned    5000
Display field IDs . . . . N (Y, N) Display extended field info N (Y, N)


CMD==>
F1= Help   F2= End    F3= Return  F4= Clear   F5= PrevOut F6= Roll  F12=PrevCmd
```

図 99. RODMView の Query (照会) パネル − EKGVQUEI

照会要求の基準として RODMView に使用させたい基準を入力し、 **Enter** を押してください。例えば、ユーザーの EKG_User クラスにおけるユーザー ID を表すオブジェクトを表示させたい場合には、582 ページの図 100 に示すような情報を入力してください。 EKG_User クラスで作成されるオブジェクトは、現在 RODM にサインオンされているユーザーを表すことに注意してください。

```
EKGVQUEI                    Simple Query  A01NV OPER2    10/18/97 12:34:56

RODM name   RODMNAME
User ID . . RODMUSER

SystemView class name _
Class name  EKG_User
Class ID    _

Object name RODMUSER
Object ID   _ (Hexadecimal value)

SystemView field name _
Field name  _
Field ID    _

Level of field detail . . DATA   (Struct, Data, Hex)
Level of subfield detail  NONE   (Struct, Data, Hex, None)
Maximum lines returned    5000
Display field IDs . . . . N (Y, N) Display extended field info N (Y, N)


CMD==>
F1= Help   F2= End    F3= Return  F4= Clear   F5= PrevOut F6= Roll  F12=PrevCmd
```

図 100. RODMView によるユーザー ID の照会

SystemView のクラス名とフィールド名を除き、RODM におけるクラス、オブジェクト、およびフィールドの名前では大文字と小文字が区別されることに注意してください。

指定されたオブジェクトが存在する場合、図 101 に示すように出力が表示されます。

```
EKGVQUEO                    Query Output  A01NV OPER2    10/18/97 12:34:56

                                                            Lines 1 to 17 of 47
-------------------------------------------------------------Matching entity ID:
 MyID (OBJECTID)
    (OBJECTID) 000F0006D3299015
               'RODMUSER'
    (CLASSID)  6
               'EKG_User'
-   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -
 MyPrimaryParentID (CLASSID)
    6
    'EKG_User'

 EKG_Status (INTEGER)
    1

 EKG_LogLevel (INTEGER)
    8

EKGV0000I Request is successful (0/0)
CMD==>
F1= Help   F2= End    F3= Return              F5= RptFind F6= Roll
```

図 101. RODMView の Query Output (照会出力) パネル

図 101 で示した「Query Output (照会出力)」パネルの右上隅には、利用可能な出力が 47 行あって、そのうちの最初の 17 行が現行パネルに表示されていることが示されています。

メッセージの戻りコードと理由コードが 0 の場合、その要求は成功しています。

探索基準に一致していることが RODMView によって検出された各クラス・エンティティーまたはフィールド・クラスについて、そのエンティティー ID が「Matching entity ID: (突き合わせエンティティー ID:)」というヘッダーの下に表示され、その後にユーザーが指定したフィールドが示されます。この例では照会基準が非常に狭く特定されているため、1 つのエンティティーだけが検出されています。このオブジェクトのすべてのフィールドを表示させたい場合には、「Field name (フィールド名)」および「Field ID (フィールド ID)」フィールドをブランクのままにしてください。

名前ではなく、数値 ID によって RODM を照会することもできます。エンティティーの ID は、名前を指定して照会することによって得られます。 ID は「Matching entity ID (突き合わせエンティティー ID)」セクションに表示され、分かりやすくするためにそのエンティティーの「MyID」フィールドにも表示されます。

数値 ID とともに、対応する名前も指定すると、数値 ID が優先されて名前は無視されます。例えば、「Class Name (クラス名)」に **EKG_System** を指定し、「Class ID (クラス ID)」に **1** を指定して照会を行うと、 ID が 1 である UniversalClass が照会の対象となります。数値 ID が存在しているため、EKG_System という名前のほうは RODM によって無視されます。

ユーザーが照会したオブジェクト上の各フィールドについて、フィールド名が表示され、そのデータ・タイプが括弧に囲まれて表示され、その値が (フィールド名の下に) 表示されます。場合によっては、フィールドに関する追加情報が自動的に得られることがあります。

例えば、RODM で定義されたデータ・タイプ ClassID は整数です。数値に対応するクラス名が分かると便利なため、RODMView はさらに RODM に照会してクラス名とその ID を突き合わせます。 582 ページの図 101 の「MyPrimaryParentID」フィールドを参照してください。

値が割り当てられていないフィールドの場合、フィールド名とフィールド・データ・タイプを含む行の後にブランク行が続きます。

照会出力パネルから PF7 および PF8 を使用するか、あるいはコマンド行で **UP** および **DOWN** コマンドを使用して、出力を上方または下方にページ移動することができます。

次の表は、「Query Output (照会出力)」パネルのコマンド行から利用可能な出力制御コマンドを要約したものです。

表 224. Query Output (照会出力) の制御コマンド

コマンド	説明
UP *n*	出力を 1 ページずつ、または任意指定により *n* 行ずつ上方にスクロールさせる。
DOWN *n*	出力を 1 ページずつ、または任意指定により *n* 行ずつ下方にスクロールさせる。
TOP	出力を先頭までスクロールさせる。
BOTTOM	出力を終わりまでスクロールさせる。

表 224. Query Output (照会出力) の制御コマンド (続き)

コマンド	説明
F *find_word*	現行パネルから出力の終わりまで、*find_word* を検索する。
F *find_word* PREV	現行パネルから出力の先頭まで、*find_word* を検索する。キーワード PREV の省略形として P を使用することができる。

注: F コマンドを使用して語を検索するときには、*find_word* には英数字の単一ストリングを指定しなければなりません。スペースは、単一引用符で囲んでも使用することはできません。

コマンド行に **F find_word** と入力することにより、現行パネルから出力の終わりまで、出力内のどこにある単一の語でも検索できます。同様に、コマンド行に **F find_word PREV** または **F find_word P** と入力することにより、パネル上の現在位置から出力の先頭まで、語を検索できます。

SystemView クラスとフィールド名による RODM の照会

RODM アプリケーションの中には、 NetView マルチシステム・マネージャーなどのように、 SystemView データ・モデル用に特別な命名規則を使用するものもあります。この規則は、SystemView 名を表すための、ピリオドで区切られた数字からなります。 RODMView は、SystemView データ・モデルのテキスト・クラス名を変換することができます。例えば、図 102 に示されている SystemView クラス名 `appnNN` と SystemView フィールド名 `usageState` を、 585 ページの図 103 に示されている、それに対応する RODM クラス名 `1.3.18.0.0.1822` とフィールド名 `2.9.3.2.7.39` に変換することができます。

```
EKGVQUEI                    Simple Query  A01NV OPER2    10/18/97 12:34:56

RODM name   RODMNAME
User ID . . RODMUSER

SystemView class name appnNN
Class name
Class ID

Object name
Object ID    (Hexadecimal value)

SystemView field name usageState
Field name
Field ID

Level of field detail . . DATA   (Struct, Data, Hex)
Level of subfield detail  NONE   (Struct, Data, Hex, None)
Maximum lines returned    5000
Display field IDs . . . . N (Y, N) Display extended field info N (Y, N)


CMD==>
F1= Help   F2= End    F3= Return  F4= Clear   F5= PrevOut F6= Roll  F12=PrevCmd
```

図 102. SystemView のクラス名とフィールド名を使用して行う RODMView 単純照会

```
EKGVQUEI                    Simple Query  A01NV OPER2    10/18/97 12:34:56

RODM name   RODMNAME
User ID . . RODMUSER

SystemView class name appnNN
Class name  1.3.18.0.0.1822
Class ID    =

Object name =
Object ID   = (Hexadecimal value)

SystemView field name usageState
Field name  2.9.3.2.7.39
Field ID    =

Level of field detail . . DATA   (Struct, Data, Hex)
Level of subfield detail  NONE   (Struct, Data, Hex, None)
Maximum lines returned    5000
Display field IDs . . . . N (Y, N) Display extended field info N (Y, N)


CMD==>
F1= Help   F2= End    F3= Return  F4= Clear   F5= PrevOut F6= Roll  F12=PrevCmd
```

図 103. RODMView 単純照会で変換された SystemView のテキスト・クラス名とフィールド名

パターン・マッチング文字を使用して行う RODM の照会

パターン・マッチング文字を使用すると、より少ない特定基準によって検索を行うことができます。例えば、探したいオブジェクトの名前が分かっていてもそのクラスが分からない場合、あるいは特定の語を含むクラス名が分かっている場合には、パターン・マッチング文字 (ワイルドカード) を使用することができます。

RODMView のパターン・マッチング文字は、照会機能の場合にだけ、「Class name (クラス名)」、「Object name (オブジェクト名)」、および「Field name (フィールド名)」入力フィールドに使用することができます。

RODMView におけるデフォルトのパターン・マッチング文字はアスタリスク (*) ですが、「Access and Control (アクセスおよび制御)」パネルを使用してユーザーが変更することができます。アスタリスクはオブジェクト名で有効な文字として使用できるため、名前にアスタリスクが含まれるオブジェクトの照会を行うと、予期しない結果が生じることがあります。パターン・マッチング文字を使用する探索ストリングの例を以下に示します。

Test* Test で始まる名前と一致します。
***Test** Test で終わる名前と一致します。
Test 任意の個所に Test が含まれている名前と一致します。
***** すべての名前と一致します。

例えば、ログに関連するフィールドで、EKG という文字で始まるクラスに定義されているすべてのフィールドを照会するには、586 ページの図 104 に示すような照会を指定してください。

```
EKGVQUEI                    Simple Query  A01NV OPER2    10/18/97 12:34:56

RODM name   RODMNAME
User ID . . RODMUSER

SystemView class name _
Class name  EKG*
Class ID    _

Object name _
Object ID   _ (Hexadecimal value)

SystemView field name _
Field name  *Log*
Field ID    _

Level of field detail . . DATA   (Struct, Data, Hex)
Level of subfield detail  NONE   (Struct, Data, Hex, None)
Maximum lines returned    5000
Display field IDs . . . . N (Y, N) Display extended field info N (Y, N)


CMD==>
F1= Help   F2= End    F3= Return  F4= Clear   F5= PrevOut F6= Roll  F12=PrevCmd
```

図 104. RODMView による Log という語を含むフィールドの照会

RODMView は、名前に Log が含まれているすべてのフィールドを検索します。RODM で定義されているすべてのクラスが検索対象になります。

図 105 は、典型的な RODM の出力パネルを示しています。

```
EKGVQUEO                    Query Output  A01NV OPER2    10/18/97 12:34:56

                                                            Lines 1 to 17 of 17
-------------------------------------------------------------Matching entity ID:
 MyID (CLASSID)
    6
    'EKG_User'
-   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -
 EKG_LogLevel (INTEGER)
    8

 EKG_MLogLevel (INTEGER)
    8
-------------------------------------------------------------Matching entity ID:
 MyID (CLASSID)
    5
    'EKG_System'
-   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -
 EKG_ExternalLogState (INTEGER)
    1
EKGV0000I Request is successful (0/0)
CMD==>
F1= Help   F2= End    F3= Return              F5= RptFind F6= Roll
```

図 105. 'Log' を含むフィールドの RODMView Query Output (照会出力)

図 105 に示すように、RODMView は、フィールド名に Log を含むクラスとして EKG_User クラスと EKG_System クラスの 2 つを検出しています。基準に一致するフィールドとして、EKG_User クラスには 2 つのフィールド EKG_LogLevel および EKG_MLogLevel が含まれています。 EKG_System クラスにはフィールド EKG_ExternalLogState が含まれています。

上の例の出力では、クラス・レベルの情報が示されています。同じ情報をオブジェクト・レベルで見たい場合には、「Object Name (オブジェクト名)」入力フィールドおよび「Class name (クラス名)」入力フィールドにパターン・マッチング文字を入力し、**Enter** を押してください。

照会によっては、特にパターン・マッチング文字を使用した場合、表示される行数が大量になることがあります。照会要求では、「Maximum lines returned (戻される最大行数)」フィールドに指定された行数を超えて表示されることはありません。 0 を指定した場合には、RODMView はデフォルト値として 5000 を使用します。照会に対する応答として「Maximum lines returned (戻される最大行数)」フィールドでの指定を超える行数が戻される場合、最後の 2 行でそのことが通知されます。

注: 「Maximum lines returned (戻される最大行数)」として 5000 を超える値を設定する場合には、注意が必要です。「Maximum lines returned (戻される最大行数)」の値を大きくすると、照会報告書で切り捨てられる行を表示させることができます。しかし、あまり大きな値を指定すると、 NetView の記憶容量を超える恐れがあります。これを避けるためには、照会要求の有効範囲を狭くしてください。 図 106 は、前の照会要求で「Maximum lines returned (戻される最大行数)」を 10 に設定したときに、照会で戻される行が 17 行である場合の結果を示しています。照会要求が正常に完了していて、超過した行が表示されないことにご注意ください。最後に表示されている 2 行は、照会報告書が切り詰められていることを示しています。 この例では、戻される最大行数の値を 17 以上に増やして、照会報告書が切り詰められないようにしてください。

```
EKGVQUEO                    Query Output  A01NV OPER2    10/18/97 12:34:56

                                                          Lines 1 to 12 of 12
-------------------------------------------------------------Matching entity ID:
 MyID (CLASSID)
    6
    'EKG_User'
-   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -
 EKG_LogLevel (INTEGER)
    8

 EKG_MLogLevel (INTEGER)
    8
****Report Truncated****
Returned Lines: 10 Total Lines: 17




EKGV0000I Request is successful (0/0)
CMD==>
F1= Help   F2= End    F3= Return              F5= RptFind F6= Roll
```

図 106. Query Output (照会出力) が大きすぎる RODMView 照会

複合照会機能

複数の基準を使用して各種の照会をさまざまなレベルの詳細さで実行するには、RODMView のメイン・メニューから「**3. Compound Query (複合照会)**」を選択してください。「Compound Query (複合照会)」パネルの外観は 588 ページの図 107 のようになっています。

クラスとオブジェクトを表示するために単純照会で使用する基準は、クラス名とオブジェクト名です。複合照会機能では、クラスとオブジェクトを同じ方法で検索できるだけでなく、他の基準に一致するクラスまたはオブジェクトだけを選択することもできます。例えば、RODM 内のオブジェクトのうちで、フィールドに特定の値が入っているすべてのオブジェクトを検索することができます。また、あるフィールドを介して他のオブジェクトとリンクされているオブジェクトのうちで、フィールドに特定の値が入っているものをすべて検索することもできます。

RODMView のメイン・メニューから「**3. Compound Query (複合照会)**」を選択してください。この照会を指定するには、次の 4 つのパネルが使用されます。

- クラス名とオブジェクト名によって検索開始位置を指定するには、複合照会パネル EKGVQA1I (図 107 に示されています) を使用してください。
- 表示されるクラスまたはオブジェクトが満たす必要のある基準を指定するには、EKGVQA2I パネル (589 ページの図 108 に示されています) を使用してください。
- リンクされたエンティティーを照会するために必要なフィールドを指定するには、パネル EKGVQA3I (589 ページの図 109 に示されています) を使用してください。探索されるフィールドで検出されたエンティティーが表示されるために満たす必要のある基準も指定することができます。
- ユーザーが指定したすべての検索条件に一致するエンティティーのフィールドのうちで表示するフィールドを指定するには (あるいは、すべてのフィールドを表示するにはブランクのままにします)、パネル EKGVQA4I (590 ページの図 110 を参照) を使用してください。

4 つの「Compound Query (複合照会)」パネル間でナビゲートするには、 PF7 と PF8 を使用してください。すべてのパネルのすべての入力フィールドをクリアするには、 **PF4** を押してください。RODMView が確認を要求してきます。

```
EKGVQA1I                  Compound Query  A01NV OPER2    10/18/97 12:34:56

RODM name    =
User ID . . =

Initial query criteria (Specify entity or entities to begin the search with):
  SystemView class name =
  Class name    =
  Class ID      =

  Object name  =
  Object ID    = (Hexadecimal value)

Output options:
  Level of field detail . . DATA  (Struct, Data, Hex)
  Level of subfield detail  NONE  (None, Struct, Data, Hex)
  Maximum lines returned    5000
  Display field IDs . . . . N (Y, N) Display extended field info  N (Y, N)

(Use PF8 to further specify query)

CMD==>
F1= Help   F2= End    F3= Return  F4= Clear   F5= PrevOut F6= Roll
           F8= Next                                          F12=PrevCmd
```

図 107. RODMView Compound Query (複合照会) パネル 1 — EKGVQA1I

```
EKGVQA2I                  Query Criteria  A01NV OPER2    10/18/97 12:34:56

Entities from the previous panel should meet the following criteria:
  SystemView field name _
  Field name _
  Operator   = (=, >, <, <>, <=, >=)
  Value . .  _

Operator between these two criteria AND (And, Or)

Entities from the previous panel should also meet the following criteria:
  SystemView field name _
  Field name _
  Operator   = (=, >, <, <>, <=, >=)
  Value . .  _




(Use PF8 to further specify query)

CMD==>
F1= Help   F2= End    F3= Return  F4= Clear   F5= PrevOut F6= Roll
```

図 108. RODMView Query Criteria (照会基準) パネル 2 — EKGVQA2I

```
EKGVQA3I         Query Traversed Criteria  A01NV OPER2    10/18/97 12:34:56

Find entities linked to the following field (leave blank to ignore):
  Traverse SystemView field name _
  Traverse field name _

Entities found in the Traverse field should meet the following criteria:
  SystemView field name _
  Field name _
  Operator   = (=, >, <, <>, <=, >=)
  Value . .  _

Operator between these two criteria AND (And, Or)

Entities found in the Traverse field should also meet the following criteria:
  SystemView field name _
  Field name _
  Operator   = (=, >, <, <>, <=, >=)
  Value . .  _
(Use PF8 to specify which fields are printed for each entity found)

CMD==>
F1= Help   F2= End    F3= Return  F4= Clear   F5= PrevOut F6= Roll
```

図 109. RODMView Query Traversed Criteria (照会探索基準) パネル 3 — EKGVQA3I

```
EKGVQA4I           Query Field Selection  A01NV OPER2    10/18/97 12:34:56

Field(s) to display of entity (or entities) found:
  SystemView field name _
  Field name _
  Field ID   _









CMD==>
F1= Help   F2= End    F3= Return  F4= Clear   F5= PrevOut F6= Roll
```

図 110. RODMView Query Field Selection (照会フィールド選択) パネル 4 — EKGVQA4I

次のセクションでは、複合照会機能を使用する例を 2 つ示します。 GMFHS サンプル・ネットワークで行われた定義が使用されています。

複合照会の例 1

最初の例では、非適合状況の集合オブジェクトを検出するために複合照会機能を使用する方法を示しています。そのためには、図 111 で示されているように、パネル EKGVQA1I で「**Class name (クラス名)**」として **GMFHS_Aggregate_Objects_Class** を指定し、「**Object name (オブジェクト名)**」としてパターン・マッチング文字 (*) を入力してください。

```
EKGVQA1I                  Compound Query  A01NV OPER2    10/18/97 12:34:56

RODM name   RODMNAME
User ID . . RODMUSER

Initial query criteria (Specify entity or entities to begin the search with):
  SystemView class name _
  Class name  GMFHS_Aggregate_Objects_Class
  Class ID    _

  Object name *
  Object ID   _ (Hexadecimal value)

Output options:
  Level of field detail . . DATA  (Struct, Data, Hex)
  Level of subfield detail  NONE  (None, Struct, Data, Hex)
  Maximum lines returned    5000
  Display field IDs . . . . N (Y, N) Display extended field info  N (Y, N)

(Use PF8 to further specify query)

CMD==>
F1= Help   F2= End    F3= Return  F4= Clear   F5= PrevOut F6= Roll
           F8= Next                                          F12=PrevCmd
```

図 111. GMFHS_Aggregate_Objects_Class での Compound Query (複合照会) の開始

非適合状況になっているオブジェクトを選択するには、最初の複合照会パネルで **PF8** を押して、2 番目の複合照会パネル EKGVQA2I にスクロールしてください。

図 112 に示すように、 DisplayStatus フィールドが 129 以外の値になるように指定してください。

```
EKGVQA2I                  Query Criteria  A01NV OPER2    10/18/97 12:34:56

Entities from the previous panel should meet the following criteria:
  SystemView field name _
  Field name DisplayStatus
  Operator   <> (=, >, <, <>, <=, >=)
  Value . .  129

Operator between these two criteria AND (And, Or)

Entities from the previous panel should also meet the following criteria:
  SystemView field name _
  Field name _
  Operator   = (=, >, <, <>, <=, >=)
  Value . .  _




(Use PF8 to further specify query)

CMD==>
F1= Help   F2= End    F3= Return  F4= Clear   F5= PrevOut F6= Roll
```

図 112. 非適合 DisplayStatus のエンティティーだけの選択

他の入力フィールドには値が指定されていないため、RODMView はそれらの入力フィールドを無視します。

基準に一致したエンティティーに関して表示されるフィールドを制限することができます。例えば、検出されたエンティティーの DisplayResourceName だけを表示させるには、**PF8** を 2 回押して 4 番目のパネル EKGVQA4I を表示させ、 図 113 に示すように入力フィールドを指定してください。

```
EKGVQA4I            Query Field Selection  A01NV OPER2    10/18/97 12:34:56

Field(s) to display of entity (or entities) found:
  SystemView field name _
  Field name DisplayResourceName
  Field ID   _









CMD==>
F1= Help   F2= End    F3= Return  F4= Clear   F5= PrevOut F6= Roll
```

図 113. DisplayResourceName フィールドのみを表示させるための選択

複合照会の指定が完了した後で、**Enter** を押して照会を実行してください。すべての GMFHS Sample Network 集合オブジェクトが非適合状況になっている場合に

は、出力は図 114 のようになります。

```
EKGVQUEO                    Query Output  A01NV OPER2    10/18/97 12:34:56

                                                            Lines 1 to 17 of 63
------------------------------------------------------------Matching entity ID:
 MyID (OBJECTID)
    (OBJECTID) 00010012457AE0AA
               'DEC'
    (CLASSID)  18
               'GMFHS_Aggregate_Objects_Class'
-   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -
 DisplayResourceName (CHARVAR)
    'DEC'
------------------------------------------------------------Matching entity ID:
 MyID (OBJECTID)
    (OBJECTID) 00010012AE51C8AB
               'BRIDGE01'
    (CLASSID)  18
               'GMFHS_Aggregate_Objects_Class'
-   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -
 DisplayResourceName (CHARVAR)
EKGV0000I Request is successful (0/0)
CMD==>
F1= Help   F2= End    F3= Return              F5= RptFind F6= Roll
```

図 114. Compound Query (複合照会) の例 1 の出力

図 114 に示すように、利用可能な出力は 63 行ありますが、出力パネルで一度に見ることのできる出力は 17 行だけです。出力をスクロールして基準を満たすすべてのエンティティーを表示するには、 PF8 を使用してください。

Compound Query (複合照会) の例 2

2 番目の例では、複合照会機能を使用して、適合状況になっている集合体のオブジェクトのうちで、 (ComposedOfPhysical リンクを介して) 物理的に接続されたすべての非適合状況のオブジェクトを検出する方法を示します。この複合照会の例では、以下の基準が使用されています。

- 開始するオブジェクト (適合状況になっているすべての集合体)
- 探索するフィールド (ComposedOfPhysical リンク)
- リンクの相手側にあるオブジェクトに適用される基準 (非適合状況)

そのためには、 593 ページの図 115 で示されているように、パネル EKGVQA1I で「**Class name (クラス名)**」として GMFHS_Aggregate_Objects_Class を指定し、「**Object name (オブジェクト名)**」としてパターン・マッチング文字 (*) を指定してください。

```
EKGVQA1I                  Compound Query  A01NV OPER2    10/18/97 12:34:56

RODM name   RODMNAME
User ID . . RODMUSER

Initial query criteria (Specify entity or entities to begin the search with):
  SystemView class name _
  Class name  GMFHS_Aggregate_Objects_Class
  Class ID    _

  Object name *
  Object ID   _ (Hexadecimal value)

Output options:
  Level of field detail . . DATA  (Struct, Data, Hex)
  Level of subfield detail  NONE  (None, Struct, Data, Hex)
  Maximum lines returned    5000
  Display field IDs . . . . N (Y, N) Display extended field info  N (Y, N)

(Use PF8 to further specify query)

CMD==>
F1= Help   F2= End    F3= Return  F4= Clear   F5= PrevOut F6= Roll
           F8= Next                                          F12=PrevCmd
```

図 115. GMFHS_Aggregate_Objects_Class での Compound Query (複合照会) の開始

非適合状況になっているオブジェクトのみを選択するには、最初の複合照会パネルで **PF8** を押して、2 番目の複合照会パネル EKGVQA2I を表示させてください。図 116 に示すように、DisplayStatus フィールドの値が 129 になるように指定してください。

```
EKGVQA2I                  Query Criteria  A01NV OPER2    10/18/97 12:34:56

Entities from the previous panel should meet the following criteria:
  SystemView field name _
  Field name DisplayStatus
  Operator   = (=, >, <, <>, <=, >=)
  Value . .  129

Operator between these two criteria AND (And, Or)

Entities from the previous panel should also meet the following criteria:
  SystemView field name _
  Field name _
  Operator   = (=, >, <, <>, <=, >=)
  Value . .  _




(Use PF8 to further specify query)

CMD==>
F1= Help   F2= End    F3= Return  F4= Clear   F5= PrevOut F6= Roll
```

図 116. 適合 DisplayStatus のエンティティーのみの選択

照会を ComposedOfPhysical リンク・フィールドに従って行う必要があり、検出されるオブジェクトの DisplayStatus が非適合になっている必要があることを指定するには、 **PF8** を押して 3 番目の複合照会パネル EKGVQA3I にスクロールしてください。このパネルは 594 ページの図 117 のように記入されています。

```
EKGVQA3I         Query Traversed Criteria  A01NV OPER2    10/18/97 12:34:56

Find entities linked to the following field (leave blank to ignore):
  Traverse SystemView field name _
  Traverse field name ComposedOfPhysical

Entities found in the Traverse field should meet the following criteria:
  SystemView field name _
  Field name DisplayStatus
  Operator   <> (=, >, <, <>, <=, >=)
  Value . .  129

Operator between these two criteria AND (And, Or)

Entities found in the Traverse field should also meet the following criteria:
  SystemView field name _
  Field name _
  Operator   = (=, >, <, <>, <=, >=)
  Value . .  _
(Use PF8 to specify which fields are printed for each entity found)

CMD==>
F1= Help   F2= End    F3= Return  F4= Clear   F5= PrevOut F6= Roll
```

図 117. ComposedOfPhysical リンク・フィールドの探索と DisplayStatus 基準の追加

4 番目のパネル EKGVQA4I を使用して、エンティティーに関して表示される出力を制限することができます。例えば、検出されたエンティティーの DisplayResourceName のみを表示するには、4 番目のパネルに図 118 のように記入してください。

```
EKGVQA4I           Query Field Selection  A01NV OPER2    10/18/97 12:34:56

Field(s) to display of entity (or entities) found:
  SystemView field name _
  Field name DisplayResourceName
  Field ID   _









CMD==>
F1= Help   F2= End    F3= Return  F4= Clear   F5= PrevOut F6= Roll
```

図 118. DisplayResourceName フィールドのみを表示させるための選択

複合照会の指定が完了した後で、**Enter** を押して照会を実行してください。なんらかの集合体ネットワーク・オブジェクトが適合状況になっていて、ComposedOfPhysical リンクに定義されているその子孫オブジェクトが非適合状況になっている場合には、出力は 595 ページの図 119 のようになります。

```
EKGVQUEO                    Query Output  A01NV OPER2    10/18/97 12:34:56

                                                          Lines 1 to 17 of 270
------------------------------------------------------------Matching entity ID:
 MyID (OBJECTID)
    (OBJECTID) 0001000E8D558A23
               'NETVIEW.T46A'
    (CLASSID)  14
               'GMFHS_Managed_Real_Objects_Class'
-   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -
 DisplayResourceName (CHARVAR)
    'T46A'
------------------------------------------------------------Matching entity ID:
 MyID (OBJECTID)
    (OBJECTID) 0001000E55D3D385
               'NETVIEW.T47A'
    (CLASSID)  14
               'GMFHS_Managed_Real_Objects_Class'
-   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -   -
 DisplayResourceName (CHARVAR)
EKGV0000I Request is successful (0/0)
CMD==>
F1= Help   F2= End    F3= Return              F5= RptFind F6= Roll
```

図 119. 照会出力の例 2

オブジェクト探索機能

索引付き (CharVar または IndexList) フィールドでデータが定義されているオブジェクトを検索するには、Locate Objects (オブジェクト探索) 機能を使用してください。

RODMView のメイン・メニューから「**4. Locate Objects (オブジェクト検索)**」を選択してください。 図 120 のような「Locate Objects (オブジェクト検索)」パネルが表示されます。

```
EKGVLOCI                   Locate Objects  A01NV OPER2    10/18/97 12:34:56

RODM name   RODMNAME
User ID . . RODMUSER

SystemView field name _
Field name  _
Field ID    _

Locate datatype CHARVAR (CharVar, INDEXList, INDEXHex)
Locate value    _

Display located entities in detail Y (Y, N)
Maximum lines returned . . . . . 5000









CMD==>
F1= Help   F2= End    F3= Return              F5= PrevOut F6= Roll  F12=PrevCmd
```

図 120. Locate Objects (オブジェクト探索) パネル

「Locate Objects (オブジェクト探索)」パネルを使用することにより、フィールド名とデータ値を指定してオブジェクトを探し出したり、オブジェクト自体を表示するのか、この値をもつ突き止められたオブジェクトの数だけを表示するのかを指定したりできます。

このパネルで指定する値は、索引付きフィールドとして作成されたものでなければなりません。例えば、「CharVar」フィールドと「IndexList」フィールドはともに、共用または共用索引付きとして作成することができます。索引機能および探索機能を使用するには、フィールドが共用索引付きフィールドでなければなりません。

索引付き CharVar フィールドに特定の値が入っているオブジェクトを探し出すには、 **Locate value (探索値)** として通常の文字を入力してください。先行または後続のブランクを含むデータを探し出すには、そのストリングを引用符で囲んでください。

IndexList フィールドに特定の値が入っているオブジェクトを探索するための探索データを指定する方法は 2 通りあります。 **INDEXLIST** を指定した場合には、文字ストリングを入力することができ、その文字ストリングは RODM に渡される前に自動的に AnonymousVar データに変換されます。データ・タイプとして **INDEXHEX** を指定した場合には、「Locate value (探索値)」行に指定するデータは、探索したい AnonymousVar 値を表す偶数桁数の 16 進数でなければなりません。文字データにはブランクを含めることができます。先行または後続のブランクを含めるには、そのストリングを引用符で囲んでください。

注: このデータは、GMFHS データ・モデルの DisplayResourceName フィールドを除き、大文字小文字が区別されます。

DisplayResourceName というフィールドの値が LANMGR.BRIDGE01 になっているすべてのオブジェクトを探し出すためには、図 121 に示すように、パネルに値を入力してください。

```
EKGVLOCI                  Locate Objects  A01NV OPER2    10/18/97 12:34:56

RODM name   RODMNAME
User ID . . RODMUSER

SystemView field name _
Field name  DisplayResourceName
Field ID    _

Locate datatype CHARVAR (CharVar, INDEXList, INDEXHex)
Locate value    LANMGR.BRIDGE01

Display located entities in detail Y (Y, N)
Maximum lines returned . . . . . . 5000









CMD==>
F1= Help   F2= End    F3= Return              F5= PrevOut F6= Roll  F12=PrevCmd
```

図 121. 索引付き CharVar フィールドによるオブジェクトの探索

フィールド・データ・タイプとして CHARVAR が指定されているため、RODMView は、「Locate value (探索値)」フィールドに入力されたデータを文字データとして解釈します。

指定された特性のオブジェクトを RODM が探し出すと、 図 122 に示すようなパネル EKGVQUEO、「Query Output (照会出力)」が表示されます。

```
EKGVQUEO                    Query Output  A01NV OPER2    10/18/97 12:34:56

                                                              Lines 1 to 7 of 7
 Number of objects located: 1

 DisplayResourceName (OBJECTIDLIST)
    (OBJECTID) 0001000ED8AD8723
               'LANMGR.BRIDGE01'
    (CLASSID)  14
               'GMFHS_Managed_Real_Objects_Class'









EKGV0000I Request is successful (0/0)
CMD==>
F1= Help   F2= End    F3= Return              F5= RptFind F6= Roll
```

図 122. オブジェクト探索の出力

図 123 に示す次の例は、「Display located entities in detail (探索されたエンティティーの詳細表示)」入力フィールドで N が指定されて、一致するデータが検出されたエンティティーの数だけが報告されるようになっていることを除き、同じ探索機能を表しています。

```
EKGVLOCI                  Locate Objects  A01NV OPER2    10/18/97 12:34:56

ROM name   RODMNAME
User ID . . RODMUSER

SystemView field name _
Field name  DisplayResourceName
Field ID    _

Locate datatype CHARVAR (CharVar, INDEXList, INDEXHex)
Locate value    LANMGR.BRIDGE01

Display located entities in detail N (Y, N)
Maximum lines returned . . . . . 5000









CMD==>
F1= Help   F2= End    F3= Return              F5= PrevOut F6= Roll  F12=PrevCmd
```

図 123. オブジェクトの詳細ではなくオブジェクトの数を表示するオブジェクトの探索

「Display located entities in detail (探索されたエンティティーの詳細表示)」フィールドで N が指定されているため、図 124 のような出力が表示されます。

```
EKGVQUEO                    Query Output  A01NV OPER2    10/18/97 12:34:56

                                                            Lines 1 to 1 of 1
 Number of objects located: 1









EKGV0000I Request is successful (0/0)
CMD==>
F1= Help   F2= End    F3= Return              F5= RptFind F6= Roll
```

図 124. オブジェクトの詳細が表示されないオブジェクト探索の出力

リンク/リンク解除機能

2 つのオブジェクトのフィールドをリンクまたはリンク解除するには、 Link/Unlink (リンク/リンク解除) 機能を使用してください。

RODMView のメイン・メニューから「**5. Link/Unlink (リンク/リンク解除)**」を選択してください。 図 125 に示すような「Link/Unlink (リンク/リンク解除)」パネルが表示されます。

```
EKGVLNKI                     Link/Unlink  A01NV OPER2    10/18/97 12:34:56

RODM name   RODMNAME                              Link/Unlink . . L (L, U)
User ID . . RODMUSER                              Trigger methods Y (Y, N, G)

Object 1 specification
  Class name   =
  Class ID     =
  Object name  =
  Object ID    = (Hexadecimal value)
  Field name   =
  Field ID     =

Object 2 specification
  Class name   =
  Class ID     =
  Object name  =
  Object ID    = (Hexadecimal value)
  Field name   =
  Field ID     =


CMD==>
F1= Help   F2= End    F3= Return                        F6= Roll  F12=PrevCmd
```

図 125. RODMView リンク・オブジェクト・パネル - EKGVLNKI

「Link/Unlink (リンク/リンク解除)」パネルを使用することにより、リンクまたはリンク解除する 2 つのオブジェクトを指定したり、あるいは、リンクまたはリンク解除の実行時に関連のメソッドを実行するのかどうかを指定することができます。

RODM 内のオブジェクト、およびそのリンクに使用されるフィールドを固有に識別できるように、十分な情報を指定する必要があります。例えば、LinkableStuffClass というクラスに、ObjectLinkList タイプの LinkToPeer というフィールドがあり、また Object1 と Object2 の 2 つのオブジェクトがある場合には、図 126 に示すようなリンク要求情報を入力してそれらのオブジェクトをリンクすることができます。

```
EKGVLNKI                    Link/Unlink  A01NV OPER2    10/18/97 12:34:56

RODM name   RODMNAME                              Link/Unlink . . L (L, U)
User ID . . RODMUSER                             Trigger methods Y (Y, N, G)

Object 1 specification
  Class name  LinkableStuffClass
  Class ID
  Object name Object1
  Object ID    (Hexadecimal value)
  Field name  LinkToPeer
  Field ID

Object 2 specification
  Class name  LinkableStuffClass
  Class ID
  Object name Object2
  Object ID    (Hexadecimal value)
  Field name  LinkToPeer
  Field ID


CMD==>
F1= Help   F2= End    F3= Return                          F6= Roll  F12=PrevCmd
```

図 126. 2 つのオブジェクトの RODMView リンク

「Link/Unlink (リンク/リンク解除)」フィールドを **L** から **U** に変更して、 2 つのオブジェクトをリンク解除することができます。リンク・フィールドに定義されている変更メソッドを含めたくない場合は、「Trigger methods (メソッドの起動)」を **Y** から **N** に変更してください。

注:

1. オブジェクトのリンクは、データ・タイプが ObjectLink または ObjectLinkList のフィールドを介してだけ行うことができます。
2. クラスはリンクまたはリンク解除できません。

この機能から得られる出力は、メッセージ行に表示される戻りコードと理由コードだけです。

GMFHS メソッド DUIFCLRT および DUIFCUAP によるリンク

Link/Unlink (リンク/リンク解除) 機能を使用して、 GMFHS のメソッド DUIFCLRT および DUIFCUAP を実行することができます。メソッド DUIFCLRT は GMFHS の表示可能オブジェクトを GMFHS のリソース・タイプ・オブジェクトにリンクします。メソッド DUIFCLRT の詳細については、 557 ページの『DUIFCLRT: リソース・タイプ・リンク・メソッド』を参照してください。メソッド DUIFCUAP は、

親から子の GMFHS 表示可能オブジェクトへの集約パスを作成します。メソッド DUIFCUAP の詳細については、 560 ページの『DUIFCUAP: 集約パス更新メソッド』を参照してください。集合オブジェクトおよび集合体の詳細については、 44 ページの『GMFHS 集合オブジェクトを定義する』を参照してください。

これらの GMFHS メソッドを実行するには、「Link/Unlink (リンク/リンク解除)」パネルの「Trigger methods (メソッドの起動)」入力フィールドに **G** を入力してください。また、「Link/Unlink (リンク/リンク解除)」入力フィールドに **L** または **U** を指定して、2 つのオブジェクトをリンクするメソッドなのか、リンク解除するメソッドなのかを指定してください。リンクまたはリンク解除の対象になる 2 つのオブジェクトに関するクラスおよびオブジェクトの情報を指定してください。RODMView が、どちらのメソッドを実行する必要があるのかを判別します。どちらかのオブジェクトが GMFHS の Displayable_Objects_Class クラスに属している場合には、メソッド DUIFCLRT (リソース・タイプ・リンク) が起動されます。それ以外の場合には、メソッド DUIFCUAP (集約パス更新) が起動されます。例えば、GMFHS の集合オブジェクト NV6000 を GMFHS の表示リソース・タイプ・オブジェクト DUIXC_RTN_MAN_AGG にリンクする場合には、「Link/Unlink (リンク/リンク解除)」パネルを図 127 のように記入します。

```
EKGVLNKI                    Link/Unlink  A01NV OPER2    10/18/97 12:34:56

RODM name   RODMNAME                              Link/Unlink . . L (L, U)
User ID . . RODMUSER                              Trigger methods G (Y, N, G)

Object 1 specification
  Class name  Display_Resource_Type_Class
  Class ID    =
  Object name DUIXC_RTN_MAN_AGG
  Object ID   = (Hexadecimal value)
  Field name  =
  Field ID    =

Object 2 specification
  Class name  GMFHS_Aggregate_Real_Objects_Class
  Class ID    =
  Object name NV6000
  Object ID   = (Hexadecimal value)
  Field name  =
  Field ID    =


CMD==>
F1= Help   F2= End    F3= Return                          F6= Roll  F12=PrevCmd
```

図 127. RODMView による GMFHS 集合オブジェクトとそのリソース・タイプのリンク

オブジェクトの一方で Displayable_Resource_Type_Class が指定されているため、メソッド DUIFCLRT が実行されます。 オブジェクトの指定順序は意味をもちません。

2 つのオブジェクト間で集約パスを確立するために、一方のオブジェクトを集約親として指定し、他方のオブジェクトを集約子として指定して DUIFCUAP が実行されます。集約子は、集約階層の中で集約親よりも下位にあります。「Trigger methods (メソッドの起動)」入力フィールドが **G** に設定されていて、両方のオブジェクトのクラス指定が GMFHS 表示可能オブジェクト・クラスになっている場合には、RODMView は DUIFCUAP メソッドを実行します。 RODMView は、最初のオブジェクト仕様が集約子であって 2 番目のオブジェクト仕様が集約親であるもの

と想定します。 GMFHS は、集約親オブジェクトが GMFHS_Aggregate_Objects_Class クラスに属していることを必要とします。例えば、GMFHS 管理実オブジェクト NETVIEW.T46A を GMFHS 集合オブジェクト NV6000 の集約子にするには、「Link/Unlink (リンク/リンク解除)」パネルを 図 128 のように記入してください。

```
EKGVLNKI                    Link/Unlink  A01NV OPER2    10/18/97 12:34:56

RODM name   RODMNAME                              Link/Unlink . . L (L, U)
User ID . . RODMUSER                              Trigger methods G (Y, N, G)

Object 1 specification
  Class name  GMFHS_Managed_Real_Objects_Class
  Class ID    =
  Object name NETVIEW.T46A
  Object ID   = (Hexadecimal value)
  Field name  =
  Field ID    =

Object 2 specification
  Class name  GMFHS_Aggregate_Real_Objects_Class
  Class ID    =
  Object name NV6000
  Object ID   = (Hexadecimal value)
  Field name  =
  Field ID    =


CMD==>
F1= Help   F2= End    F3= Return                          F6= Roll  F12=PrevCmd
```

図 128. NETVIEW.T46A と NV6000 の間の集約パスの更新

フィールド変更機能

クラスまたはオブジェクトのフィールドに入っている特定のタイプのデータを変更するには、フィールド変更機能を使用してください。

RODMView のメイン・メニューから「**6. Change field (フィールド変更)**」を選択してください。 602 ページの図 129 に示すような「Change field (フィールド変更)」パネルが表示されます。

```
EKGVCHGI                    Change Field  A01NV OPER2    10/18/97 12:34:56

RODM name   RODMNAME                                  Trigger methods Y (Y, N)
User ID . . RODMUSER

SystemView class name _
Class name  _
Class ID    _

Object name _
Object ID   _ (Hexadecimal value)

SystemView field name _
Field name  _
Field ID    _

Field data type _ (Anon, Ber, Char, Float, INDex, INT, Small, Time)
Field data  _
            _
The following two input fields are used ONLY with the IndexList datatype:
Update type ADD (Add, Del, Replace)  Data is CHARVAR (Anon, CharVar)

CMD==>
F1= Help   F2= End    F3= Return                        F6= Roll  F12=PrevCmd
```

図 129. RODMView Change Field (フィールド変更) パネル － EKGVCHGI

エンティティーの名前または ID とともにフィールドの名前または ID、フィールド・データ・タイプ、およびコピーする新しいデータを指定することにより、エンティティーのフィールドの値を変更することができます。変更を行う前に関連する変更メソッドを起動する必要があるかどうかを指定することもできます。変更できるのは、以下のデータ・タイプのフィールドです。

- AnonymousVar
- BERVar
- CharVar
- Floating
- IndexList
- Integer
- Smallint
- TimeStamp

例えば、変更するクラス、オブジェクト、およびフィールドと、そのフィールドにコピーする新しい値を指定することにより、GMFHS 管理オブジェクトの表示状況 (カラー) を変更することができます。 GMFHS 管理実オブジェクト NETVIEW.T46A の表示状況を 129 に変更するには、 603 ページの図 130 に示すようにパネル EKGVCHGI に記入してください。

```
EKGVCHGI                    Change Field  A01NV OPER2    10/18/97 12:34:56

RODM name   RODMNAME                                  Trigger methods Y (Y, N)
User ID . . RODMUSER

SystemView class name _
Class name  GMFHS_Managed_Real_Objects_Class
Class ID

Object name NETVIEW.T46A
Object ID   _ (Hexadecimal value)

SystemView field name _
Field name  DisplayStatus
Field ID    _

Field data type INTEGER (Anon, Ber, Char, Float, INDex, INT, Small, Time)
Field data  129
            _
The following two input fields are used ONLY with the IndexList datatype:
Update type ADD (Add, Del, Replace)  Data is CHARVAR (Anon, CharVar)

CMD==>
F1= Help   F2= End    F3= Return                        F6= Roll  F12=PrevCmd
```

図 130. RODMView のフィールド変更

注:

1. 「Field data (フィールド・データ)」入力フィールドの最大長は 134 文字です。2 行の入力は、データを RODM に送るときに 1 行に連結されます。
2. パネルの下部にある入力フィールド「Update type (更新タイプ)」および「Data is (データは)」は、IndexList データ・タイプのフィールドだけで使用されます。その他のデータ・タイプでは、これらの入力フィールドを指定しても無視されます。

604 ページの表 225 には、フィールド変更の規則がデータ・タイプごとにリストされています。

表 225. 特定のデータ・タイプ変更の規則

データ・タイプ	規則
AnonymousVar および BERVar	• 入力されたフィールド・データは 16 進数として解釈されます。 • フィールド・データ値は、16 進ストリングが含まれているかどうか検査されます。 16 進ストリングが含まれていない場合には、次のメッセージが表示されます。 `EKGV8052E The Field data value is not a valid hex value` • 16 進データを入力するときには、`X'001122'` などの特殊な表記を使用しないでください。 `001122` のように数値部分だけを入力してください。 • AnonymousVar および BERVar データ・タイプのフィールドには、実際のデータの前に 2 バイト長が含まれています。値を入力するときには、2 バイト長を含めないでください。 RODMView が、データを構文解析した後でこの値を計算します。
CharVar	文字を受け入れます。
Floating	実数を受け入れます。
IndexList	『IndexList フィールドの変更』を参照してください。
Integer	整数を受け入れます。
TimeStamp	• このストリングは、Lillian 秒数を表す 8 バイト (16 桁) の 16 進数値として解釈されます。 • EKG_System クラスの EKG_Name フィールドを `HEX` レベルのサブフィールドの詳しさで照会し、この値の例を調べてください。

IndexList フィールドの変更

IndexList フィールドのエレメントを追加または削除するには、 Change Field (フィールド変更) 機能を使用してください。 IndexList フィールドの例として、ExceptionViewList フィールドがあります。 ExceptionViewList フィールドの値を動的に変更するには、 RODMView の Change Field (フィールド変更) 機能を使用してください。例えば、'TCPIP ' および 'LAN27 ' というビューを集合オブジェクト NV6000 の例外ビューのリストに追加するには、 605 ページの図 131 に示すようにパネル EKGVCHGI に記入してください。

```
EKGVCHGI                    Change Field  A01NV OPER2    10/18/97 12:34:56

RODM name   RODMNAME                                  Trigger methods Y (Y, N)
User ID . . RODMUSER

SystemView class name _
Class name  GMFHS_Aggregate_Objects_Class
Class ID

Object name NV6000
Object ID   _ (Hexadecimal value)

SystemView field name _
Field name  ExceptionViewList
Field ID    _

Field data type INDEXLIST (Anon, Ber, Char, Float, INDex, INT, Small, Time)
Field data  'TCPIP    ' 'LAN27    '
            _
The following two input fields are used ONLY with the IndexList datatype:
Update type ADD (Add, Del, Replace)  Data is CHARVAR (Anon, CharVar)

CMD==>
F1= Help   F2= End    F3= Return                          F6= Roll  F12=PrevCmd
```

図 131. 文字形式による IndexList フィールドへの複数の値の追加

注:

1. このリストに他の値が含まれていない場合にも、2 つのビュー名がリストに追加されます。
2. リスト内に値がすでに存在している場合、その値は重複されません。
3. 複数の入力値はスペースで区切る必要があります。例えば、'TCPIP ' 'LAN27 ' のようにします。
4. 値にスペースが含まれている場合、その値を単一引用符で囲んでください。例えば、 'TCPIP ' のようにします。

パネルに指定したデータによって索引リストの内容を置き換えるためには、「**Update type (更新タイプ)**」入力フィールドを REPLACE に変更してください。

サブフィールド・アクション機能

Subfield Actions (サブフィールド・アクション) 機能は、以下の指定を行うときに使用します。

- サブフィールドのタイプ (Value、Query、Change、Notify、Prev_value、または Timestamp)
- 実行したいアクション (作成、削除、または継承値への復帰)
- そのサブフィールドと関連付けられるフィールド

RODMView のメイン・メニューからオプション 7 の「Subfield Actions (サブフィールド・アクション)」を指定してください。 606 ページの図 132 に示すような「Subfield Actions (サブフィールド・アクション)」パネルが表示されます。

```
EKGVSUBI                Subfield Actions  A01NV OPER2    10/18/97 12:34:56

RODM name   RODMNAME
User ID . . RODMUSER

SystemView class name _
Class name  _
Class ID    _

Object name _
Object ID   _ (Hexadecimal value)

SystemView field name _
Field name  _
Field ID    _

Subfield type _ (Value, Query, Change, Notify, Prev_value, Time)
Action  . . . _ (Create, Delete, Revert)




CMD==>
F1= Help   F2= End    F3= Return                          F6= Roll  F12=PrevCmd
```

図 132. RODMView Subfield Actions (サブフィールド・アクション) パネル − EKGVSUBI

アクションによっては、特定のサブフィールドでは実行できないものがあります。例えば、RODM では、ユーザーが Timestamp サブフィールドを継承された値に復帰させることはできません。

サブフィールドは、クラスのフィールド上だけで作成または削除することができます。例えば、ExtrememlyImportantClass クラスにある VeryImportantField というフィールド上に notify サブフィールドを作成したい場合には、図 133 に示すように「Subfield Action (サブフィールド・アクション)」パネルに情報を入力してください。

```
EKGVSUBI                Subfield Actions  A01NV OPER2    10/18/97 12:34:56

RODM name   RODMNAME
User ID . . RODMUSER

SystemView class name _
Class name  ExtremelyImportantClass
Class ID    _

Object name _
Object ID   _ (Hexadecimal value)

SystemView field name _
Field name  VeryImportantField
Field ID    _

Subfield type notify (Value, Query, Change, Notify, Prev_value, Time)
Action  . . . create (Create, Delete, Revert)




CMD==>
F1= Help   F2= End    F3= Return                          F6= Roll  F12=PrevCmd
```

図 133. RODMView での Notify サブフィールドの作成

注:

1. RODMView を使用して notify サブフィールドの値を変更することはできません。このサブフィールドのタイプは MethodSpec です。
2. サブフィールドはオブジェクトの親クラス上に作成しなければなりません。サブフィールドの存在および初期内容は、クラスからオブジェクトに継承されます。Notify サブフィールドの場合には、ヌル値が継承されます。
3. サブフィールドが存在しているクラスがクラスまたはオブジェクト子である場合には、サブフィールドをそのクラスから削除することはできません。
4. サブフィールドの削除は、それが定義されたクラスから行う必要があります。
5. Notify、Prev_value、および Timestamp サブフィールドは継承された値に復帰させることはできません。

作成アクション機能

クラス、オブジェクト、またはクラス上のフィールドを作成するには、 Create Actions (作成アクション) 機能を使用してください。どの場合にも、作業を行うための*親クラス* と呼ばれるクラスを指定しなければなりません。

RODMView のメイン・メニューから「**8. Create Actions (作成アクション)**」を選択してください。 図 134 に示すような「Create Actions (作成アクション)」パネルが表示されます。

```
EKGVCREI                  Create Actions  A01NV OPER2    10/18/97 12:34:56

RODM name   RODMNAME
User ID . . RODMUSER

Parent Class information
  Class name  _
  Class ID    _

Child Class to create (optional)
  Child class _

OR Object to create (optional)
  Object name _

OR Field to create on the Parent Class (optional)
  Field name  _
  Field data type _
  Field inherits  _ (PUblic, PRivate, Indexed)



CMD==>
F1= Help   F2= End    F3= Return                          F6= Roll  F12=PrevCmd
```

図 134. RODMView Create Actions (作成アクション) パネル — EKGVCREI

608 ページの表 226 には、子クラス、オブジェクト、またはフィールドを作成するために指定する必要のある情報が示されています。

表 226. エンティティーを作成するための指定事項

作成対象	以下の入力フィールドだけに記入します。
子クラス	「Class name (クラス名)」または「Class ID (クラス ID)」 「Child Class name (子クラス名)」
オブジェクト	「Class name (クラス名)」または「Class ID (クラス ID)」 「Object name (オブジェクト名)」
フィールド	「Class name (クラス名)」または「Class ID (クラス ID)」 「Field name (フィールド名)」または「Field ID (フィールド ID)」 フィールド・データ・タイプ 「Field inherits (フィールド継承)」

RODMView は、RODM がパネルで指定されたとおりにエンティティーを作成することを要求します。ユーザーが作成不可能なものを作成しようとしていること (例えば、オブジェクト上でのフィールドの作成) を RODM が検出すると、メッセージが表示されます。

Object3 という名前の CreatableStuffClass にオブジェクトを作成したい場合には、図 135 に示すように「Create Actions (作成アクション)」パネルに情報を入力してください。

```
EKGVCREI                  Create Actions  A01NV OPER2    10/18/97 12:34:56

RODM name   RODMNAME
User ID . . RODMUSER

Parent Class information
  Class name  CreatableStuffClass
  Class ID    _

Child Class to create (optional)
  Child class _

OR Object to create (optional)
  Object name Object3

OR Field to create on the Parent Class (optional)
  Field name  _
  Field data type _
  Field inherits  _ (PUblic, PRivate, Indexed)



CMD==>
F1= Help   F2= End    F3= Return                          F6= Roll  F12=PrevCmd
```

図 135. RODMView によるオブジェクトの作成

CreatableStuffClass クラスに NewCharVarField という名前の専用フィールドを作成したい場合には、図 136 に示すように「Create Actions (作成アクション)」パネルに情報を入力してください。

「Object name (オブジェクト名)」フィールドに値が指定されていないことにご注意ください。

```
EKGVCREI                  Create Actions  A01NV OPER2    10/18/97 12:34:56

RODM name   RODMNAME
User ID . . RODMUSER

Parent Class information
  Class name  CreatableStuffClass
  Class ID    _

Child Class to create (optional)
  Child class _

OR Object to create (optional)
  Object name _

OR Field to create on the Parent Class (optional)
  Field name  NewCharVarField
  Field data type charvar
  Field inherits  public (PUblic, PRivate, Indexed)



CMD==>
F1= Help   F2= End    F3= Return                          F6= Roll  F12=PrevCmd
```

図 136. RODMView によるフィールドの作成

「Field data type (フィールド・データ・タイプ)」および「Field inherits (フィールド継承)」入力フィールド内のデータは、フィールドを作成するためにフィールド名が作成されていなければ無視されます。

図 136 で示されている例の場合、この要求に対する出力はメッセージ行に表示される戻りコードと理由コードのみです。

削除アクション機能

クラス、オブジェクト、またはクラス上のフィールドを削除するには、 Delete Actions (削除アクション) 機能を使用してください。

RODMView のメイン・メニューから「**9. Delete Actions (削除アクション)**」を選択してください。 607 ページの図 134 に示すような「Delete Actions (削除アクション)」パネルが表示されます。

```
EKGVDELI                  Delete Actions  A01NV OPER2    10/18/97 12:34:56

RODM name   RODMNAME
User ID . . RODMUSER

Class information
  Class name   =
  Class ID     =

Object to delete
  Object name  =
  Object ID    = (Hexadecimal value)

Field to delete from a class
  Field name   =
  Field ID     =









CMD==>
F1= Help   F2= End    F3= Return                          F6= Roll  F12=PrevCmd
```

図 137. RODMView Delete Actions (削除アクション) パネル － EKGVDELI

表 227 には、子クラス、オブジェクト、またはフィールドを削除するために指定する必要のある情報が示されています。

表 227. エンティティーを削除するための指定事項

削除対象	以下の入力フィールドだけに記入します。
クラス	「Class name (クラス名)」または「Class ID (クラス ID)」
オブジェクト	「Class name (クラス名)」、「Class ID (クラス ID)」、または「Object name (オブジェクト名)」
オブジェクト	オブジェクト ID
フィールド	「Class name (クラス名)」、「Class ID (クラス ID)」、または「Field ID (フィールド ID)」

`DeletableStuffClass` クラスから DeletableObject という名前のオブジェクトを削除したい場合には、611 ページの図 138 に示すように「Delete Actions (削除アクション)」パネルに情報を入力してください。

```
EKGVDELI                  Delete Actions  A01NV OPER2    10/18/97 12:34:56

RODM name   RODMNAME
User ID . . RODMUSER

Class information
  Class name  DeletableStuffClass
  Class ID    _
              =

Object to delete
  Object name DeletableObject
  Object ID   _ (Hexadecimal value)
              =

Field to delete from a class
  Field name  _
              =
  Field ID    _
              =









CMD==>
F1= Help   F2= End    F3= Return                          F6= Roll  F12=PrevCmd
```

図 138. RODMView によるクラスからのフィールドの削除

RODMView が削除要求を送信する前に、削除要求を確認するようにユーザーに対してプロンプトが出されます。

注:

1. クラスを削除する場合、そのクラスにクラス子またはオブジェクト子があってはなりません。
2. オブジェクトを削除する場合、そのオブジェクトに別のオブジェクトとのリンクが含まれていてはなりません。
3. クラスからフィールドを削除する場合、そのクラスにクラス子またはオブジェクト子があってはなりません。
4. フィールドをオブジェクトから直接削除することはできません。フィールドは、親クラスから削除しなければなりません。

メソッド・アクション機能

以下のことを行うときには、Method Actions (メソッド・アクション) 機能を使用してください。

- あるメソッドをオブジェクト独立またはオブジェクト特有 (名前付き) メソッドとして起動する。
- メソッドをインストールする。
- メソッドを削除する。
- メソッド・コードを置き換える。

RODMView のメイン・メニューから「**10. Method Actions (メソッド・アクション)**」を選択してください。 612 ページの図 139 に示すような「Method Actions (メソッド・アクション)」パネルが表示されます。

```
EKGVMETI                  Method Actions  A01NV OPER2    10/18/97 12:34:56

RODM name   RODMNAME
User ID . . RODMUSER

Method name
Method type   (Named, Object independent)

Action  . . TRIGGER   (Trigger, Install, Delete, Replace)

Additional information for Named Methods only
Class name
Class ID

Object name
Object ID     (Hexadecimal value)

Field name
Field ID



CMD==>
F1= Help   F2= End    F3= Return                          F6= Roll  F12=PrevCmd
```

図 139. RODMView Method Actions (メソッド・アクション) パネル ― EKGVMETI

RODMView を使用すると、オブジェクト独立メソッドは短命パラメーターなしで実行されます。ただし名前付きメソッドは、ユーザーが指定した (MethodSpec タイプの) フィールドで定義された、短命パラメーターを受け取ります。

例えば、MethodSpecField という名前の MethodSpec タイプのフィールドが UsefulClass クラスに存在していて、 MethodSpecField の値に USFLMETH というメソッドが含まれているものとします。このメソッドを実行するには、図 140 に示すように「Method Actions (メソッド・アクション)」パネルに情報を入力してください。

```
EKGVMETI                  Method Actions  A01NV OPER2    10/18/97 12:34:56

RODM name   RODMNAME
User ID . . RODMUSER

Method name usflmeth
Method type named   (Named, Object independent)

Action  . . TRIGGER   (Trigger, Install, Delete, Replace)

Additional information for Named Methods only
Class name  UsefulClass
Class ID

Object name
Object ID     (Hexadecimal value)

Field name  MethodSpecField
Field ID



CMD==>
F1= Help   F2= End    F3= Return                          F6= Roll  F12=PrevCmd
```

図 140. RODMView による名前付きメソッドの起動

MethodSpecField フィールドで定義された短命パラメーターにより、メソッド USFLMETH が実行されます。

メソッドの実行が終了すると、そのメソッド自体から戻された戻りコードと理由コードが RODMView によってメッセージ行に表示されます。この説明で使用している例の結果は、図 141 のパネルのようになります。

```
EKGVMETI                  Method Actions  A01NV OPER2    10/18/97 12:34:56

RODM name   RODMNAME
User ID . . RODMUSER

Method name USFLMETH
Method type NAMED  (Named, Object independent)

Action  . . TRIGGER   (Trigger, Install, Delete, Replace)

Additional information for Named Methods only
Class name  UsefulClass
Class ID    =

Object name =
Object ID   = (Hexadecimal value)

Field name  MethodSpecField
Field ID    =


EKGV8037E RODM return code/reason code is (8/60000)
CMD==>
F1= Help   F2= End    F3= Return                          F6= Roll  F12=PrevCmd
```

図 141. 起動されたメソッドから戻される RODMView の戻りコードと理由コード

前の例で起動されたメソッドはユーザー作成のメソッドです。このメソッドは、完了した後で戻りコード/理由コードの組み合わせ 8/60000 を戻します。この組み合わせは特定の RODMView メッセージには変換されません。したがって、RODMView は次のようなメッセージを表示します。

```
EKGV8037E RODM return code/reason code is (return_code/reason_code)
```

注: 612 ページの図 140 のメソッド名は小文字で入力されていますが、 図 141 で RODMView パネルが更新されると、メソッド名が大文字に変換されることにご注意ください。 RODM で定義されたヌル・メソッド NullMeth の名前には大文字と小文字が含まれていますが、 RODM 内にコードとして存在するすべてのメソッドの名前は、大文字になっていなければなりません。 RODMView がメソッド名を自動的に大文字に変換します。

RODM アンロード機能

RODM アンロード機能による RODM クラス構造の照会は、深さ優先で行われます。各クラスごとに、クラスとともにその固有なフィールドを作成するために RODM 高水準構文ステートメントが書かれます。クラス・レベルの作成ステートメントは、CLASSES ファイルに書き込まれます。いずれかのクラス・フィールドにローカル定義の値が含まれている場合、その値は CLASSVAL ファイルに書き込まれます。

RODM アンロード機能は、システム・クラス (UniversalClass およびすべての EKG*xxxx* クラス) のシステム定義フィールドの値をアンロードしません。ユーザー定義フィールドを検出すると、RODM アンロード機能は、フィールドを作成するためのプリミティブ・ステートメントを作成し、ヌルでない値が現在存在している場合にはフィールドに値を割り当てるためのプリミティブ・ステートメントを作成します。

クラスのアンロード中に検査が行われ、そのクラスにオブジェクト子があるかどうかが調べられます。各オブジェクト子が順番に検査され、それを作成するための OBJECTS ファイルに RODM 低水準プリミティブ・ステートメントが書き込まれます。ローカル値が入っているフィールド内のすべてのデータは、OBJVAL ファイルに書き込まれます。

アンロードされたデータ・セットが再び正しくロードされるようにするために、これらのデータ・セットを RODM ロード機能 EKGIN3 ステートメント内で次の順序で連結する必要があります。

1. CLASSES
2. OBJECTS
3. CLASSVAL
4. OBJECTVAL
5. LINKS

このような順序にすることにより、サブフィールドに含まれているデータが、ロードされていないものを参照することがなくなります。

サンプル EKGKUJCL で説明されているデータ・セット・スキーマをすると、 JCL を実行する RODM ロード機能の EKGIN1 DD 連結は図 142 のようになります。

```
//EKGIN1     DD DSN=EKG.RODMUNLD.CLASSES,DISP=SHR
//            DD DSN=EKG.RODMUNLD.OBJECTS,DISP=SHR
//            DD DSN=EKG.RODMUNLD.CLASSVAL,DISP=SHR
//            DD DSN=EKG.RODMUNLD.OBJVAL,DISP=SHR
//            DD DSN=EKG.RODMUNLD.LINKS,DISP=SHR
```

図 142. EKGIN1 用のサンプル JCL

データ・タイプ FieldID および Anonymous(*N*) は、RODM アンロード機能を使用してアンロードすることはできません。

RODM アンロード機能の操作は、RODM データが静的で変更されないことを前提として行われます。 RODM アンロード機能の実行中には、RODM データが変更される可能性があります。このような場合、アンロードされたデータ・セットには、現行の RODM データと矛盾するデータが含まれることがあります。したがって、RODM アンロード機能の実行は、RODM の活動が活発でないときに行います。

RODM アンロード機能の開始

RODM アンロード機能を開始するには、ジョブ EKGKUJCL を実行依頼してください。

RODM アンロード機能のカスタマイズ

このセクションには、RODM アンロード機能のカスタマイズに必要な情報が記載されています。

1. EKGKUCDS ジョブをカスタマイズしてください。

   EKGKUCDS ジョブは RODM アンロード機能用の出力データ・セットを割り振ります。 NETVIEW.V5R3M0.CNMSAMP (EKGKUCDS) ジョブを編集して、出力データ・セットの場所を示すようにしてください。
2. EKGKUCDS を実行して、RODM アンロード機能の出力データ・セットを割り振ってください。
3. EKGKUJCL ジョブを修正してください。

   インストール先の要件に合わせてパラメーターを修正してください。このジョブは NETVIEW.V5R3M0.CNMSAMP データ・セットに入っています。

RODM アンロード機能は JCL によって実行されます。入力パラメーターは、JCL の SYSIN DD ファイルで指定されたファイルに入れて RODM アンロード機能に渡されます。 図 143 には、サンプル JCL の一部が示されています。単純化のために、SYSIN DD ファイルは JCL にインライン化されています。

```
 ...
//SYSIN     DD *
  RODM=
  CLASS=
  OBJECT=
  DEPTH=
  REPORTONLY=
  WRITEMODE=
  WHITESPACE=
 ...
```

図 143. JCL のサンプル SYSIN DD ファイル

表 228 には、この SYSIN DD のパラメーターが説明されています。

表 228. SYSIN DD パラメーターの説明

パラメーター	説明
RODM	アンロードする RODM の名前を指定します。通常は、RODM の開始に使用する z/OS プロシージャーと同じです。
CLASS	• アンロード・プロセスを開始させるクラスを指定します。 • ブランクのままにすると、UniversalClass が開始点になります。 • 各行に 1 つのクラスを指定して複数の行でこのパラメーターを繰り返すことにより、複数のクラスを指定することができます。 • このパラメーターでは、大文字小文字が区別されます。

表 228. SYSIN DD パラメーターの説明 (続き)

パラメーター	説明
OBJECT	• アンロード対象の特定のオブジェクトを指定します。 • 各行に 1 つのオブジェクトを指定して複数の行でこのパラメーターを繰り返すことにより、複数のオブジェクトを指定することができます。 • ブランクのままにした場合、または指定を省略した場合には、すべてのオブジェクトがアンロードされます。 • このパラメーターでは、大文字小文字が区別されます。
DEPTH	• ALL または ONE を指定します。 • DEPTH=ALL を指定すると、CLASS= パラメーターで指定されたクラスと、その子孫にあたるすべてのクラスがアンロードされます。 • DEPTH=ONE を指定すると、CLASS= パラメーターで指定された個々のクラスだけがアンロードされます。
REPORTONLY	• YES または NO を指定することができます。 • REPORTONLY=YES を指定すると、定義されたすべてのクラス、オブジェクト、フィールド、およびリンクの要約報告書が作成されますが、 RODM ロード機能互換の出力が実際に作成されることはありません。これは、RODM の現行の容量情報を抽出するために使用すると便利です。 • REPORTONLY=NO を指定すると、要約報告書とともに RODM ロード機能互換の出力が作成されます。
WRITEMODE	• APPEND または OVERWRITE を指定することができます。 • WRITEMODE=APPEND を指定すると、開始 JCL で指定されたデータ・セットの終わりに、生成されたすべての出力が追加されます。 • WRITEMODE=OVERWRITE を指定すると、それまでデータ・セット内に存在していたすべてのデータが破棄され、RODM アンロード機能によって作成された新しい出力がその代わりに書き込まれます。
WHITESPACE	• これにより、RODM ロード機能互換出力に組み込むホワイト・スペース (ブランク行) のレベルが指定されます。 • LOW または HIGH を指定することができます。WHITESPACE=HIGH を指定すると、最も読みやすい出力が得られ、 WHITESPACE=LOW を指定すると、出力の合計行数が約半分になります。 • 出力の実際のデータ内容は、LOW を指定した場合にも HIGH を指定した場合にも同じです。

この JCL では 5 つの出力データ・セットが指定されています。出力データ・セットとその内容は次のとおりです。

CLASSES クラス構造作成用の高水準構文が含まれます

CLASSVAL クラス・サブフィールドの作成用および値設定用のプリミティブ・ステートメントが含まれます。

OBJECTS	オブジェクト作成用のプリミティブ・ステートメントが含まれます。
OBJVAL	オブジェクト・サブフィールドの値設定用のプリミティブ・ステートメントが含まれます。
LINKS	リンク・プリミティブ・ステートメントが含まれています。

RODM アンロード機能は、JCL からデータ・セットの DCB 指定を読み取り、アンロード機能自体を修正します。サンプルに含まれている DCB 指定をそのまま使用してください。 RODM アンロード機能は、80 文字を超える幅の DCB が指定されている場合にも、常に最大 80 文字の幅で出力を作成します。

RODM アンロード機能は、EKGKUJCL ジョブを実行することによって開始します。

RODM アンロード機能の実行

RODM アンロード機能を使用して、あるバージョンの RODM から別のバージョンにマイグレーションすることができます。これは、既存の RODM をアンロードし、RODM アンロード機能からの出力を指定して新しいバージョンの RODM をロードすることによって行います。 RODM を完全にアンロードするには、図 144 で示すように EKGKUJCL ジョブの SYSIN パラメーターを変更して、このジョブを実行してください。 OBJECT= パラメーターがサンプル JCL から削除されていることにご注意ください。

```
RODM=(rodmname)
CLASS=UniversalClass
DEPTH=All
REPORTONLY=No
WRITEMODE=Overwrite
WHITESPACE=Low
```

図 144. RODM を完全にアンロードするための EKGKUJCL SYSIN パラメーター

GMFHS データ・モデル内のネットワーク・モニター可能 (実および集合体) リソースを表すすべてのオブジェクトをアンロードするには、図 145 に示すように EKGUJCL に対する SYSIN パラメーターを変更します。

```
RODM=(rodmname)
CLASS=GMFHS_Monitorable_Objects_Class
DEPTH=All
REPORTONLY=No
WRITEMODE=Overwrite
WHITESPACE=Low
```

図 145. ネットワーク・モニター可能オブジェクトをアンロードするための EKGKUJCL SYSIN パラメーター

特定のオブジェクトのクラスが不明な場合にそのオブジェクトに関する RODM 定義を入手するには、618 ページの図 146 に示すように EKGKUJCL ジョブの SYSIN パラメーターを変更してください。

```
RODM=(rodmname)
CLASS=UniversalClass
OBJECT=DesiredObject
DEPTH=All
REPORTONLY=No
WRITEMODE=Overwrite
WHITESPACE=High
```

図 146. クラスが不明なときにオブジェクトをアンロードするための EKGKUJCL SYSIN パラメーター

オブジェクトが定義されているクラスが分かっているときには、そのクラスを直接指定すると処理時間が節約できます。 図 147 に示すように CLASS=、OBJECT=、および DEPTH= パラメーターを設定してください。

```
RODM=(rodmname)
CLASS=SpecificClass
OBJECT=DesiredObject
DEPTH=One
REPORTONLY=No
WRITEMODE=Overwrite
WHITESPACE=High
```

図 147. クラスが分かっているときにオブジェクトをアンロードするための EKGKUJCL SYSIN パラメーター

特定の 2 つのクラスだけのすべてのオブジェクトに関する RODM 定義を入手するには、 図 148 に示すように EKGKUJCL ジョブのパラメーターを変更してください。

```
RODM=(rodmname)
CLASS=SpecificClass1
CLASS=SpecificClass2
DEPTH=One
REPORTONLY=No
WRITEMODE=Overwrite
WHITESPACE=Low
```

図 148. 2 つのクラスのオブジェクト定義を判別するための EKGKUJCL SYSIN パラメーター

FLCARODM

このセクションでは、FLCARODM の使用方法を説明します。以下のトピックについて説明します。

- ステム構築ルーチンの使用
- FLCARODM コマンド
- FLCARODM 関数
- 結果ステム
- オブジェクト・データ・ストリーム

概要

FLCARODM は、RODM ユーザー・アプリケーション・プログラミング・インターフェース (UAPI) に、REXX インターフェースを提供します。 FLCARODM は、1 回の呼び出しで 1 つ以上のオブジェクト上で複数の操作を実行し、 RODM UAPI

使用時の複雑さの多くをなくします。この高速インターフェースを使用して、RODM 中のオブジェクトを作成、更新、照会、検索、および削除します。

FLCARODM には、次の 2 つの使用法があります。
- 低レベル・データ・ストリームを使用してデータと操作を指定します。詳細については、662 ページの『オブジェクト・データ・ストリームの詳細』を参照してください。
- NetView が提供するステム構築サブルーチンを使用して、REXX ステム変数を作成します。

ステム構築サブルーチン

このセクションでは、REXX ステム変数で、REXX オブジェクト・データ・ストリームを作成するために提供されるサブルーチンを説明します。これらのサブルーチンは**ステム構築サブルーチン**と呼ばれ、 FLCARODM コマンドを使用して FLCARODM に渡される REXX ステム変数の内容を作成します。

ステム構築サブルーチンは、サンプル FLCSSTEM で提供されます。これらのサブルーチンは、FLCARODM で使用される REXX ステム変数を操作します。これらのサブルーチンが操作するステム変数は以下の 3 つです。
- FLCARODM への入力として使用される RodmStem
- FLCARODM からの出力を保持するのに使用される RodmResult
- RodmResult から取り出される照会済み情報を保持するのに使用される QueryStem

Retcode という変数もあり、これはエラーが発生したかどうかを示すため、すべてのサブルーチンが使用します。 Retcode 変数の非ゼロ値は、処理が停止することを示します。

FLCARODM は、クラス、オブジェクト、およびフィールド ID を、入力ステム変数でサポートします。名前ではなく数値 ID を指定するには、ID の先頭に # を付けます。例えば、オブジェクトのクラスが 12 であると分かっている場合、入力ステム変数のエレメントを `input.x = '#12'` として指定できます。

AddAttr サブルーチン

AddAttr サブルーチンは、現行のオブジェクト上の新規または既存のフィールドを指定するのに使用します。

仕様:

```
call AddAttr fieldname fieldtype fieldvalue
```

オペランド記述: ここで、

fieldname
フィールドの名前を表します。

fieldtype
フィールドのデータ・タイプ

fieldvalue
フィールドの新規または変更済みの値。

使用上の注意:

- AddAttr は、BUILD および UPDATE 関数と共に使用します。
- AddAttr は、Addlink の前に指定することが必要です。

例: サンプル FLCSX7 からの次のコードは、タイプが Integer で、 InActive の値を持つ、DispStat という名前のフィールドを作成する、 AddAttr サブルーチンを呼び出します。

```
call AddAttr DispStat Integer InActive
```

注: DispStat は、次の割り当てステートメントを使用してサンプル FLCSSTEM で定義される、 DisplayStatus の短縮されたバージョンです。

```
DispStat  = 'DisplayStatus'
```

AddAttrForQuery サブルーチン

AddAttrForQuery サブルーチンは、QUERY 機能を使用して照会されるフィールド、または XREF=1STFIELD パラメーターを使用して関数が指定される場合の最初のフィールドの名前の、どちらかを指定するのに使用します。

仕様:

```
call AddAttrForQuery 'fieldname'
```

オペランド記述: ここで、

fieldname
照会するフィールドの名前、または XREF=1STFIELD パラメーターに参照されるフィールドの名前です。

使用上の注意:

- AddAttrForQuery サブルーチンは、QUERY 関数、または XREF=1STFIELD パラメーターで指定される場合は次の関数と一緒に使用します。
  - DELINKA
  - DELOBJ
  - QUERY
  - UPDATE

例: サンプル FLCSXS02 からの次のコードは、AddAttrForQuery サブルーチンを呼び出し、照会される RAgeClass の RealAgent オブジェクト上で 4 つのフィールドを指定します。

```
call StartObject RAgeClass RealAgent
call AddAttrForQuery MyName
call AddAttrForQuery DispName
call AddAttrForQuery RealAgeNam
call AddAttrForQuery RealSerNam
call MakeRODMCall 'QUERY'
```

サンプル FLCSX19 からの次のコードは、AddAttrForQuery サブルーチンを呼び出し、除去されるオブジェクト・リンクを識別するのに使用される ALmnClass クラスの Demo_Lan オブジェクトで、 2 つのフィールドを指定します。

```
call StartObject ALnmClass 'Demo_Lan'

call AddAttrForQuery  Member
call AddAttrForQuery  PhyConn
call MakeRODMCall 'DELINKA' 'XREF=1STFIELD'
```

AddAttrForQuery Member は、 Member フィールドが指定するすべてのオブジェクトが識別されることを指定し、 AddAttrForQueryPhyConn は、 PhyicalConPP フィールドが指定するすべてのリンクが除去されることを指定します。

AddLink サブルーチン

AddLink サブルーチンは、リンク先のフィールドを指定するのに使用します。フィールドは、次のデータ・タイプのうちいずれかであるべきです。

- ObjectLink
- ObjectLinkList
- ObjectIdList

仕様:

```
call AddLink 'linkfldname' 'classofobj' 'nameofobj' 'fldofobj'
```

オペランド記述: ここで、

linkfldname
リンク先のフィールドの名前。

classofobj
リンク先のオブジェクトのクラス。

nameofobj
リンク先のオブジェクトの名前。

fldofobj
リンク先のオブジェクト上のフィールド。

使用上の注意:

- AddAttr サブルーチンへの呼び出しは、 AddLink への呼び出しが指定されるより前に指定することが必要です。

例: サンプル FLCSX11 からの次のコードは、AddLink サブルーチンを使用して、Bridge_1 オブジェクトの PhysicalConnPP フィールド、および Segment_1 と Segment_2 オブジェクトの PhysicalConnPP フィールドを指定します。 DELINKAB 関数は、PhysicalConnPP フィールドが定義するリンクを除去します。

```
call StartObject ABrgClass 'Bridge_1'

call AddLink PhyConn ASegClass 'Segment_1' PhyConn
call AddLink PhyConn ASegClass 'Segment_2' PhyConn

call MakeRODMCall 'DELINKAB'
```

AddLinkForDelete サブルーチン

AddLinkForDelete サブルーチンは、指定されたオブジェクト上でリンクを指定するのに使用します。

仕様:

```
call AddLinkForDelete fldname
```

オペランド記述: ここで、

fldname
削除されるリンクを定義する、指定済みオブジェクト上のフィールドの名前です。

例: サンプル FLCSX10 からの次のコードは、Bridge_1 という名前の ABrgClass のオブジェクト上で、 PhysicalConnPP を指定する AddLinkForDelete サブルーチンを呼び出します。 DELINKA 関数は、PhysicalConnPP フィールドが定義するリンクを除去します。

```
call StartObject ABrgClass 'Bridge_1'

call AddLinkForDelete PhyConn
call MakeRODMCall 'DELINKA'
```

CheckChildrenUpdate サブルーチン

CheckChildrenUpdate サブルーチンは、UPDATE または DELINKA 関数が、CHILDREN=ONLY パラメーターを使って指定される場合に、受け入れ可能な戻りコードを RodmResult ステム変数から除去するのに使用します。

受け入れ可能な戻りコードは、次のいずれかを示しています。
- 集合オブジェクトが存在していない。
- 子オブジェクトが存在していない。
- 子オブジェクトに、指定されたフィールドが存在していない。

受け入れ不能な戻りコードの場合は次のとおりです。
- メッセージ FLC070E が発行される。
- 戻りコードがログに書き込まれる。
- Retcode ステム変数が 16 に設定される。

仕様:

```
call CheckChildrenUpdate
```

- このサブルーチンは、CHILDREN=ONLY パラメーターを使って UPDATE および DELINKA 関数を指定する場合のみ使用します。他の関数とパラメーターの組み合わせはサポートされません。

CheckDelinkResponse サブルーチン

CheckDeLinkResponse サブルーチンは、DELOBJ または DELINKA 関数が指定される場合に、受け入れ可能な戻りコードを RodmResult ステム変数から除去するのに使用します。

受け入れ可能な戻りコードは、次のいずれかを示しています。
- 集合オブジェクトが存在していない。
- 子オブジェクトが存在していない。
- 子オブジェクトに、指定されたフィールドが存在していない。

受け入れ不能な戻りコードの場合は次のとおりです。
- メッセージ FLC070E が発行される。

- 戻りコードがログに書き込まれる。
- Retcode ステム変数が 16 に設定される。

仕様:

```
call CheckDelinkResponse
```

使用上の注意:

- このサブルーチンは、DELOBJ および DELINKA 関数を指定する場合のみ使用します。他の関数はサポートされません。

InitRODMConstants サブルーチン

InitRODMConstants サブルーチンは、サンプル FLCSSTEM で指定される定数を初期化するのに使用します。

仕様:

```
call InitRODMConstants
```

使用上の注意:

- どの変数が使用できるかを確認するため、コードを読む必要があります。

InitRODMStem サブルーチン

InitRODMStem サブルーチンは、RODMStem 変数を初期化するために使用します。

仕様:

```
call InitRODMStem
```

使用上の注意:

- InitRODMStem は、初めて FLCSSTEM を使用するときに指定します。その後は MakeRODMCall サブルーチンが InitRODMStem を呼び出すので、InitRODMStem への後続の呼び出しは必要ありません。

MakeRODMCall サブルーチン

MakeRODMCall サブルーチンは、RODMStem 変数を入力として使用して、FLCARODM コマンドを発行するのに使用します。

仕様:

```
call MakeRODMCall function functparm1 functparm2
```

オペランド記述: ここで、

function
実行される関数を指定します。詳細については、631 ページの『FLCARODM 関数』を参照してください。

functparm1
最初の関数パラメーターを指定します。

functparm2
2 番目の関数パラメーターを指定します。

例: サンプル FLCSXF1 からの次のコードは、 XREF および FILTER パラメーターを使って QUERY コマンドを呼び出します。

```
call MakeRODMCall 'QUERY' 'XREF=2.9.3.2.7.42' 'FILTER=1STFIELD'
```

SetIndexList サブルーチン

SetIndexList サブルーチンは、タイプ IndexList のフィールドの値を更新するのに使用します。

仕様:

```
call SetIndexList fieldvalue fieldname
```

オペランド記述: ここで、

fieldvalue
フィールドの値を指定します。

fieldname
フィールドの名前を指定します。

使用上の注意:

- SetIndexList サブルーチンは、タイプ IndexList のフィールドのみの値を更新するのに使用します。
- SetIndexList 関数の使用時は、注意を要します。これは、フィールドの値が上書きされ、前の値が回復できないためです。

例: サンプル FLCSX22 からの次のコードは、SetIndexList サブルーチンを呼び出し、 Demo_Lan オブジェクトで ExceptionViewList フィールドを変更します。

```
call StartObject ALnmClass 'Demo_Lan'

my_String = 'testing'
call SetIndexList my_String ExceptionViewList

call MakeRODMCall 'UPDATE'
```

StartObject サブルーチン

StartObject サブルーチンは、新規または既存のオブジェクトを指定するのに使用します。後続のサブルーチン指定 (例えば AddAttr) は、別のオブジェクトが StartObject によって指定されるか、または MakeRODMCall サブルーチンが指定されるまで、現行のオブジェクトに適用されます。

仕様:

```
call StartObject classname objectname
```

オペランド記述: ここで、

classname
指定されるオブジェクトのクラスの名前。

objectname
指定されるオブジェクトの名前。

使用上の注意:

- クラスは StartObject を使用して作成することはできません。
- StartObject サブルーチンは、FLCARODM 関数すべてと一緒に使用します。
- オブジェクト名は、単一引用符 (' ') の間に指定することが必要です。

例: サンプル FLCSX09 からの次のコードは、 Demo_Lan という名前の ALnmClass のオブジェクトの作成時に StartObject サブルーチンを呼び出します。

```
call StartObject ALnmClass 'Demo_Lan'
```

サンプル FLCSX09 の実行時に Demo_Lan という名前のオブジェクトが存在しない場合、新しいオブジェクトが作成されることに注意してください。 Demo_Lan というオブジェクトがすでに存在している場合、その既存のオブジェクトが使用されます。

例について

この付録で使用される例は、NetView 製品によってサンプル・コードとして提供されています。例ではマルチシステム・マネージャーおよび GMFHS データ・モデルを使用していますが、 FLCARODM は、RODM にロードされるデータ・モデルであればすべてサポートします。

例では、FLCARODM コマンドへの入力として使用されるステム変数を作成します。ステートメント *call MakeRODMCall* ***function*** は、指定された関数を使用して FLCARODM コマンドを呼び出します。例えば、次のステートメントは、BUILD 関数を使って FLCARODM コマンドを発行します。

```
call MakeRODMCall 'BUILD'
```

サンプルの使用

NetView 製品が提供するサンプル・コードを使用するには、以下のタスクを実行します。

- RODMNAME の値を、使用している RODM の名前に変更します。
- RODMAPPL の値を、使用している RODM アプリケーション ID に変更します。
- サンプル FLCSSTEM の内容を、作成中の REXX コードの最後尾に付加します。 FLCSSTEM は、サンプルが使用するサブルーチンおよび定数定義を提供します。

FLCARODM コマンド

FLCARODM コマンドは、RODM との間でデータを入力したり読み取ったりするのに使用します。

FLCARODM コマンドは、NetView PIPE コマンドの NETVIEW ステージを使用して発行することが必要です。したがって、このコマンドは 2 つのソース、つまり PIPE データ・ストリーム、およびコマンドの発行時に使用されるパラメーターから実行される関数についての情報を受け取ります。 626 ページの図 149 は、FLCARODM コマンドの発行例です。

```
PIPE STEM object_data
  | COLLECT
  | NETVIEW FLCARODM parameters
  | stem result
```

図 149. FLCARODM コマンドの発行

ここで、

object_data
入力として使用される REXX ステム変数。

parameters
FLCARODM コマンドのパラメーター。

result FLCARODM から戻りコードまたはデータを受け取る REXX ステム変数。

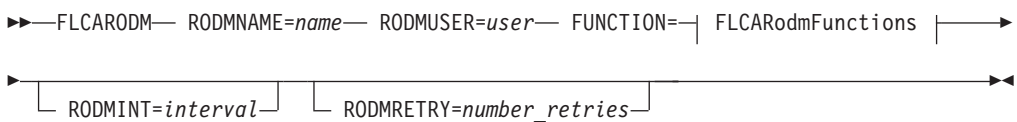
662 ページの『オブジェクト・データ・ストリームの詳細』で説明されるオブジェクト・データ・ストリームを使用してデータを指定している場合、図 149 で示されるフォーマットを使用してください。

また、NetView 製品は、FLCARODM コマンドを使用する別の方法も提供します。コマンドを直接指定する代わりに、MakeRODMCall サブルーチンを使用します。MakeRODMCall サブルーチンの詳細、および REXX オブジェクト・データ・ストリームを作成するのに使用できる他のサブルーチンについては、 619 ページの『ステム構築サブルーチン』を参照してください。

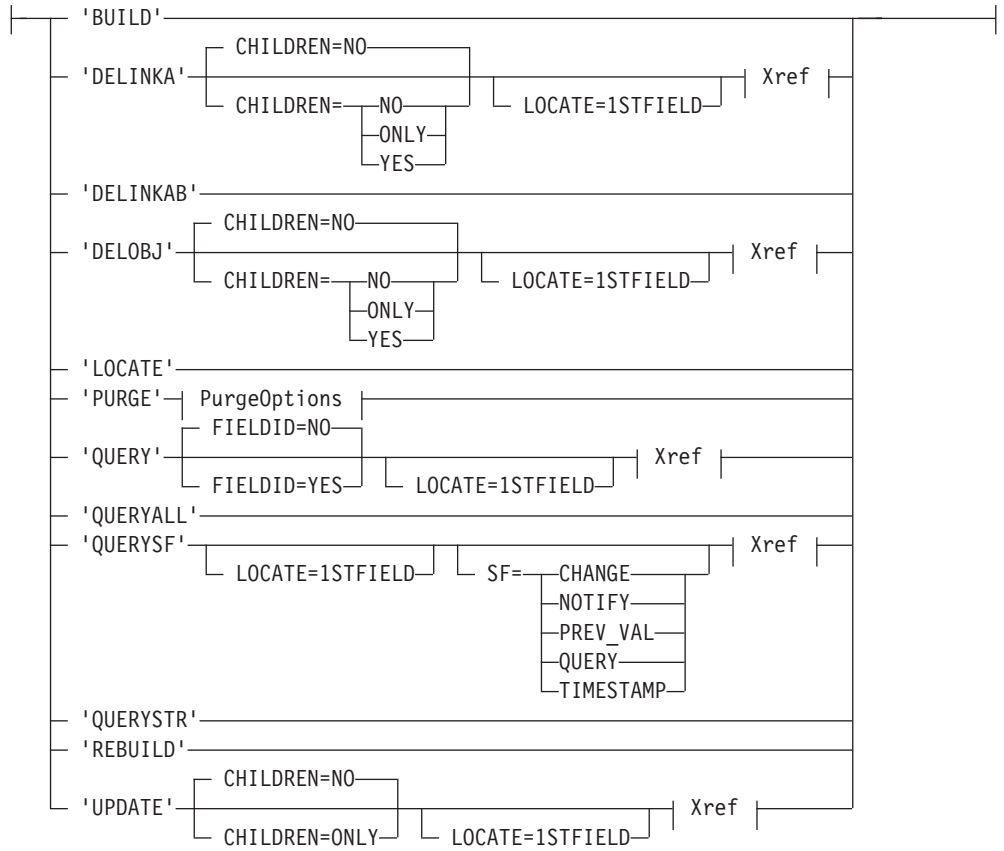
次のセクションでは、FLCARODM コマンドのフォーマットを説明します。説明には、オペランドのフォーマットおよび記述、および使用上の注意が含まれます。

FLCARODM

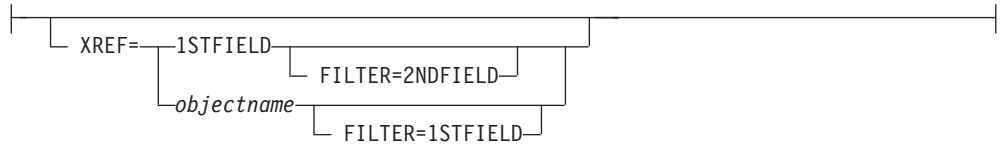
構文:



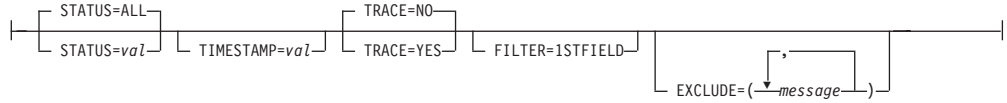
FLCARodmFunctions:



Xref:



PurgeOptions:



オペランド記述:

CHILDREN
指定された子オブジェクトに操作を適用するかどうかを指定します。
CHILDREN パラメーターは、XREF パラメーターが指定される場合は指定できません。

CHILDREN パラメーターは、次の関数と一緒に使用します。
- UPDATE
- DELINKA
- DELOBJ

NO
指定されたオブジェクトで関数が実行されるが、その子では実行されないことを示します。

ONLY
指定された子オブジェクトで関数が実行され、そのオブジェクト自体では実行されないことを示します。

YES
指定されたオブジェクトおよびその子で関数が実行されることを示します。

注:

1. YES は、UPDATE 関数では無効です。
2. UPDATE 関数の場合、第 1 レベルの子のみが更新されます。

EXCLUDE
PURGE 関数とともに使用します。これは、TRACE=YES を指定した場合にのみ指定可能です。 EXCLUDE オプションは、除去メッセージ (FLC040I、FLC041I、および FLC042I) のうち、除去処理時に発行されてはならないメッセージはどれかを指示します。その下に多数のオブジェクトを持つ集約オブジェクトの除去を試みる場合は、除去が正常に実行されたことを示すメッセージ FLC040I および FLC042I は受け取るが、除去が失敗したことを示すメッセージ FLC041I は抑制するようにしたいことがあります。そのような場合は、EXCLUDE=FLC041I と指定します。これを行わないと、無用の FLC041I メッセージをいくつも受け取る可能性があります。上記の除去メッセージのうち、1 個から 3 個を指定することができます。上記以外のメッセージはどれも指定できません。

FIELDID

QUERY 関数が、フィールド名と共にフィールド ID を戻すかどうかを示します。

NO
フィールド ID が戻されないことを示します。

YES
フィールド ID が戻されることを示します。

FILTER
XREF パラメーターを指定して使用し、操作が実行されているオブジェクトのリストにフィルターを掛けます。

FILTER パラメーターは、次の関数と一緒に使用します。

- DELOBJ
- DELINKA
- PURGE
- QUERY
- QUERYSF
- UPDATE

PURGE 関数を除いて、XREF パラメーターは、FILTER パラメーターが指定される前に指定される必要があります。 PURGE 関数の場合、FILTER は XREF パラメーターがなくても指定できます。

各オブジェクト仕様の最初のフィールドは、フィルター基準のフィールド名、タイプ、および値でなければなりません。 FILTER 値は、他のすべての関数およびパラメーターが処理された後、初めて適用されます。 FILTER は、全体が一致するか、または部分的に一致する値を戻します。例えば、フィールド値 Segment が指定され、既存のオブジェクトに値 Seg がある場合、フィルターは一致し、オブジェクトが戻されます。

FILTER=1STFIELD は、 XREF=1STFILELD が指定されない限り必ず指定する必要があります。 XREF=1STFILELD が指定される場合、FILTER=2NDFIELD も指定しなければなりません。フィールド記述は、以下に示される順序で、次の情報を指定することが必要です。

1. フィールド名
2. フィールド・データ・タイプ
3. フィールド値

FUNCTION
実行される関数を指定します。各関数については、 631 ページの『FLCARODM 関数』を参照してください。

LOCATE
オブジェクトのリストを作成するための基準として、最初のフィールド定義を使用することを指定します。

LOCATE=1STFIELD を指定する必要があります。また、フィールド記述は、以下に示される順序で次の情報を指定することが必要です。

1. フィールド名
2. フィールド・データ・タイプ
3. フィールド値

LOCATE パラメーターは、次の関数と一緒に使用します。

- DELINKA
- DELOBJ
- QUERY
- QUERYSF
- UPDATE

RODMINT
RODM のチェックポイント中の場合、FLCARODM が要求の再試行するまでに待機する時間の長さ (秒単位)。デフォルト値は 5 秒です。

RODMRTRY
RODM がチェックポイント中の場合、FLCARODM が要求を再試行する回数。

デフォルト値は 3 回です。 FLCARODM が、指定された回数だけ要求を再試行した後も RODM がまだチェックポイントを実行している場合、エラーがアプリケーションに戻されます。

RODMNAME
使用される RODM の名前。

RODMUSER
RODM への接続に使用されるアプリケーション名。 FLCARODM を呼び出す REXX プログラムを実行する、複数の NetView オペレーターが、同じ RODMUSER 値を使用することができます。しかし、アクセスは、 RODM に接続する他のアプリケーション (例えば RODMVIEW) と同じ RODMUSER 値を使用することはできません。

RODMUSER 値は、NetView ドメイン名と 3 文字の ID を連結して作成します。例えば、マルチシステム・マネージャーは、NetView ドメイン・ネーム CNMO1 を、マルチシステム・マネージャーの ID と連結し、RODMUSER 値を作成します。例えば、NetView ドメイン・ネームが CNM01 である場合、マルチシステム・マネージャーは CNM01MSM という RODMUSER 値を作成します。

SF 照会されるサブフィールドを示します。以下の値のいずれかを指定します。
- CHANGE
- NOTIFY
- PREV_VAL
- QUERY
- TIMESTAMP

STATUS
オブジェクトが PURGE 関数によって除去されるかどうかを判別するのに使用される、 DisplayStatus フィールド値。

ALL
オブジェクトが、DisplayStatus 値に関係なく除去されることを示します。TIMESTMP パラメーターは、STATUS の値が ALL である場合は指定できません。

val 削除されるオブジェクトの DisplayStatus フィールド値。デフォルト値は 132 (不明) です。

TIMESTAMP
秒単位で指定される、パージされるオブジェクトの経過時間基準。デフォルトは 84400、つまり 24 時間を秒数で表したものです。

TRACE
PURGE 関数をトレース・モードで実行するかどうかを指定します。トレース・モードでは、メッセージはパージされるオブジェクトごとに発行されます。

NO
PURGE 関数をトレース・モードで実行しないことを指定します。

YES
PURGE 関数をトレース・モードで実行することを指定します。

XREF
関数が、動的に獲得されたオブジェクトのリストについて実行されることを指定

します。オブジェクトのリストは、指定されるフィールドにより定義されます。フィールドは、次のデータ・タイプのうちいずれかであるべきです。
- ObjectIdList
- ObjectLink
- ObjectLinkList

XREF パラメーターは、次の関数と一緒に使用します。
- DELINKA
- DELOBJ
- QUERY
- UPDATE

XREF パラメーターは、CHILDREN パラメーターが使用される場合、指定できません。

XREF パラメーターは大文字小文字混合文字を含む場合があるので、 ADDRESS NETVASIS を指定することが必要です。

1STFIELD
オブジェクトで定義される最初のフィールドが指定されることを指定します。

objectname
使用されるフィールドの名前を示します。小数点付き 10 進表記の名前をもつオブジェクトの場合、小数点付き 10 進数名を使用する必要があります。例えば、メンバー・フィールドを指定するには、2.9.3.2.7.42 を指定します。

FLCARODM 関数

このセクションでは、FLCARODM コマンドの FUNCTION パラメーターが提供する関数を説明します。

以下の情報が各関数ごとに提供されます。
- 各関数の説明およびいつ使用するか。
- マルチシステム・マネージャーが提供するサンプルのセットに基づく例。
- 適用できれば、関数の結果の説明。

このセクションで説明されるサンプルの使用に関する詳細は、 625 ページの『例について』を参照してください。

BUILD 関数

BUILD 関数は、以下の機能を実行するのに使用します。
- 新規オブジェクトの作成
- 既存オブジェクトの変更
- フィールドの作成とフィールド値の割り当て
- オブジェクト間の関係の定義

以下のデータ・タイプは、BUILD および UPDATE 関数によってサポートされます。

データ・タイプ	データ・タイプ ID
CHARVAR	4

INTEGER	10
SELFDEFINING	19
SMALLINT	21
FIELDID	26
ANONYMOUSVAR	30

サンプル FLCSX1 からの次のコードは、BUILD 関数を使用して、 RODM でオブジェクトを作成する方法を示しています。

⋮

```
/*********************************************************************/
/* Start the first object.  This is the top object and is of type    */
/* Network_View_Class.  Its name is Hometown                        */
/*********************************************************************/
call StartObject NetClass 'Hometown'
                                /* Start creating Hometown object  */

call AddAttr Annotate CharVar 'This is the Hometown City View'
                                /* Add an Annotation or label      */

/*********************************************************************/
/* Add a second object to the list.  This object are inside the  */
/* Hometown class.  It is called Main_Street, is of type             */
/* GMFHS_Aggregate_Objects_Class.                                    */
/*********************************************************************/
call StartObject AggClass 'Main_Street'

     /*******************************************************/
     /* Now add a label which says 'Constructed in 1889 to   */
     /* the object.                                         */
     /*******************************************************/
call AddAttr DispOther CharVar 'Constructed in 1889'

     /*******************************************************/
     /* Add a link to the object which tells the Display     */
     /* ResourceType and Display_Resource_Type_Class are     */
     /* linked to the DUIXC_RTN_HOST_AGG                    */
     /*******************************************************/
call AddLink DispType DispClass HtAgg_Icon 'Resources'

     /*******************************************************/
     /* Now add another link to link the object to the       */
     /* Hometown view                                       */
     /*******************************************************/
call AddLink ConView NetClass 'Hometown' ConObjs

call MakeRODMCall 'BUILD'
/*********************************************************************/
/* Start the third object in the group.  This one is called          */
/* '1000_Main_Street' and is contained in the 'Main_Street' object   */
/*********************************************************************/
call InitRODMStem

call StartObject AggClass '1000_Main_Street'

     /*******************************************************/
     /* Add some information to the object                  */
     /*******************************************************/
call AddAttr DispOther CharVar '3 Bedroom Ranch'
call AddAttr DispStat Integer Active

      /*******************************************************/
      /* Now link it to its parent and to its class          */
      /*******************************************************/
```

```
call AddLink DispType DispClass HtAgg_Icon 'Resources'
call AddLink PartOf AggClass 'Main_Street' COMPPHY


call MakeRODMCAll 'BUILD'  /* make the FLCARODM call           */
⋮
```

BUILD 関数の実行結果: 以下のオブジェクトが、BUILD 関数によって RODM で作成されました。

- Hometown という名前のネットワーク・ビューを表すビュー・オブジェクト。
- Main_Street を表す集合オブジェクト。
- 1000_Main_Street という名前の Main_Street にある家を表す実オブジェクト。

UPDATE 関数

UPDATE 関数は、既存のオブジェクトでフィールドの値を変更するのに使用します。 UPDATE 関数はオブジェクトを作成しません。存在しないオブジェクトでフィールドを更新しようとすると、エラーが戻されます。

サンプル FLCSX2 からの次のコードは、UPDATE 関数を使用して、 RODM でオブジェクトを変更する方法を示しています。

```
⋮

call StartObject AggClass '1000_Main_Street' /*Which object we are   */
                                             /*referring to.         */
call AddAttr DispStat Integer InActive       /*Update display status */
call MakeRODMCall 'UPDATE'                 /*Call RODM               */
⋮
```

UPDATE 関数の実行結果: 1000_Main_Street という名前を表す実オブジェクトで DisplayStatus フィールドの値が、 132 (不良) に変更されます。

QUERY 関数

QUERY 関数は、1 つ以上のオブジェクトで 1 つ以上の値を判別するのに使用します。フィールドまたはオブジェクトが存在しない場合、エラーが戻されます。フィールド・タイプとフィールド値は、各オブジェクトのフィールドごとに戻されます。

フィールド・タイプはフィールドの照会時には指定されませんが、 FLCARODM は以下のデータ・タイプにのみ値を戻します。

CLASSID	1
CHARVAR	4
INTEGER	10
OBJECTID	14
OBJECTIDLIST	15
OBJECTLINK	16
OBJECTLINKLIST	17
OBJECTNAME	18
SELFDEFINING	19
SMALLINT	21
SMALLINT	23
FIELDID	26
ANONYMOUSVAR	30

QUERY 関数の使用例: このセクションでは、QUERY 関数の使用例をいくつか示します。

サンプル FLCSX3 からの次のコードは、Main_Street オブジェクトで、DisplayResourceOtherData フィールドを照会します。

```
⋮
call StartObject AggClass 'Main_Street'    /*Which object we are     */
                                           /*referring to.           */
call AddAttrForQuery DispOther             /*Query contents of       */
                                           /*DisplayResourceOtherData*/
call MakeRODMCall 'QUERY'                  /*Call RODM               */
⋮
```

FLCSX3 からの結果ステムには、指定された順序で次の情報が入っています。
- ステム中のエレメント数
- FLCARODM 戻りコード、続いて RODM 戻りコードおよび理由コード
- フィールドの値

サンプル FLCSX3 の実行時に戻される結果ステムの例を、以下に部分的に示します。

```
3
FLCARODM:0,0,0
4
Constructed In 1889
⋮
```

表 229 は、サンプル FLCSX3 に戻された結果ステムを説明しています。

表 229.

エレメント番号	エレメント値	説明
0	3	結果ステムに 3 つのエレメントが含まれることを示します。
1	FLCARODM:0,0,0	FLCARODM 戻りコード、RODM 戻りコードおよび理由コード
2	4	フィールドのデータ・タイプ (charvar)
3	Constructed In 1889	フィールドの値

指定されたフィールドのフィールド ID の値を知っておくことが役立つ場合もあります。例えば、表にフィールドを保管している場合、長いファイル名ではなく、4 バイトのフィールド ID を保存するとスペースを節約できます。

YES の値で FIELDID パラメーターを指定すると、 FLCARODM は、QUERY 関数が戻したフィールドにフィールド ID の値を戻します。

注:

1. フィールド ID は RODM がコールド・スタートする時に変わることがあるので、前に保管したフィールド ID に関する情報は使用されません。
2. FIELDID パラメーターは、LOCATE、XREF、または CHILDREN パラメーターと一緒には使用できません。

サンプル FLCSX3 からの次のコードは、FIELDID=YES を指定することによって変更され、 DisplayResourceOtherData フィールドのフィールド ID を戻します。

```
⋮

call StartObject AggClass 'Main_Street'    /*Which object we are     */
                                           /*referring to.           */
call AddAttrForQuery DispOther             /*Query contents of       */
                                           /*DisplayResourceOtherData*/
call MakeRODMCall 'QUERY' 'FIELDID=YES'   /*Call RODM               */
⋮
```

変更済みサンプル FLCSX3 の実行時に戻される結果ステムの例を、以下に部分的に示します。

```
4
FLCARODM:0,0,0
4
60
Constructed In 1889
⋮
```

表 230 は、変更済みのサンプル FLCSX3 に戻された結果ステムを説明しています。

表 230.

エレメント番号	エレメント値	説明
0	3	結果ステムに 3 つのエレメントが含まれることを示します。
1	FLCARODM:0,0,0	FLCARODM 戻りコード、RODM 戻りコードおよび理由コード
2	4	フィールドのデータ・タイプ (charvar)
3	60	フィールドのフィールド ID を示します。
4	Constructed In 1889	フィールドの値

サンプル FLCSX4 を実行する前に、FLCSX1、FLCSX2、および FLCSX3 を実行してください。

サンプル FLCSX4 は、タスクを達成する 2 つの照会の使用例を提供し、クラス上でフィールド値を判別する方法を例示します。これは、デフォルトのフィールド値を照会したり、特定のクラスのすべてのオブジェクトを獲得するのに役立ちます。この例の場合、サンプル FLCSX1 の実行前に RODM が空だったと想定します。サンプル FLCXS4 の最初の部分は、 RODM 中のすべての GMFHS_Aggegrate_Objects_Class オブジェクトを照会します。

```
⋮

call StartObject AggClass '.'                /*Which object we are   */
                                            /*referring to.         */
call AddAttrForQuery 'MyObjectChildren'

Say ''
Say 'Result from MyObjectChildren query:'
call MakeRodmCall 'QUERY'
'PIPE STEM RodmResult. | CONSOLE'
```

⋮

FLCSX4 からの結果ステムには、指定された順序で次の情報が入っています。

- ステム中のエレメント数
- FLCARODM 戻りコード、続いて RODM 戻りコードおよび理由コード
- フィールドのデータ・タイプ
- リスト中のオブジェクト ID の数
- オブジェクトのオブジェクト ID

サンプル FLCSX4 の最初の部分が戻す結果ステムの例を、以下に示します。

```
4
FLCARODM:0,0,0
15
1
00010012E05C2A1E
```

表 231 は、サンプル FLCSX4 の最初の部分に戻された結果ステムを説明しています。

表 231.

エレメント番号	エレメント値	説明
0	4	結果ステムに 4 つのエレメントが含まれることを示します。
1	FLCARODM:0,0,0	FLCARODM 戻りコード、RODM 戻りコードおよび理由コード
2	15	フィールドのデータ・タイプ (objectidlist) を示します。
3	1	リスト中のオブジェクト ID の数
4	00010012E05C2A1E	オブジェクトの 16 進数オブジェクト ID

サンプル FLCSX4 の 2 番目の部分は、最初の照会から戻されたオブジェクト ID の名前、および状況を照会します。

⋮

```
/*********************************************************************/
/* Query the name and status of the object.                          */
/*********************************************************************/
call InitRODMStem              /*Get ready for next set of operations*/

call StartObject AggClass '.'  /*Use Object ID from previous call*/
call AddAttrForQuery 'MyName'
call AddAttrForQuery 'DisplayStatus'

Say ''
Say 'Result from MyName and DisplayStatus query:'
call MakeRodmCall 'QUERY'
⋮
```

サンプル FLCSX4 の 2 番目の部分が戻す結果ステムの例を、以下に示します。

```
6
FLCARODM:0,0,0
18
Main_Street
FLCARODM:0,0,0
10
132
```

表 232 は、サンプル FLCSX4 の 2 番目の部分に戻された結果ステムを説明しています。

表 232.

エレメント番号	エレメント値	説明
0	6	結果ステムに 6 つのエレメントが含まれることを示します。
1	FLCARODM:0,0,0	照会された最初のフィールドへの FLCARODM 戻りコード、RODM 戻りコードおよび理由コード
2	18	フィールドのデータ・タイプを示します。
3	Main_Street	リスト中のオブジェクト ID の数
4	FLCARODM:0,0,0	照会された 2 番目のフィールドへの FLCARODM 戻りコード、RODM 戻りコードおよび理由コード
5	10	フィールドのデータ・タイプ (整数)
6	132	フィールドの値

注: FLCSX4 の QUERY 関数は、MakeRODMCall サブルーチンを使って、FLCARODM への 2 回の呼び出しによって実行されました。どちらの関数も、XREF パラメーターを使った FLCARODM の 1 回の呼び出しを使用して実行することができます。詳細については、625 ページの『FLCARODM コマンド』を参照してください。

DELOBJ 関数

DELOBJ は、1 つ以上のオブジェクトを削除するために使用します。オブジェクトの削除時、他のすべてのオブジェクトへのリンクは削除されます。フィールドおよびリンクは、DELOBJ 関数を使用して指定できないことに注意してください。

他のアプリケーションまたはユーザーが必要とするオブジェクトが削除される可能性があるので、 DELOBJ 関数の使用には注意が必要です。代わりに PURGE 関数を使用することを考慮してください。この関数は、他のアプリケーションと関連したオブジェクトを削除させないようにするオブジェクトを除去する手段を提供します。

サンプル FLCSX5 からの次のコードは、DELOBJ を使用して、 1000 Main Street オブジェクトを削除します。

```
⋮
call StartObject AggClass '1000_Main_Street' /*Which object we are   */
                                             /*referring to.        */
```

```
call MakeRODMCall 'DELOBJ'                    /*Call RODM            */
⋮
```

DELOBJ 関数の実行結果: このプログラムの実行後、1000_Main_Street オブジェクト、Main_Street へのリンク、およびオブジェクトが除去されます。

DELINKA 関数

DELINKA 関数は、オブジェクト上で指定されたフィールドへのすべてのリンクを削除するのに使用します。 DELINKA 関数は存在するリンクを判別し、それをすべて除去するので、リンクを指定する必要はありません。

DELINKA 関数の使用例に関しては、649 ページの『オブジェクトのリンク解除』を参照してください。

DELINKAB 関数

DELINKAB 関数は、オブジェクト間で指定されたリンクを削除します。

タイプ ObjectLink のフィールドを使用してリンクされたほとんどのオブジェクトの場合、新しいリンクの定義前にオブジェクト間のリンクを除去する必要はありません。その代わり、UPDATE 関数を使用して、古いリンクを最初に除去してから新しいリンクを定義します。しかし、リンクの除去を実行するメソッドが必要なフィールド (例えば DisplayResourceType) の場合、 DELINKAB 関数を使用することが必要です。

タイプ ObjectLinkList のフィールド (例えば Resources) が定義するリンクの場合、UPDATE 関数は新しいリンクを追加するだけで、前に定義されたリンクを削除しないので、 DELINKAB 関数を使用する必要があります。

DELINKAB 関数の使用例に関しては、649 ページの『オブジェクトのリンク解除』を参照してください。

PURGE 関数

PURGE 関数は、RODM からオブジェクトを除去するのに使用します。 RODM からオブジェクトを除去する方法として、PURGE 関数の代わりに **RemvObjs** コマンドを使用することを検討してください。 **RemvObjs** コマンドの詳細は、「*IBM Tivoli NetView for z/OS* マルチシステム・マネージャー ユーザーズ・ガイド」を参照してください。

LOCATE 関数

LOCATE 関数は、指定されたストリングを、public_indexed として作成されている、タイプ CharVar または IndexList のすべてのフィールドで検索するのに使用します。公的に索引を付けられるフィールドの例は、DisplayResourceName です。

LOCATE 関数は、指定されたストリングに一致する値を含む、オブジェクトのオブジェクト ID を戻します。検索では大文字小文字が区別されないことに注意してください。

サンプル FLCSXL01 からの次のコードでは、 DisplayResourceName フィールドが CPU_UTILIZATION の値を持つ RODM のオブジェクトをすべて検索します。

```
⋮
call StartObject '' '' /*Can not specify a class or an object for    */
```

```
                        /*This function                                */
call AddAttr DispName CharVar 'CPU_utilization'  */
call MakeRODMCall 'LOCATE'                    /*Call RODM             */
⋮
```

LOCATE 関数にクラスまたはオブジェクトを指定できないことに注意してください。したがって、StartObject '' '' が指定されます。つまり、すべてのクラスですべてのオブジェクトが検索されるということです。

FLCSXL01 からの結果ステムには、DisplayResourceName が比較ストリング NOT_LOGGED_IN と一致するオブジェクトのオブジェクト ID のリストが入っています。例えば、この基準に 1 つのオブジェクトが一致した場合、次の結果ステムが戻されます。

```
4
FLCARODM:0,0,0
15
1
000100012E05C2A1E
```

表 233 は、1 つのオブジェクトが検索基準を満たす場合、サンプル FLCSXL01 に戻された結果ステムを説明しています。

表 233.

エレメント番号	エレメント値	説明
0	4	結果ステムに 4 つのエレメントが含まれることを示します。
1	FLCARODM:0,0,0	FLCARODM 戻りコード、RODM 戻りコードおよび理由コード
2	15	戻りデータのデータ・タイプ (objectidlist) を示します。
3	1	見つかった一致の数
4	000100012E05C2A1E	検索基準に一致したオブジェクトのオブジェクト ID

比較基準に一致したフィールドを持つ RODM にオブジェクトがない場合、FLCARODM は、次のようなエレメントがゼロのオブジェクト ID リストを戻します。

```
3
FLCARODM:0,0,0
15
0
```

表 234 に、検索条件を満たすオブジェクトがない場合に、サンプル FLCSXL01 に戻される結果ステムを説明します。

表 234.

エレメント番号	エレメント値	説明
0	3	結果ステムに 3 つのエレメントが含まれることを示します。

表 234. (続き)

エレメント番号	エレメント値	説明
1	FLCARODM:0,0,0	FLCARODM 戻りコード、RODM 戻りコードおよび理由コード
2	15	戻りデータのデータ・タイプ (objectidlist) を示します。
3	0	一致が見つからなかったことを示します。

RodmResult.0 3
RodmResult.1 FLCARODM:0,0,0
RodmResult.2 15
RodmResult.3 0

QUERYALL 関数

QUERYALL 関数は、指定されたオブジェクトで定義されるすべてのフィールドに、フィールド名、フィールド・タイプ、および値を戻します。例えば、次の例は、Main_Street オブジェクトでフィールドを照会します。

```
⋮
call StartObject AggClass 'MainStreet'
call MakeRODMCall 'QUERYALL'
⋮
```

QUERYALL 関数の実行結果: FLCSXQ2 からの結果ステムには、指定された順序で次の情報が入っています。

- ステム中のエレメント数。
- FLCARODM 戻りコード、続いて RODM 戻りコードおよび理由コード。
- オブジェクトで定義されたフィールド数。
- フィールド仕様のシーケンス。フィールドごとに、フィールド仕様には、指定された順序で次の情報が入っています。
  - 戻りコード
  - 名前
  - ID
  - 値

  フィールド仕様情報は、各フィールドごとに繰り返されます。

FLCSXQ2 からの結果ステムには、ステム中のエレメント数、戻りコード、オブジェクトで定義されたフィールドの数、およびフィールド仕様のシーケンスが含まれます。各フィールド仕様には、以下の情報が含まれています。

サンプル FLCSXQ2 の実行時に戻される結果ステムの例を、以下に部分的に示します。

```
212
FLCARODM:0,0,0
51
FLCARODM:0,0,0
IsPartOf
17
0
FLCARODM:0,0,0
IsBusNode
17
```

```
0
⋮
```

表 235 は、サンプル FLCSXQ2 に戻された結果ステムを説明しています。

表 235.

エレメント番号	エレメント値	説明
0	212	結果ステムに 212 のエレメントが含まれることを示します。
1	FLCARODM:0,0,0	FLCARODM 戻りコード、RODM 戻りコードおよび理由コード
2	51	オブジェクトで定義されたフィールド数を示します。
3	FLCARODM:0,0,0	FLCARODM 戻りコード、RODM 戻りコードおよび理由コード
4	IsPartOf	オブジェクトで定義される最初のフィールド名
5	17	IsPartOf フィールドのデータ・タイプ (objectlinklist)
6	0	IsPartOf フィールドの値
7	FLCARODM:0,0,0	FLCARODM 戻りコード、RODM 戻りコードおよび理由コード
8	IsBusNode	オブジェクトで定義される 2 番目のフィールド名
9	17	IsBusNode フィールドのデータ・タイプ (objectlink)
10	0	IsBusNode フィールドの値

前の例では、結果ステムの最初の 2 つのフィールドを説明しています。エレメント 11 から 212 は、同じフォーマットを使用する残りのフィールドを説明します。

QUERYSTR 関数

QUERYSTR 関数は、オブジェクト・クラスの構造体を判別するために使用します。各クラスごとに、フィールド名、フィールド ID タイプ、およびクラス上で定義される各フィールドごとの継承状況ビットマップが戻されます。例えば、次のサンプルは、GMFHS_Aggregate_Objects_Class クラスの構造体を照会します。

```
⋮
call StartObject AggClass ''
call MakeRODMCall 'QUERYSTR'
⋮
```

QUERYSTR 関数の実行結果: FLCSXQ1 からの結果ステムには、指定された順序で次の情報が入っています。

- ステム中のエレメント数
- FLCARODM 戻りコード、続いて RODM 戻りコードおよび理由コード
- オブジェクトで定義されたフィールド数
- フィールド仕様のシーケンス。フィールドごとに、フィールド仕様には、指定された順序で次の情報が入っています。
  - 名前
  - ID
  - タイプ
  - 継承状況ビットマップ

フィールド仕様情報は、各フィールドごとに繰り返されます。

サンプル FLCSXQ1 の実行時に戻される結果ステムの例を、以下に部分的に示します。

```
214
FLCARODM:0,0,0
53
AggrgationChild
121
17
00
UpdateAggregationCounters
122
13
00
⋮
```

表 236 は、サンプル FLCSXQ1 に戻された結果ステムを説明しています。

表 236.

エレメント番号	エレメント値	説明
0	214	結果ステムに 214 のエレメントが含まれることを示します。
1	FLCARODM:0,0,0	FLCARODM 戻りコード、RODM 戻りコードおよび理由コード。
2	53	オブジェクトで定義されたフィールド数を示します。
3	AggregationChild	オブジェクトで定義される最初のフィールド名
4	121	フィールド ID を表します。
5	17	フィールドのデータ・タイプ (objectlinklist)
6	00	継承状況ビットマップ
7	UpdateAggregationCounters	オブジェクトで定義される 2 番目のフィールド名
8	122	フィールド ID を表します。
9	13	フィールドのデータ・タイプ (methodspec)
10	00	継承状況ビットマップ

前の例では、結果ステムの最初の 2 つのフィールドを説明しています。エレメント 11 から 214 は、同じフォーマットを使用する残りのフィールドを説明します。

QUERYSF 関数

QUERYSF 関数は、指定されたオブジェクト上のフィールドに指定されたサブフィールドの値を照会するのに使用します。次のサブフィールドが照会されます。

- VALUE
- QUERY
- CHANGE
- NOTIFY
- TIMESTAMP

- PREV_VAL

サンプル FLCSXQ3 からの次のコードは、 1000 Main Street オブジェクトの DisplayStatus フィールドのサブフィールドに、前の値を戻します。

```
⋮

call StartObject AggClass '1000_Main_Street' /*Which object we are   */
                                             /*referring to.         */
call AddAttrForQuery DispStat                /*Query this field      */
call MakeRODMCall 'QUERYSF' 'SF=PREV_VAL'    /*Call RODM             */
⋮
```

QUERYSF 関数の実行結果: FLCSXQ3 からの結果ステムには、指定された順序で次の情報が入っています。

- ステム中のエレメント数
- FLCARODM 戻りコード、続いて RODM 戻りコードおよび理由コード
- サブフィールドのデータ・タイプ
- サブフィールド値

注: サンプル FLCSX1 および FLCSX2 を、サンプル FLCSXQ3 の実行前に実行してください。

サンプル FLCSXQ3 の実行時に戻される結果ステムの例を、以下に示します。

```
3
FLCARODM:0,0,0
10
129
```

表 237 は、サンプル FLCSXQ3 に戻された結果ステムを説明しています。

表 237.

エレメント番号	エレメント値	説明
0	3	結果ステムに 3 つのエレメントが含まれることを示します。
1	FLCARODM:0,0,0	FLCARODM 戻りコード、RODM 戻りコードおよび理由コード
2	10	サブフィールドのデータ・タイプ (整数)
3	129	フィールドの以前の値

注: FLCSX1 が値を 129 に設定してから、FLCSX2 が値を 130 に変更したので、以前の値は 129 でした。

REBUILD 関数

REBUILD 関数は、オブジェクト間のリンクが変更された場合にオブジェクトを変更するのに使用します。 REBUILD 関数で指定されたすべての各オブジェクトで、すべての指定済みフィールドが更新され、すべての指定済みリンクが定義され、以前定義されたリンクがすべて除去されます。ただし、以下は例外です。

- LayoutParmList
- DetailLayoutParmList
- 2.9.3.2.7.42 (member)

- 1.3.18.0.0.2217 (memberArcs)
- ComposedOfPhysical
- ComposedOfLogical
- AggregationChild

上記にリストされる関係は、親オブジェクトを定義していない RODM のオブジェクトを回避するため、除去されません。

まとめ

このセクションでは、関数およびパラメーターの使用例を提供する、サンプル・ファイルを説明します。

サンプルで使用されるサブルーチンの詳細は、 619 ページの『ステム構築サブルーチン』を参照してください。

オブジェクトの構築

次のサンプルは、StartObject および AddLink ルーチンを使用して、以下のオブジェクトを作成し、リンクします。

- Demo_Lan という名前の集合オブジェクト
- LAN セグメントを表す 2 つのオブジェクト
- セグメントを接続するブリッジを表すオブジェクト

```
⋮

call StartObject NetClass 'Advanced'         /*Which object?         */
/*********************************************************************/
/* Start creating LAN object in the Advanced Operations View         */
/*********************************************************************/
call StartObject ALnmClass 'Demo_Lan'
call AddLink DispType DispClass 'DUIXC_RTN_LAN_AGG' 'Resources'
call AddLink ConView NetClass 'Advanced' ConObjs

/*********************************************************************/
/* Create the Segment_1 object                                       */
/*********************************************************************/
call StartObject RSegClass 'Segment_1'
call AddLink DispType DispClass 'DUIXC_RTN_TR_SEGMENT' 'Resources'
call AddLink MemberOf ALnmClass 'Demo_Lan' Member

/*********************************************************************/
/* Add a Bridge called Bridge_1                                      */
/* Add a link to hook it to Segment_1.                               */
/*********************************************************************/
Call StartObject ABrgClass 'Bridge_1'
Call AddLink DispType DispClass 'DUIXC_RTN_BRIDGE_APPL' 'Resources'
Call AddLink MemberOf ALnmClass 'Demo_Lan' Member
Call AddLink PhyConn RSegClass 'Segment_1' Phyconn

/*********************************************************************/
/* Create the second segment, called Segment_2                       */
/* Add a link to connect it to Bridge_1                              */
/*********************************************************************/
call StartObject RSegClass 'Segment_2'
call AddLink DispType DispClass 'DUIXC_RTN_TR_SEGMENT' 'Resources'
call AddLink MemberOf ALnmClass 'Demo_Lan' Member
call AddLink PhyConn ABrgClass 'Bridge_1' Phyconn

call MakeRODMCall 'BUILD'                    /*Call RODM             */
⋮
```

図 150. サンプル FLCSX6

オブジェクトの更新

次のサンプルは、UPDATE 関数を使用してオブジェクトを変更する例を提供します。

CHILDREN パラメーターを指定して UPDATE 関数を使用する: 646 ページの図 151 は、UPDATE 関数を使って Demo_Lan 集合オブジェクトの表示状況を変更します。 CHILDREN=ONLY が指定されるために、すべての Demo_Lan の子が更新されることに注意してください。しかし、CHILDREN パラメーターは、子の第 1 レベルを更新するだけです。

```
⋮

call StartObject ALnmClass 'Demo_Lan'        /*Which object we are   */
                                             /*referring to.         */
call AddAttr DispStat Integer InActive       /*Update display status */

call MakeRODMCall 'UPDATE' 'CHILDREN=ONLY' /*Call RODM               */
                                           /*Update only the children*/
⋮
```

図 151. サンプル FLCSX7

XREF パラメーターを指定して UPDATE 関数を使用する: XREF パラメーターは、次のタイプのフィールドを指定するのに使用できます。

- ObjectLink
- ObjectLinkList
- ObjectIdList

次のサンプルは、これらのタイプのフィールドを使用してオブジェクトを見つける方法を示します。

図 152 では、UPDATE 関数を使用して図 151 と同じタスクを実行します。しかし、CHILDREN パラメーターを指定する代わりに、XREF パラメーターを使用してフィールド 2.9.3.2.7.42 (member) が定義するリンクを指定します。

```
⋮

call StartObject ALnmClass 'Demo_Lan'        /*Which object we are   */
                                             /*referring to.         */
call AddAttr DispStat Integer InActive       /*Update display status */
call MakeRODMCall 'UPDATE' 'XREF=2.9.3.2.7.42'    /*Call RODM        */
⋮
```

図 152. サンプル FLCSX14

図 153 は、XREF パラメーターを指定して UPDATE 関数を使用し、ComposedOfPhysical フィールドが定義するリンクを使用して、更新されるオブジェクトのリストを判別する必要があることを指定します。

```
⋮

call StartObject AggClass 'Main_Street'      /*Which object we are   */
                                             /*referring to.         */
call AddAttr DispStat Integer Active         /*Update display status */
call MakeRODMCall 'UPDATE' 'XREF=ComposedOfPhysical' /* Call RODM    */
⋮
```

図 153. サンプル FLCSX15

647 ページの図 154 は、サンプル FLCSX14 および FLCSX15 と同じ関数を実行し、単一の関数呼び出しで、複数の関数を実行できることを示します。サンプル FLCSX16 は、XREF パラメーターを指定して UPDATE 関数を使用し、指定される最初のフィールドが定義するリンクを使って、更新されるオブジェクトのリストを判別する必要があることを指定します。例えば、サンプル FLCSX16 は次のとおりです。

```
call StartObject ALnmClass 'Demo_Lan'
call AddLink Member DispStat Integer InActive
```

Demo_Lan オブジェクトで定義される最初のフィールドは Member フィールドなので、定義されるリンクは、更新されるオブジェクトを判別するために使用されます。

```
⋮

call StartObject ALnmClass 'Demo_Lan'        /*Which object we are   */
                                             /*referring to.         */
call AddLink Member DispStat Integer InActive /*Update display status*/
                                             /*Cross Reference Member*/
                                             /*Field. Anything that  */
                                             /*has is a Member of the*/
                                             /*Demo_Lan gets changed */

call StartObject AggClass 'Main_Street'      /*Which object we are   */
                                             /*referring to.         */
call AddLink COMPPHY DispStat Integer InActive/*Update display status*/
                                  /*Cross Reference the COMPPHY field */
                                  /*in the Main_street to find out    */
                                  /*which objects have their Display  */
                                  /*status changed.                   */

call MakeRODMCall 'UPDATE' 'XREF=1STFIELD'   /*Call RODM             */
⋮
```

図 154. サンプル FLCSX16

図 155 は、タイプ ObjectIdList で、クラスのオブジェクト ID のリストを含む MyObjectChildren フィールドを使用することによって、クラスの子オブジェクトをすべて更新する方法を示します。

```
⋮

call StartObject RealClass ''                /*Which object we are   */
                                             /*referring to.         */
call AddAttr DispStat Integer InActive       /*Update display status */
call MakeRODMCall 'UPDATE' 'XREF=MyObjectChildren' /*Call RODM       */
⋮
```

図 155. サンプル FLCSX17

オブジェクトの照会

このセクションでは、QUERY 関数の使用法を説明します。各サンプルごとに、照会仕様が説明され、結果ステムのサンプルが提供されます。結果ステムの詳細については、651 ページの『結果ステム』を参照してください。

648 ページの図 156 は、すべての Demo_Lan オブジェクトの名前を照会します。名前は MyName フィールドに入っており、照会されるオブジェクトのリストは、フィールド 2.9.3.2.7.42 (member) によって定義されます。

```
⋮

call StartObject ALnmClass 'Demo_Lan'        /*Which object we are   */
                                             /*referring to.         */
call AddAttrForQuery MyName                  /*Update display status */
call MakeRODMCall 'QUERY' 'XREF=2.9.3.2.7.42'   /*Call RODM          */
⋮
```

図 156. サンプル FLCSX18

次の結果ステムが戻されます。

RodmResult.0	11
RodmResult.1	FLCARODM:0,0,0
RodmResult.2	3
RodmResult.3	FLCARODM:0,0,0
RodmResult.4	18
RodmResult.5	Segment_1
RodmResult.6	FLCARODM:0,0,0
RodmResult.7	18
RodmResult.8	Bridge_1
RodmResult.9	FLCARODM:0,0,0
RodmResult.10	18
RodmResult.11	Segment_2

FLCARODM:0,0,0 は、相互参照フィールド field 2.9.3.2.7.42 の照会が正常に実行されたことを示します。

図 157 は、RODM 中のすべてのオブジェクトを照会して、 LNM_NETWORKS という表示名を持つオブジェクトを判別します。 `call StartObject '' ''` が RODM にあるすべてのオブジェクトを意味することに注意してください。

```
⋮

call StartObject '' ''                    /*Which object we are     */
                                          /*referring to.           */
call AddAttr DispName CharVar 'LNM_Networks' /*Look at all objects,  */
call AddAttrForQuery 'MyName'             /*Return all with MyName= */
                                          /*LNM_Networks            */
call MakeRODMCall 'QUERY' 'LOCATE=1STFIELD' /*Call RODM            */
```

図 157. サンプル FLCSXL02

次の結果ステムが戻されます。

RodmResult.0	5
RodmResult.1	FLCARODM:0,0,0
RodmResult.2	1
RodmResult.3	FLCARODM:0,0,0
RodmResult.4	18
RodmResult.5	2.9.3.2.7.4=LNM_Networks

2 番目のステム変数は、基準に一致するオブジェクトが 1 つあったことを示します。 5 番目のステム変数は、オブジェクトの名前を提供します。

図 158 は、Segment という語を含むすべての Demo_Lan オブジェクトの名前を表示します。 FILTER パラメーターが XREF パラメーターと一緒に使用され、照会を洗練していることに注意してください。

```
⋮
call StartObject ALNMClass 'Demo_Lan'

call AddAttr MyName ObjectName 'Segment'
call AddAttrForQuery MyName

call MakeRODMCall 'QUERY' 'XREF=2.9.3.2.7.42' 'FILTER=1STFIELD'
⋮
```

図 158. サンプル FLCSXF1

次の結果ステムが戻されます。

RodmResult.0	8
RodmResult.1	FLCARODM:0,0,0
RodmResult.2	2
RodmResult.3	FLCARODM:0,0,0
RodmResult.4	18
RodmResult.5	Segment_1
RodmResult.6	FLCARODM:0,0,0
RodmResult.7	18
RodmResult.8	Segment_2

2 番目のステム変数は、XREF および FILTER 基準に一致する 2 つのリソースがあったことを示します。名前は、RodmResult.5 および RodmResult.8 に入っています。

注: XREF 値が 1STFIELD を使用して指定される場合、フィルター基準は `FILTER=2NDFIELD` でなければなりません。

オブジェクトのリンク解除

このセクションでは、DELINKA および DELINKAB 関数を使用してオブジェクト間のリンクを除去する方法を説明します。

また、図 159 では、DELINKA 関数を使用して、 Bridge_1 オブジェクトの PhysicalConnPP フィールドが定義するすべてのリンクを削除します。

```
⋮

call StartObject ABrgClass 'Bridge_1'        /*Which object we are   */
                                             /*referring to.         */
call AddLinkForDelete PhyConn                /*Add the link to Delete*/
call MakeRODMCall 'DELINKA'                  /*Call RODM             */
⋮
```

図 159. サンプル FLCSX10

図 159 と同様、 650 ページの図 160 では、DELINKA 関数を使用して、 Bridge_1 オブジェクトの PhysicalConnPP フィールドが定義するすべてのリンクを削除します。しかし、CHILDREN=ONLY パラメーターは、削除されるリンクを判別するの

に使用されます。

```
⋮

call StartObject ALnmClass 'Demo_Lan'        /*Which object we are   */
                                             /*referring to.         */
call AddLinkForDelete PhyConn
call MakeRODMCall 'DELINKA' 'CHILDREN=ONLY'  /*Call RODM             */
                                             /*Only do the CHILDREN  */
⋮
```

図 160. サンプル FLCSX9

```
⋮

call StartObject ALnmClass 'Demo_Lan'        /*Which object we are   */
                                             /*referring to.         */
call AddAttrForQuery  Member
call AddAttrForQuery  PhyConn
call MakeRODMCall 'DELINKA' 'XREF=1STFIELD'  /*Call RODM             */
⋮
```

図 161. サンプル FLCSX19

図 162 では、DELINKAB 関数を使用して、Bridge_1 オブジェクトへの特定のリンクを除去します。

```
⋮

call StartObject ABrgClass 'Bridge_1'        /*Which object we are   */
                                             /*referring to.         */
call AddLink PhyConn ASegClass 'Segment_1' PhyConn
call AddLink PhyConn ASegClass 'Segment_2' PhyConn
                                  /* Remove PhyConn links between the*/
                                  /* Bridge and the 2 Segments       */
call MakeRODMCall 'DELINKAB'                 /*Call RODM             */
⋮
```

図 162. サンプル FLCSX11

オブジェクトの削除

651 ページの図 163 では、DELOBJ 関数を使って Demo_Lan オブジェクトを削除します。 CHILDREN パラメーターは、Demo_Lan オブジェクトの子オブジェクトも削除されることを指定します。

```
⋮

call StartObject ALnmClass 'Demo_Lan'        /*Which object we are   */
                                             /*referring to.         */
call MakeRODMCall 'DELOBJ' 'CHILDREN=YES'    /*Call RODM             */
⋮
```

図 163. サンプル FLCSX8

IndexList フィールドの処理

SetIndexList サブルーチンは、IndexList フィールドを変更するのに使用します。

図 164. は、IndexList タイプ・フィールドの変更例です。 Demo_Lan オブジェクトの ExceptionViewList フィールドは、テストの値で更新されます。

注: IndexList タイプ・フィールドの更新時には注意してください。この関数はフィールドの前の値を上書きし、その前の値が失われるためです。

```
⋮
call StartObject ALnmClass 'Demo_Lan'      /*Which object we are    */
                                           /*referring to.          */
my_String = 'testing'
call SetIndexList my_String ExceptionViewList

call MakeRODMCall 'UPDATE'                 /*Call RODM              */
⋮
```

図 164. サンプル FLCSX22

結果ステム

結果ステムは、FLCARODM コマンドが実行されるたびに戻されます。結果ステムのフォーマットは、実行される操作、および操作が正常に完了したかどうかに依存します。

どの結果ステムでも、最初の 2 つのエレメント (0 および 1) には、常に同じ情報が含まれます。 0 エレメント (RodmResult.0) には、ステム中のエレメントの合計数、 1 エレメントには指定された順序で以下の情報が含まれます。

1. FLCARODM 戻りコード
2. RODM 戻りコード
3. RODM 理由コード

例えば、FLCARODM コマンドが BUILD 関数を指定して発行され、コマンドがエラーもなく正常に完了したと想定します。次の結果ステムが戻されます。

```
1
FLCARODM:0,0,0
```

1 は、結果ステムに 1 つのエレメントが含まれていること、 FLCARODM:0,0,0 は FLCARODM コマンドが FLCARODM または RODM エラーなしで完了したことを示します。

FLCARODM 戻りコードの説明については、 659 ページの『戻りコード』を参照してください。 RODM 戻りコードおよび理由コードの説明については、 515 ページの『RODM の戻りコードと理由コード』を参照してください。

以下のセクションでは、操作が正常に実行された場合と、失敗した場合の結果ステムを説明しています。

正常に完了した操作の結果ステム

このセクションでは、エラーを出さずに完了する操作を説明しています。エラー条件の詳細については、 655 ページの『ERROR CONDITIONS』を参照してください。

正常に完了した BUILD、UPDATE、DELETE、および PURGE 操作の結果ステム: エラーを出さない BUILD、UPDATE、DELETE、および PURGE 操作の場合、結果ステムのフォーマットは次のとおりです。

エレメント	エレメント値
RodmResult.0	1
RodmResult.1	FLCARODM:0,0,0

1 は、結果ステムに 1 つのエレメントが含まれていること、 `FLCARODM:0,0,0` は FLCARODM コマンドが FLCARODM または RODM エラーなしで完了したことを示します。

正常に完了した照会操作の結果ステム: 正常に完了した照会操作の結果ステムの構造は、照会されるフィールドのデータ・タイプ、および XREF パラメーターが指定されたかどうかに依存します。

QUERY 関数の実行中にエラーが発生せず、XREF パラメーターが指定されなかった場合、結果ステムのフォーマットは次のとおりです。

表 238.

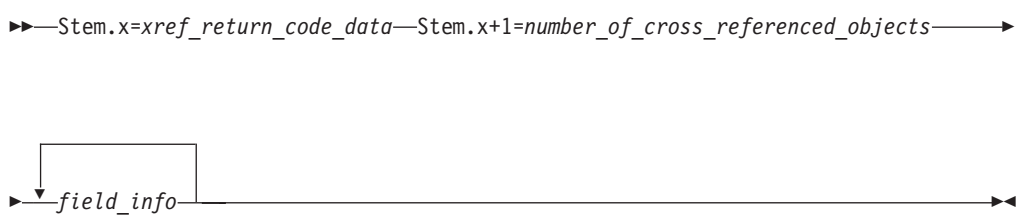
エレメント	エレメント値	説明
RodmResult.0	*x*	結果ステム中のエレメント数
RodmResult.1	FLCARODM:0,0,0	FLCARODM 戻りコード、RODM 戻りコードおよび理由コード
RodnResult.1	10	フィールドに含まれるデータ

QUERY 関数の実行中にエラーが発生せず、XREF パラメーターが指定された場合、結果ステムのフォーマットは少し異なります。各オブジェクトごとに、相互参照フィールド照会が正常か失敗かを示す付加的な戻りコードがあり、続いて相互参照されたオブジェクトの数が示されます。

ここで、

elements
結果ステム中のエレメントの合計数。

xref_field_info
相互参照されるフィールドの戻りコード・データを含む構造、相互参照されるオブジェクトの数、および各オブジェクトの照会結果。 req_field_info 構造のフォーマットは次のとおりです。



ここで、

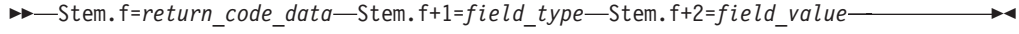
xref_return_code_data
相互参照フィールドの照会に関する戻りコード・データ。

number_of_cross_referenced_objects
相互参照フィールドの照会の結果のオブジェクト数を示す数値。

field_info
各相互参照オブジェクトで照会された各フィールドの戻りコード・データ、フィールド ID、およびフィールド値を含む構造。 field_info 構造は、照会されたフィールドのフィールド・タイプに依存します。このフィールド・タイプは、常に field_info 構造の 2 番目のエレメントにあります。

数値、および文字データの場合、field_info フォーマットは次のとおりです。

数値および文字:



ここで、

return_code_data
エラーが発生しなかったことを示すデータ

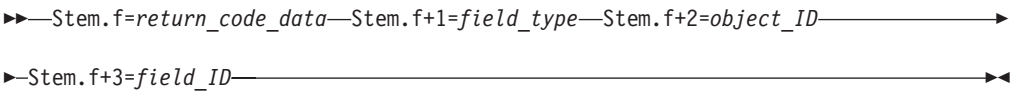
field_type
INTEGER (10) などの数値タイプ、または CHARVAR (4) などの文字タイプのどちらかを示す 10 進値

field_value
フィールドに含まれる数値または文字データ

例えば、オブジェクトの他のデータ・フィールドを照会した結果は次のようになります。

```
FLCARODM:0,0,0
4
Constructed In 1889
```

OBJECTLINK: OBJECTLINK データ・タイプのフィールドの場合、結果ステムのフォーマットは次のとおりです。



ここで、

return_code_data
エラーが発生しなかったことを示すデータ。

field_type
データ・タイプが OBJECTLINK であることを示す 10 進値 (16)。

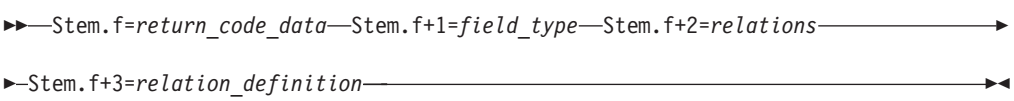
object_ID
フィールドのリンク先のオブジェクトの、16 進数でのオブジェクト ID。

field_ID
照会されたフィールドのリンク先のフィールドの、10 進数でのフィールド ID。

例えば、オブジェクトの objectlink フィールドを照会した結果は次のようになります。

```
FLCARODM:0,0,0
16
00010012E05C2A1E
5
```

OBJECTLINKLIST: OBJECTLINKLIST データ・タイプのフィールドの場合、結果ステムのフォーマットは次のとおりです。



ここで、

return_code_data
エラーが発生しなかったことを示すデータ

field_type
データ・タイプが OBJECTLINKLIST であることを示す 10 進値 (17)。

relations
照会されたフィールドへの関係の数

relation definition
照会されたフィールドを使って、オブジェクトにリンクされるオブジェクトに関する情報

フォーマットは次のとおりです。

オブジェクト ID とフィールド ID は、示される関係の数が表示されるまで繰り返すことができます。

ここで、

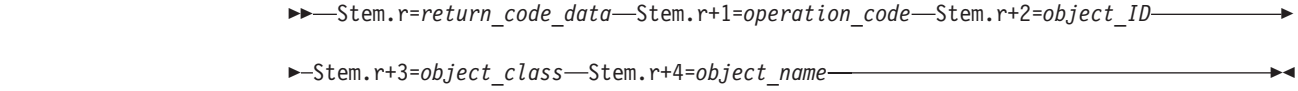
object_ID
フィールドの関連先のオブジェクトの、16 進数でのオブジェクト ID。

field_ID
照会されたフィールドの関連先のフィールドの、10 進数でのフィールド ID。

例えば、オブジェクトの ObjectLinkList フィールドを照会した結果は次のようになります。

```
FLCARODM:0,0,0
17
2
00010012E05C2A1E
5
00010012E05C2A1F
6
```

ERROR CONDITIONS: エラー条件の場合、結果ステムのフォーマットは、実行された操作、およびエラーの発生位置に依存します。エラーの状態にかかわらず、次の 5 種類の情報が常に戻されます。



ここで、

return_code_data
次のフォーマットです。

```
FLCARODM FLCARODM_return_code RODM_return_code RODM_reason_code
```

FLCARODM_return_code は、FLCARODM コマンド・プロセッサーからの戻りコードです。値 2000 はエラーが RODM で発生したこと、および RODM_return_code と RODM_reason_code を検査する必要があることを示します。他の戻りコード値の定義については、659 ページの『戻りコード』を参照してください。詳細については、「*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。

operation_code
エラーの発生時に、FLCARODM が実行しようとしていた操作。
FLCARODM は、要求される関数ごとにいくつかの異なる操作を実行することがあります。 FLCARODM 操作については後述します。

object_ID
FLCARODM 操作が失敗したオブジェクトの、16 進数での RODM オブジェクト ID。不明の場合、NULL です。

object_class
FLCARODM 操作が失敗したオブジェクトの RODM オブジェクト・クラス。不明の場合、NULL です。

object_name
FLCARODM 操作が失敗したオブジェクトの RODM オブジェクト名。不明の場合、NULL です。

Locate: Locate の結果ステムのフォーマットは、オブジェクト ID リストの QUERY 関数の結果ステムと同じです。 Locate のエラー条件は、オブジェクト・クラスおよびオブジェクト名が NULL 値であることを除けば、同じです。

マルチシステム・マネージャーの操作

FLCARODM が実行する操作は次のとおりです。

```
Operation Id                          Operation
------------          ------------------------------------------------
    000               No Operation Determined
    100               Create An Object
    101               Delete An Object
    102               Delete An Object And Its Children
    103               Delete An Object's Children
    104               Execute Purge Against An Object
    200               Change A Field, Creating The Object If Necessary
    201               Change A Field, Only If The Object Exists
    202               Query A Field On An Object
    203               Change A Field On A Child Object
    300               Define A Relation, Creating The Object If Necessary
    301               Define A Relation, Only If The Object Exists
    302               Delete A Relation
    303               Delete All Relations To A Field On Children Objects
    304               Delete All Relations To A Field On An Object
    401               Locate
```

操作 ID 000、100、101、102、103、および 104 の場合、前に説明された情報以外の追加情報はありません。例えば、存在しないオブジェクト・クラスで、単一のオブジェクトの BUILD を次のように試みたとします。

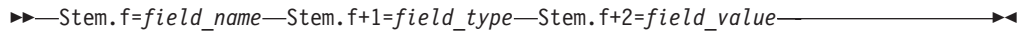
```
⋮

call StartObject 'NoClass' 'Dave'           /*Which object we are   */
                                            /*referring to.         */
call MakeRODMCall 'BUILD'                   /*Call RODM             */
⋮
```

次のエラー・ステムが戻されます。

```
FLCARODM:2000,8,52
100
0000000000000000
No_Class
Dave
```

戻される情報は、(No_Class) クラスで (Dave) という名前のオブジェクトを作成 (100) しようとしたときに、エラーが発生した (2000,8,52) ことを示します。戻りコード 2000 は、エラーは RODM エラーであったことを示します。 RODM 戻りコード/理由コード (8/52) の記述は、参照されるオブジェクト・クラス No_Class が存在しないことを表します。したがって、発生したエラーの完全な記述が戻されます。この単純な例の場合、これは必要以上の情報に見えるかもしれませんが、FLCARODM は複数のフィールドおよび関係を使って、複数のオブジェクトで複数の操作をサポートするため、より複雑な呼び出しにはこの程度詳細な説明が必要になります。

操作 ID 200、201、および 203 の場合、操作されたフィールドに関連する詳細も戻されます。フィールド情報のフォーマットは次のとおりです。



ここで、

field_name
操作が実行されるフィールド名またはフィールド ID

field_type
操作が実行されるフィールドのデータ・タイプ

field_value
操作が実行されるフィールドに指定されるフィールド値

エラーの例を次に示します。

```
FLCARODM:1048,0,0
200
0000000000000000
GMFHS_Managed_Real_Objects_Class
1000_Main_Street
DisplayStatus
Integer
129
```

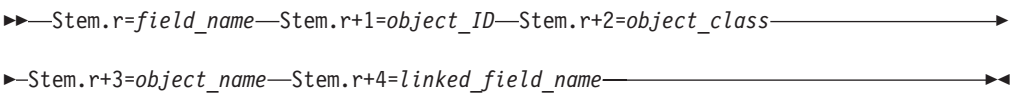
戻される情報は、タイプ (Integer) のフィールド (DisplayStatus) を、(GMFHS_Managed_Real_Objects_Class) クラスにある (1000_Main_Street) という名前のオブジェクト上で値 (129) に変更 (200) しようとしていた時にエラーが発生した (1048,0 0) ことを示します。戻りコード 1048 は、指定されたフィールド・タイプが無効であることを示します。フィールド・タイプはデータ・タイプを表す 10 進値のはずですが、ワード Integer が指定されました。これは誤りです。 10 進値の 10 を使用します。

操作 ID 202、303、および 304 の場合、操作が実行されていたフィールドも戻されます。唯一の追加データはフィールド名またはフィールド ID で、フィールド・タイプおよびフィールド値はこれらの操作に適用されないため表示されません。次に、存在しないフィールドを照会しようとした場合に戻される結果を示します。

```
FLCARODM:2000,4,56
202
0000000000000000
GMFHS_Managed_Real_Objects_Class
1000_Main_Street
My_New_Field
```

戻される情報は、(GMFHS_Managed_Real_Objects_Class) クラスにあるフィールド (My_New_Field) を照会 (202) しようとしたときに、エラーが発生した (2000,4 56) ことを示します。 RODM エラーは、フィールドが GMFHS_Managed_Real_Objects_Class クラスに定義されないために発生しました。

操作 ID 300、301、および 302 の場合、操作が実行されていた関係も戻されます。関係データのフォーマットは次のとおりです。



ここで、

field_name
別のオブジェクトに関連させるのに使用されているフィールド名またはフィールド ID

object_ID
前のオブジェクトに関連するオブジェクトのオブジェクト ID

object_class
前のオブジェクトに関連するオブジェクトのクラス

object_name
前のオブジェクトに関連するオブジェクトの名前

linked_field_name
前のオブジェクトに関連させるのに使用されているオブジェクト上のフィールドの名前

次に、存在しない別のオブジェクトにオブジェクトを関連付けようとした場合に戻される結果を示します。

```
FLCARODM:2000,?,??
301
00010012E05C2A1E
GMFHS_Managed_Real_Objects_Class
1000_Main_Street
PhysicalConnPP
0000000000000000
GMFHS_Managed_Real_Objects_Class
Not_Defined_Yet
PhysicalConnPP
```

戻される情報は、物理関係 (PhysicalConnPP) を定義する、 2 つの実際のオブジェクト (1000_Main_Street) および (Not_Defined_Yet) をリンク (301) しようとしたときに、エラーが発生した (2000,?,??) ことを示します。

前述のとおり、エラー情報が非常に詳細なのは、 RODM エラー (2000 から 2999 の間の FLCARODM 戻りコード) が見つかった場合に FLCARODM が先行するためです。 FLCARODM 自体が入力データが破壊されていることを判別する場合、または内部エラーが発生する場合は先行しません。したがって、一度の FLCARODM 呼び出しで次のエラーが出力されます。

```
FLCARODM:2000,8,52
100
0000000000000000
No_Class
Dave
FLCARODM:2000,4,56
202
0000000000000000
GMFHS_Managed_Real_Objects_Class
1000_Main_Street
My_New_Field
FLCARODM:2000,?,??
301
```

```
00010012E05C2A1E
GMFHS_Managed_Real_Objects_Class
1000_Main_Street
PhysicalConnPP
0000000000000000
GMFHS_Managed_Real_Objects_Class
Not_Defined_Yet
PhysicalConnPP
```

これは、FLCARODM 要求の処理中に 3 つのエラーが発生したことを示します。FLCARODM 命令コードは続くデータのフォーマットを定義するので、呼び出しアプリケーションはこの情報をデコードすることができます。

エラーが発生しない場合、前述のとおり、FLCARODM は戻りコード FLCARODM:0,0,0 を、 QUERY を除くすべての操作に送信するだけです。 QUERY の場合、個々の戻りコードは照会されるすべてのフィールドに送信され、正常に処理が行われ、データが入っていることを示すか、または失敗したことをその理由と共に示します。これによって、呼び出しアプリケーションは、照会が正常に行われたフィールドと、失敗したフィールドとを判別することができます。その後、必要に応じて、アプリケーションは正常に行われた照会の情報を取り出し、失敗した照会を処理することができます。例えば、次のようになります。

```
FLCARODM:0,0,0
4
Constructed In 1889
FLCARODM:0,0,0
16
00010012E05C2A1E
5
FLCARODM:2000,4,56
202
0000000000000000
GMFHS_Managed_Real_Objects_Class
1000_Main_Street
My_New_Field
FLCARODM:0,0,0
17
2
00010012E05C2A1E
5
00010012E05C2A1F
6
```

これは、最初のフィールドが正常に照会され、Constructed In 1889 という値を持つ文字フィールドがあることを示します。 2 番目の照会済みフィールドはオブジェクト・リンクで、オブジェクト ID およびフィールド ID が戻されます。 3 番目の照会済みフィールドの結果はエラー (2000,4,56) で、エラー情報が戻されます。 4 番目の照会済みフィールドはオブジェクト・リンク・リストで、オブジェクトに関連する情報が戻されます。 3 番目のフィールドの照会がエラーだった場合でも、FLCARODM は続行し、4 番目のフィールドに関するデータを戻すことに注意してください。

戻りコード

FLCARODM 戻りコードを以下に説明します。

1000 オブジェクト・データが見つかりませんでした。コマンドが NetView PIPE コマンドを使って発行されなかったか、または PIPE データ・ストリームで何も見つからなかったかのどちらかです。

1004 無効な関数が要求されました。有効な関数は次のとおりです。
- BUILD
- DELINKA
- DELINKAB
- DELOBJ
- PURGE
- QUERY
- UPDATE

1012 指定された RODM 名が NULL だったか、またはその長さが 8 文字を超えました。

1016 指定されたアプリケーション名が NULL だったか、またはその長さが 8 文字を超えました。

1020 指定されたクラスが無効でした。考えられる理由は次のとおりです。
- クラス名の場合、長さが 64 文字を超えていたか、または長さがゼロで、オブジェクト ID が指定されていませんでした。
- クラス ID の場合、# に続く値が数値でなかったか、または値が 4 バイトに収まりませんでした。

1024 指定されたオブジェクトが無効でした。考えられる理由は次のとおりです。
- オブジェクト名の場合、長さが 254 文字を超えていたか、または長さがゼロで、オブジェクト・クラスが指定されていませんでした。
- オブジェクト ID の場合、 # に続くデータの値が 16 進値を表す 16 EBCDIC 文字ではありませんでした。

1028 指定されたオブジェクトの数が正しくないか、または大きすぎます。

1032 指定されたフィールドの数が正しくないか、または大きすぎます。

1036 指定された関係の数が正しくないか、または大きすぎます。

1044 指定されたフィールドが無効でした。考えられる理由は次のとおりです。
- フィールド名の場合、長さが 64 文字を超えていたか、または長さがゼロでした。
- フィールド ID の場合、# に続く値が数値でなかったか、または値が 4 バイトに収まりませんでした。

1048 指定されたフィールド・タイプが正しくないか、または大きすぎます。

1052 指定されたフィールド値が無効でした。フィールド・タイプがフィールド値が数値であることを示す場合、フィールド値が正しくないか、または大きすぎます。フィールド・タイプがフィールド値が文字データであることを示す場合、フィールド値の長さが 254 文字を超えています。

1056 UPDATE または QUERY 操作で、フィールドの値および関係の両方がゼロでした。 UPDATE では、更新のために少なくとも 1 つのフィールドまたはリンクが必要であり、 QUERY では照会のために 1 つのフィールドが必要です。

1060 指定されたフィールド名が NULL だったか、またはその長さが 64 文字を超えました。

1064 指定されたクラス名が NULL だったか、またはその長さが 64 文字を超えました。

1068 指定されたオブジェクト名が NULL だったか、またはその長さが 254 文字を超えました。

1072 指定されたフィールド名が NULL だったか、またはその長さが 64 文字を超えました。

1076 指定された関数の場合、許可されるフィールドはありません。

1080 指定された関数の場合、許可される関係はありません。

1084 照会されたフィールドに戻されるデータ・タイプが、FLCARODM でサポートされていません。

1088 RODMRTRY パラメーターに指定された値が無効です。

1092 RODMINT パラメーターに提供された値が無効です。

1096 CHILDREN パラメーターに提供された値が無効です。

1100 STATUS パラメーターに提供された値が無効です。

1104 TIME パラメーターに提供された値が無効です。

1108 TRACE パラメーターに提供された値が無効です。

1112 指定されたパラメーターが、指定された関数に対して無効か、または権限がありません。

1116 見つかったオブジェクト定義の数が、指定されたオブジェクトの数を下回っています。

1120 すべての予期されるデータが処理されましたが、まだデータが存在します。

1124 オブジェクト定義が完了しませんでした。

1128 見つかったフィールド定義の数が、指定されたフィールドの数を下回っています。

1132 フィールド定義が完了しませんでした。

1136 見つかった関係の数が、指定された数を下回っています。

1140 関係定義が不完全です。

1144 指定されたフィールドの数が、XREF 関数には間違っています。

1148 LOCATE パラメーターに提供された値が無効です。

1152 SF パラメーターに提供された値が無効です。

1156 FIELDID パラメーターに提供された値が無効です。

1160 FILTER パラメーターに提供された値が無効です。

1164 指定された関数に指定されたフィールド定義が多すぎます。

19XX 1900 から 1999 のすべてのエラー・コードは、オブジェクト・データの処理中に、FLCARODM で内部エラーが発生したことを示します。この戻りコードを、関連エラー情報と共に、適切なサービス担当者に報告してください。

2000 要求の処理中に RODM でエラーが発生しました。 RODM 戻りコードおよび理由コードが、より詳細な情報を提供しています。

2004 指定されたオブジェクトには子がありませんでした。 XREF オプションを指定した関数の場合、この戻りコードは、走査すべき関係がなかったことを意味します。

2008 オブジェクトの子で変更されることになっているフィールドが、子オブジェクトに存在しません。 XREF オプションを指定した関数の場合、この戻りコードは、相互参照されたどのオブジェクトにも、フィールドが存在しなかったことを意味します。

4000 示された操作を実行しようとしていたときに、FLCARODM で内部エラーが発生しました。この戻りコードを、関連エラー情報と共に、適切なサービス担当者に報告してください。

4004 FLCARODM は、必要なストレージを取得できません。

4008 FLCARODM が、発生してはならない条件を検出しました。この戻りコードを、関連エラー情報と共に、適切なサービス担当者に報告してください。

4012 リンクを削除する試みがなされましたが、指定されたフィールドのデータ・タイプは、タイプ ObjectLink または ObjectLinkList ではありませんでした。

4016 指定されたオブジェクトで Member または MemberArcs フィールドが定義されていないので、関数をオブジェクトの子で実行することはできません。

4020 フィルター・エラーです。

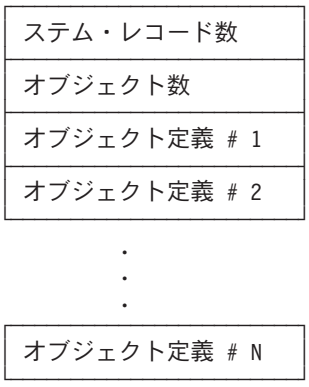
オブジェクト・データ・ストリームの詳細

データ・ストリームは、オブジェクトの作成および更新のために、 RODM にデータを指定するための低レベル手段です。ステム構築ルーチンを使用する開発者は、この低レベルでのデータ・ストリームを指定する必要はありません。

データ・ストリームの解説

データ・ストリームのフォーマットは、REXX ステム (X.0) のレコードの合計数、定義されるオブジェクトの数、各オブジェクト定義の順で構成されます。

データ・ストリームの形式



各オブジェクト定義は、オブジェクト・クラス、オブジェクト名、定義されるフィールドと関係の数、続いてフィールドと関係の定義から構成されます。

各オブジェクト定義の形式

オブジェクト・クラス
オブジェクト名
フィールド数
関係の数
フィールド定義 #1
フィールド定義 #2

.
.
.

フィールド定義 #M
関係定義 #1
関係定義 #2

.
.
.

関係定義 #P

各フィールド定義は、フィールドの名前、フィールド値のデータ・タイプ、およびフィールド値から構成されます。

各フィールド定義の形式

フィールド名
フィールド値のデータ・タイプ
フィールド値

各関係定義は、別のオブジェクトに関連するこのオブジェクトに存在するフィールドの名前、フィールドが関連付けられる先のクラスおよびオブジェクト名、続いて関連オブジェクト上のフィールドから構成されます。

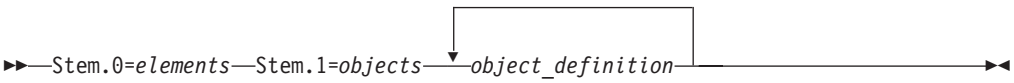
各関係定義の形式

関係に使用するフィールド
関連オブジェクトのクラス
関連オブジェクト名
関連オブジェクト上のフィールド

データ・ストリームは、個々のデータ・ステムから構成されます。

データ・ステムの詳細

このセクションでは、REXX オブジェクト・データ・ステムの形式を詳述します。構造は次の形式です。



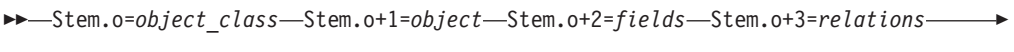
ここで、

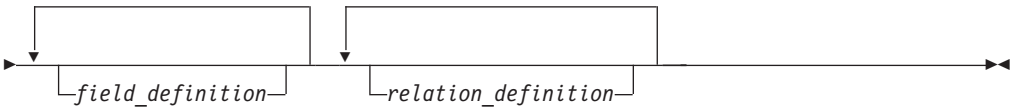
elements
ステム変数に定義されるエレメントの合計数。

objects 操作が実行されるオブジェクトの数。この値は、最低 1 でなければなりません。

object_definition
変更されるオブジェクトを定義します。オブジェクト定義は繰り返すことができ、オブジェクト定義の数は、 *objects* が示す数に等しくなければなりません。

オブジェクト定義: オブジェクト定義の形式は次のとおりです。注: 実際のステム値は変化するので、ステム変数に文字 'o' が使用されています。





ここで、

object_class
操作が行われるオブジェクト・クラス。 オブジェクト ID を指定する場合は、ブランクでなければなりません。 Locate 関数が指定される場合は、NULL でなければなりません。

object 操作が行われるオブジェクトの名前またはオブジェクト ID。オブジェクト ID は、# を接頭部とし、直後に 16 進オブジェクト ID 値を続けることによって指定されます。最初の文字が # でない場合、データはオブジェクト名として解釈されます。オブジェクト ID が指定される場合、オブジェクト・クラスは無視されます。 NULL が指定されると ('')、操作はクラスで実行されます。これは QUERY 操作だけで有効です。 Locate 関数が指定される場合は、NULL でなければなりません。

fields 変更または照会されるオブジェクト上のフィールド数。

relations
作成または除去されるオブジェクト上の関係の数。

field_definition
変更または照会されるフィールドを定義します。フィールド定義は繰り返すことができ、フィールド定義の数は、 *fields* が示す数に等しくなければなりません。

relation_definition
オブジェクト間で作成または削除される関係を定義します。関係定義は繰り返すことができ、定義の数は、 *relations* が示す数に等しくなければなりません。

フィールド定義: フィールド定義の形式は次のとおりです。注: 実際のステム値は変化するので、ステム変数に文字 f が使用されています。

ここで、

field 変更または照会されるフィールドの名前またはフィールド ID。フィールド ID は、# を接頭部とし、直後に 10 進数フィールド ID 値を続けることによって指定されます。最初の文字が # でない場合、データはフィールド名として解釈されます。

field_type
フィールドのデータ・タイプ ID に対応する 10 進整数値。次のデータ・タイプが、BUILD および UPDATE についてサポートされます。

データ・タイプ	データ・タイプ ID
CHARVAR	4
INTEGER	10
SELFDEFINING	19
SMALLINT	21
FIELDID	26
ANONYMOUSVAR	30

BUILD および UPDATE 関数がサポートするデータ・タイプのリストは、631 ページの『BUILD 関数』にあります。

QUERY 関数がサポートするデータ・タイプのリストは、633 ページの『QUERY 関数』にあります。

field_value
フィールドに割り当てられる値。

フィールド・タイプおよびフィールド値は、BUILD、UPDATE、および LOCATE 関数のフィールド定義に必要なコンポーネントです。他の関数には指定してはなりません。 XREF パラメーターが値 1STFIELD を使って指定される場合 (BUILD および UPDATE 関数のみ)、フィールド・タイプおよびフィールド値を、各オブジェクトの最初のフィールドに指定してはなりません。 LOCATE 関数の場合、field_value は比較ストリングです。

関係定義: 関係定義の形式は次のとおりです。注: 実際のステム値は変化するので、ステム変数に文字 r が使用されています。

```
►►—Stem.r=field_to_link—Stem.r+1=object_class_to_link_to———————————►
►—Stem.r+2=object_to_link_to—Stem.r+3=field_to_link_to———————————►◄
```

ここで、

field_to_link
他のフィールドに関係付けられる *object* 上の名前、またはフィールド ID。フィールド ID は、# を接頭部とし、直後に 10 進数フィールド ID 値を続けることによって指定されます。最初の文字が # でない場合、データはフィールド名として解釈されます。

object_class_to_link_to
定義中のオブジェクトに関係付けられるオブジェクトのオブジェクト・クラスの名前。

object_to_link_to
定義中のオブジェクトに関係付けられるオブジェクトの名前またはオブジェクト ID。オブジェクト ID は、# を接頭部とし、直後に 16 進オブジェクト ID 値を続けることによって指定されます。最初の文字が # でない場合、データはオブジェクト名として解釈されます。オブジェクト ID が指定される場合、object_class_to_link_to は無視されます。

field_to_link_to
object で *field_link_to* に関係付ける必要のある *object_to_link_to* の名前またはフィールド ID。フィールド ID は、# を接頭部とし、直後に 10 進数フィールド ID 値を続けることによって指定されます。最初の文字が # でない場合、データはフィールド名として解釈されます。

BLDVIEWS

BLDVIEWS は、集合オブジェクトおよびカスタマイズされたビューを作成するための、REXX EXEC です。 BLDVIEWS を使って、以下のタイプのビューを作成できます。

- 構成バックボーン・ビュー
- 構成論理ビュー
- 構成物理ビュー
- 構成対等機能ビュー
- 例外ビュー
- より詳細な論理ビュー
- より詳細な物理ビュー
- ネットワーク

BLDVIEWS は、制御ステートメントを使って、作成したいビューと集合体の名前、およびビューと集合体を入れたいリソースを指定します。制御ステートメントは、キーワードと値を使用してパラメーターを指定します。リソースの指定時に、RODM クラスまたは RODM 名の形式を知る必要はありません。リソースを指定するには、表示されるリソースの名前 (RODM DisplayResourceName フィールド) を入力します。 ALL またはワイルドカードの名前を指定することもできます。

BLDVIEWS は、RODM 中の既存のリソース (オブジェクト) をビューおよび集合オブジェクトにリンクしたり、既存のリソース上で最も一般的に使用されるサブセ

ットを変更したりするのに使用します。新しいビューおよび集合体を作成したり、既存のビューおよび集合体を更新したりすることができます。 BLDVIEWS は、マルチシステム・マネージャーおよび SNA トポロジー・マネージャーが作成する RODM オブジェクトをサポートします。しかし、BLDVIEWS はこれらのクラスではオブジェクトを作成しません。 BLDVIEWS を使って、GMFHS クラスでリソースを作成してください。

制御ステートメントは、次の方法のいずれかを使用して BLDVIEWS に渡されます。

- DSIPARM メンバー (例えば BLDVIEWS MYMEMBER)
- 完全に修飾されたカタログ付き順次データ・セット (例えば BLDVIEWS ESP.GAF.DATA(MYDEFS))
- PIPE コマンドを使用して収集され、渡されるステム配列 (例えば MyStem.0=2; MyStem.1=VIEW ....; MyStem.2=BRIDGE ...; 'PIPE STEM MyStem. | COLLECT | NETV BLDVIEWS | CONSOLE')

BLDVIEWS は、指定された接頭部を持つビューまたはビューのグループを削除するための、 DELVIEWS と呼ばれる REXX EXEC も提供します。

始める前に

Visual BLDVIEWS (VBV) を使用して、BLDVIEWS 制御ステートメントを生成できます。 VBV は、RODM のビューおよび情報の管理を単純化するアプリケーションです。 VBV は、BLDVIEWS および RODMView に対する、グラフィカルなドラッグ・アンド・ドロップ・インターフェースを提供します。既存の BLDVIEWS ファイルを VBV にインポートできることに注意してください。 VBV の詳細については、VBV のオンライン・ヘルプを参照してください。

BLDVIEWS 制御ステートメントのサンプルは、CNMSAMP データ・セットにあるメンバー FLCVBLDS に入っています。 FLCVBLDS には、さまざまなパラメーターを使用した制御ステートメントのコーディング例があります。

BLDVIEWS 処理

BLDVIEWS は、RODM で指定されたオブジェクトを照会してから、これらのオブジェクトを、指定されたビューまたは集合オブジェクトにリンクします。
BLDVIEWS は、RODM のどのクラスのオブジェクトの特定のフィールドも変更することができ、また GMFHS クラスでオブジェクトを作成することができます。

BLDVIEWS が実行するすべての処理は静的です。 BLDVIEWS の実行時に RODM にあったリソースだけが処理されます。リソースが後で RODM に追加されたり削除されたりする場合、 BLDVIEWS を再実行して、変更をビューに取り込みます。

RODM コレクション・マネージャーは、完全に動的なビューの作成とメインテナンスを提供し、 BLDVIEWS 制御ステートメントと互換性があります。 FLCV2RCM については NetView コマンドのオンライン・ヘルプ、 RODM コレクション・マネージャーについては NetView 管理コンソール オンライン・ヘルプを参照してください。

クラスのすべての組み合わせがサポートされています。

ビュー

BLDVIEWS は、以下のタイプのビューをサポートしています。
- ネットワーク
- 構成対等機能
- 構成バックボーン
- 構成接続性
- より詳細

BLDVIEWS によって、サポートされるビュー・レイアウト・タイプを指定できますが、使用するのは以下のビュー・レイアウト・タイプだけです。
- 階層
- 楕円
- グリッド

集合オブジェクト

AGGregate 制御ステートメントおよび AGGChild 制御ステートメントは、自分の集合リソースを作成し、集合体をリンクさせたいオブジェクトを指定するのに使用します。 BLDVIEWS は、AggregationParent と AggregationChild フィールド、および ComposedOfLogical と IsPartOf フィールドの両方をリンクすることによって、AGGChild リソースを AGGregate リソースにリンクします。

BLDVIEWS 制御ステートメント

以下の制御ステートメントがサポートされています。

ADaPter	LNM アダプター・リソースを指定します。
AGENT	RODM にオブジェクトを作成したマルチシステム・マネージャー・エージェントを指定します。
AGG	集合リソース (GMFHS 集合オブジェクト) を指定します。集合リソースとして、既存のリソースを指定することも、または集合を作成して次の AGGChild 制御ステートメントでリソースを集合リソースにリンクさせ、それを指定することもできます。
AGGCHILD	事前に定義されている集合リソースにリンクさせる、集約子を指定します。
BBVIEW	後に続く制御ステートメント上のリソースを含む構成バックボーン・ビューを定義します。
BRidge	LNM ブリッジの実リソースまたは集合リソースを指定します。
CAU	LNM CAU の実リソースまたは集合リソースを指定します。
CDRM	VTAM CDRM リソースを指定します。
CDRSC	VTAM CDRSC リソースを指定します。
CIRCUIT	APPN 伝送グループ回線、およびサブエリア回線を指定します。

CLASS 後に続く他の制御ステートメント上のリソースを含むグローバル RODM クラスを指定します。この制御ステートメントは他の制御ステートメントに対してのみ使用されて、 RODM クラスを他の制御ステートメントごとに指定する代わりにグローバルに指定します。

CLUSTER MultiSystem Manager または APPNTAM クラスター集合リソースを指定します。

DOMAIN APPN ドメインを指定します。

ENODE APPN エンド・ノードを指定します。

EVIEW 例外ビューを定義します。

注: EVIEW ステートメントの後に指定したオブジェクトが例外ビューに加えられるのは、EVIEW ステートメントに CREATE=Y または CREATE=B を指定した場合のみです。 CREATE=N を指定した場合は、それらのオブジェクトは無視され、例外ビューには加えられません。

GW_NCP NCP ゲートウェイ・リソースを指定します。

HOST_NODE ホスト PU (PU タイプ 5 のノード) を指定します。

IC_NODE APPN 交換ノードを指定します。

INTERFACE TCP/IP アダプター・リソースを指定します。

IP_BRIDGE TCP/IP ブリッジ集合リソースを指定します。

IP_HOST TCP/IP ホスト集合リソースを指定します。

IP_HUB TCP/IP ハブ集合リソースを指定します。

IP_LINK TCP/IP インターフェース・リンク・リソースを指定します。

IP_LOCATION TCP/IP ロケーション集合リソースを指定します。

IP_ROUTER TCP/IP ルーター集合リソースを指定します。

IP_SEGMENT TCP/IP セグメント集合リソースを指定します。

IP_SUBNET TCP/IP サブネットワーク集合リソースを指定します。

IPSPname 後に続く制御ステートメント上のリソースを管理する NetView for AIX サービス・ポイントの VTAM PU、LU、または CP 名を指定します。

LAN_PORT LNM Port を指定します。これは、LNM V2 のためのものです。

LANSPname 後に続く制御ステートメント上のリソースを管理する LNM サービス・ポイントの VTAM PU、LU、または CP 名を指定します。

LCVIEW	後に続く制御ステートメント上のリソースを含む構成の論理接続ビューを定義します。
LINE	VTAM 回線を指定します。
LLINK	論理リンクを指定します。
LNODE	APPN LEN ノードを指定します。
LU	VTAM 論理装置を指定します。
LU_GROUP	VTAM 論理グループを指定します。
MAJNODE	VTAM メジャー・ノードを指定します。
MDLVIEW	後に続く制御ステートメント上のリソースを含む、より詳細な論理ビューを定義します。
MDPVIEW	後に続く制御ステートメント上のリソースを含む、より詳細な物理ビューを定義します。
MIG_DATA_HOST	マイグレーション中のデータ・ホストを指定します。
NCP	NCP リソースを指定します。
NETWORK	MultiSystem Manager または APPNTAM ネットワーク集合リソースを指定します。
NNODE	APPN ネットワーク・ノードを指定します。
NONSNA	非 SNA リソース (GMFHS 管理の実リソース) を指定します。
OTHER	ユーザー作成またはマルチシステム・マネージャー・オープン・クラスのリソースを指定します。
PCVIEW	後に続く制御ステートメント上のリソースを含む構成の物理接続ビューを定義します。
PU	VTAM 物理装置を指定します。
PVIEW	後に続く制御ステートメント上のリソースを含む構成対等機能ビューを定義します。
SEGment	LNM セグメントの実リソースまたは集合リソースを指定します。
SNA	VTAM SNA シャドー・リソースを指定します。
SNA_DOMAIN	後に続く制御ステートメント上のリソースを所有するグローバル VTAM ドメインを定義します。
SNA_PORT	SNA ポートを指定します。
SNALOCALTOPO	APPN SNA ローカル・トポロジー・リソースを指定します。
SYSTEM	システム集合リソースを指定します。
TG	APPN 伝送グループを指定します。
TME_MONITOR	TME® モニター・リソースを指定します。

TME_POLICYREGION	TME ポリシー・リージョン・リソースを指定します。
TME_TMR	処理対象の TME 管理リージョン・リソースを指定します。
TMESPname	後に続く制御ステートメント上のリソースを管理する TME サービス・ポイントの IP 名またはアドレスを指定します。
VIEW	後に続く制御ステートメント上のリソースを含むネットワーク・ビューを定義します。
VRN	APPN 仮想ルーティング・ノードを指定します。
WILDCARD	制御ステートメントにワイルドカード名をコーディングする際に使用するワイルドカード文字を定義します。

制御ステートメントの構文

BLDVIEWS 制御ステートメントには、キーワードと値を使用するフリー・フォームの構文があります。コーディングは任意のカラムから開始できます。先頭ブランクと末尾ブランクは無視されます。特定の制御ステートメントを 1 つ以上の行に渡って指定することができます。継続の方法としては、次の 2 つのタイプがあります。

- 制御ステートメントを複数のステートメントに分割して、keyword=value の後で中断する方法。 keyword=value の後にコンマをコーディングして、残りのパラメーターの指定を次のステートメントで続けます。例えば、次のようになります。

```
BRIDGE=ALL,
       TYPE=AGG,AGGTHRESH=(20%,60%,80%),
       SP=A19SRVCP
```

- 制御ステートメントを複数のステートメントに分割して、コーディング内の任意の場所で中断する方法。 (このタイプの継続は、keyword=value の全体を 1 つのステートメント上にコーディングできない場合に必要となります)。記号 || をコーディングすることにより、キーワードまたは値の中間で中断することができます。例えば、次のようになります。

```
BRIDGE=ALL,
       TYPE=||,
       AGG,
       AGG||,
       THRESH=(20%,||,
       60%,80%),
       SP=A19||,
       SRVCP
```

これらのステートメントは連結されて、記号は除去されます。

注: RODM Collection Manager のインタープリターは、ビューに名前を表示するときに MyName を基にした名前を使用する場合と DisplayResourceName を基にした名前を使用する場合とを区別するための二重等号 (==) の使用をサポートしています。例えば、以下の制御ステートメントは、ビューを作成し、オブジェクトの DisplayResourceName を基にしてオブジェクトをビューに追加します。

```
VIEW=NewView,CREATE=Yes
GENERIC==CommonName,CLASS=My_Object_Class
```

制御ステートメントは以下の場所にコーディングできます。

- NetView DSIPARM メンバー
- 完全修飾カタログ式データ・セット
- NetView の PIPE コマンドを使用して MLWTO に集められてから BLDVIEWS に渡される REXX ステム配列

注: z/OS システムのシンボリックは、BLDVIEWS が処理する制御ステートメントで使用できます。

キーワードは任意のケース (大文字、小文字、または混合) で指定することができ、省略形の使用も可能です。省略構文は、制御ステートメントごとに大文字で記して定義されています。

制御ステートメントが NetView DSIPARM メンバーまたは完全修飾データ・セットでコーディングされている場合、各レコードの最大長は 80 文字です。カラム 73 から 80 は無視されます。制御ステートメントが NetView の PIPE コマンドを使用して BLDVIEWS に渡された場合、レコードのサイズには制限がなく、無視される列はありません。

次のリソース名は常に大文字に変換されます。

- ADaPter
- AGGCHILD (NONSNA、AGG、CLUSTER、およびマルチシステム・マネージャー TCP/IP リソース以外のすべてのリソース)
- nnDomainNetworkCluster 以外のすべての APPNTAM リソース
- すべての SNA トポロジー・マネージャー・リソース
- BRidge
- CAU
- IPSPname
- LANSPname
- TMESPname
- SEGment
- SNA
- SNA_DOMAIN

他のすべてのリソース名は、NMC や各種のエレメント・マネージャーによってコーディングされたとおりに表示されるため、すべて大文字またはすべて小文字でコーディングしてください。

キーワード値は大文字小文字混合でコーディングできます。値がそのまま使用されることもあれば、値が大文字に変換されることもあります。次のキーワードの値は大文字に変換されません。

- CONSOLE
- CORRELATER (NetView V1R3 以降)
- DISPLAY_NAME
- DOMAIN
- 総称コマンド (ACTIVATE、DEACTIVATE、RECYCLE、DISPLAY)
- OTHER_DATA
- USER_DATA

コメントも使用できますが、それは別個のステートメントとして記述しなければなりません。コメント・ステートメントは、カラム 1 に * をコーディングします。

```
* NETA NCPs
NCP=NETA*
```

リソースをビューにリンクしたい場合、 VIEW ステートメントの後にビューにリンクしたいリソース・ステートメントをコーディングします。

```
VIEW=NEWVIEW,CREATE=YES
IP_ROUTER=rtr1.company.com
IP_ROUTER=rtr2.company.com
```

リソースを集合にリンクしたい場合、AGGregate ステートメントの後に、集合にリンクしたい AGGChild リソース・ステートメントをコーディングします。

```
AGGREGATE=NEWAGG,CREATE=YES
AGGCHILD=rtr1.company.com,type=IP_ROUTER
AGGCHILD=rtr2.company.com,type=IP_ROUTER
```

共通の制御ステートメント・パラメーター

次のパラメーターは多くの BLDVIEWS 制御ステートメントに共通のもので、ここに記述された後、それらをサポートする制御ステートメントによって文書内で参照されます。

- AGGPRI
- AGGTHRESH
- COLUMN
- CONSOLE
- CORRELATER
- DISPLAY_NAME
- DISPLAY_STATUS
- OTHER_DATA
- ROW
- TYPE
- UNLINK
- USER_DATA
- ユーザー状況
  - MARK
  - AUTO_IN_PROGRESS
  - SUSPEND
  - SUSPEND_WITH_AUTO_CLEAR
- 総称コマンド:
  - ACTIVATE
  - DEACTIVATE
  - DISPLAY
  - RECYCLE

AGGPRI:

説明: AGGPRI キーワードを使用して、実リソースの集合優先順位を設定または変更します。集合優先順位は、実リソースが不十分になると (集合しきい値には関係なく) 状況が即時に低下する、集合リソースのレベル数です。これにより、重要なリソースに高い優先順位を設定できます。

構文:

```
AGGPRI=x
```

-2	DisplayResourceType デフォルト値を使用します
-1	集約を行いません
0	集約してから、即時に 0 レベル低下させます
1	即時に 1 レベル低下させます
2	即時に 2 レベル低下させます
3	即時に 3 レベル低下させます
4	即時に 4 レベル低下させます
5	即時に 5 レベル低下させます
6	即時に 6 レベル低下させます
7	即時に 7 レベル低下させます
8	即時に 8 レベル低下させます
9	即時に 9 レベル低下させます

例: `AGGPRI=2`

AGGTHRESH:

説明: AGGTHRESH キーワードを使用して、集合リソースの集合しきい値を設定します。集合しきい値は、基礎となるリソースの状況を反映するように集合の状況を変更するときを決めるために使用されます。集合しきい値には、次の 3 つがあります。

- ThresholdDegraded (状況の色は黄色)
- ThresholdSeverelyDegraded (状況の色はピンク)
- ThresholdUnsatisfactory (状況の色は赤)

集合リソースにしきい値が指定されて、集合の状況が変更されるための、集合の下にある不良な実リソースの最小数を示します。

パーセンテージを指定した場合、BLDVIEWS は集合の TotalRealResourceCount フィールドを照会してその値と指定されたパーセンテージとを乗算することにより、新規のしきい値を算出します。

構文:

```
AGGTHRESH=(xxx,yyy,zzz)
```

xxx	1 から 3 桁の ThresholdDegraded
yyy	1 から 3 桁の ThresholdSeverelyDegraded
zzz	1 から 3 桁の ThresholdUnsatisfactory

例: AGGTHRESH=(10#,25%,75%)

使用上の注意:

- しきい値をパーセンテージで指定するためには、数値の前か後に % を付けます。 BLDVIEWS はその値と集合にリンクされた実リソースの合計数とを乗算して、しきい値を算出します。

  しきい値を実際の数で指定するためには、数値の前か後に # を付けます。指定されたしきい値が集合の下にある実リソースの総数よりも大きい場合、しきい値は集合の下にある実リソースの総数に設定されます。

AGGTHRESH キーワード内に実際の数とパーセンテージとを混在させることもできます。

- BLDVIEWS を実行した後にリソースが追加または削除された場合、 BLDVIEWS を再実行してしきい値を再調整する必要があります。

COLUMN:

説明: グリッド・ビュー (レイアウト・タイプ 9) を作成する場合、リソースを画面上のどのカラムに表示するかを指定できます。 COLUMN キーワードを使用してカラムを指定します。

構文: COLUMN=column_on_screen

例: COLUMN=3

使用上の注意:

- COLUMN キーワードがサポートされるのは、ビュー制御ステートメントに続くリソース制御ステートメント上にレイアウト・タイプ 9 (グリッド) で指定された場合だけです。
- COLUMN を指定した場合、ROW も指定しなければなりません。

CONSOLE:

説明: リソースをクリックしてからコマンドを発行できる TELNET などのリモート・コンソール・サポートを活用するか、またはリソースにリモートでログオンすることができます。これはリモート・コンソール・サポートと呼ばれていますが、任意のコマンドを指定することができます。コマンドは NMC コンソール・ワークステーション上で実行します。 BLDVIEWS は指定されたコマンドを RemoteConsole = # および # で囲んでから、 DisplayResourceUserData フィールドを設定します。ユーザーが行う必要があるのは、コマンドを指定することだけです。

DisplayResourceUserData が設定された RODM 内のどのリソースに対しても、リモート・コンソール・フィールドを設定することができます。詳細については、「*IBM Tivoli NetView for z/OS NetView 管理コンソール ユーザーズ・ガイド*」を参照してください。

構文:

```
CONSOLE='command'
```

例:

```
CONSOLE='TELNET.EXE %name%'
```

使用上の注意:

- DisplayResourceUserData フィールドが定義されたどのオブジェクトに対しても、リモート・コンソール・サポートを設定することができます。
- BLDVIEWS は指定されたコマンドを、そのコマンドが正しく実行されるために必要となる適切な制御情報で囲みます。コマンドを完全修飾名 (ドライブおよびパス) で指定するか、またはコマンドが見つかるように PATH が設定されていなければなりません。

- CONSOLE と USER_DATA とは相互に排他的です。
- BLDVIEWS はコマンド・テキスト内の任意の位置にコーディングできる制御変数を提供します。それらの変数は次のとおりです。

  %NAME% リソースの名前に置換されます。この変数はすべてのリソースについてサポートされています。

  %RANDOM% 1 から 5 桁の乱数に置換されます。この変数はすべてのリソースについてサポートされています。

  %SEGMENT% リソースが存在するセグメント番号に置換されます。

  この変数がサポートされるのは、次の制御ステートメントで識別される次のリソースだけです。
  - ADAPTER
  - CAU
  - IP_HOST
  - INTERFACE

  %IPADDRESS% リソースのインターネット・アドレスに置換されます。
- BLDVIEWS を使用すると、次の NetView 変数をコマンド・テキスト内の任意の位置にコーディングできます。

  netid() VTAM ネットワーク ID
  domain() 現行の NetView ドメイン
  opid() NetView オペレーターまたはタスク ID
  cursys() 現行のオペレーティング・システム名
  ecvtpseq() オペレーティング・システムのレベル
  vtam() VTAM のバージョンおよびリリース
  netview() NetView のバージョンおよびリリース
  mvslevel() z/OS バージョンおよびリリース
  opsystem() オペレーティング・システムのタイプ
  sysplex() MVS シスプレックスの名前
- 単一引用符 (') または二重引用符 (") を区切り文字として使用できます。

Correlater:

説明: CORRELATER キーワードを使用して、オブジェクトの Correlater フィールドを設定します。

BLDVIEWS を使用すると、次の NetView 変数を相関テキスト内の任意の位置にコーディングできます。

netid() VTAM ネットワーク ID
domain() 現行の NetView ドメイン
opid() NetView オペレーターまたはタスク ID
cursys() 現行のオペレーティング・システム名
ecvtpseq() オペレーティング・システムのレベル
vtam() VTAM のバージョンおよびリリース
netview() NetView のバージョンおよびリリース
mvslevel() MVS バージョンおよびリリース

opsystem()	オペレーティング・システムのタイプ
sysplex()	MVS シスプレックスの名前

構文: CORRELATER='USA VA RICHMOND'

```
CORRELATER='text'
```

使用上の注意: 単一引用符 (') または二重引用符 (") を区切り文字として使用できます。

DISPLAY_NAME:

説明: SNA、NONSNA、および AGGREGATE ステートメント上にコーディングされたリソースに、 DisplayResourceName フィールドを設定します。 DisplayResourceName フィールドを使用して、リソースにより記述的で使いやすい名前を定義することができます。 DisplayResourceName がリソースに定義されている場合、リソースの実際の RODM 名 (MyName) の代わりにそれが表示されます。

BLDVIEWS 置換変数 %NAME% を新規の DisplayResourceName 値の一部として使用します。 %NAME% 変数はリソースの名前に置換されます。これにより、複数のリソース名を 1 つの制御ステートメントで一度に再フォーマットすることができます。名前の前または後にテキストを追加することができます。

構文:

```
DISPLAY_NAME=xxx
```

例:

```
DISPLAY_NAME=NCP_1
```

DISPLAY_STATUS:

説明: DISPLAY_STATUS キーワードを使用してオブジェクトの状況を設定できます。

構文:

```
DISPLAY_STATUS=xxx
```

129	適合
144	中程度に適合
145	低程度に適合
130	不良
160	中程度に不良
161	低程度に不良
131	中間
132	不明
133	劣化
134	重大劣化
136 から 143	ユーザー状況
152 から 159	ユーザー状況

例:

```
DISPLAY_STATUS=130
```

使用上の注意: 表示状況値 131 は集合オブジェクトではサポートされていません。

表示状況値 133 および 134 は実オブジェクトではサポートされていません。

OTHER_DATA:

説明: OTHER_DATA キーワードを使用して、オブジェクトの RODM DisplayResourceOtherData フィールドを設定します。 DisplayResourceOtherData フィールドは任意の値に設定できます。このフィールドの値は NMC Data1 フィールドに表示されます。

構文:

```
OTHER_DATA='other_data'
```

例:

```
OTHER_DATA='Call 1-800-IBM-HELP for support'
```

使用上の注意: BLDVIEWS を使用すると、以下の NetView 変数を他のデータ・テキスト内の任意の位置にコーディングできます。それらの変数は次のとおりです。

netid()	VTAM ネットワーク ID
domain()	現行の NetView ドメイン
opid()	NetView オペレーターまたはタスク ID
cursys()	現行のオペレーティング・システム名
vtam()	VTAM のバージョンおよびリリース
netview()	NetView のバージョンおよびリリース
mvslevel()	MVS バージョンおよびリリース
opsystem()	オペレーティング・システムのタイプ
sysplex()	MVS シスプレックスの名前

単一引用符 (') または二重引用符 (") を区切り文字として使用できます。

ROW:

説明: 階層ビュー (レイアウト・タイプ 6) またはグリッド・ビュー (レイアウト・タイプ 9) を作成する場合、リソースを画面上のどの行に表示するかを指定できます。 ROW キーワードを使用して、リソースを表示したい画面上の行を指定します。

ROW キーワードがサポートされるのは、ビュー制御ステートメントに続くリソース制御ステートメント上にレイアウト・タイプ 6 (階層) または 9 (グリッド) で指定された場合だけです。

構文:

```
ROW=row_on_screen
```

例:

```
ROW=2
```

UNLINK:

説明: UNLINK キーワードを使用すると、ビューまたは集合を削除して再作成しなくても、ビューまたは集合からリソースを除去することができます。

構文:

```
Syntax: UNLINK
```

例:

```
View=myview
Agg=myagg,unlink
```

USER_DATA:

説明: USER_DATA キーワードを使用して、オブジェクトの DisplayResourceUserData フィールドを設定します。このフィールドの内容は NMC Data2 フィールドに表示されます。 DisplayResourceUserData が設定されたどのリソースに対しても、 User Data フィールドを設定することができます。さらに、DisplayResourceUserData フィールドを任意の値に設定することができます。

BLDVIEWS を使用すると、以下の NetView 変数をユーザー・データ・テキスト内の任意の位置にコーディングできます。それらの変数は次のとおりです。

netid()	VTAM ネットワーク ID
domain()	現行の NetView ドメイン
opid()	NetView オペレーター ID またはタスク ID
cursys()	現行のオペレーティング・システム名
vtam()	VTAM のバージョンおよびリリース
netview()	NetView のバージョンおよびリリース
mvslevel()	MVS バージョンおよびリリース
opsystem()	オペレーティング・システムのタイプ
sysplex()	MVS シスプレックスの名前

構文:

```
Syntax: USER_DATA='user_data'
```

例:

```
USER_DATA=Call x45108 for support
```

使用上の注意:

- 単一引用符 (') または二重引用符 (") を区切り文字として使用できます。
- RODM 内の DisplayResourceUserData フィールドが占有されるために、リモート・コンソール・サポートを使用している場合はこの機能を使用できません。

ユーザー状況:

説明: 次のユーザー状況キーワードを使用して、UserStatus フィールド内の対応するビットを設定します。

- Mark
- Automation in progress
- Suspend

MARK キーワードを使用すると、UserStatus フィールドが定義されたすべてのリソースで、 UserStatus フィールドにマーク・ビットをセットまたはクリアすることが

できます。リソースは RODM に存在していなければなりません。オブジェクトを作成したい場合、まずリソースを作成するために制御ステートメントをコーディングしてから、リソースを更新するために制御ステートメントをコーディングしなければなりません。

AUTO_IN_PROGRESS キーワードを使用すると、 UserStatus フィールドが定義された RODM 内のすべてのリソースで、 UserStatus フィールドに Automation in Progress ビットをセットまたはクリアすることができます。リソースは RODM に存在していなければなりません。オブジェクトを作成したい場合、まずリソースを作成するために制御ステートメントをコーディングしてから、リソースを更新するために制御ステートメントをコーディングしなければなりません。

SUSPEND および SUSPEND_WITH_AUTO_CLEAR キーワードを使用すると、 UserStatus フィールドが定義された RODM 内のすべてのリソースで、 UserStatus フィールドに Suspend ビットを設定することができます。リソースは RODM に存在していなければなりません。オブジェクトを作成したい場合、まずリソースを作成するために制御ステートメントをコーディングしてから、リソースを更新するために制御ステートメントをコーディングしなければなりません。

suspend ユーザー状況フラグをセットすると、集合および例外ビューへの参加に関してリソースが使用不可になります。 Suspend ビットを SUSPEND_WITH_AUTO_CLEAR キーワードと共に UserStatus フィールドにセットした場合、 GMFHS はリソースが満足な状況に戻ったときに Suspend ビットを自動的にクリアします。 Suspend ビットを SUSPEND キーワードと共に UserStatus フィールドにセットした場合、 Suspend ビットを NMC コンソールから手動でクリアするか、または BLDVIEWS を使用する必要があります。

UserStatus ビットの状況は、「Resource Information (リソース情報)」ポップアップ・ウィンドウに表示されます。

総称コマンド:

説明: BLDVIEWS では、オブジェクトに総称コマンドを指定できます。 NMC 総称コマンド機能により、 NMC オペレーターはリソースを選択して次の総称コマンドの 1 つを発行することができます。

- Current Status (DisplayStatusCommandText)
- Activate (ActivateCommandText)
- Inactivate (DeactivateCommandText)
- Recycle (RecycleCommandText)

発行される実際のコマンドは、オブジェクト上のフィールドから検索されます。例えば、Activate コマンドのコマンド・テキストは ActivateCommandText フィールドから検索されます。

構文:

```
ACTivate='activate_command'
DEACTivate'=deactivate_command'
RECYcle='recycle_command'
DISPlay='display_command'
```

例:

```
ACTIVATE='BRG LINK NAME=%NAME%'
DISPLAY=BRG QUERY NAME=%NAME%
```

使用上の注意:

- MultiSystem Manager トークンリング・リソースでは、 BLDVIEWS はコマンドに FLCVBLDV のオペレーター ID および固有の相関値を追加します。
- BLDVIEWS はコマンド・テキスト内の任意の位置にコーディングできる次の制御変数を提供します。

%NAME%	リソースの名前に置換されます。この変数はすべてのリソースについてサポートされています。
%RANDOM%	1 から 5 桁の乱数に置換されます。この変数はすべてのリソースについてサポートされています。
%SEGMENT%	リソースが存在するセグメント番号に置換されます。この変数がサポートされるのは、次の制御ステートメントで識別される次のリソースだけです。 – ADAPTER – CAU – IP_HOST – INTERFACE
%IPADDRESS%	リソースのインターネット・アドレスに置換されます。この変数がサポートされるのは、NWSERVER 制御ステートメントだけです。

- BLDVIEWS を使用すると、次の NetView 変数をコマンド・テキスト内の任意の位置にコーディングできます。

netid()	VTAM ネットワーク ID
domain()	現行の NetView ドメイン
opid()	NetView オペレーターまたはタスク ID
cursys()	現行のオペレーティング・システム名
vtam()	VTAM のバージョンおよびリリース
netview()	NetView のバージョンおよびリリース
mvslevel()	MVS バージョンおよびリリース
opsystem()	オペレーティング・システムのタイプ
sysplex()	MVS シスプレックスの名前

- 単一引用符 (') または二重引用符 (") を区切り文字として使用できます。

ワイルドカード文字の定義

WILDCARD 制御ステートメントを使用して、ワイルドカード文字を定義します。

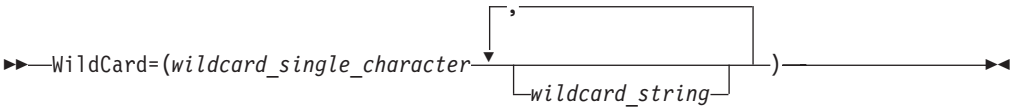
WILDCARD 制御ステートメント:

説明: WILDCARD 制御ステートメントは、RESOURCE 制御ステートメントにワイルドカード・パターン・マッチング名をコーディングする際に使用するワイルドカード文字を定義します。

wildcard_single_character と *wildcard_string* は、ワイルドカード・パターン・マッチング名の定義に使用する特殊文字です。

wildcard_single_character および *wildcard_string* のデフォルト値は * です。

WildCard



パラメーター:

wildcard single_character および *wildcard_string*
ワイルドカード・パターン・マッチング名の定義に使用する特殊文字。

コンマ ',' または等号 '=' 以外の任意の文字を指定できます。
wildcard_single_character と *wildcard_string* のデフォルト文字は、どちらもアスタリスク (*) です。

wildcard_single_character

1 つの文字に対してワイルドカードをマッチングする場合に指定します。ワイルドカード・パターン・マッチング名の特定の位置にコーディングされた *wildcard_single_character* は、リソース名の対応する位置にある文字に常にマッチングします。

wildcard_string がワイルドカード・パターン・マッチング名の最後以外の位置に指定された場合、それは *wildcard_single_character* (指定された位置にある 1 文字に対してワイルドカード・マッチングを実行する) として扱われます。

wildcard_string

リソース名の最後にある残っている複数文字のストリングに対してワイルドカードをマッチングする場合に指定します。ワイルドカード・パターン・マッチング名の最後にコーディングされた *wildcard_string* は、リソース名の対応する位置にある複数文字のストリングと常にマッチングします。

wildcard_string がワイルドカード・パターン・マッチング名の最後以外の位置に指定された場合、それは *wildcard_single_character* (指定された位置にある 1 文字に対してワイルドカード・マッチングを実行する) として扱われます。

例: 次に示すパターン・マッチングの例では、WILDCARD=(?,*) が指定されていると想定します。

パターン・マッチングの例	検出されるマッチング
BRIDGE=A001B*	名前が A001B で始まるすべてのブリッジ・リソースにマッチングします。
BRIDGE=????B001	名前が 8 文字の長さであり、B001 で終わるすべてのブリッジ・リソースにマッチングします。
SEGMENT=?C?0	名前の中で C が位置 2 にあり 0 が位置 4 にあるすべてのセグメント・リソースにマッチングします。

ADP=??SERV*	名前が 6 文字以上の長さであり、 SERV が位置 3 から 6 にあるすべてのアダプター・リソースにマッチングします。
ADP=??PRINTER0?	名前が 10 または 11 文字の長さであり、 PRINTER0 が位置 3 から 10 にあるすべてのアダプター・リソースにマッチングします。

次に示すパターン・マッチングの例では、WILDCARD = *,* (デフォルト) を想定しています。

パターン・マッチングの例	検出されるマッチング
BRIDGE=A001B*	名前が A001B で始まるすべてのブリッジ・リソースにマッチングします。
BRIDGE=**B001**	名前が 8 文字の長さであり、B001 で終わるすべてのブリッジ・リソースにマッチングします。
SEGMENT=*C*0	名前の中で C が位置 2 にあり 0 が位置 4 にあるすべてのセグメント・リソースにマッチングします。
ADP=SERV***	名前が 6 文字以上の長さであり、 SERV が位置 3 から 6 にあるすべてのアダプター・リソースにマッチングします。
ADP=PRINTER0***	名前が 10 文字以上の長さであり、PRINTER0 が位置 3 から 10 にあるすべてのアダプター・リソースにマッチングします。

選択的制御ステートメント

次に示す選択的制御ステートメントによって、BLDVIEWS によって処理されるリソースをより選択的に指定すること、または RODM 内にリソースを配置するために使用する共通情報を指定することが可能になります。この種類の制御ステートメントでは、ワイルドカードを使用できません。

サービス・ポイント制御ステートメント:

説明: サービス・ポイント制御ステートメントは、サービス・ポイント制御ステートメントに続く制御ステートメント上のリソースを管理するサービス・ポイントを指定します。サービス・ポイント制御ステートメントによって、 BLDVIEWS によって処理されるリソースをより選択的に指定することができます。このサービス・ポイント名は、SPname キーワードを使用して個別の制御ステートメント上で指定変更することができます。

次のサービス・ポイント制御ステートメントが使用可能です。

- LANSPname
- TMESPname
- IPSPname

構文:

ATMSPname

パラメーター:

service_point
1 から 8 文字の VTAM PU、LU、CP 名、または IP ホスト名

All
すべてのサービス・ポイントからのリソースを含みます。 All はデフォルトです。

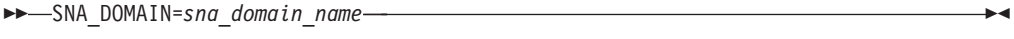
使用上の注意:
- リソース名 ALL の制御ステートメントをコーディングした場合、処理されるリソースはサービス・ポイント制御ステートメントが以前に指定されたかどうかに依存します。
- 以前にサービス・ポイント・ステートメントが指定されていない場合、リソース制御ステートメントがリソース名 ALL でコーディングされていれば、すべてのリソースが処理されます。
- 以前にサービス・ポイント・ステートメントが指定されていない場合、リソース制御ステートメントがワイルドカードのリソース名でコーディングされていれば、ワイルドカード名にマッチングするすべてのリソースが処理されます。
- 以前にサービス・ポイント・ステートメントが指定されている場合、リソース制御ステートメントがリソース名 ALL でコーディングされていれば、そのサービス・ポイントによって管理されるすべてのリソースが処理されます。
- 以前にサービス・ポイント・ステートメントが指定されている場合、リソース制御ステートメントがワイルドカードのリソース名でコーディングされていれば、ワイルドカード名にマッチングして、そのサービス・ポイントによって管理されるすべてのリソースが処理されます。

SNA_DOMAIN 制御ステートメント:

説明: SNA_DOMAIN 制御ステートメントは、SNA_DOMAIN 制御ステートメントに続く制御ステートメント上の SNA トポロジー・マネージャー・リソースを所有する SNA ドメインを指定します。 SNA ドメインを使用して、 RODM 内の SNA トポロジー・マネージャー・リソースを位置指定することができます。デフォルトは ALL です。この値は、SNA_DOMAIN キーワードを使用して個別の制御ステートメント上で指定変更することができます。

構文:

SNA_DOMAIN



パラメーター:

sna_domain_name
network.host_pu_name の形式による、1 から 17 文字の SNA ドメイン名。

network
1 から 8 文字の VTAM ネットワーク名 (VTAM 開始リスト ATCSTRxx 内の NETID パラメーター)

host_pu_name
1 から 8 文字の VTAM ホスト PU 名 (VTAM 開始リスト ATCSTRxx 内の HOSTPU パラメーター)

sna_domain_name が指定されていない場合、ローカル SNA ドメインが使用されます (BLDVIEWS が実行されるドメイン)。

次の SNA トポロジー・リソースには、SNA ドメインが必要です。
- VTAM メジャー・ノード (MAJNODE 制御ステートメント)
- CDRM (CDRM 制御ステートメント)
- CDRSC (CDRSC 制御ステートメント)
- 論理装置 (LU 制御ステートメント)
- 論理装置グループ (LU_GROUP 制御ステートメント)

SNA ドメイン名は、SNA_DOMAIN キーワードを使用してそれらの制御ステートメントに指定することもできます。その場合、それは SNA_DOMAIN 制御ステートメントを指定変更します。

ビュー制御ステートメント

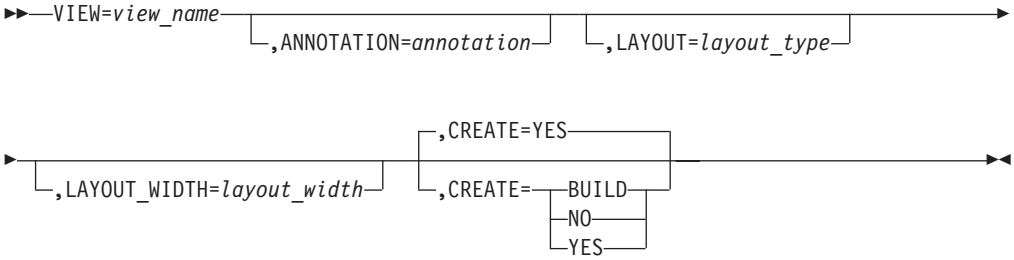
以下のビュー制御ステートメントは、作成するビューのタイプを定義します。

VIEW 制御ステートメント:

説明: VIEW 制御ステートメントは、後に続く制御ステートメント上のリソースを含むネットワーク・ビューを定義します。

構文:

VIEW



パラメーター:

view_name
1 から 32 文字のビューの名前。これは、ネットワーク・ビュー・オブジェクトの MyName です。

annotation
1 から 32 文字のビューの注釈。

layout
ビューの作成時に使用するレイアウト・アルゴリズムを決める、1 桁のレイアウト・タイプ指定。 BLDVIEWS はレイアウト・タイプをサポートしています。しかし、次の値のみが使用されます。
- 6 - 階層 (CREATE=YES のデフォルト)
- 7 - 楕円
- 9 - グリッド

layout_width
ビュー内の 1 行に水平に表示されるリソース・オブジェクトの数を指定する整数。デフォルト値は 0 であり、この場合グリッドは正方形に似たものとなります。これはレイアウト・タイプ 9 のみに適用されます。

CREATE
指定されたリソース上で実行するアクションを指定します。

YES このビューに対する新規のオブジェクトを作成します。古いオブジェクトが存在する場合、それは削除されます。

デフォルトは YES です。

NO このビューに対する新規のオブジェクトを作成しません。既存のオブジェクトを更新します。オブジェクトが存在しない場合、エラーが生じます。

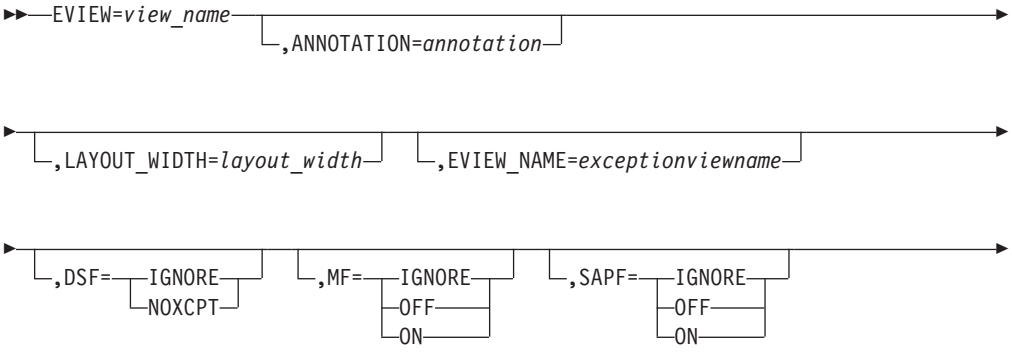
BUILD
このビューに対するオブジェクトが存在しない場合、オブジェクトを新規に作成します。存在する場合、オブジェクトを更新します。

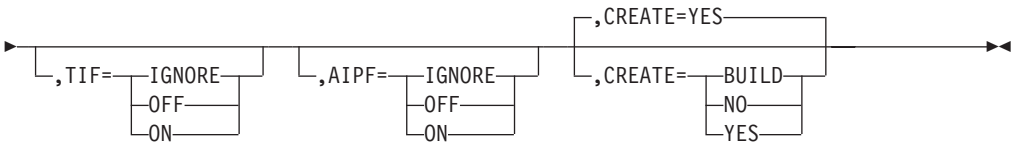
EVIEW 制御ステートメント:

説明: EVIEW 制御ステートメントは、例外ビューを定義します。

構文:

EVIEW





パラメーター:

view_name
1 から 32 文字のビューの名前。これは、Exception View オブジェクトの MyName です。

annotation
1 から 32 文字のビューの注釈。

layout_width
ビュー内の 1 行に水平に表示されるリソース・オブジェクトの数を指定する整数。デフォルト値は 0 であり、この場合グリッドは正方形に似たものとなります。これはレイアウト・タイプ 9 のみに適用されます。

exceptionviewname
例外ビューに関連した 1 から 8 文字の名前。この名前が ExceptionViewList フィールドにあるリソース・オブジェクトは、関連する例外ビューに表示される候補とされます。このフィールドは、例外ビューごとに固有のものでなければなりません。指定されない場合、BLDVIEWS が固有の exceptionviewname を作成します。

DSF
例外ビューに DisplayStatus フィルター・オプションを指定します。

IGNORE
フィルター操作は行われず、DisplayStatus は無視されます。マップされた表示状況が XCPT または NOXCPT のオブジェクトは、このビューの候補です。

NOXCPT
例外状況にマップされない すべてのオブジェクトをフィルターで除外します。

MF
例外ビューに UserStatus Mark フィルター・オプションを指定します。

IGNORE
フィルター操作は行われません。 UserStatus Mark は無視されます。

ON Mark の UserStatus ビットがオンになっているオブジェクトをフィルターで除外します。オブジェクトの UserStatus ビットがオンになっている場合、そのオブジェクトはビューに表示されません。

OFF Mark の UserStatus ビットがオフになっているオブジェクトをフィルターで除外します。オブジェクトの UserStatus ビットがオフになっている場合、そのオブジェクトはビューに表示されません。

SAPF
例外ビューに UserStatus SNA Alert Pending フィルター・オプションを指定します。

IGNORE
フィルター操作は行われません。 UserStatus SNA Alert Pending は無視されます。

ON SNA Alert Pending の UserStatus ビットがオンになっているオブジェクトをフィルターで除外します。オブジェクトの UserStatus ビットがオンになっている場合、そのオブジェクトはビューに表示されません。

OFF SNA Alert Pending の UserStatus ビットがオフになっているオブジェクトをフィルターで除外します。オブジェクトの UserStatus ビットがオフになっている場合、そのオブジェクトはビューに表示されません。

TIF
例外ビューに UserStatus Threshold Inconsistency フィルター・オプションを指定します。

IGNORE
フィルター操作は行われません。 UserStatus Threshold Inconsistency は無視されます。

ON Threshold Inconsistency の UserStatus ビットがオンになっているオブジェクトをフィルターで除外します。オブジェクトの UserStatus ビットがオンになっている場合、そのオブジェクトはビューに表示されません。

OFF Threshold Inconsistency の UserStatus ビットがオフになっているオブジェクトをフィルターで除外します。オブジェクトの UserStatus ビットがオフになっている場合、そのオブジェクトはビューに表示されません。

AIPF
例外ビューに UserStatus Automation In Progress フィルター・オプションを指定します。

IGNORE
フィルター操作は行われません。 UserStatus Automation In Progress は無視されます。

ON Automation In Progress の UserStatus ビットがオンになっているオブジェクトをフィルターで除外します。オブジェクトの UserStatus ビットがオンになっている場合、そのオブジェクトはビューに表示されません。

OFF Automation In Progress の UserStatus ビットがオフになっているオブジェクトをフィルターで除外します。オブジェクトの UserStatus ビットがオフになっている場合、そのオブジェクトはビューに表示されません。

CREATE
指定されたリソース上で実行するアクションを指定します。

YES このビューに対する新規のオブジェクトを作成します。古いオブジェクトが存在する場合、それは削除されます。

デフォルトは YES です。

NO このビューに対する新規のオブジェクトを作成しません。既存のオブジェクトを更新します。オブジェクトが存在しない場合、エラーが生じます。

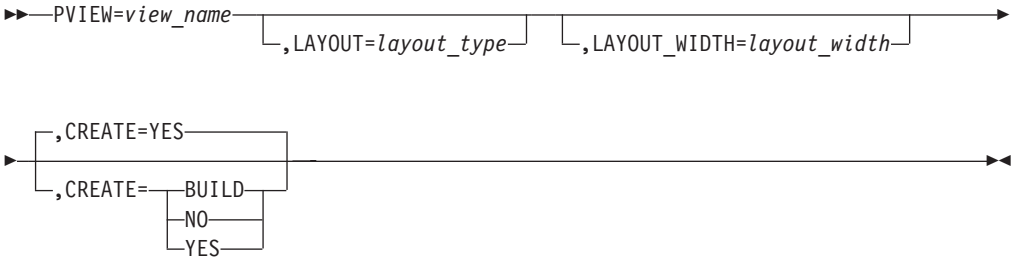
BUILD
このビューに対するオブジェクトが存在しない場合、オブジェクトを新規に作成します。存在する場合、オブジェクトを更新します。

PVIEW 制御ステートメント:

説明: PVIEW 制御ステートメントは、後に続く制御ステートメント上のリソースを含む構成対等機能ビューを定義します。

構文:

PVIEW



パラメーター:

view_name
1 から 32 文字のビューの名前。これは、構成対等機能ビュー・オブジェクトの MyName です。

layout
ビューの作成時に使用するレイアウト・アルゴリズムを決める、1 桁のレイアウト・タイプ指定。 BLDVIEWS はすべてのレイアウト・タイプをサポートしています。しかし、次の値のみが使用されます。
- 6 - 階層 (CREATE=YES のデフォルト)
- 7 - 楕円
- 9 - グリッド

layout_width
ビュー内の 1 行に水平に表示されるリソース・オブジェクトの数を指定する整数。デフォルト値は 0 であり、この場合グリッドは正方形に似たものとなります。これはレイアウト・タイプ 9 のみに適用されます。

CREATE
指定されたリソース上で実行するアクションを指定します。

YES このビューに対する新規のオブジェクトを作成します。古いオブジェクトが存在する場合、それは削除されます。

デフォルトは YES です。

NO このビューに対する新規のオブジェクトを作成しません。既存のオブジェクトを更新します。オブジェクトが存在しない場合、エラーが生じます。

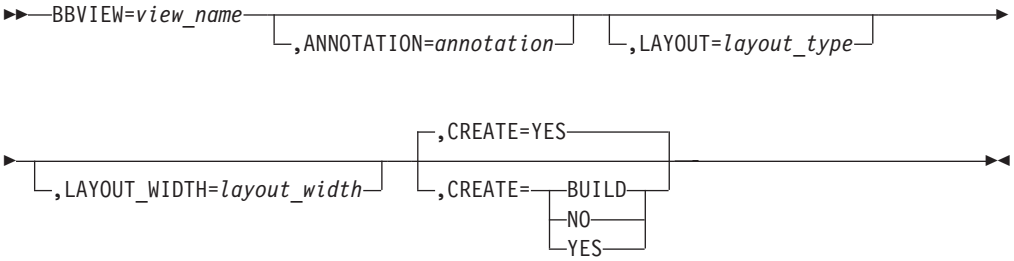
BUILD
このビューに対するオブジェクトが存在しない場合、オブジェクトを新規に作成します。存在する場合、オブジェクトを更新します。

BBVIEW 制御ステートメント:

説明: BBVIEW 制御ステートメントは、後に続く制御ステートメント上のリソースを含む構成バックボーン・ビューを定義します。

構文:

BBVIEW



パラメーター:

view_name
1 から 32 文字のビューの名前。これは、構成バックボーン・ビュー・オブジェクトの MyName です。

annotation
1 から 32 文字のビューの注釈。

layout
ビューの作成時に使用するレイアウト・アルゴリズムを決める、1 桁のレイアウト・タイプ指定。 BLDVIEWS はすべてのレイアウト・タイプをサポートしています。しかし、次の値のみが使用されます。
- 1 - リンク・タイプによる放射レイアウト (CREATE=YES のデフォルト)
- 6 - 階層
- 7 - 楕円
- 9 - グリッド

layout_width
ビュー内の 1 行に水平に表示されるリソースの数を指定する整数。デフォルト値は 0 であり、この場合グリッドは正方形に似たものとなります。これはレイアウト・タイプ 9 のみに適用されます。

CREATE
指定されたリソース上で実行するアクションを指定します。

YES このビューに対する新規のオブジェクトを作成します。古いオブジェクトが存在する場合、それは削除されます。

デフォルトは YES です。

NO このビューに対する新規のオブジェクトを作成しません。既存のオブジェクトを更新します。オブジェクトが存在しない場合、エラーが生じます。

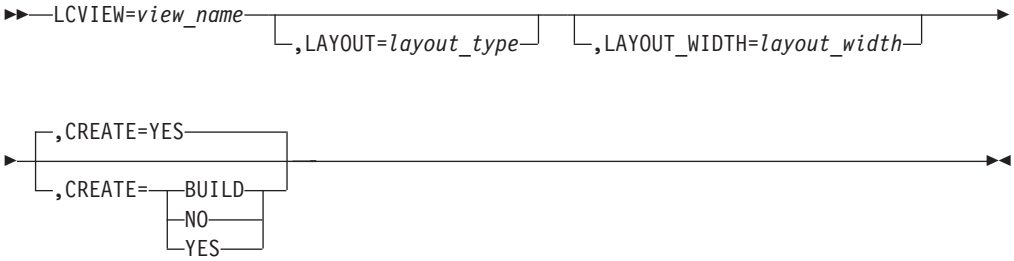
BUILD
このビューに対するオブジェクトが存在しない場合、オブジェクトを新規に作成します。存在する場合、オブジェクトを更新します。

LCVIEW 制御ステートメント:

説明: LCVIEW 制御ステートメントは、後に続く制御ステートメント上のリソースを含む構成論理接続ビューを定義します。

構文:

LCVIEW



パラメーター:

view_name
1 から 32 文字のビューの名前。これは、構成論理接続ビュー・オブジェクトの MyName です。

layout
ビューの作成時に使用するレイアウト・アルゴリズムを決める、1 桁のレイアウト・タイプ指定。 BLDVIEWS はすべてのレイアウト・タイプをサポートしています。しかし、次の値のみが使用されます。
- 1 - リンク・タイプによる放射レイアウト (CREATE=YES のデフォルト)
- 6 - 階層
- 7 - 楕円
- 9 - グリッド

layout_width
ビュー内の 1 行に水平に表示されるリソースの数を指定する整数。値は 0 であり、この場合グリッドは正方形に似たものとなります。これはレイアウト・タイプ 9 のみに適用されます。

CREATE
指定されたリソース上で実行するアクションを指定します。

YES このビューに対する新規のオブジェクトを作成します。古いオブジェクトが存在する場合、それは削除されます。

デフォルトは YES です。

NO このビューに対する新規のオブジェクトを作成しません。既存のオブジェクトを更新します。オブジェクトが存在しない場合、エラーが生じます。

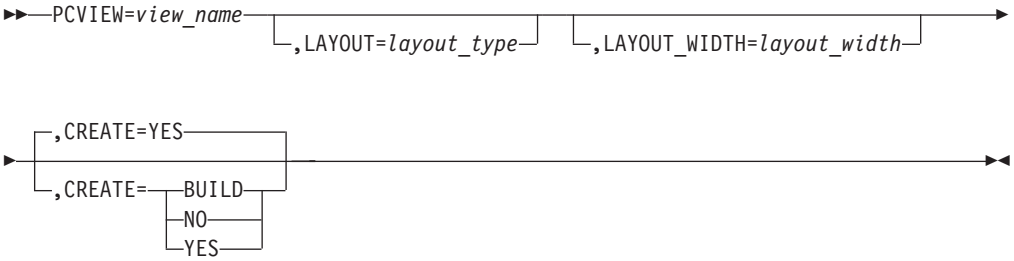
BUILD
このビューに対するオブジェクトが存在しない場合、オブジェクトを新規に作成します。存在する場合、オブジェクトを更新します。

PCVIEW 制御ステートメント:

説明: PCVIEW 制御ステートメントは、後に続く制御ステートメント上のリソースを含む構成の物理接続ビューを定義します。

構文:

PCVIEW



パラメーター:

view_name
1 から 32 文字のビューの名前。これは、構成物理接続ビュー・オブジェクトの MyName です。

layout
ビューの作成時に使用するレイアウト・アルゴリズムを決める、1 桁のレイアウト・タイプ指定。 BLDVIEWS はすべてのレイアウト・タイプをサポートしています。しかし、次の値のみが使用されます。
- 1 - リンク・タイプによる放射レイアウト (CREATE=YES のデフォルト)
- 6 - 階層
- 7 - 楕円
- 9 - グリッド

layout_width
ビュー内の 1 行に水平に表示されるリソースの数を指定する整数。値は 0 であり、この場合グリッドは正方形に似たものとなります。これはレイアウト・タイプ 9 のみに適用されます。

CREATE
指定されたリソース上で実行するアクションを指定します。

YES このビューに対する新規のオブジェクトを作成します。古いオブジェクトが存在する場合、それは削除されます。

デフォルトは YES です。

NO このビューに対する新規のオブジェクトを作成しません。既存のオブジェクトを更新します。オブジェクトが存在しない場合、エラーが生じます。

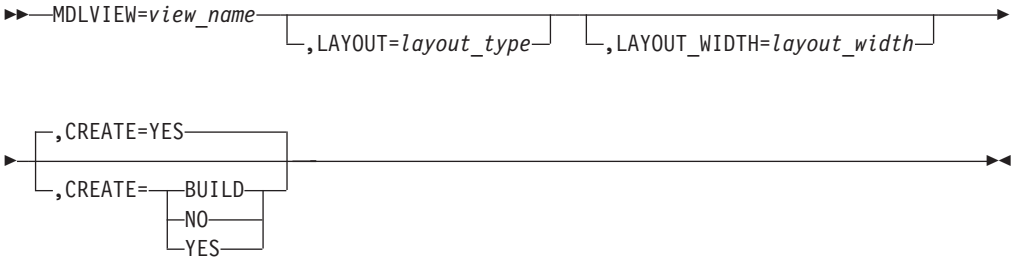
BUILD
このビューに対するオブジェクトが存在しない場合、オブジェクトを新規に作成します。存在する場合、オブジェクトを更新します。

MDLVIEW 制御ステートメント:

説明: MDLVIEW 制御ステートメントは、後に続く制御ステートメント上のリソースを含む、より詳細な論理ビューを定義します。

構文:

MDLVIEW



パラメーター:

view_name
1 から 32 文字のビューの名前。これは、より詳細な論理ビュー・オブジェクトの MyName です。

layout
ビューの作成時に使用するレイアウト・アルゴリズムを決める、1 桁のレイアウト・タイプ指定。 BLDVIEWS はすべてのレイアウト・タイプをサポートしています。しかし、次の値のみが使用されます。
- 1 - リンク・タイプによる放射レイアウト (CREATE=YES のデフォルト)
- 6 - 階層
- 7 - 楕円
- 9 - グリッド

layout_width
ビュー内の 1 行に水平に表示されるリソースの数を指定する整数。値は 0 であり、この場合グリッドは正方形に似たものとなります。これはレイアウト・タイプ 9 のみに適用されます。

CREATE
指定されたリソース上で実行するアクションを指定します。

YES このビューに対する新規のオブジェクトを作成します。古いオブジェクトが存在する場合、それは削除されます。

デフォルトは YES です。

NO このビューに対する新規のオブジェクトを作成しません。既存のオブジェクトを更新します。オブジェクトが存在しない場合、エラーが生じます。

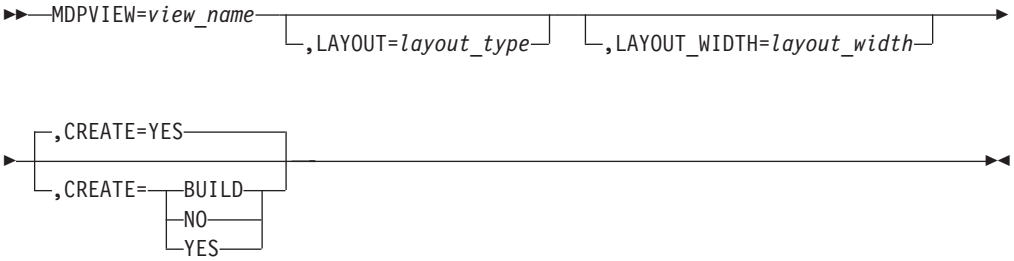
BUILD
このビューに対するオブジェクトが存在しない場合、オブジェクトを新規に作成します。存在する場合、オブジェクトを更新します。

MDPVIEW 制御ステートメント:

説明: MDPVIEW 制御ステートメントは、後に続く制御ステートメント上のリソースを含む、より詳細な物理ビューを定義します。

構文:

MDPVIEW



パラメーター:

view_name
1 から 32 文字のビューの名前。これは、より詳細な物理ビュー・オブジェクトの MyName です。

layout
ビューの作成時に使用するレイアウト・アルゴリズムを決める、1 桁のレイアウト・タイプ指定。 BLDVIEWS はすべてのレイアウト・タイプをサポートしています。しかし、次の値のみが使用されます。
- 1 - リンク・タイプによる放射レイアウト (CREATE=YES のデフォルト)
- 6 - 階層
- 7 - 楕円
- 9 - グリッド

layout_width
ビュー内の 1 行に水平に表示されるリソースの数を指定する整数。値は 0 であり、この場合グリッドは正方形に似たものとなります。これはレイアウト・タイプ 9 のみに適用されます。

CREATE
指定されたリソース上で実行するアクションを指定します。

YES このビューに対する新規のオブジェクトを作成します。古いオブジェクトが存在する場合、それは削除されます。

デフォルトは YES です。

NO このビューに対する新規のオブジェクトを作成しません。既存のオブジェクトを更新します。オブジェクトが存在しない場合、エラーが生じます。

BUILD
このビューに対するオブジェクトが存在しない場合、オブジェクトを新規に作成します。存在する場合、オブジェクトを更新します。

リソース制御ステートメント

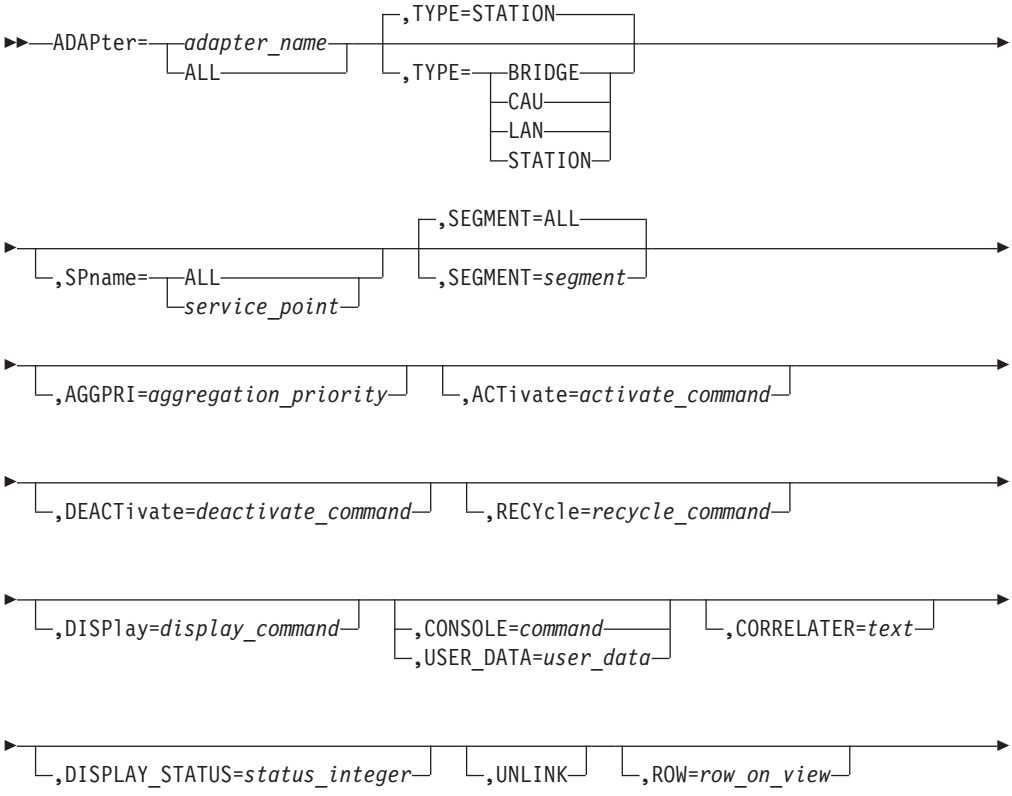
以下のリソース制御ステートメントは、BLDVIEWS によって処理されるリソースを指定します。

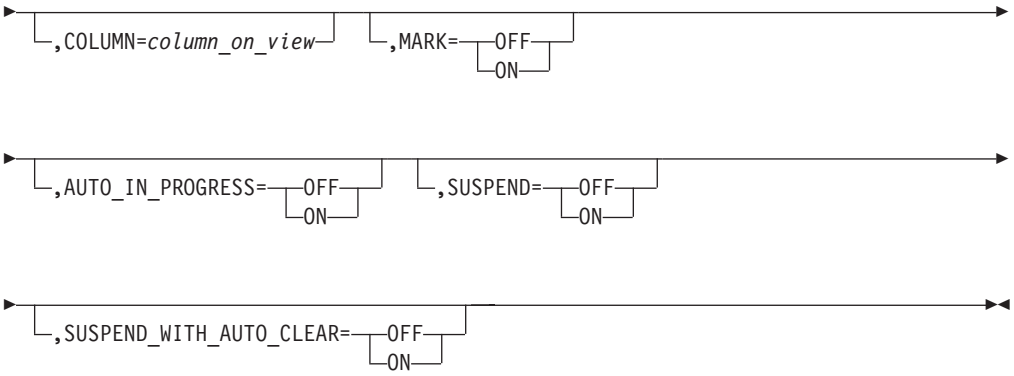
ADAPter 制御ステートメント:

説明: ADAPter 制御ステートメントは、処理されるマルチシステム・マネージャー LNM アダプター・リソースを指定します。 LNM ポート (LAN Network Manager V2 でサポートされている) については、 LAN_PORT 制御ステートメントを参照してください。

構文:

ADAPter





パラメーター:

adapter_name

STATION、BRIDGE、CAU、または LAN

- 1 から 12 文字のアダプター MAC アドレス、または
- 1 から 16 文字のアダプター MAC 名

ALL またはワイルドカード名を指定できます。

TYPE

アダプター・リソースのタイプを指定します。値は次のとおりです。

- STATION - アダプターはステーション・アダプターです (デフォルト)
- BRIDGE - アダプターはブリッジ・アダプターです
- CAU - アダプターは Controlled Access Unit アダプターです
- LAN - アダプターは LNM アダプターです (STATION、BRIDGE、または CAU にすることが可能)

segment_name

STATION、BRIDGE、CAU、または LAN、セグメント番号 (3 から 4 文字) またはセグメント名 (SEGxxxx など)。

ALL も指定可能で、これがデフォルトです。

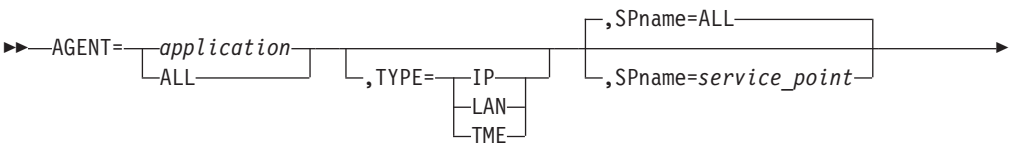
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

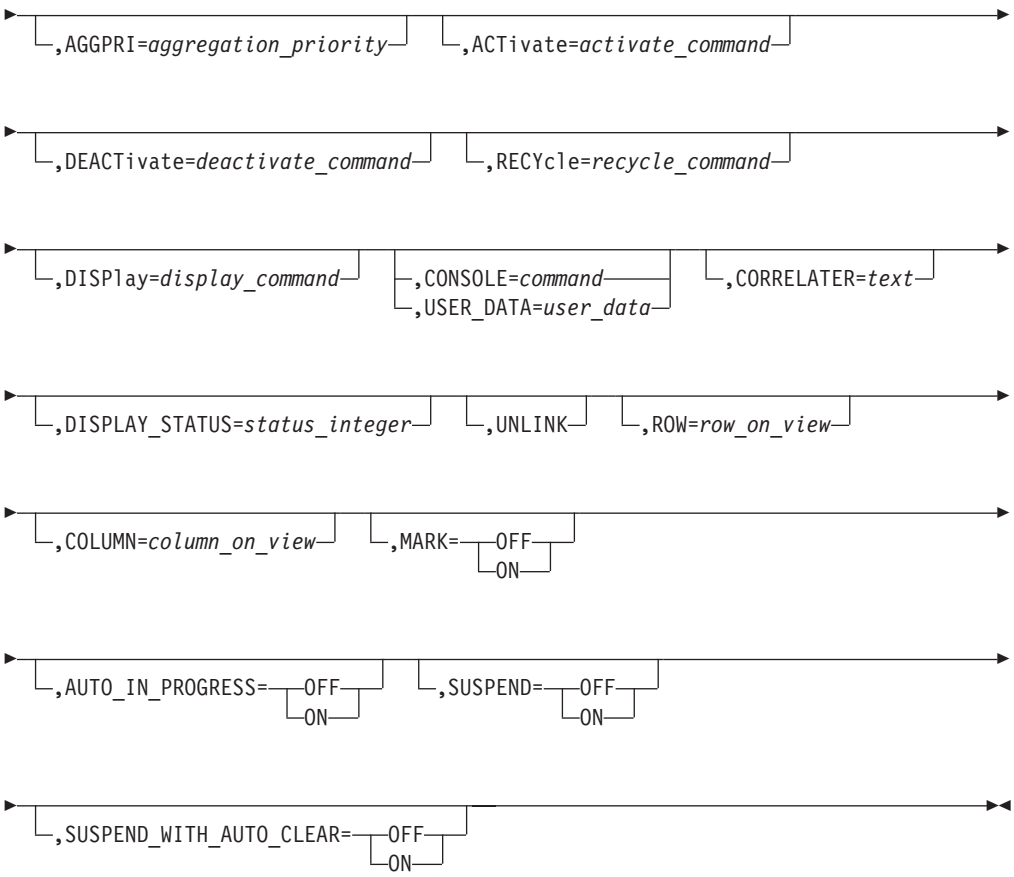
AGENT 制御ステートメント:

説明: AGENT 制御ステートメントは、処理されるマルチシステム・マネージャー・エージェント・リソースを指定します。

構文:

AGENT





パラメーター:

application
1 から 8 文字のサービス・ポイント・アプリケーション名。
- LAN Network Manager エージェントのアプリケーション名は LANMGR です。
- NetView for AIX のエージェント・アプリケーション名は、 AIX NetView サービス・ポイントに登録された名前です。
- TME エージェントのアプリケーション名は MSMTME です。
- ALL またはワイルドカード名を指定できます。

TYPE
エージェントのタイプを指定します。エージェント名として ALL またはワイルドカード名が指定された場合、TYPE は無視されます。

LAN	LAN Network Manager IBM エージェント
IP	NetView for AIX IBM エージェント
TME	TME IBM エージェント

service_point
エージェントの VTAM PU、LU、または CP 名。

デフォルトは ALL です。

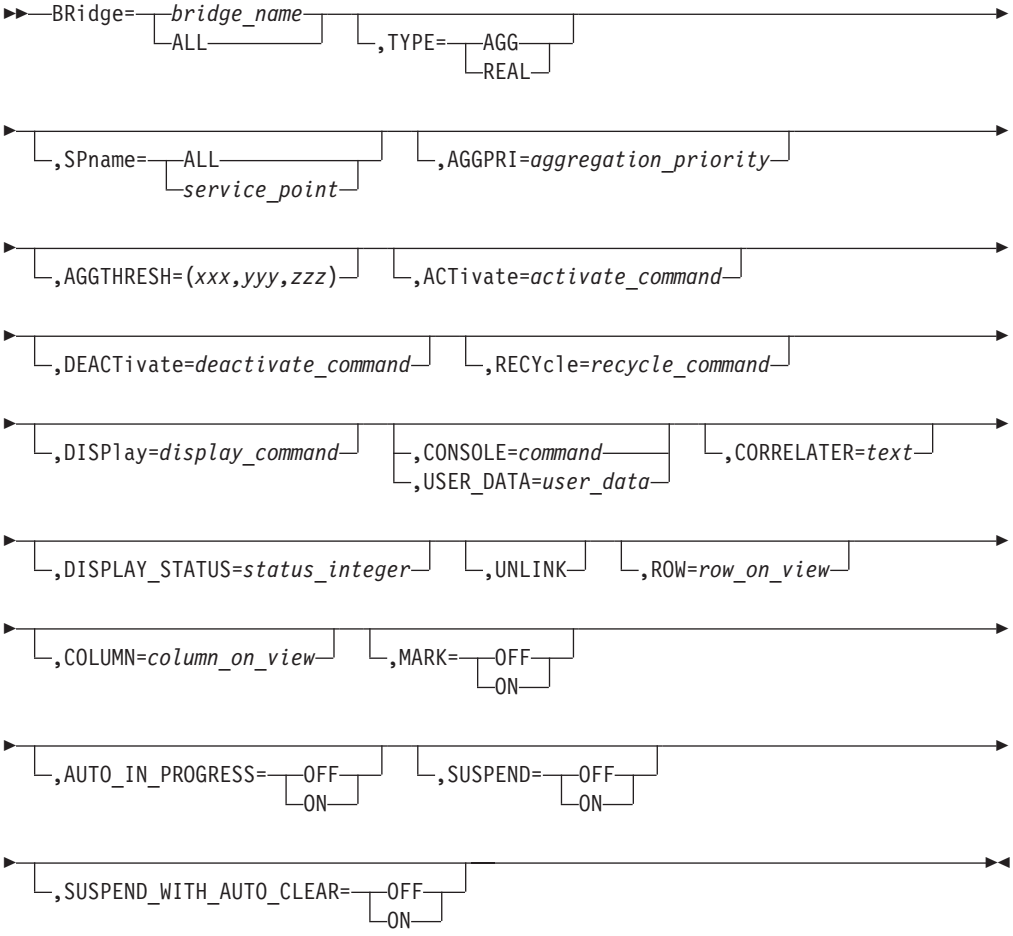
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

BRidge 制御ステートメント:

説明: BRidge 制御ステートメントは、処理されるマルチシステム・マネージャー LNM ブリッジ・リソースを指定します。

構文:

BRidge



パラメーター:

bridge_name
1 から 8 文字のブリッジ名。 ALL またはワイルドカード名を指定できます。

TYPE
ブリッジ・リソースのタイプを指定します。値は次のとおりです。

REAL	実ブリッジ・リソース
AGG	集合ブリッジ・リソース

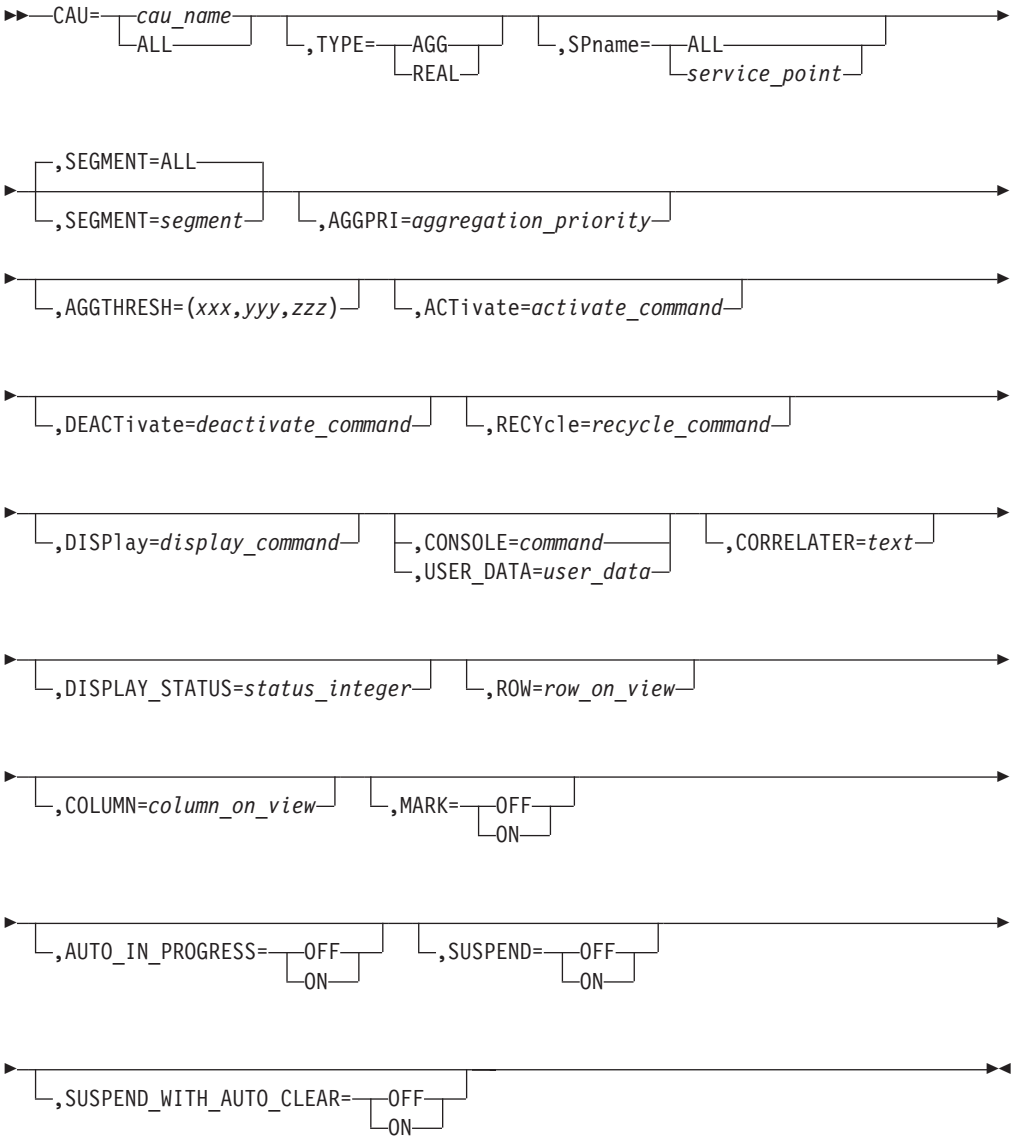
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

CAU 制御ステートメント:

説明: CAU 制御ステートメントは、処理されるマルチシステム・マネージャー LNM Controlled Access Unit リソースを指定します。

構文:

CAU



パラメーター:

cau_name
1 から 8 文字の Controlled Access Unit 名。 ALL またはワイルドカード名を指定できます。

TYPE
CAU リソースのタイプを指定します。値は次のとおりです。

REAL 実 CAU リソース

AGG 集合 CAU リソース

segment_name
セグメント番号 (3 から 4 文字) またはセグメント名 (SEGxxxx など)。 ALL も指定可能で、これがデフォルトです。

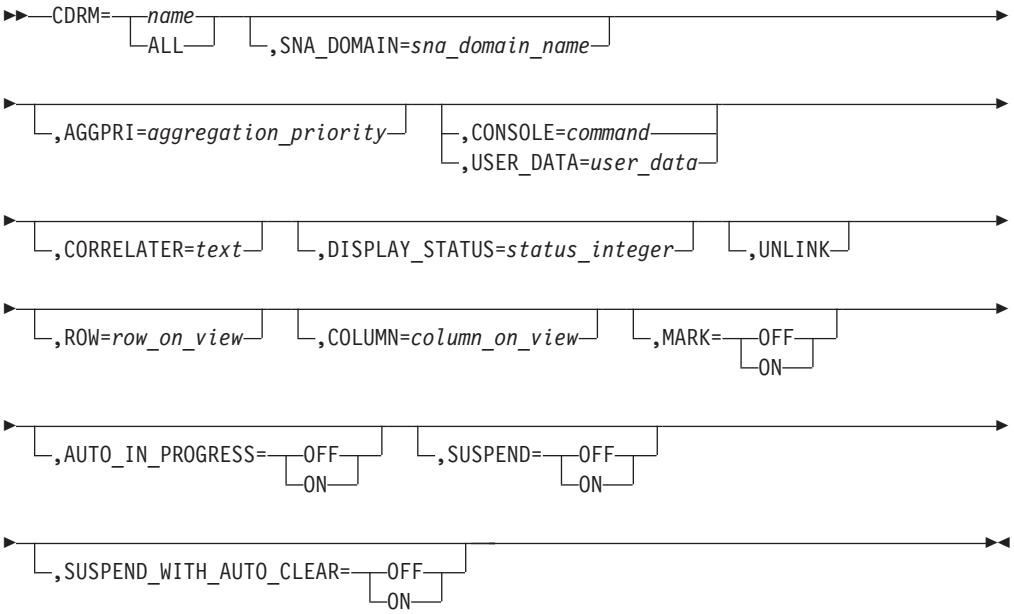
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

CDRM 制御ステートメント:

説明: CDRM 制御ステートメントは、処理される VTAM CDRM リソースを指定します。

構文:

CDRM



パラメーター:

name
snaNetID.snaNodeName の形式による 1 から 17 文字の VTAM CDRM 名。 ALL またはワイルドカード名を指定できます。

sna_domain_name
CDRM リソースを所有する VTAM SNA ドメインを指定します。これは、SNA_DOMAIN 制御ステートメントで指定した値をオーバーライドします。名前の形式は、network.domain です。

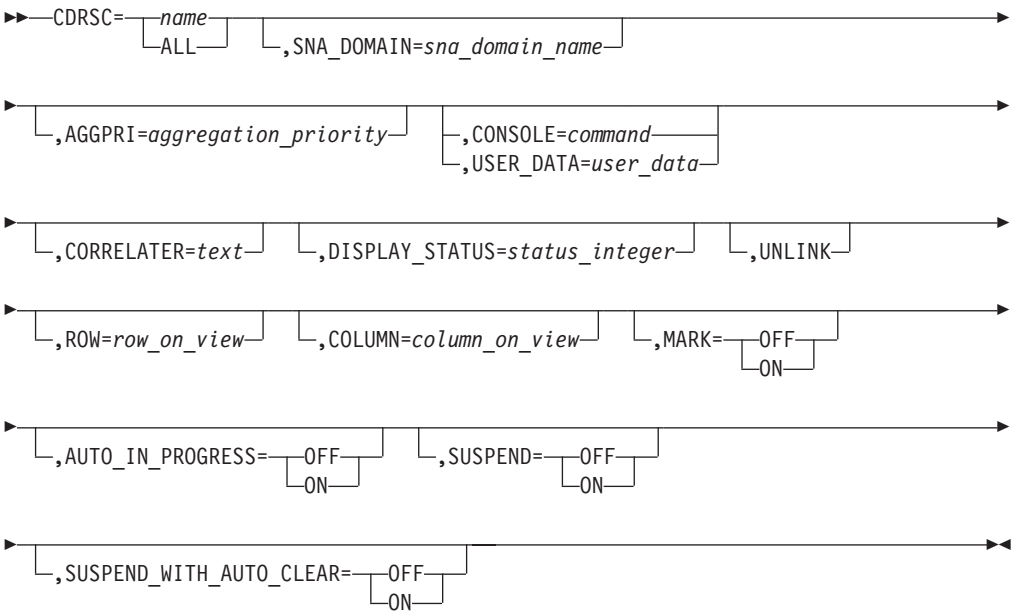
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

CDRSC 制御ステートメント:

説明: CDRSC 制御ステートメントは、処理される VTAM CDRSC リソースを指定します。

構文:

CDRSC



パラメーター:

name
: snaNetID.snaNodeName の形式による 1 から 17 文字の VTAM CDRSC 名。NETID パラメーターと共に定義されていない CDRSC については、 CDRSCS 名のネットワーク部分が省略されることがあります。 ALL またはワイルドカード名を指定できます。

sna_domain_name
: CDRM リソースを所有する VTAM SNA ドメインを指定します。これは、SNA_DOMAIN 制御ステートメントで指定した値をオーバーライドします。名前の形式は、*network.domain* です。

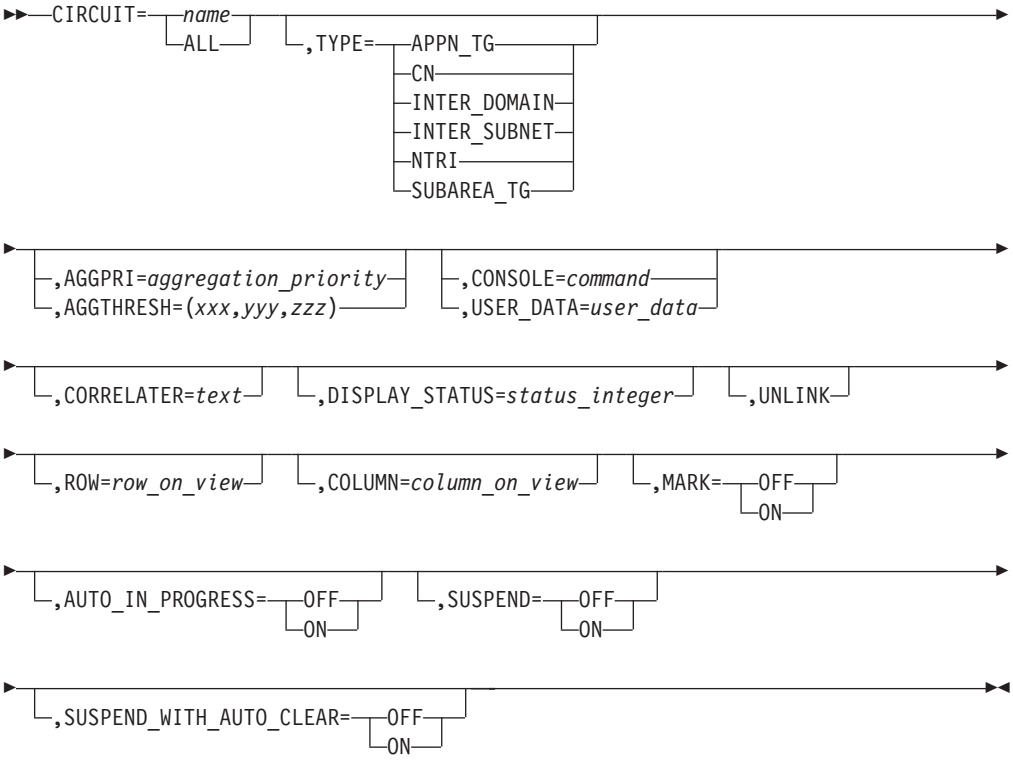
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

CIRCUIT 制御ステートメント:

説明: CIRCUIT 制御ステートメントは、処理される回線リソースを指定します。これには、タイプ 2.1 ノードに接続された APPN 伝送グループ回線、複合ノードに接続された APPN 伝送グループ回線、 NTRI に類似したノードに接続された APPN 伝送グループ回線、 APPN 伝送グループのドメイン間回線、APPN 伝送グループのサブネットワーク間回線、およびサブエリア伝送グループ回線が含まれます。

構文:

CIRCUIT



パラメーター:

name

snaNetID.circuitID の形式による、SNA 回線名。この名前は、リソースが NMC に表示されるときと同じフォーマット (DisplayResourceName) になります。ALL またはワイルドカード名を指定できます。

TYPE

回線のタイプを指定します。

APPN_TG
タイプ 2.1 ノードに接続された回線

CN 複合ノードに接続された回線

NTRI NTRI に類似したノードに接続された回線

INTER_SUBNET
サブネットワーク間回線

INTER_DOMAIN
ドメイン間回線

SUBAREA_TG
サブエリア伝送グループ回線

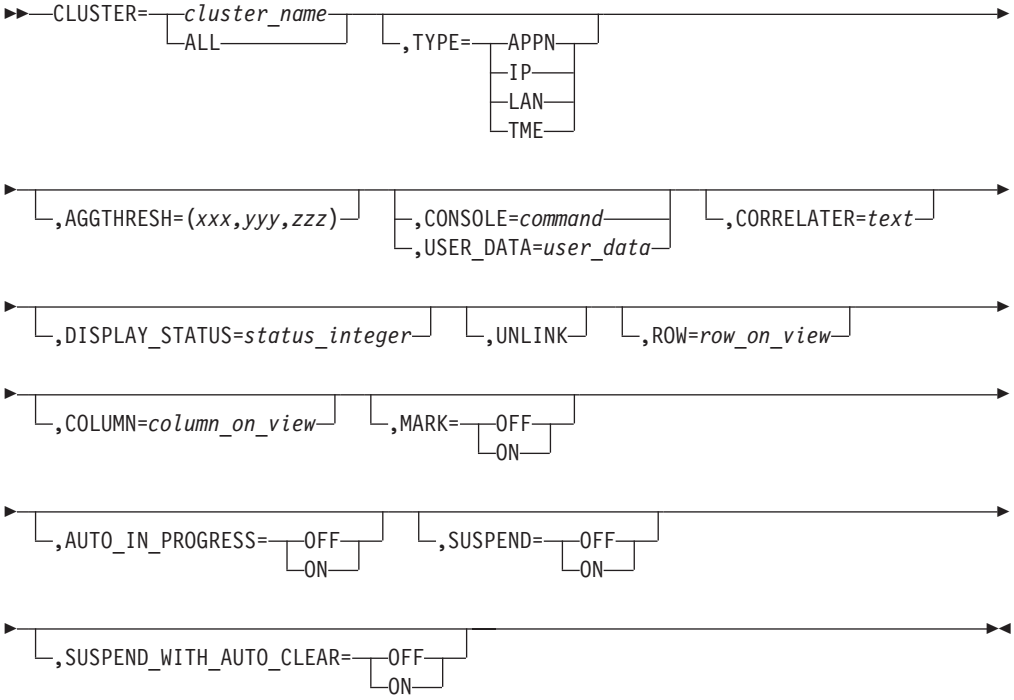
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

CLUSTER 制御ステートメント:

説明: CLUSTER 制御ステートメントは、処理されるマルチシステム・マネージャー または APPN Cluster 集合リソースを指定します。この集合には、1 つ以上のネットワーク集合を含めることができます。

構文:

CLUSTER



パラメーター:

cluster_name
CLUSTER 集合リソースの名前。

TYPE=LAN、TYPE=IP、または TYPE=TME では、この名前は GETTOPO コマンドまたはステートメントの NETWORK_AGG_OBJECT に指定された値になります。

TYPE=APPN では、この名前はトポロジー・マネージャーが存在する NetView ドメインのネットワーク ID である、snaNetid.systemId の形式になります。

ALL またはワイルドカード名を指定できます。

TYPE
CLUSTER 集合リソースのタイプを指定します。値は次のとおりです。

LAN	LAN Network Manager (LNM)
IP	TCP/IP
APPN	APPN
TME	Tivoli Managed Enterprise

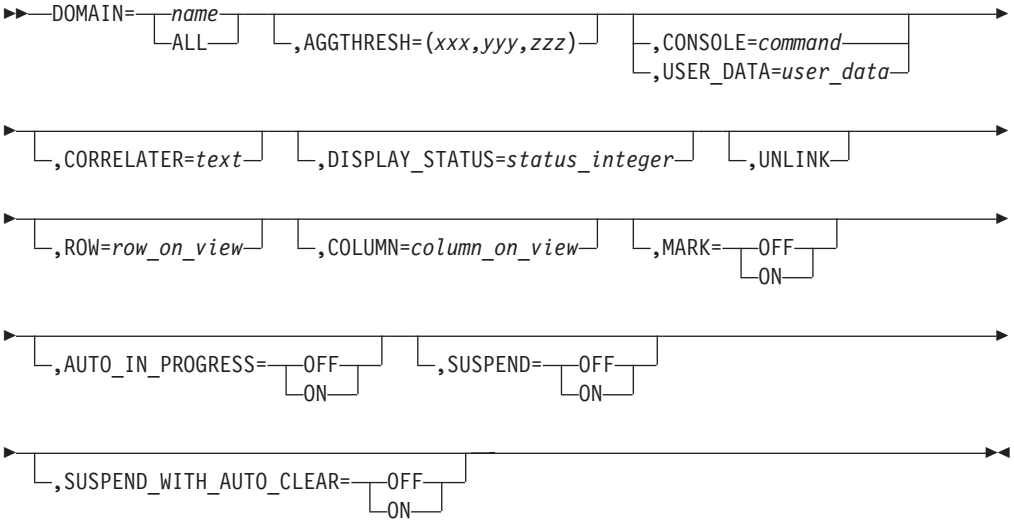
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

DOMAIN 制御ステートメント:

説明: DOMAIN 制御ステートメントは、処理される APPN ドメイン・リソースを指定します。

構文:

DOMAIN



パラメーター:

name
snaNetID.snaNodeName の形式による、1 から 17 文字の APPN ネットワーク・ノードのドメイン名。 ALL またはワイルドカード名を指定できます。

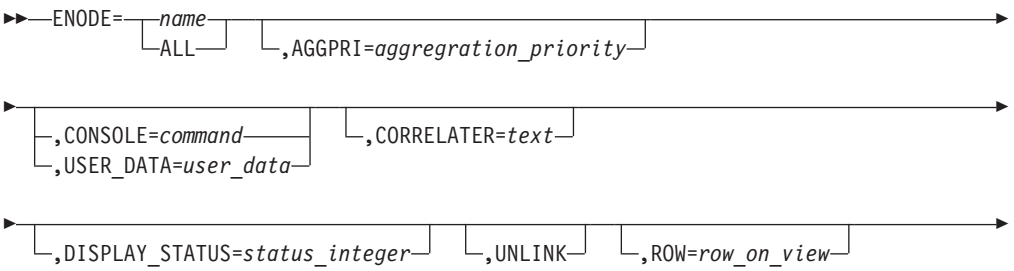
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

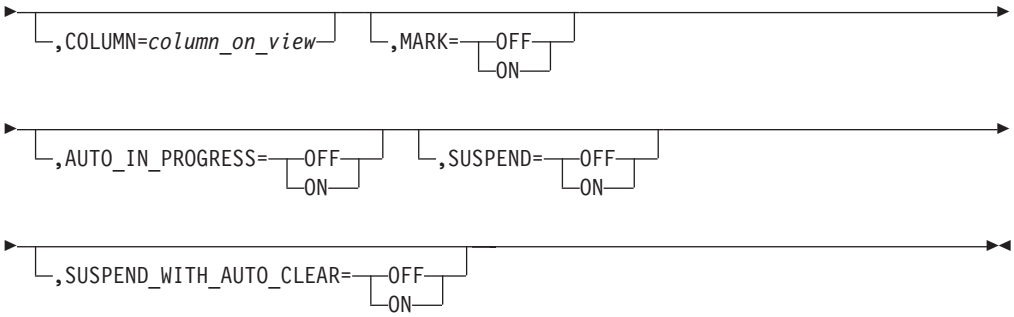
ENODE 制御ステートメント:

説明: ENODE 制御ステートメントは、処理される APPN エンド・ノード・リソースを指定します。

構文:

ENODE





パラメーター:

name
snaNetID.snaNodeName の形式による、1 から 17 文字の SNA エンド・ノードのリソース名。 ALL またはワイルドカード名を指定できます。

サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

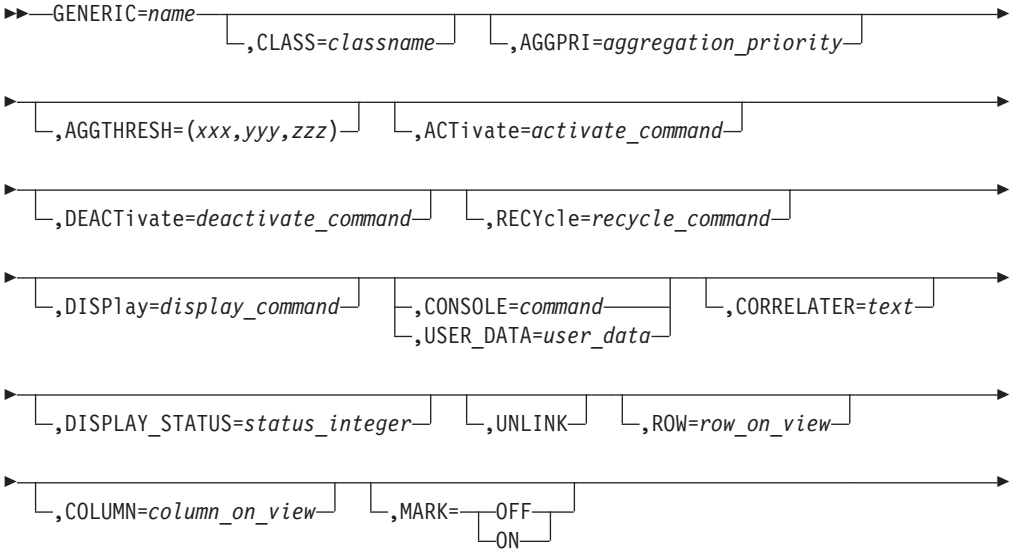
GENERIC 制御ステートメント:

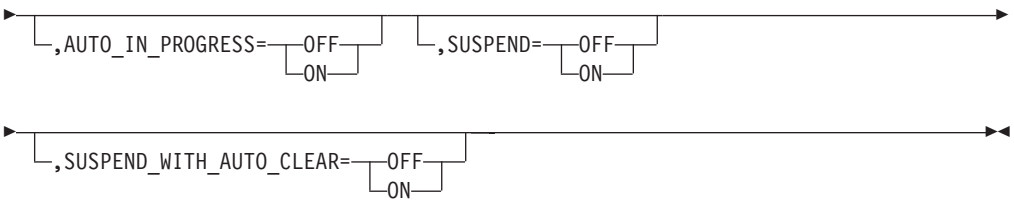
説明: GENERIC 制御ステートメントは、処理される、ユーザー定義クラスからの実リソースまたは集合リソースを指定します。

注: BLDVIEWS インタープリター (FLCVBLDV) と RODM Collection Manager インタープリター (FLCV2RCM) とでは、*name* パラメーターの扱いに若干の違いがあります。以下の *name* パラメーターの説明を参照してください。

構文:

GENERIC





パラメーター:

name
BLDVIEWS インタープリター (FLCVBLDV) は、一致するオブジェクト名を探す際に RODM MyName 属性と DisplayResourceName 属性の両方を検索します。 RODM Collection Manager インタープリター (FLCV2RCM) は、一致する名前を探す際に RODM DisplayResourceName 属性のみを検索します。

classname
オブジェクトが含まれている RODM クラスの名前。

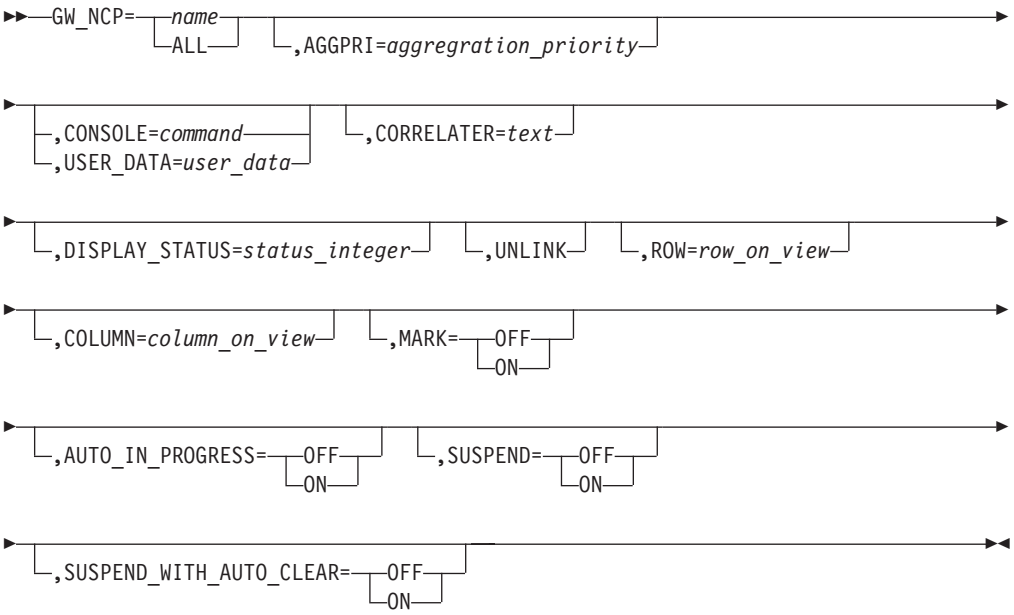
サポートされる他のキーワードについては、673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

GW_NCP 制御ステートメント:

説明: GW_NCP 制御ステートメントは、処理されるゲートウェイとして機能している SNA 通信コントローラー・ノードのリソースを指定します。

構文:

GW_NCP



パラメーター:

name
snaNetID.snaNodeName の形式による 1 から 17 文字の SNA 通信コントローラー・ノード。 ALL またはワイルドカード名を指定できます。

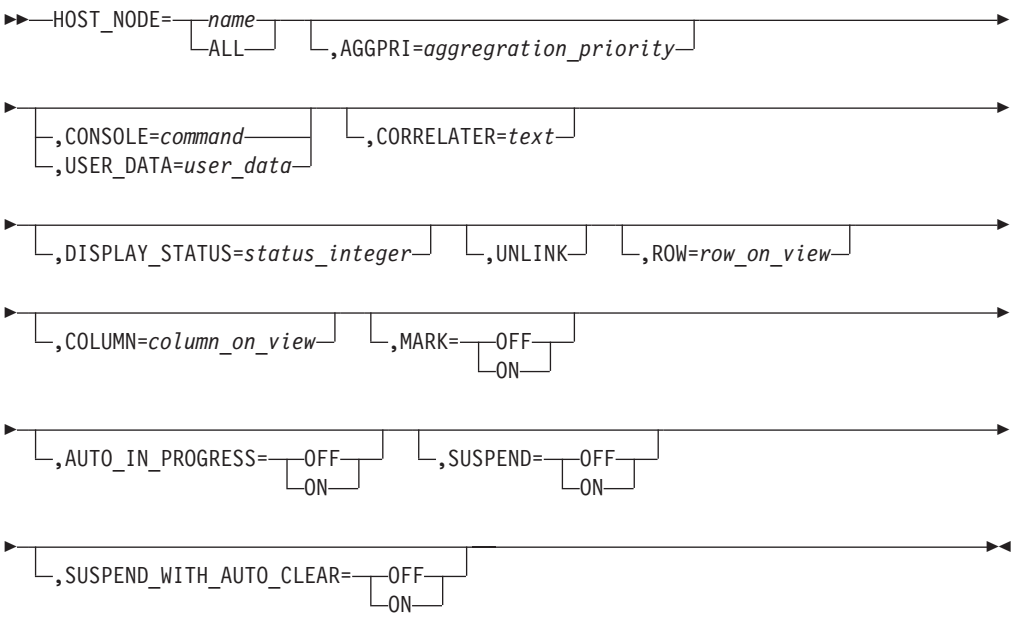
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

HOST_NODE 制御ステートメント:

説明: HOST_NODE 制御ステートメントは、処理される SNA タイプ 5 ノード・リソースを指定します。タイプ 5 ノードは、SSCP を含み、タイプ 4 ノードおよび周辺ノードを階層的に制御するサブエリア・ノードです。

構文:

HOST_NODE



パラメーター:

name
snaNetID.snaNodeName の形式による、1 から 17 文字の SNA ホスト・ノード名。 ALL またはワイルドカード名を指定できます。

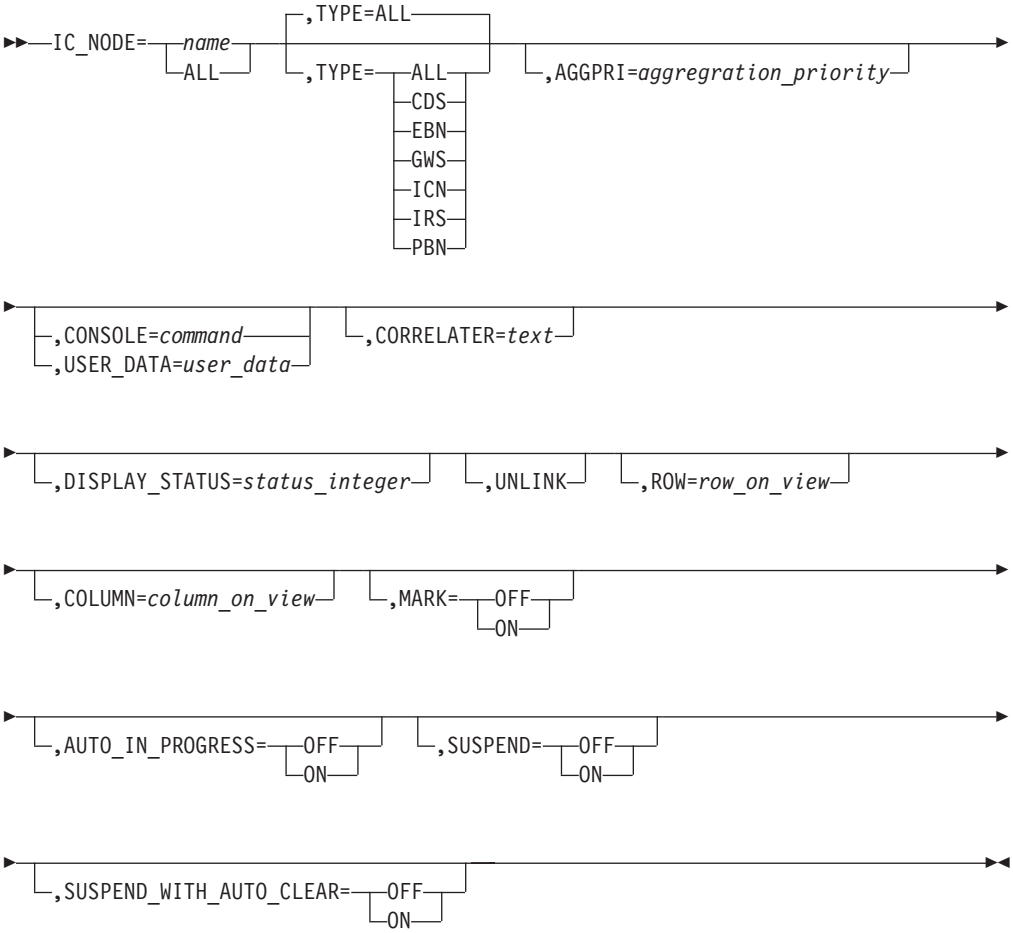
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

IC_NODE 制御ステートメント:

説明: IC_NODE 制御ステートメントは、処理される SNA 交換ノード・リソースを指定します。

構文:

IC_NODE



パラメーター:

name
snaNetID.snaNodeName の形式による、1 から 17 文字の SNA 交換ノード名。ALL またはワイルドカード名を指定できます。

TYPE
ネットワーク・ノード・リソースのタイプを指定します。値は次のとおりです。

GWS	ゲートウェイ・サービスのあるノード
CDS	中央ディレクトリー・サービスのあるノード
IRS	中間ルーティング・サービスのあるノード
PBN	周辺ボーダー・ノードであるノード
EBN	拡張ボーダー・ノードであるノード
ALL	すべての IC_NODE タイプ (デフォルト)

TYPE
正確なリソース名を指定した場合、この値は無視されます。これがサポートされるのは、名前が ALL またはワイルドカード名である場合のみです。

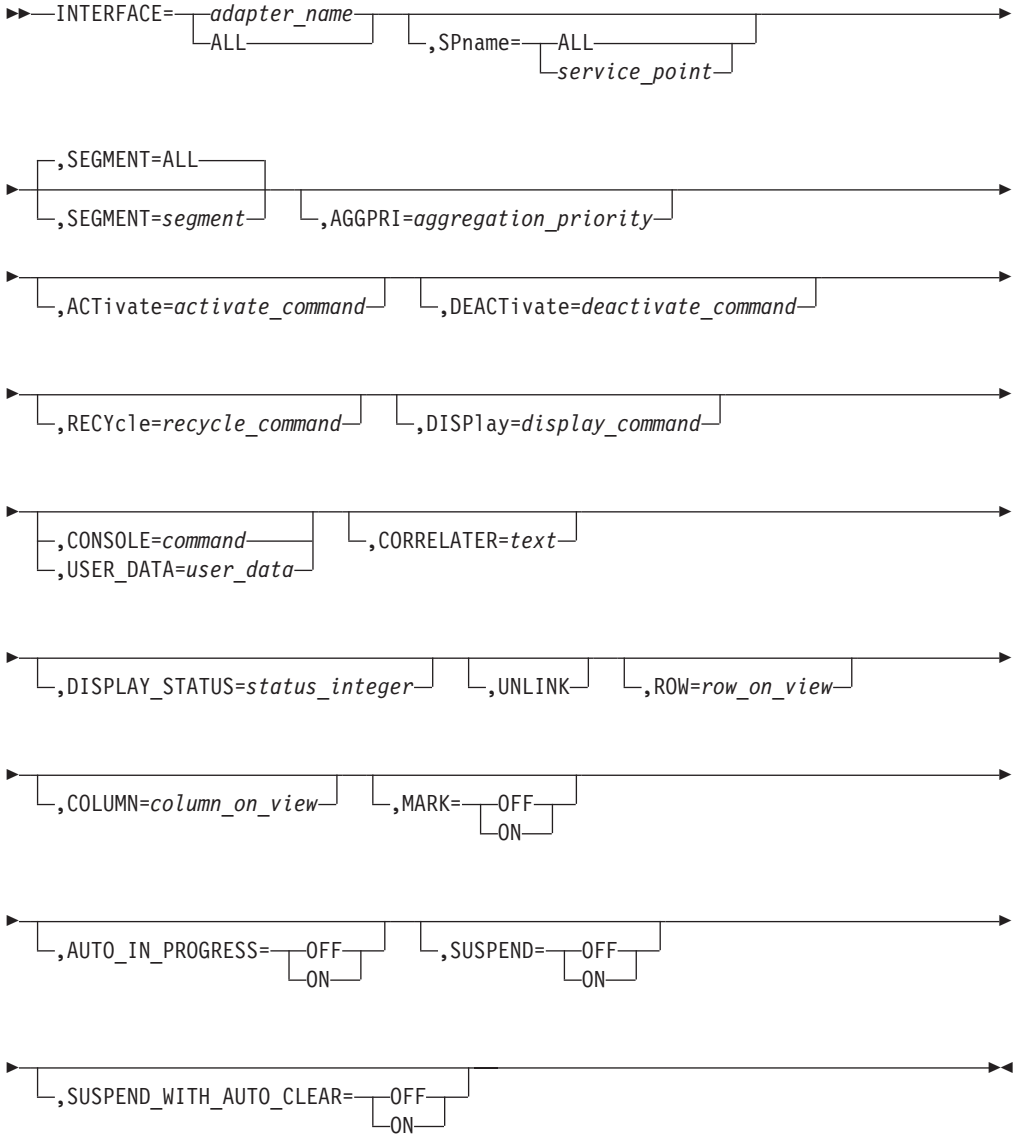
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

INTERFACE 制御ステートメント:

説明: INTERFACE 制御ステートメントは、処理されるマルチシステム・マネージャー TCP/IP アダプター・リソースを指定します。

構文:

INTERFACE



パラメーター:

adapter_name
TCP/IP インターフェース・アダプター名。

ALL またはワイルドカード名を指定できます。

segment_name
セグメント名。

ALL も指定可能で、これがデフォルトです。

サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

IP_BRIDGE 制御ステートメント:

説明: IP_BRIDGE 制御ステートメントは、処理されるマルチシステム・マネージャー TCP/IP ブリッジ・リソースを指定します。

構文:

IP_BRIDGE

パラメーター:

name
TCP/IP ブリッジ名。 ALL またはワイルドカード名を指定できます。

サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

IP_HOST 制御ステートメント:

説明: IP_HOST 制御ステートメントは、処理されるマルチシステム・マネージャー TCP/IP ホスト集合リソースを指定します。

構文:

IP_HOST

パラメーター:

name
TCP/IP ホスト名。 ALL またはワイルドカード名を指定できます。

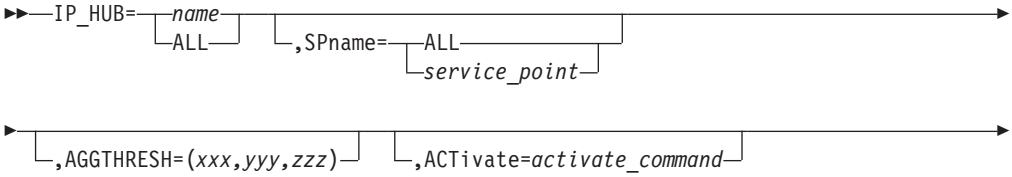
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

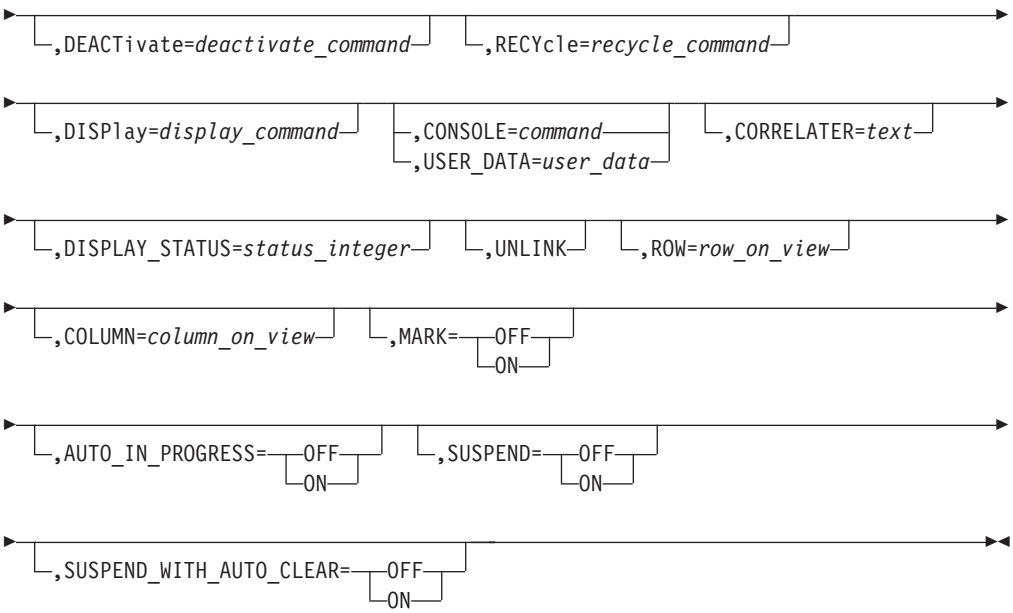
IP_HUB 制御ステートメント:

説明: IP_HUB 制御ステートメントは、処理されるマルチシステム・マネージャー TCP/IP ハブ集合リソースを指定します。

構文:

IP_HUB





パラメーター:

name
TCP/IP ハブ名。 ALL またはワイルドカード名を指定できます。

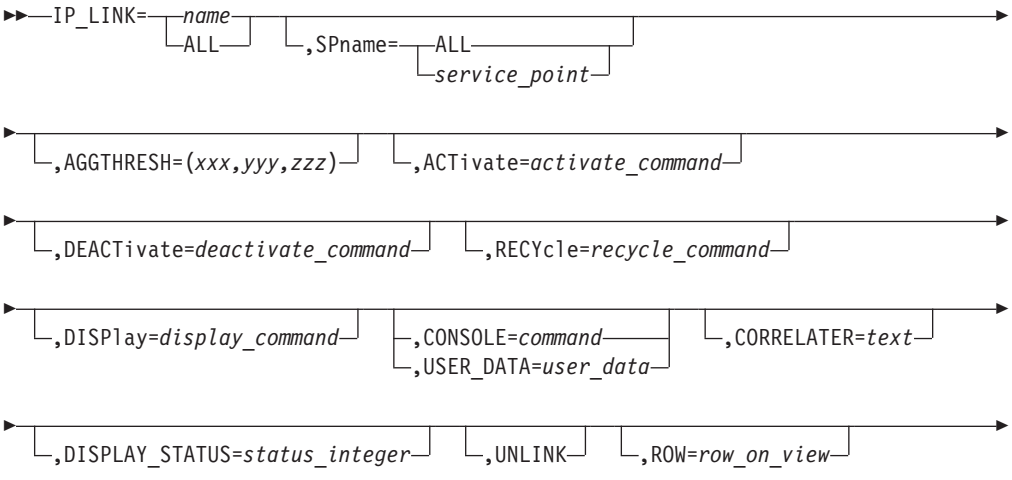
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

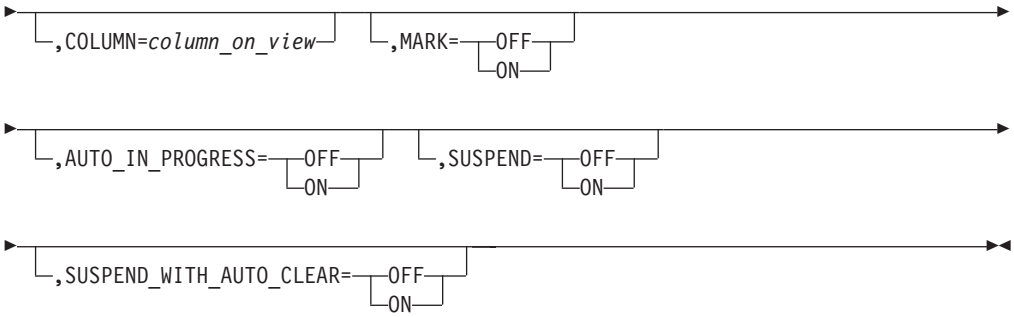
IP_LINK 制御ステートメント:

説明: IP_LINK 制御ステートメントは、処理されるマルチシステム・マネージャー TCP/IP インターフェース・リンク集合リソースを指定します。

構文:

IP_LINK





パラメーター:

name
TCP/IP リンク名。 ALL またはワイルドカード名を指定できます。

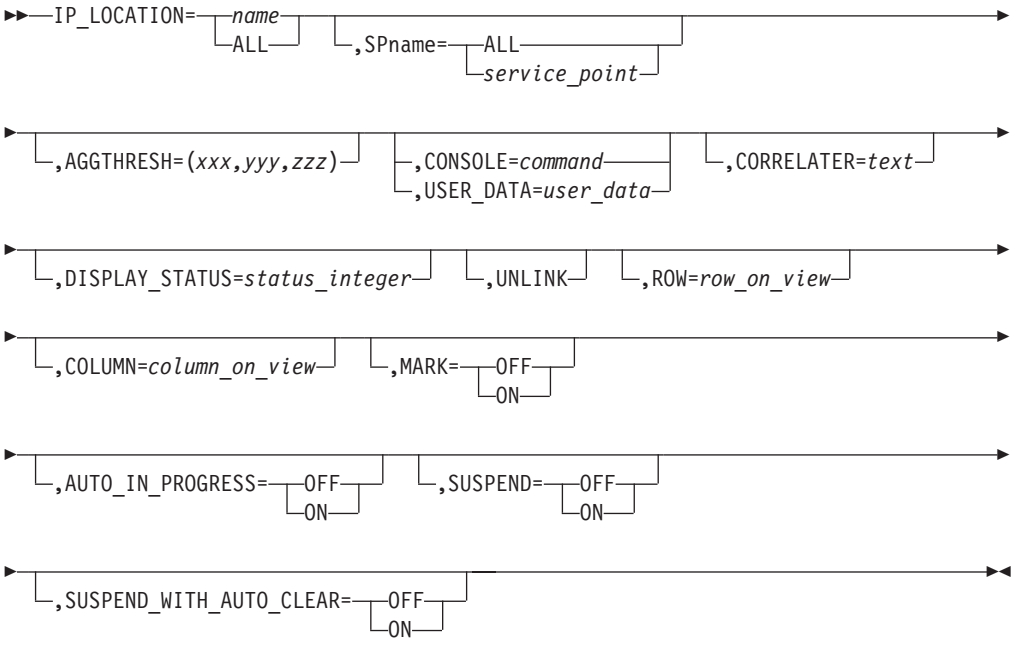
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

IP_LOCATION 制御ステートメント:

説明: IP_LOCATION 制御ステートメントは、処理されるマルチシステム・マネージャー TCP/IP ロケーション・リソースを指定します。

構文:

IP_LOCATION



パラメーター:

name
TCP/IP ロケーション名。 ALL またはワイルドカード名を指定できます。

サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

IP_ROUTER 制御ステートメント:

説明: IP_ROUTER 制御ステートメントは、処理されるマルチシステム・マネージャー TCP/IP ルーター集合リソースを指定します。

構文:

IP_ROUTER

パラメーター:

name
TCP/IP ルーター名。 ALL またはワイルドカード名を指定できます。

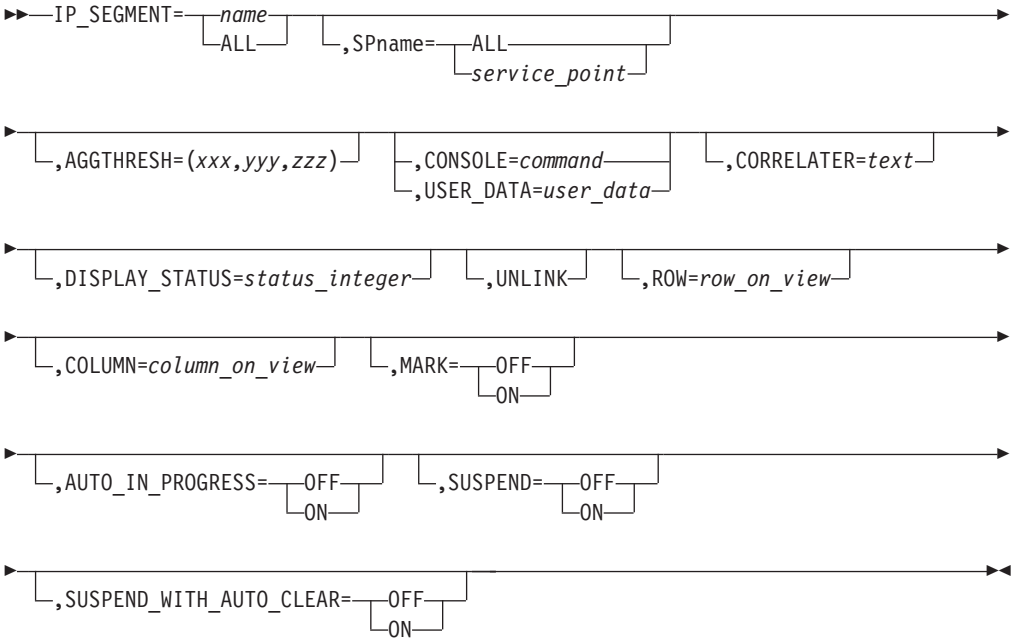
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

IP_SEGMENT 制御ステートメント:

説明: IP_SEGMENT 制御ステートメントは、処理されるマルチシステム・マネージャー TCP/IP セグメント集合リソースを指定します。

構文:

IP_SEGMENT



パラメーター:

name
TCP/IP セグメント名。 ALL またはワイルドカード名を指定できます。

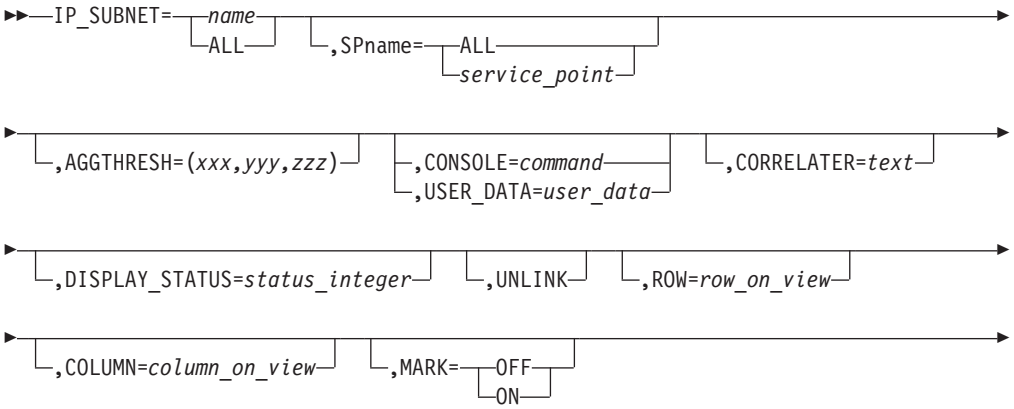
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

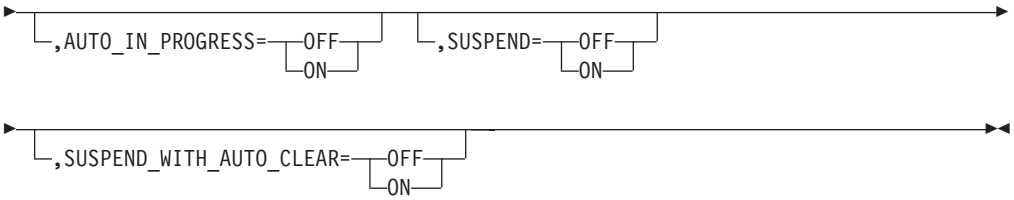
IP_SUBNET 制御ステートメント:

説明: IP_SUBNET 制御ステートメントは、処理されるマルチシステム・マネージャー TCP/IP サブネットワーク集合リソースを指定します。

構文:

IP_SUBNET





パラメーター:

name
TCP/IP サブネットワーク名。 ALL またはワイルドカード名を指定できます。

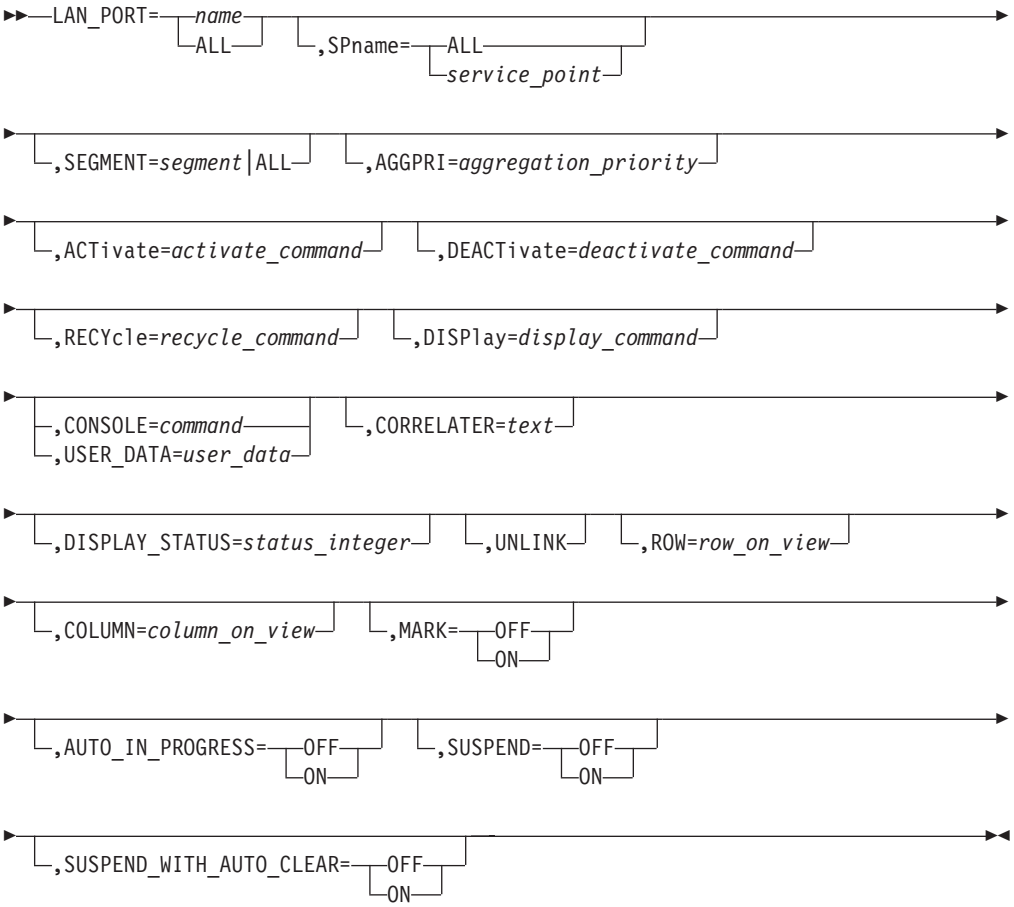
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

LAN_PORT 制御ステートメント:

説明: LAN_PORT 制御ステートメントは、処理される LAN ドメイン・リソースを指定します。これは、LNM V2 のためのものです。

構文:

LAN_PORT



パラメーター:

name
LAN Network Manager V2 によって決められる (そして DisplayResourceName を使用してリソースについて NMC に表示される) LNM Port リソース名。 ALL またはワイルドカード名を指定できます。

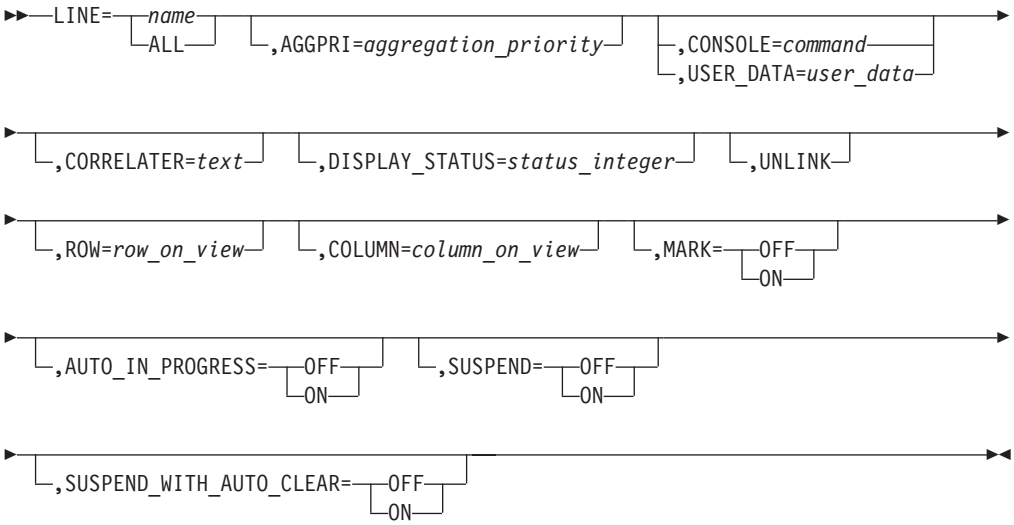
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

LINE 制御ステートメント:

説明: LINE 制御ステートメントは、処理される SNA 回線リソースを指定します。

構文:

LINE



パラメーター:

name
snaNetID.snaNodeName の形式による、1 から 17 文字の SNA 回線名。 ALL またはワイルドカード名を指定できます。

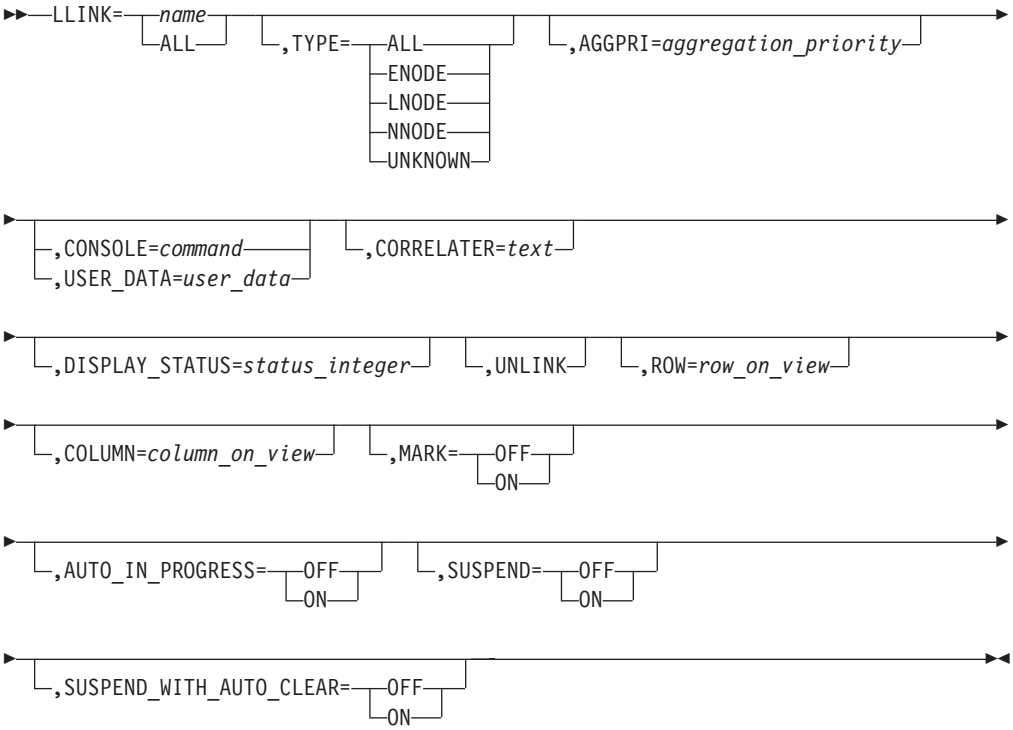
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

LLINK 制御ステートメント:

説明: LLINK 制御ステートメントは、処理される論理リンク・リソースを指定します。

構文:

LLINK



パラメーター:

name
network.resource.link の形式による、SNA 論理リンク・リソース名。 ALL またはワイルドカード名を指定できます。

TYPE
論理リンクのタイプを指定します。正確なリソース名を指定した場合、TYPE は無視されます。これがサポートされるのは、名前が ALL またはワイルドカード名である場合のみです。値は次のとおりです。

NNODE	ネットワーク・ノード
ENODE	エンド・ノード
LNODE	LEN ノード
UNKNOWN	論理リンクのタイプは不明です。
ALL	すべての論理リンク

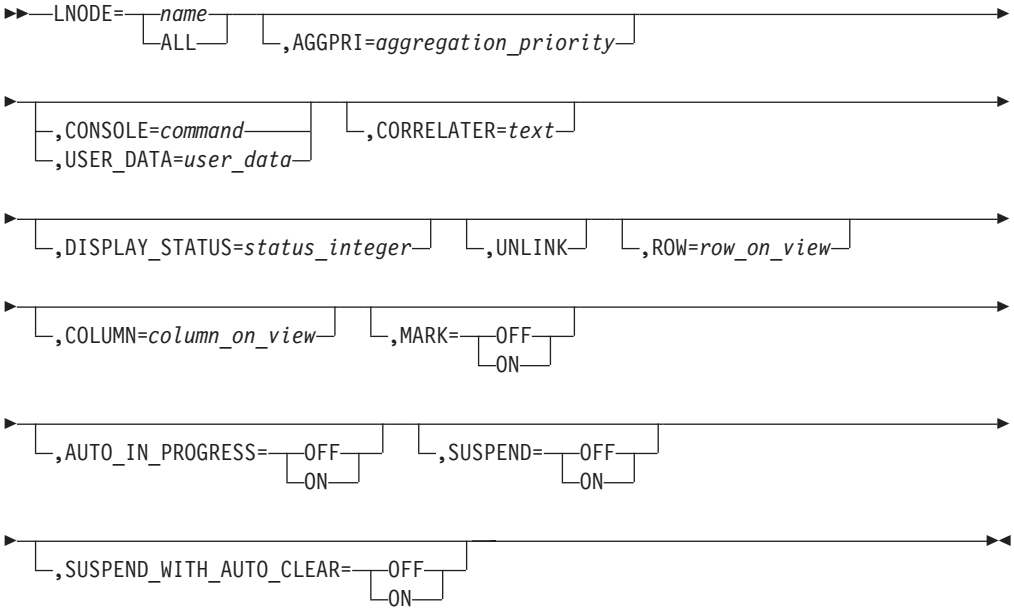
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

LNODE 制御ステートメント:

説明: LNODE 制御ステートメントは、処理される APPN LEN ノード・リソースを指定します。

構文:

LNODE



パラメーター:

name

snaNetID.snaNodeName の形式による、1 から 17 文字の SNA LEN ノードのリソース名。 ALL またはワイルドカード名を指定できます。

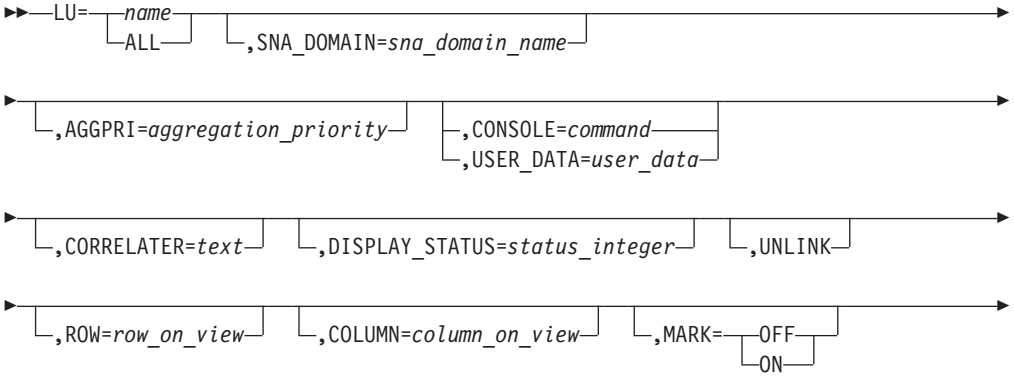
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

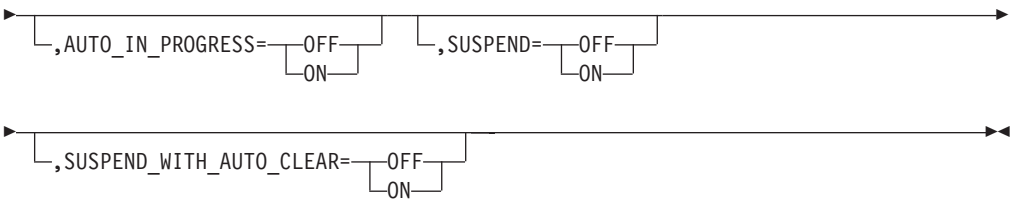
LU 制御ステートメント:

説明: LU 制御ステートメントは、処理される SNA 論理装置リソースを指定します。

構文:

LU





パラメーター:

name
snaNetID.snaNodeName の形式による、1 から 17 文字の SNA 論理装置名。ALL またはワイルドカード名を指定できます。

sna_domain_name
論理装置リソースを所有する VTAM SNA ドメインを指定します。これは、SNA_DOMAIN 制御ステートメントで指定した値をオーバーライドします。名前の形式は、network.domain です。

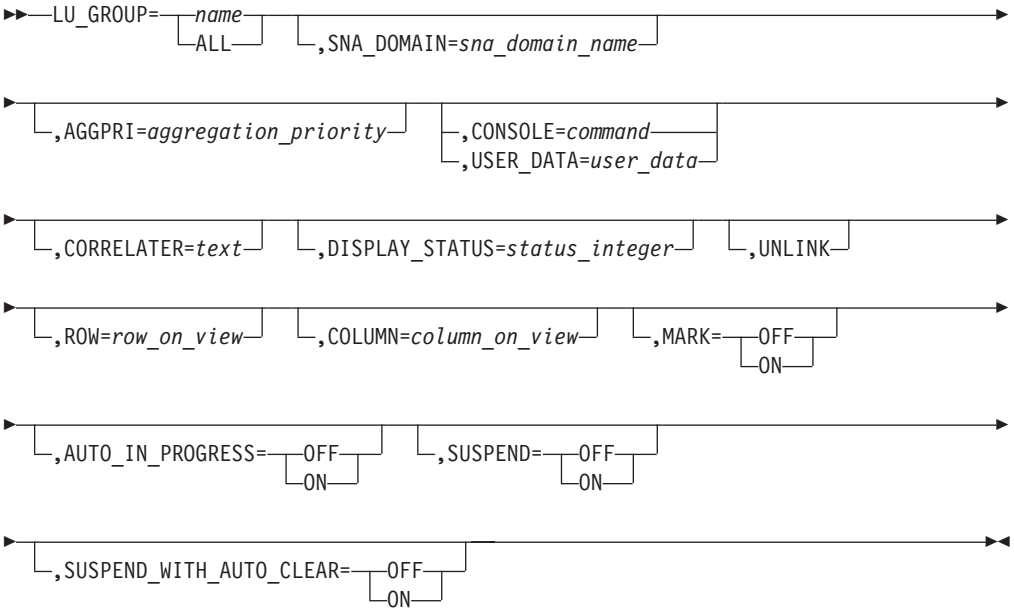
サポートされる他のキーワードについては、673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

LU_GROUP 制御ステートメント:

説明: LU_GROUP 制御ステートメントは、処理される SNA 論理装置グループ・リソースを指定します。

構文:

LU_GROUP



パラメーター:

name
luGroupName の形式による、1 から 17 文字の SNA 論理装置グループ名。ALL またはワイルドカード名を指定できます。

sna_domain_name
論理装置グループ・リソースを所有する VTAM SNA ドメインを指定します。これは、SNA_DOMAIN 制御ステートメントで指定した値をオーバーライドします。名前の形式は、network.domain です。

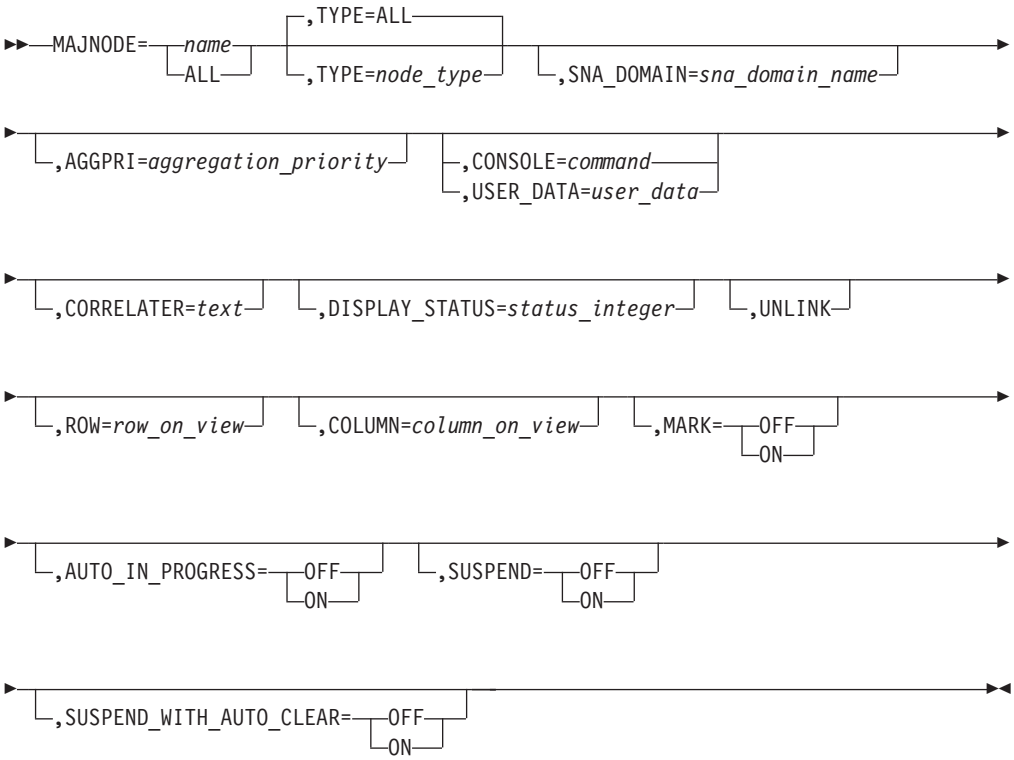
サポートされる他のキーワードについては、673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

MAJNODE 制御ステートメント:

説明: MAJNODE 制御ステートメントは、処理される VTAM メジャー・ノード・リソースを指定します。

構文:

MAJNODE



パラメーター:

name
snaNodeName の形式による、1 から 8 文字の VTAM メジャー・ノード名。ALL またはワイルドカード名を指定できます。

sna_domain_name
メジャー・ノード・リソースを所有する VTAM SNA ドメインを指定します。

これは、SNA_DOMAIN 制御ステートメントで指定した値をオーバーライドします。名前の形式は、network.domain です。

TYPE
VTAM メジャー・ノードのタイプを指定します。値は次のとおりです。

APPL	アプリケーション・メジャー・ノード
CA	チャネル・メジャー・ノード
CDRM	CDRM メジャー・ノード
CDRSC	CDRSC メジャー・ノード
LAN	ローカル・エリア・ネットワーク・メジャー・ノード
LCLNONSNA	ローカル非 SNA メジャー・ノード
LOCALSNA	ローカル SNA メジャー・ノード
LUGROUP	LU グループ・メジャー・ノード
NCP	NCP メジャー・ノード
PACKET	パケット・メジャー・ノード
SWITCHED	交換回線メジャー・ノード
TRL	トークンリング LAN メジャー・ノード
XCA	XCA メジャー・ノード
ALL	すべてのメジャー・ノード・タイプ (デフォルト)

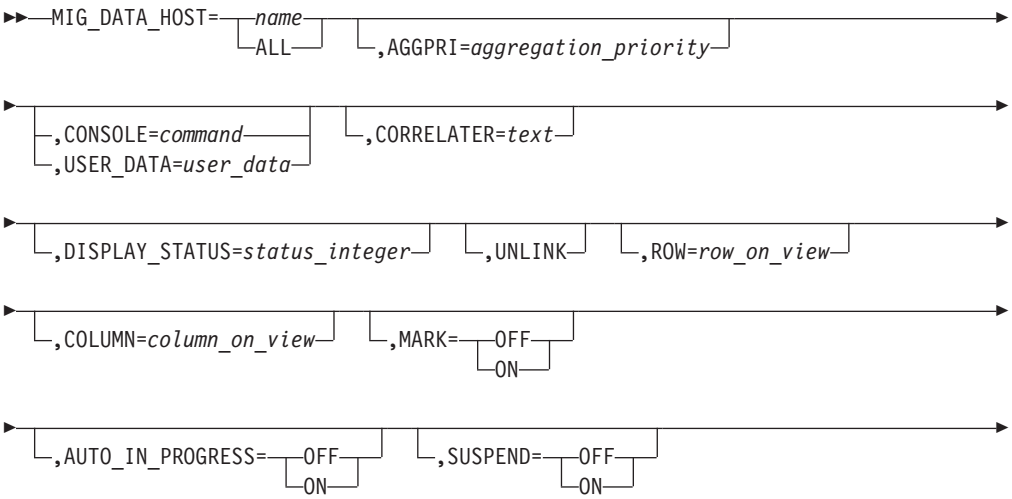
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

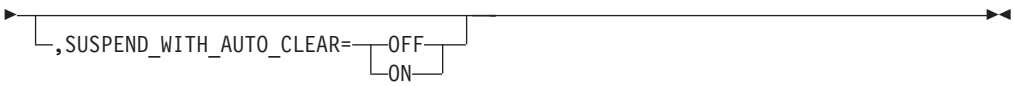
MIG_DATA_HOST 制御ステートメント:

説明: MIG_DATA_HOST 制御ステートメントは、処理される SNA Migration Data Host ノードのリソースを指定します。

構文:

MIG_DATA_HOST





パラメーター:

name
network.name の形式による、1 から 17 文字の SNA マイグレーション中のデータ・ホスト・ノード。 ALL またはワイルドカード名を指定できます。

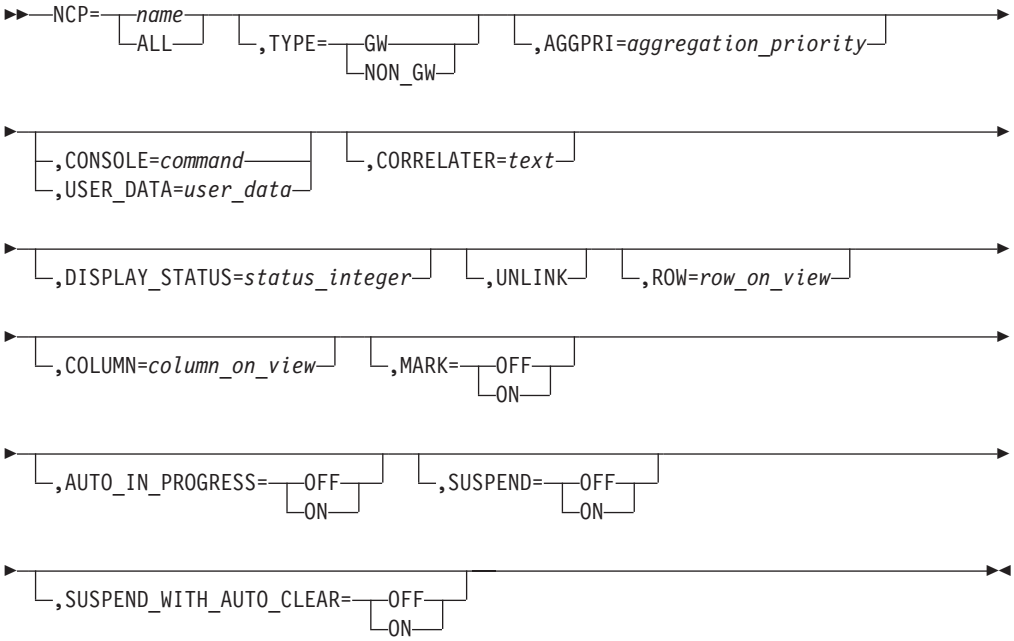
サポートされる他のキーワードについては、673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

NCP 制御ステートメント:

説明: NCP 制御ステートメントは、処理される SNA 通信コントローラー・ノードのリソースを指定します。

構文:

NCP



パラメーター:

name
snaNetID.snaNodeName の形式による 1 から 17 文字の SNA 通信コントローラー・ノード。 ALL またはワイルドカード名を指定できます。

TYPE
SNA 通信コントローラーのタイプを指定します。 TYPE は必須キーワードです。値は次のとおりです。

GW	ゲートウェイ通信コントローラー
NON_GW	非ゲートウェイ通信コントローラー

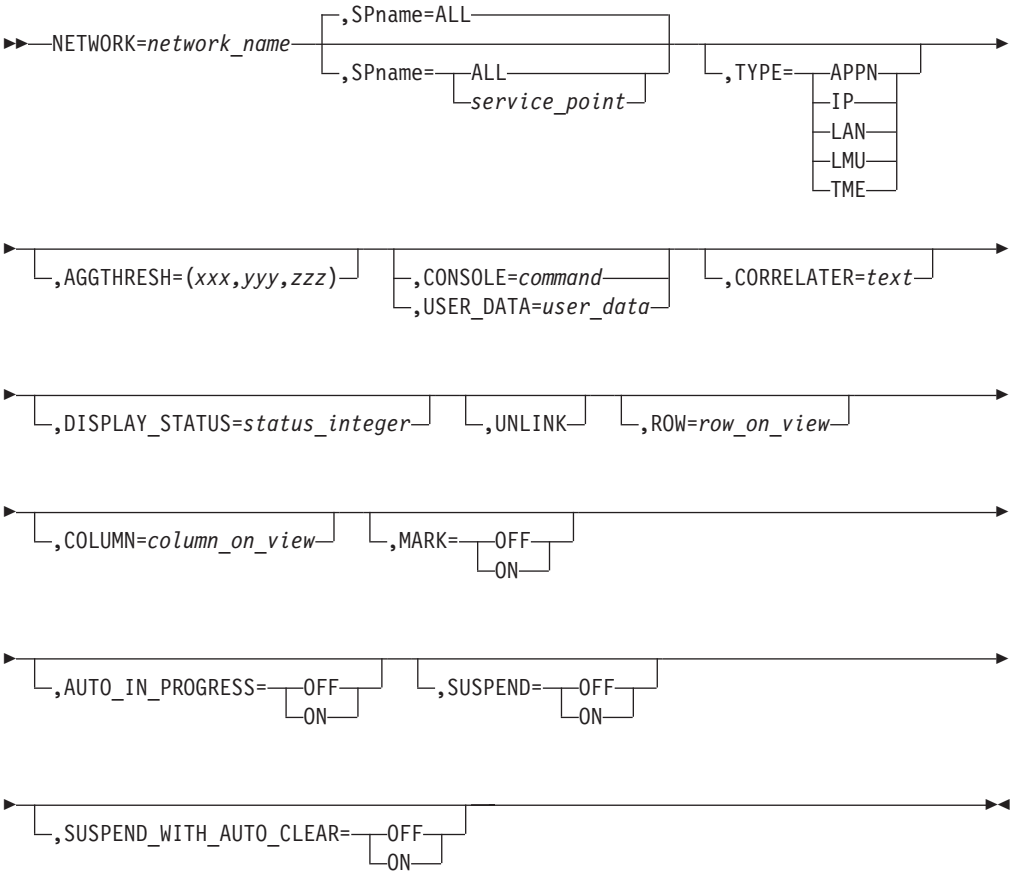
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

NETWORK 制御ステートメント:

説明: NETWORK 制御ステートメントは、処理されるマルチシステム・マネージャーまたは APPN ネットワーク集合リソースを指定します。この集合は、1 つのサービス・ポイントによって管理されるネットワークを表します。

構文:

NETWORK



パラメーター:

network_name
ネットワーク集合リソースの名前。

TYPE=LAN、TYPE=IP、または TYPE=LMU では、 network_name は 1 から 8 文字のサービス・ポイント・アプリケーション名です。

- LAN Network Manager のアプリケーション名は LANMGR です。
- NetView for AIX のエージェント・アプリケーション名は、 AIX NetView サービス・ポイントに登録された名前です。
- LMU のアプリケーション名は REMOTEOP.LMU です。

TYPE=APPN では、この名前は snaNetid.n の形式 (n は数字の増分子) になります。 ALL またはワイルドカード名を指定できます。

service_point
LAN、IP、または LMU エージェントの VTAM PU、LU、または CP 名。 これは TYPE=APPN ではサポートされずに、無視されます。

デフォルトは ALL です。

TYPE
NETWORK 集合リソースのタイプを指定します。値は次のとおりです。

LAN	LAN Network Manager (LNM)
IP	TCP/IP
APPN	APPN
TME	TME

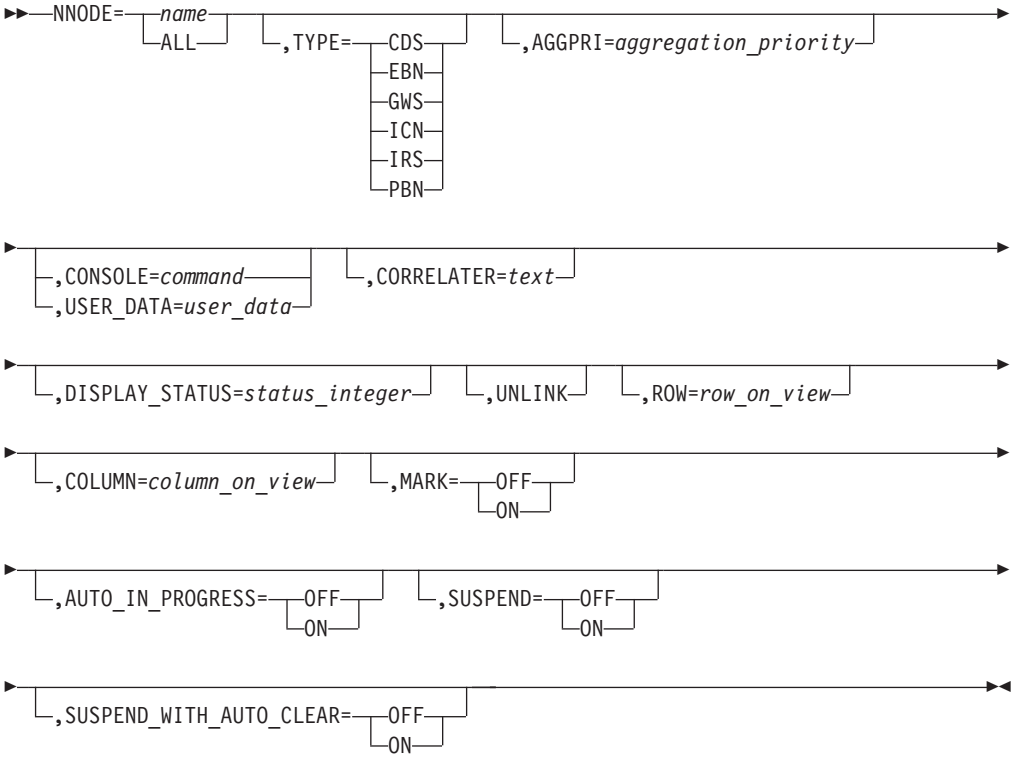
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

NNODE 制御ステートメント:

説明: NNODE 制御ステートメントは、処理される APPN ネットワーク・ノード・リソースを指定します。

構文:

NNODE



パラメーター:

name
snaNetID.snaNodeName の形式による、1 から 17 文字の SNA ネットワーク・ノードのリソース名。 ALL またはワイルドカード名を指定できます。

TYPE
ネットワーク・ノード・リソースのタイプを指定します。正確なリソース名を指定した場合、TYPE は無視されます。これがサポートされるのは、名前が ALL またはワイルドカード名である場合のみです。値は次のとおりです。

GWS	ゲートウェイ・サービスのあるノード
CDS	中央ディレクトリー・サービスのあるノード
IRS	中間ルーティング・サービスのあるノード
PBN	周辺ボーダー・ノードであるノード
ICN	交換ノードであるノード
EBN	拡張ボーダー・ノードであるノード

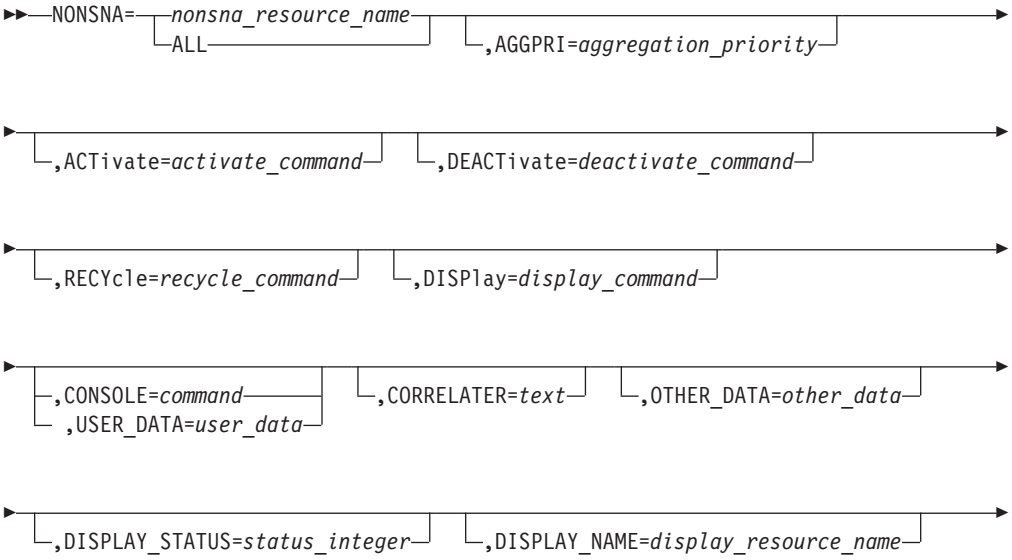
サポートされる他のキーワードについては、673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

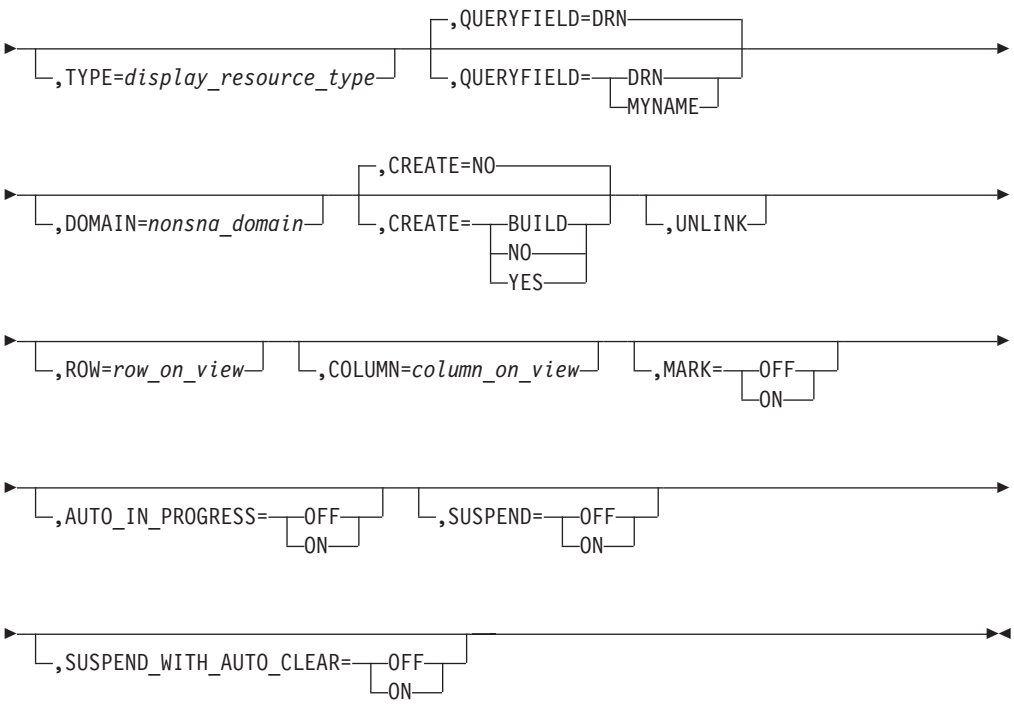
NONSNA 制御ステートメント:

説明: NONSNA 制御ステートメントは、処理される非 SNA リソース (GMFHS 管理の実リソース) を指定します。非 SNA ドメインは、NONSNA ステートメント上にコーディングされたリソースに対して設定できます。これにより、非 SNA リソースが、対応する非 SNA ドメインにリンクされます。非 SNA ドメインはリンクが作成される前に存在していなければなりません。

構文:

NONSNA





パラメーター:

nonsna_resource_name
非 SNA リソース名。 CREATE=NO では、ALL またはワイルドカード名を指定できます。

DISPLAY_NAME
オブジェクトの RODM DisplayResourceName を指定します。この値は、リソースに関して RODM resource_name に代わる値として NMC ワークステーション上に表示されます。

注: BLDVIEWS は、値の中の任意の位置にコーディングできる %NAME% 置換変数を提供します。これを使用して、複数のリソースの DisplayResourceName を 1 つの制御ステートメントで再フォーマットできます。

TYPE
非 SNA リソースのタイプを指定します。 CREATE=YES では TYPE が必要とされ、他の値では無視されます。 TYPE 値は、非 SNA オブジェクトについて RODM に設定する DisplayResourceType 値を決めます。本書に記載されている有効な非 SNA の DisplayResourceType 値を指定できます。

QUERYFIELD
NONSNA リソース・クラス (GMFHS_Managed_Real_Objects_Class) から RODM オブジェクト照会で使用するフィールドを指定します。 QUERYFIELD=DRN を指定すると、 DisplayResourceName フィールドを使用してオブジェクトを取り出します。 QUERYFIELD=MYNAME を指定すると、 MyName フィールドを使用してオブジェクトを取り出します。 QUERYFIELD が NONSNA 制御ステートメント上に指定されていない場合、DRN がデフォルトとなります。

DOMAIN
このリソースにリンクさせたい非 SNA ドメイン・リソースの名前を指定します。非 SNA ドメイン・リソースは RODM に存在していなければなりません。

CREATE
指定されたリソース上で実行するアクションを指定します。

YES	このリソースの新規オブジェクトを作成します。古いオブジェクトが存在する場合、削除されます。
NO	このリソースの新規オブジェクトを作成しません。作成する代わりに、オブジェクトを更新します。オブジェクトが存在しない場合、エラーが発生します。 NO がデフォルトです。
BUILD	このリソースの新規オブジェクトが存在しない場合、作成します。存在する場合、オブジェクトを更新します。

サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

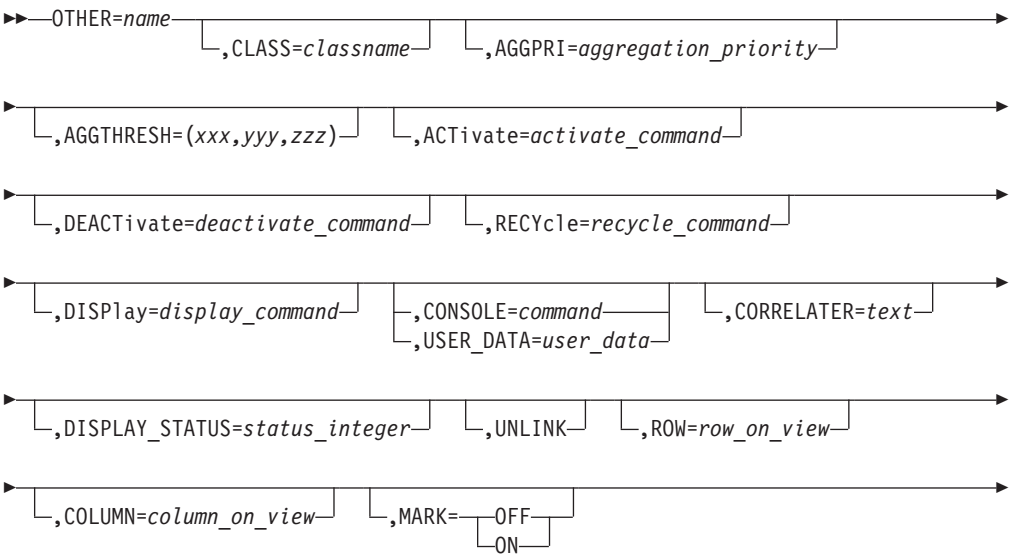
OTHER 制御ステートメント:

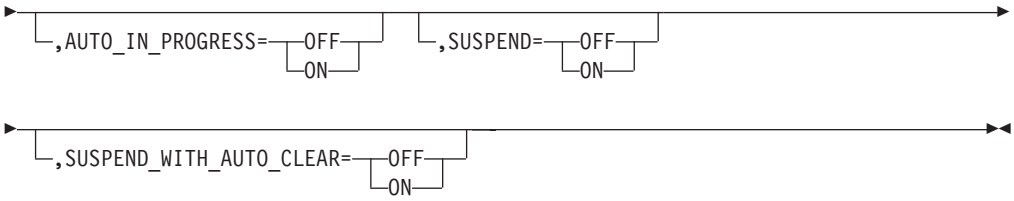
説明: OTHER 制御ステートメントは、処理される、ユーザー定義クラスからの実リソースまたは集合リソースを指定します。

注: BLDVIEWS インタープリター (FLCVBLDV) と RODM Collection Manager インタープリター (FLCV2RCM) とでは、*name* パラメーターの扱いに若干の違いがあります。以下の *name* パラメーターの説明を参照してください。

構文:

OTHER





パラメーター:

name
BLDVIEWS インタープリター (FLCVBLDV) は、一致するオブジェクト名を探す際に RODM MyName 属性と DisplayResourceName 属性の両方を検索します。 RODM Collection Manager インタープリター (FLCV2RCM) は、一致する名前を探す際に RODM MyName 属性のみを検索します。

classname
オブジェクトが含まれている RODM クラスの名前。

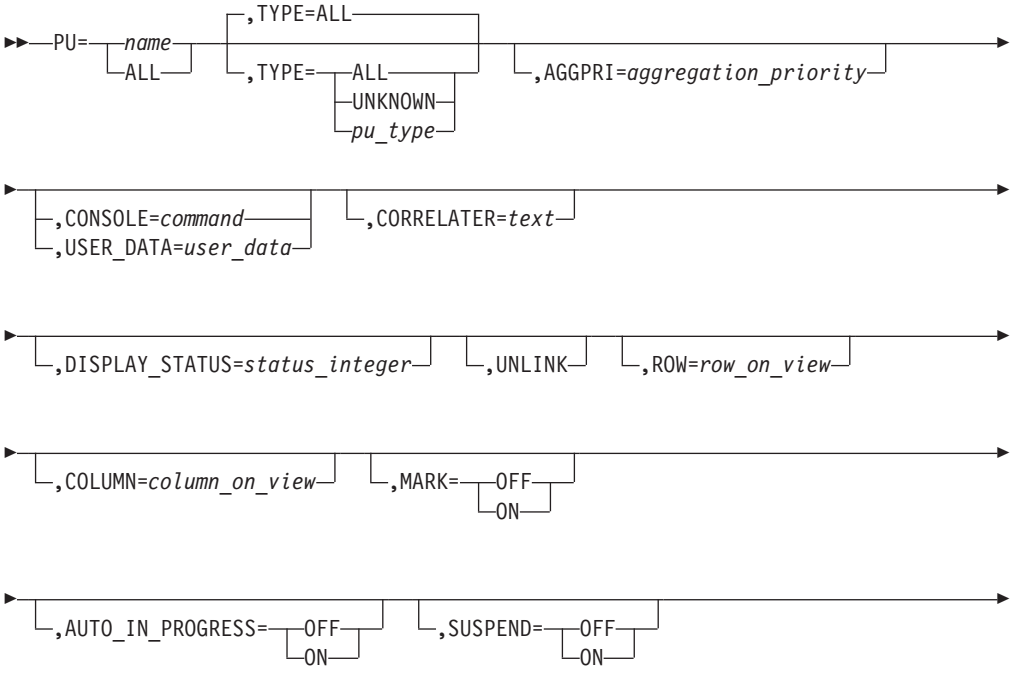
サポートされる他のキーワードについては、673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

PU 制御ステートメント:

説明: PU 制御ステートメントは、処理される SNA 物理装置リソースを指定します。

構文:

PU



パラメーター:

name
snaNetID.snaNodeName の形式による、1 から 17 文字の SNA 物理装置名。ALL またはワイルドカード名を指定できます。

TYPE
SNA 物理装置のタイプを指定します。値は次のとおりです。

1	PU タイプ 1
2	PU タイプ 2
2.1	PU タイプ 2.1
4	PU タイプ 4
5	PU タイプ 5
UNKNOWN	PU タイプは不明です。
ALL	すべての PU タイプ (デフォルト)

TYPE
正確なリソース名を指定した場合、この値は無視されます。これがサポートされるのは、名前が ALL またはワイルドカード名である場合のみです。

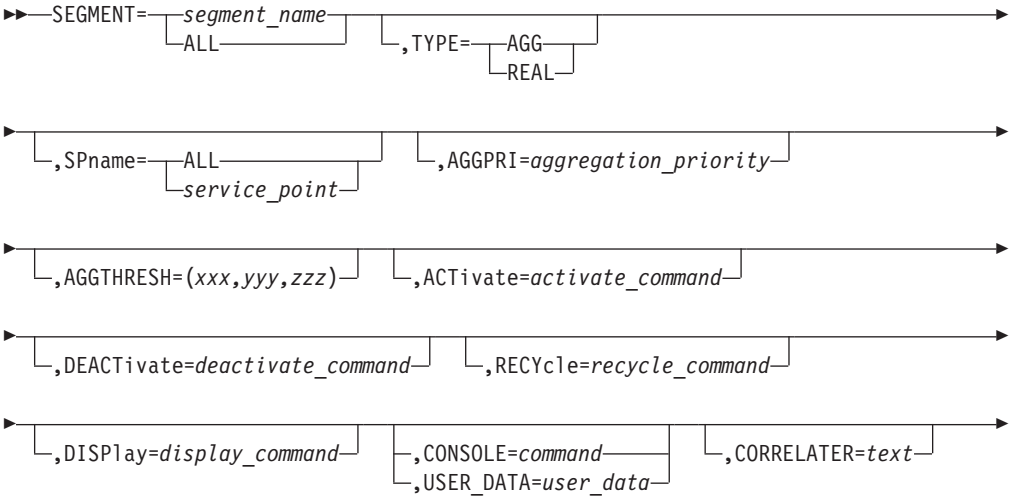
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

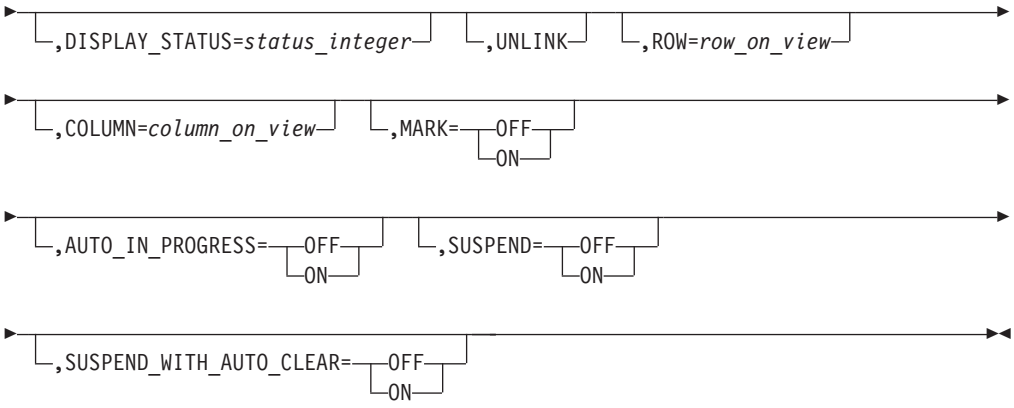
SEGment 制御ステートメント:

説明: SEGment 制御ステートメントは、処理されるマルチシステム・マネージャー LNM セグメント・リソースを指定します。

構文:

SEGMENT





パラメーター:

segment_name
セグメント番号 (3 から 4 文字) またはセグメント名 (SEGxxxx など)。 ALL またはワイルドカード名を指定できます。

TYPE
セグメント・リソースのタイプを指定します。値は次のとおりです。

REAL	実セグメント・リソース
AGG	集合セグメント・リソース

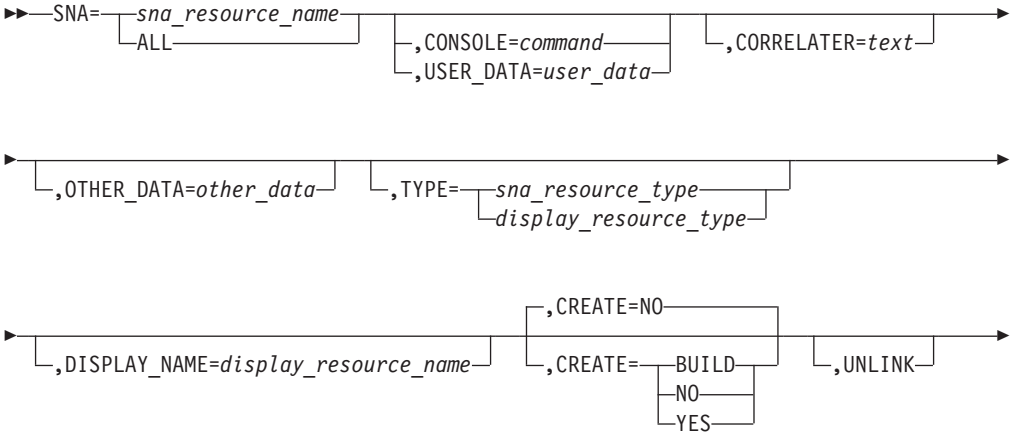
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

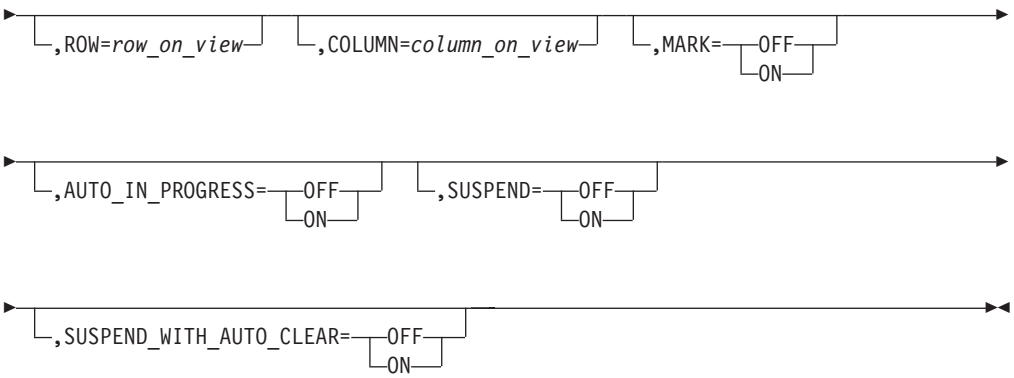
SNA 制御ステートメント:

説明: SNA 制御ステートメントは、処理される SNA (GMFHS シャドー) リソースを指定します。

構文:

SNA





パラメーター:

sna_resource_name
network.resource の形式による、1 から 17 文字の SNA のリソース名。 CREATE=NO では、ALL またはワイルドカード名を指定できます。

TYPE
SNA リソースのタイプを指定します。 CREATE=YES では TYPE が必要とされ、他の値では無視されます。 TYPE 値は、SNA オブジェクトについて RODM に設定する DisplayResourceType 値を決めます。次に示す値の 1 つ、または本書に記載されている有効な DisplayResourceType の値を指定できます。

HOST	DUIXC_RTS_HOST
GATEWAY_NCP	DUIXC_RTS_GATEWAY_NCP
NCP	DUIXC_RTS_PU4
PU4	DUIXC_RTS_PU4
APPL	DUIXC_RTS_APPL
CDRM	DUIXC_RTS_CDRM
CDRSC	DUIXC_RTS_CDRSC
LINK	DUIXC_LTS_GENERIC_LINK
PU21	DUIXC_RTS_PU21
PU20	DUIXC_RTS_PU20
PU1	DUIXC_RTS_PU1
PU	DUIXC_RTS_GENERIC_PU
LU	DUIXC_RTS_LU

DISPLAY_NAME
オブジェクトの RODM DisplayResourceName を指定します。この値は、リソースに関して sna_resource_name に代わる値として NMC ワークステーション上に表示されます。

注: BLDVIEWS は、値の中の任意の位置にコーディングできる %NAME% 置換変数を提供します。これを使用して、複数のリソースの DisplayResourceName を 1 つの制御ステートメントで再フォーマットできます。

CREATE
指定されたリソース上で実行するアクションを指定します。

YES	このリソースの新規オブジェクトを作成します。オブジェクトが存在する場合、最初にそれが削除されます。
NO	このリソースの新規オブジェクトを作成しません。作成する代わりに、オブジェクトを更新します。オブジェクトが存在しない場合、エラーが発生します。 NO がデフォルトです。
BUILD	このリソースの新規オブジェクトが存在しない場合、作成します。存在する場合、オブジェクトを更新します。

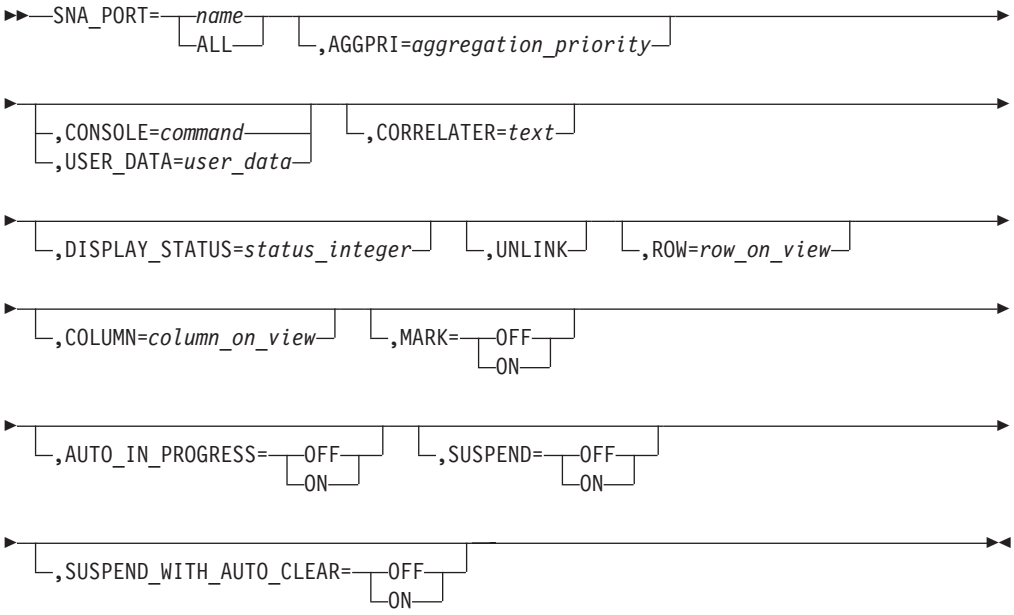
サポートされる他のキーワードについては、673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

SNA_PORT 制御ステートメント:

説明: SNA_PORT 制御ステートメントは、処理される SNA ドメイン・リソースを指定します。

構文:

SNA_PORT



パラメーター:

name

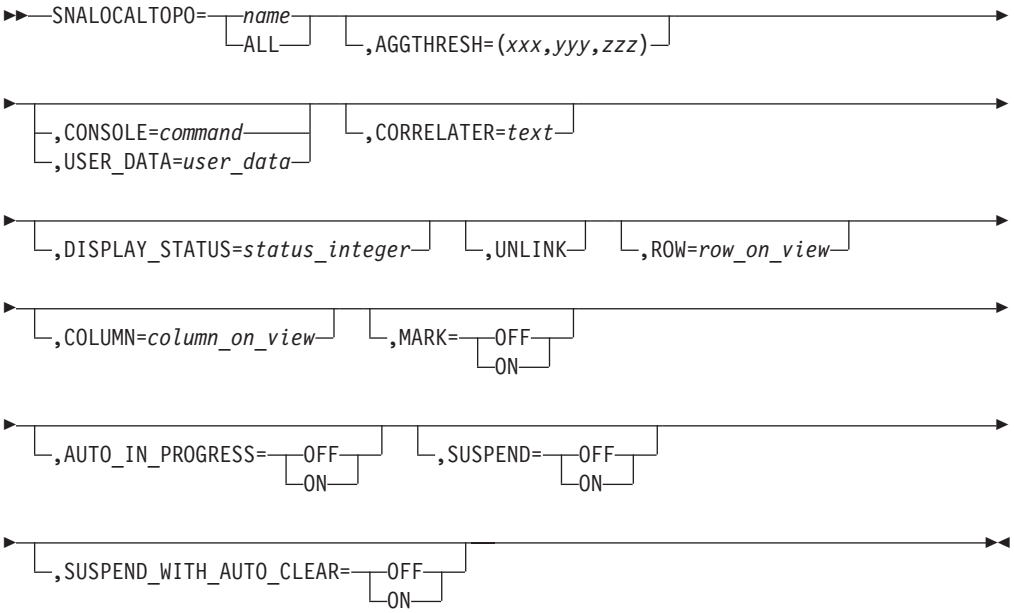
snaNetID.portId の形式による SNA ポート・リソース名。 ALL またはワイルドカード名を指定できます。

SNALOCALTOPO 制御ステートメント:

説明: SNALOCALTOPO 制御ステートメントは、処理される APPN SNA ローカル・トポロジー・リソースを指定します。

構文:

SNALOCALTOPO



パラメーター:

name
snaNetID.snaNodeName の形式による、APPN SNA ローカル・トポロジー・リソース名。 ALL またはワイルドカード名を指定できます。

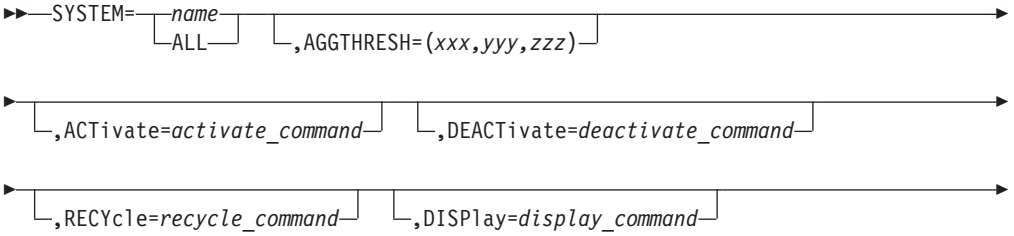
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

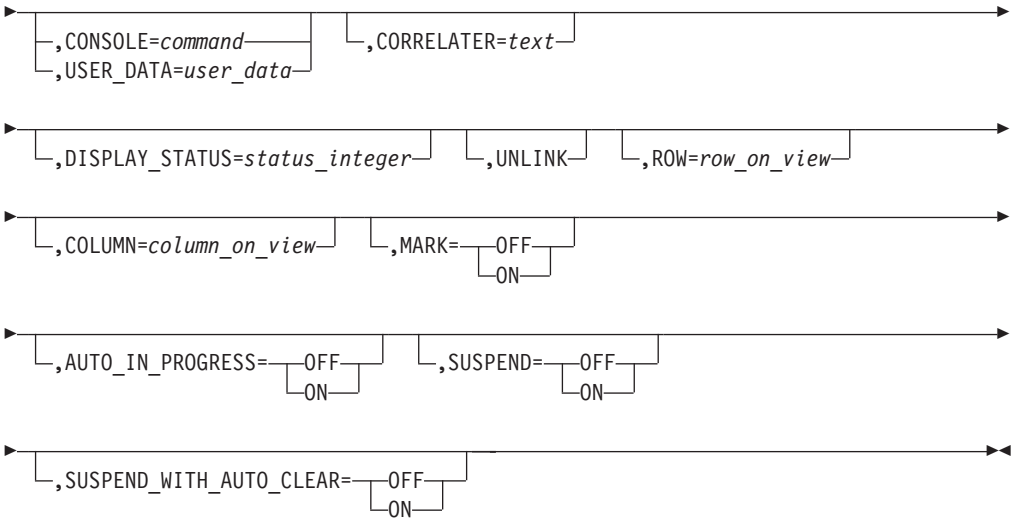
SYSTEM 制御ステートメント:

説明: SYSTEM 制御ステートメントは、処理されるワークステーション・システム集合リソースを指定します。

構文:

SYSTEM





パラメーター:

name
システムの名前。この名前は、ワークステーションのタイプに応じて次のいずれかになります。
- ニックネーム
- コンピューター名 (IBMLAN.INI ファイルにある物理名)
- MAC アドレス
- IPX アドレス

ALL またはワイルドカード名を指定できます。

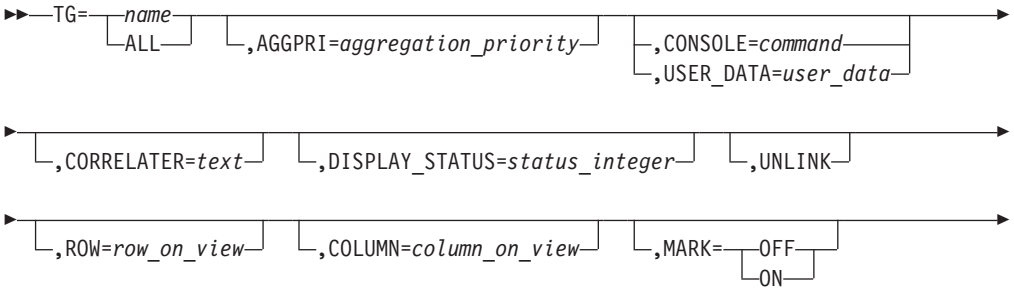
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

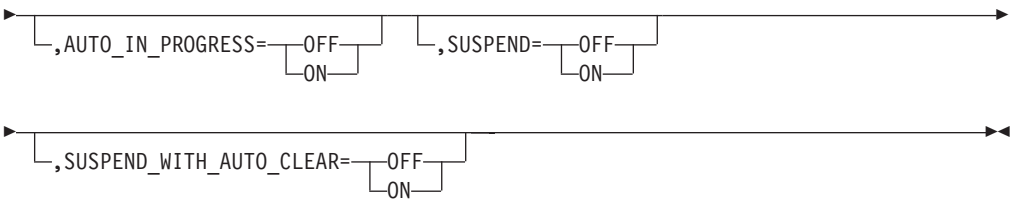
TG 制御ステートメント:

説明: TG 制御ステートメントは、処理される APPN 伝送グループ・リソースを指定します。

構文:

TG





パラメーター:

name
次の形式のいずれかによる、APPN 伝送グループ・リソース名。
- snaNetID.snaNodeName.tgn{.adj_snaNetID}.adj_snaNodeName
- snaNetID.vrnNodeName.tgn{.adj_snaNetID}.adj_snaNodeName
- snaNetID.snaNodeName.tgn{.adj_snaNetID}.adj_vrnNodeName

この名前は、リソースが NMC に表示されるときと同じ形式 (DisplayResourceName) になります。 ALL またはワイルドカード名を指定できます。

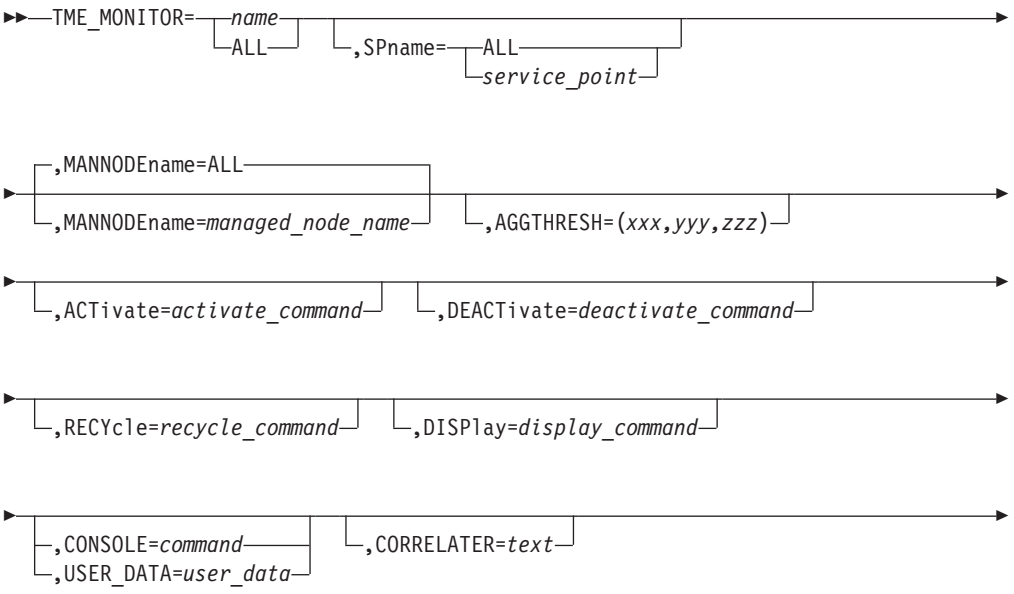
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

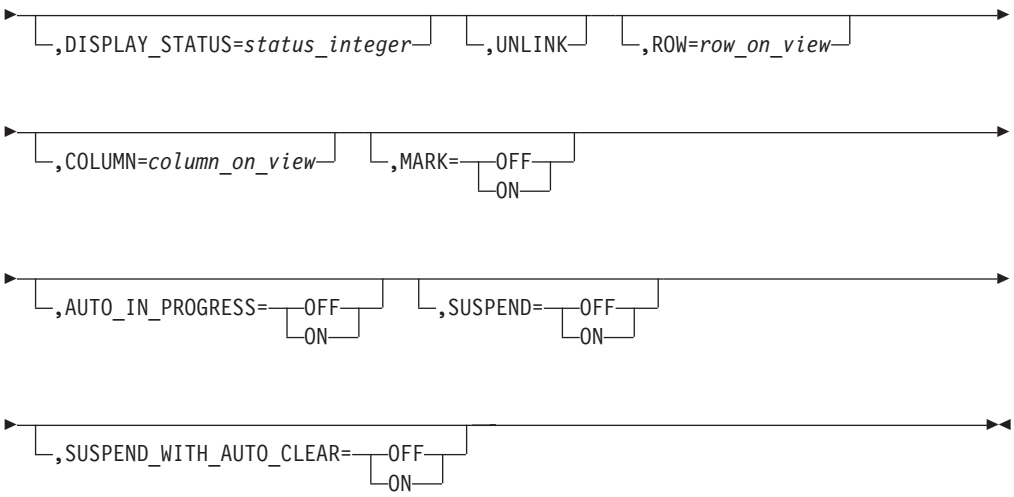
TME_MONITOR 制御ステートメント:

説明: TME_MONITOR 制御ステートメントは、処理されるマルチシステム・マネージャー TME モニター・リソースを指定します。

構文:

TME_MONITOR





パラメーター:

name
TME モニター・リソース名。

ALL またはワイルドカード名を指定できます。

managed_node_name
管理対象ノードとして TMR に定義された名前。

ALL またはワイルドカード名を指定できます。

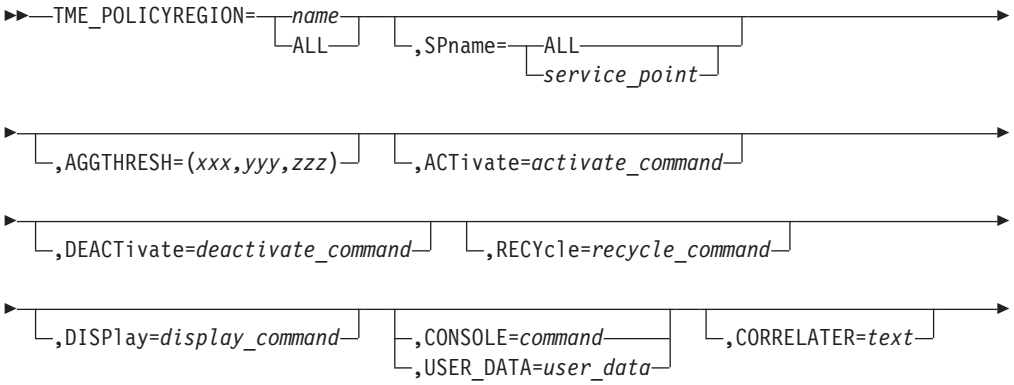
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

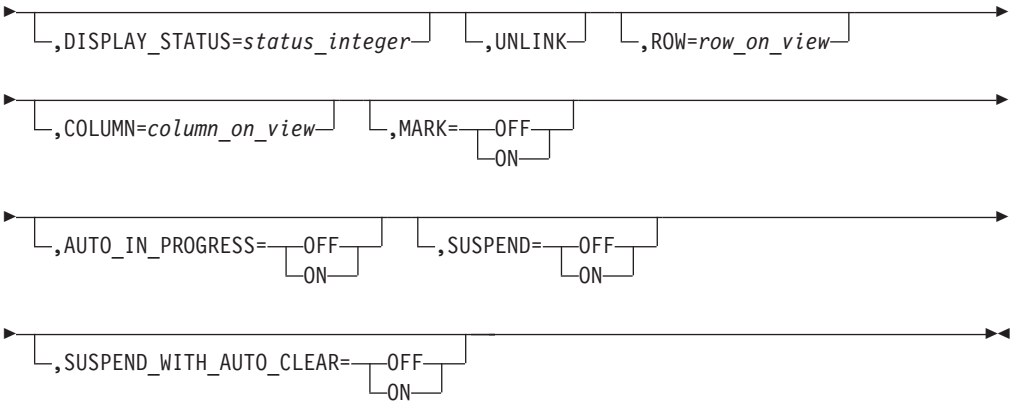
TME_POLICYREGION 制御ステートメント:

説明: TME_POLICYREGION 制御ステートメントは、処理されるマルチシステム・マネージャー TME ポリシー・リージョン・リソースを指定します。

構文:

TME_POLICYREGION





パラメーター:

name
TME ポリシー・リージョン名。ALL またはワイルドカード名を指定できます。

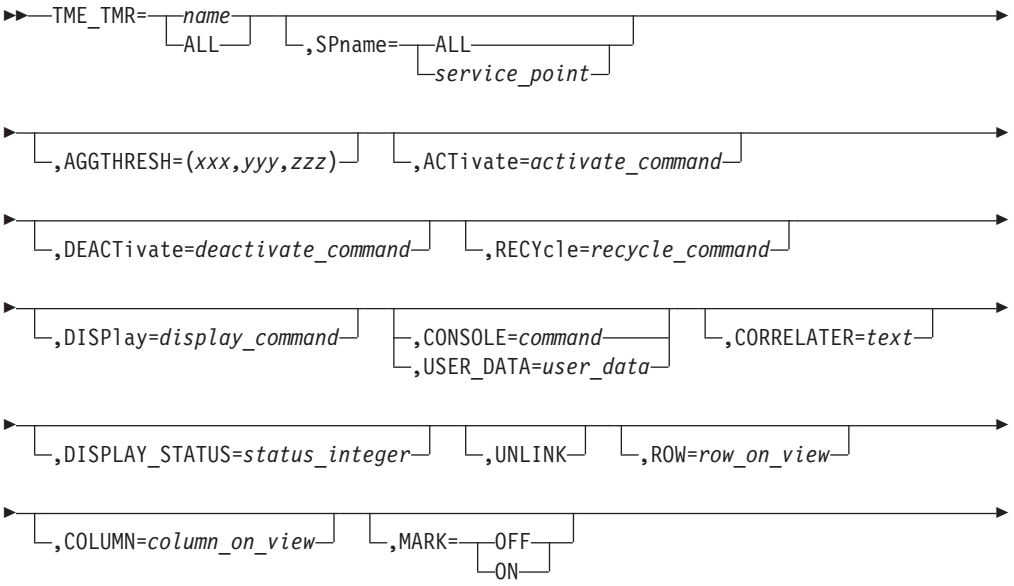
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

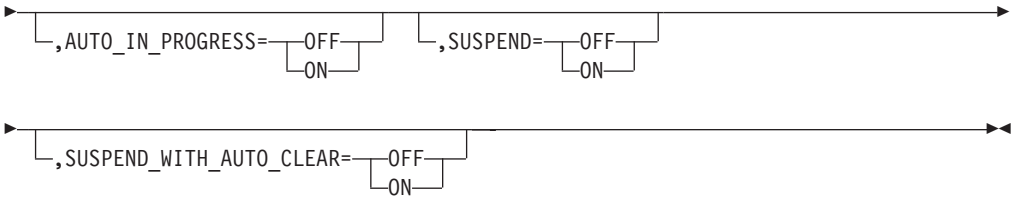
TME_TMR 制御ステートメント:

説明: TME_TMR 制御ステートメントは、処理されるマルチシステム・マネージャー TME 管理リージョン・リソースを指定します。

構文:

TME_TMR





パラメーター:

name
TME 管理リージョン名。 ALL またはワイルドカード名を指定できます。

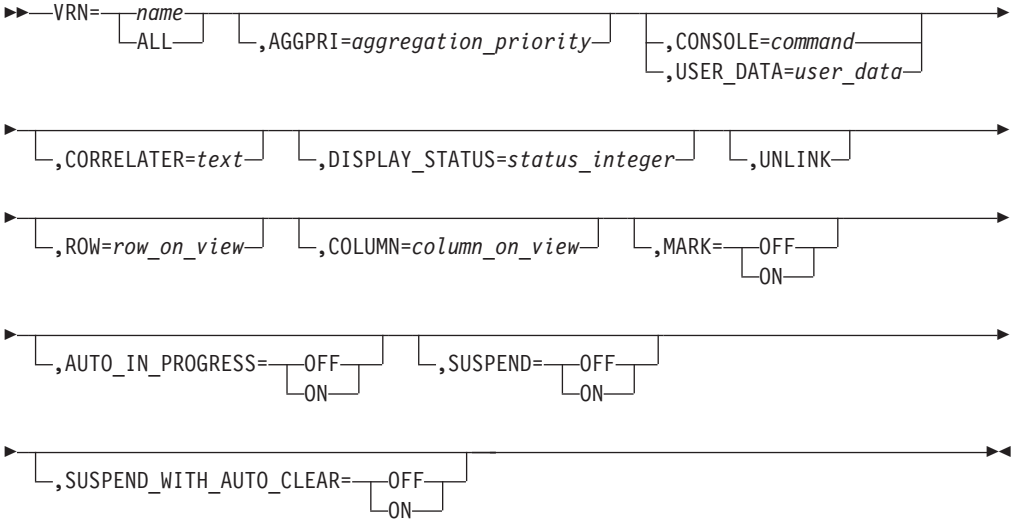
サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

VRN 制御ステートメント:

説明: VRN 制御ステートメントは、処理される APPN 仮想ルーティング・ノードを指定します。

構文:

VRN



パラメーター:

name
1 から 17 文字の SNA 仮想ルーティング・ノードのリソース名で、形式は snaNetID.snaNodeName です。 ALL またはワイルドカード名を指定できます。

サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

集約制御ステートメント

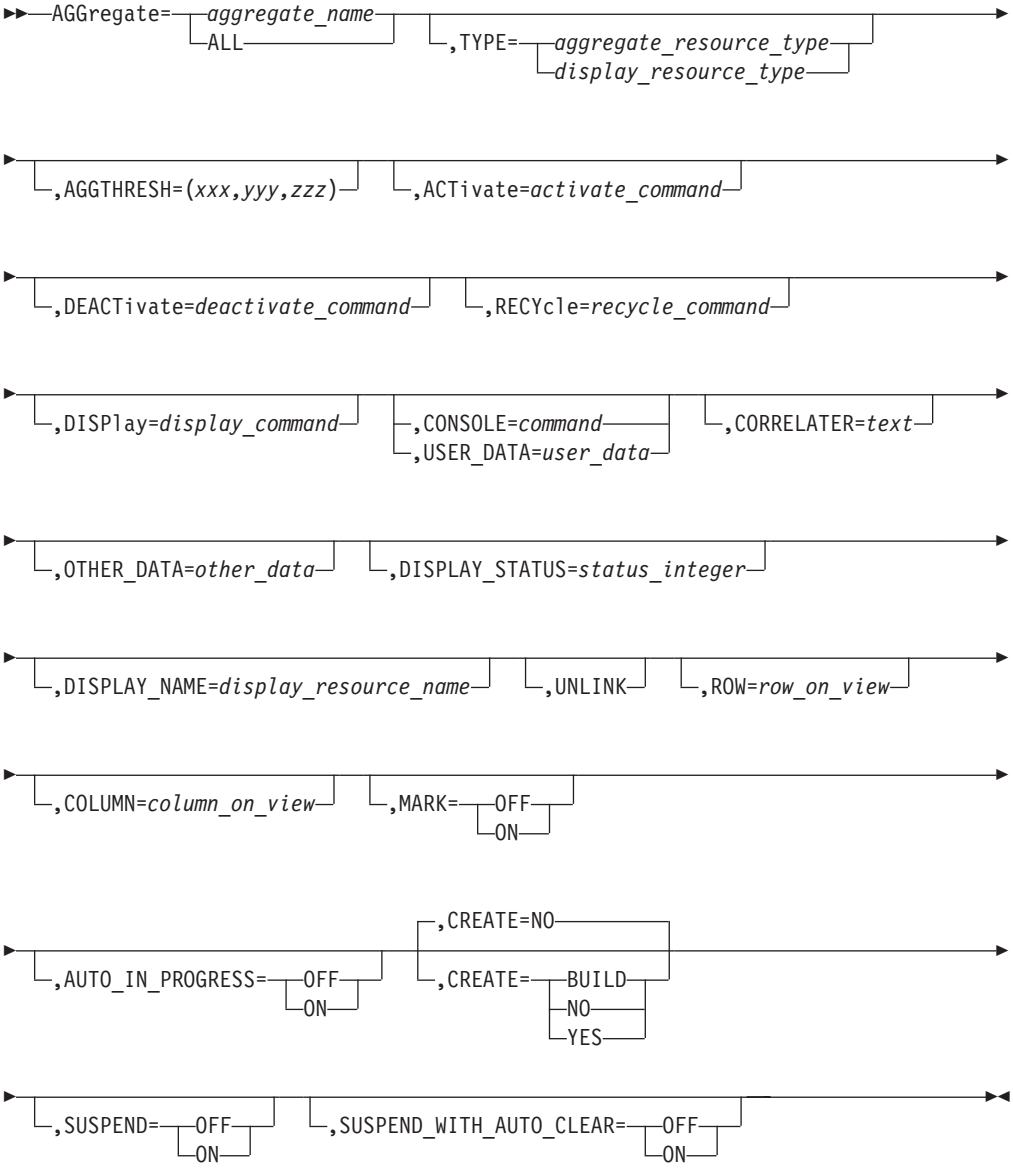
以下の制御ステートメントは、作成される、または更新される集合リソース、および集合リソースを構成するリソースを指定します。

AGGregate 制御ステートメント:

説明: AGGregate 制御ステートメントは、処理される Aggregate (GMFHS Aggregate) リソースを指定します。

構文:

AGGregate



パラメーター:

aggregate_name
集合リソース名。

CREATE=NO では、ALL またはワイルドカード名を指定できます。

TYPE

集合リソースのタイプを指定します。 CREATE=YES では TYPE が必要とされ、他の値では無視されます。 TYPE 値は、集合オブジェクトの RODM で設定する DisplayResourceType の値を決定します。次に示す値の 1 つ、または本書に記載されている有効な DisplayResourceType の値を指定できます。

LAN_CLUSTER
DUIXC_RTN_LAN_NETWORK_AGG

LAN_NETWORK
DUIXC_RTN_LAN_AGG

TME_TMR DUIXC_RTN_MANAGED_REGION_AGG

TME_POLICYREGION
DUIXC_RTN_POLICY_REGION_AGG

TME_MONITOR
DUIXC_RTN_MONITOR

SEGMENT DUIXC_RTN_TR_SEGMENT_AGG

BRIDGE DUIXC_RTN_BRIDGE_AGG

CAU DUIXC_RTN_LAN_CONCENT_AGG

IP_CLUSTER DUIXC_RTN_INTERNET_CLUSTER

IP_NETWORK
DUIXC_RTN_INTERNET_MGMT_DOMAIN_AGG

IP_SUBNET DUIXC_RTN_INTERNET_SUBNET_AGG

IP_SEGMENT DUIXC_RTN_INTERNET_SEGMENT_AGG’

IP_LOCATION
DUIXC_RTN_INTERNET_LOCATION_AGG

IP_ROUTER DUIXC_RTN_INTERNET_ROUTER_AGG

IP_HUB - DUIXC_RTN_INTERNET_HUB_AGG

IP_BRIDGE - DUIXC_RTN_INTERNET_BRIDGE_AGG

IP_HOST DUIXC_RTN_INTERNET_HOST_AGG

IP_LINK DUIXC_RTN_LTN_IP_LINK_AGG

SYSTEM DUIXC_RTN_OPEN_SYSTEM_AGG

APPN_DOMAIN
DUIXC_RTN_NN_DOMAIN_AGG

APPN_NETWORK
DUIXC_RTN_NN_DOMAIN_NETWORK

APPN_CLUSTER
DUIXC_RTN_NN_DOM_NET_CLUSTER

SNALOCALTOPO
DUIXC_RTN_NN_LOCAL_TOP_AGG

USER DUIXC_RTN_NODE_AGG_USER1

CREATE

指定されたリソース上で実行するアクションを指定します。

YES このリソースの新規オブジェクトを作成します。古いオブジェクトが存在する場合、削除されます。

NO このリソースの新規オブジェクトを作成しません。代わりに、オブジェクトを更新します。オ

ブジェクトが存在しない場合、エラーが発生します。 NO がデフォルトです。

BUILD
このリソースの新規オブジェクトが存在しない場合、作成します。存在する場合、オブジェクトを更新します。

サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

注: AGGregate 制御ステートメントは、新規集合体を作成するか、または GMFHS_Aggregate_Objects_Class クラスに属する既存の集合体を参照します。

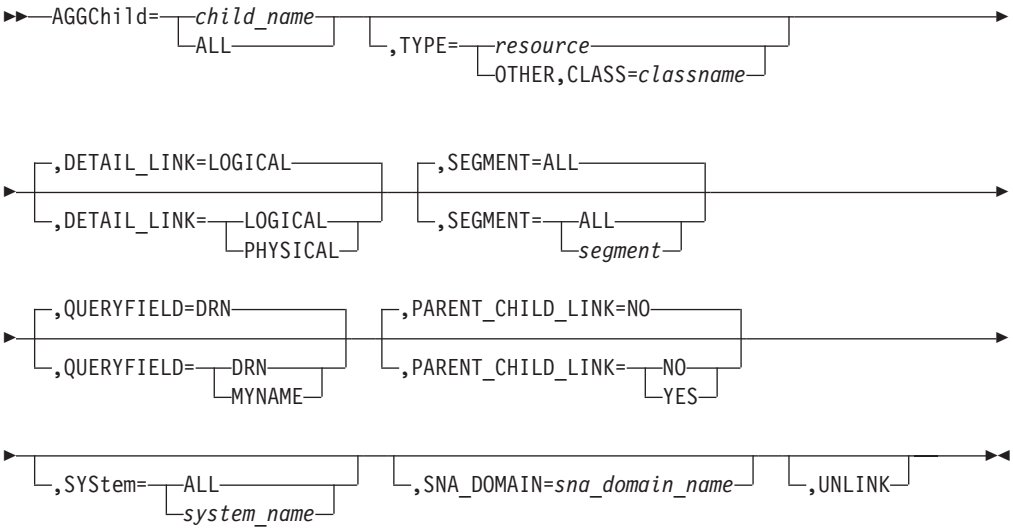
AGGChild 制御ステートメントが AGGregate 制御ステートメントに続く場合、AGGChild 制御ステートメントで指定されたリソースは、AGGCHILD 制御ステートメントが UNLINK=YES を指定しなければ、 AGGregate 制御ステートメントで指定された集合体にリンクされます。

AGGChild 制御ステートメント:

説明: AGGChild 制御ステートメントは、 AGGChild 制御ステートメントに先行する AGGregate ステートメントの集合リソースとリンクさせたい、またはリンク解除させたい集約子リソースを指定します。

構文:

AGGChild



パラメーター:

name
リソースの名前。名前の形式および長さは、リソースのタイプに依存します。

ALL またはワイルドカード名を指定できます。

TYPE

リソースのタイプを指定します。タイプは、特定のリソース制御ステートメントに対応します。

- LAN_CLUSTER
- LAN_NETWORK
- LAN_AGENT
- TME_CLUSTER
- TME_NETWORK
- TME_AGENT
- BRIDGE
- BRIDGE_AGG
- SEGMENT
- SEGMENT_AGG
- CAU
- CAU_AGG
- ADAPTER | ADP
- STATION_ADAPTER
- BRIDGE_ADAPTER
- CAU_ADAPTER
- LAN_ADAPTER
- TME_TMR
- TME_POLICYREGION
- TME_MONITOR
- IP_CLUSTER
- IP_NETWORK
- IP_AGENT
- IP_SUBNET
- IP_LOCATION
- IP_SEGMENT
- IP_ROUTER
- IP_HUB
- IP_BRIDGE
- IP_HOST
- IP_LINK
- INTERFACE
- SYSTEM
- NONSNA
- APPN_CLUSTER
- APPN_NETWORK
- SNALOCALTOPO

- NNODE
- ENODE
- LNODE
- LINE
- SNA_PORT
- LAN_PORT
- DOMAIN
- LLINK
- TG
- APPN_VRN
- APPN_TG_CIRCUIT
- INTER_DOMAIN_CIRCUIT
- INTER_SUBNETWORK_CIRCUIT
- CN_CIRCUIT
- NTRI_CIRCUIT
- SUBAREA_TG_CIRCUIT
- AGG
- APPL_MAJNODE
- CDRSC_MAJNODE
- CDRM_MAJNODE
- LAN_MAJNODE
- LCLNONSNA_MAJNODE
- LOCALSNA_MAJNODE
- LUGROUP_MAJNODE
- NCP_MAJNODE
- PACKET_MAJNODE
- SWITCHED_MAJNODE
- TRL_MAJNODE
- XCA_MAJNODE
- HOST_NODE
- IC_NODE
- MIG_DATA_HOST
- GW_NCP
- NCP_GW
- NCP_NON_GW
- CDRM
- CDRSC
- PU
- LU
- LU_GROUP

- CA_MAJNODE

DETAIL_LINK
集合体子と集合体の間に確立する接続のタイプを指定します。

LOGICAL	論理接続 (DEFAULT) を使用して、集合体の子を集合体にリンクします。
PHYSICAL	物理接続を使用して、集合体の子を集合体にリンクします。

segment_name
(STATION_ADAPTER、BRIDGE_ADAPTER、CAU_ADAPTER、または LAN_ADAPTER) セグメント番号 (3 から 4 文字) またはセグメント名 (例えば、SEGxxxx)。 ALL も指定可能で、これがデフォルトです。

segment_name
(INTERFACE) セグメント名 (1 から 64 文字)。 ALL を指定することができ、これがデフォルトです。

sna_domain_name
メジャー・ノード・リソースを所有する VTAM SNA ドメインを指定します。これは、SNA_DOMAIN 制御ステートメントで指定した値をオーバーライドします。名前の形式は、network.domain です。

QUERYFIELD
NONSNA リソース・クラス (GMFHS_Managed_Real_Objects_Class) から RODM オブジェクト照会で使用するフィールドを指定します。QUERYFIELD=DRN を指定すると、 DisplayResourceName フィールドを使用してオブジェクトを取り出します。 QUERYFIELD=MYNAME を指定すると、MyName フィールドを使用してオブジェクトを取り出します。 QUERYFIELD が NONSNA 制御ステートメント上に指定されていない場合、DRN がデフォルトとなります。

PARENT_CHILD_LINK
ヌル・リンクを使用して、集合の子を集合の親にリンクするオプションを使用可能にします。パラメーターは以下のようにコーディングされます。

PARENT_CHILD_LINK=YES (NO がデフォルトです)

サポートされる他のキーワードについては、 673 ページの『共通の制御ステートメント・パラメーター』を参照してください。

BLDVIEWS の実行

BLDVIEWS に、指定するビューおよび集合体の作成を指示する BLDVIEWS 制御ステートメントをコーディングします。制御ステートメントは、NetView DSIPARM メンバーで、または完全に修飾されたカタログ式順次データ・セット (メンバーとともに指定された PDS を含む) で、あるいは REXX ステム配列でコーディングされ、 NetView PIPE コマンドを使用して BLDVIEWS に渡されます。

NetView DSIPARM メンバーでの制御ステートメントのコーディング

制御ステートメントが DSIPARM メンバーでコーディングされる場合、構文は次のようになります。

```
BLDVIEWS dsiparm_member {RODM=rodmname}
                       {TEST=YES|NO}
                       {ECHO=YES|NO}
                       {QUIET=YES|NO}
                       {OPTIMIZE=CPU|STORage}
```

dsiparm_member
BLDVIEWS 制御ステートメントを含む NetView DSIPARM メンバー名。

rodmname
接続先の RODM 名。 *rodmname* はオプションです。これが指定されない場合、マルチシステム・マネージャー共通グローバル FLC_RODMNAME が使用されます。

TEST=YES
BLDVIEWS は、制御ステートメントの構文検査しか行いません。アクションは実行されません。 RODM はアクティブである必要はありません。デフォルトは、TEST=NO です。

ECHO=YES
BLDVIEWS は、制御ステートメントが読み取られる際、また制御ステートメントが処理される前に、一度に 1 つの制御ステートメントを表示します。デフォルトは、ECHO=NO です。

QUIET=YES
BLDVIEWS はエラー・メッセージ以外のすべてのメッセージを抑制します。デフォルトは、QUIET=NO です。

OPTIMIZE

CPU BLDVIEWS の結果は、ストレージ内の REXX 配列に、クラス全体を照会した結果を保管します。これが行われることによって、 BLDVIEWS の実行時に複数回クラスを照会するために必要なサイクルが削減されます。これによってサイクルが省かれますが、ストレージにデータを保持するために余分のストレージが使用されます。これはデフォルトです。ストレージに制約がある場合、OPTIMIZE=STORage を指定する必要があるかもしれません。

STORage BLDVIEWS は、ストレージ内の REXX 配列に、クラス全体を照会した結果を保管しません。これにより、これらのクラスにあるリソースが同じ BLDVIEWS の実行中に後で再び必要となる場合に、ストレージが節約されますが CPU の消費は増えます。

完全修飾データ・セットでの制御ステートメントのコーディング

制御ステートメントがカタログ式データ・セットでコーディングされる場合、構文は次のようになります。

```
BLDVIEWS data_set {RODM=rodmname}
                  {TEST=YES|NO}
                  {ECHO=YES|NO}
                  {QUIET=YES|NO}
                  {OPTIMIZE=CPU|STORage}
```

data_set
BLDVIEWS 制御ステートメントを含む、完全に修飾されたカタログ式データ・

セットの名前。データ・セットは、順次ファイルか、またはメンバーとともに指定された区分データ・セットになります。

rodmname
接続先の RODM 名。これはオプションです。これが指定されない場合、マルチシステム・マネージャー共通グローバル FLC_RODMNAME が使用されます。

TEST=YES
BLDVIEWS は、制御ステートメントの構文検査しか行いません。アクションは実行されません。 RODM はアクティブである必要はありません。デフォルトは、TEST=NO です。

ECHO=YES
BLDVIEWS は、制御ステートメントが読み取られる際、また制御ステートメントが処理される前に、一度に 1 つの制御ステートメントを表示します。デフォルトは、ECHO=NO です。

QUIET=YES
BLDVIEWS はエラー・メッセージ以外のすべてのメッセージを抑制します。デフォルトは、QUIET=NO です。

OPTIMIZE

CPU BLDVIEWS の結果は、ストレージ内の REXX 配列に、クラス全体を照会した結果を保管します。これが行われることによって、 BLDVIEWS の実行時に複数回クラスを照会するために必要なサイクルが削減されます。これによってサイクルが省かれますが、ストレージにデータを保持するために余分のストレージが使用されます。これはデフォルトです。ストレージに制約がある場合、OPTIMIZE=STORage を指定する必要があるかもしれません。

STORage BLDVIEWS は、ストレージ内の REXX 配列に、クラス全体を照会した結果を保管しません。これにより、これらのクラスにあるリソースが同じ BLDVIEWS の実行中に後で再び必要となる場合に、ストレージが節約されますが CPU の消費は増えます。

例:

```
BLDVIEWS ESP.NV24.BLDVIEWS(MYVIEWS)

BLDVIEWS ESP.NV24.BLDVIEWS.DATA1
```

REXX ステム配列での制御ステートメントのコーディング

制御ステートメントが REXX ステム配列でコーディングされる場合、構文は次のようになります。

```
'PIPE STEM stem_array. | COLLECT | NETV BLDVIEWS',
                                  '{RODM=rodmname}',
                                  '{TEST=YES|NO}',
                                  '{ECHO=YES|NO}',
                                  '{QUIET=YES|NO}',
                                  '{OPTIMIZE=CPU|STORage} | ....'
```

stem_array
BLDVIEWS 制御ステートメントを含む REXX ステム配列変数の名前。 *stem.array.0* には、配列内の項目の数が含まれます。

rodmname
接続先の RODM 名。これはオプションです。これが指定されない場合、rodmname にはマルチシステム・マネージャー共通グローバル FLC_RODMNAME が使用され、 RODM userid には共通グローバル FLC_RODMAPPL が使用されます。

rodmname が指定される場合、BLDVIEWS を実行するタスクの NetView オペレーター ID が、 RODM ユーザー ID として使用されます。このユーザー ID には、RODM への適切な SAF アクセス権がなければなりません。

TEST=YES
BLDVIEWS は、制御ステートメントの構文検査しか行いません。アクションは実行されません。 RODM はアクティブである必要はありません。デフォルトは、TEST=NO です。

ECHO=YES
BLDVIEWS は、制御ステートメントが読み取られる際、また制御ステートメントが処理される前に、一度に 1 つの制御ステートメントを表示します。デフォルトは、ECHO=NO です。

QUIET=YES
BLDVIEWS はエラー・メッセージ以外のすべてのメッセージを抑制します。デフォルトは、QUIET=NO です。

OPTIMIZE

CPU BLDVIEWS の結果は、ストレージ内の REXX 配列に、クラス全体を照会した結果を保管します。これが行われることによって、 BLDVIEWS の実行時に複数回クラスを照会するために必要なサイクルが削減されます。これによってサイクルが省かれますが、ストレージにデータを保持するために余分のストレージが使用されます。これはデフォルトです。ストレージに制約がある場合、OPTIMIZE=STORage を指定する必要があるかもしれません。

STORage BLDVIEWS の結果は、ストレージ内の REXX 配列に、クラス全体を照会した結果を保管しません。これにより、これらのクラスにあるリソースが同じ BLDVIEWS の実行中に後で再び必要となる場合に、ストレージが節約されますが CPU の消費は増えます。

例:

```
/* REXX  */

statement.1="VIEW=My_View,ANNOTATION='This is my View',",
statement=2=' CREATE=YES'
statement.3='NONSNA='resource',CREATE=YES,',
       ||'TYPE=DUIXC_RTN_HOST'


statement.0=3


'PIPE STEM statement. | COLLECT | NETV FLCVBLDV | CONSOLE'
exit
```

BLDVIEWS 制御ステートメントの例

このセクションには、BLDVIEWS 制御ステートメントのコーディング例が示されています。 表 239 の記述を利用して、要件に最も良く合う例を判別してください。

表 239. BLDVIEWS 制御ステートメントの例

例	説明	ページ
1	マルチシステム・マネージャー・オブジェクトの集約しきい値を変更する。	749
2	マルチシステム・マネージャー・オブジェクト用の総称コマンドを RODM で設定する。	749
3	マルチシステム・マネージャー・オブジェクトに対して RODM での総称コマンドを設定し、DisplayStatusCommandText が rping を行うように、また DisplayResourceUserData が TELNETPM を行うように設定する。	750
4	DisplayResourceName を非 SNA リソースに対して設定する。	750
5	サービス・ポイントによって管理されるブリッジ集合リソースすべてを含むビューを作成する。	750
6	サービス・ポイント A19SRVCP によって管理される特定のブリッジおよびセグメント・リソースを含むビューを作成する。	750
7	2 つの新規集合オブジェクトを含むビューを作成する。	751
8	レイアウト・タイプ 6 (階層) を持つビュー、および特定の行のビューのリソースを作成する。	751
9	ブリッジ・リソースをビューからリンク解除する。	752
10	TME 管理リージョン集合リソースすべてを含むビューを作成する。	752
11	RTP で始まる TME ポリシー・リージョン・リソースすべてを含むビューを作成する。	752

BLDVIEWS の例 1:

この例では、LNM リソースおよび TCP/IP リソースの、マルチシステム・マネージャー・クラスターおよびネットワーク集合用の集約しきい値を変更します。集約しきい値は、25%、50%、および 75% に変更されています。

```
NETWORK=ALL,AGGTHRESH=(25%,50%,75%),TYPE=LAN
CLUSTER=ALL,AGGTHRESH=(25%,50%,75%),TYPE=LAN
NETWORK=ALL,AGGTHRESH=(25%,50%,75%),TYPE=IP
CLUSTER=ALL,AGGTHRESH=(25%,50%,75%),TYPE=IP
```

BLDVIEWS の例 2:

この例では、マルチシステム・マネージャー、アダプター、ブリッジ、および制御アクセス・ユニットに対して RODM での総称コマンドを設定します。マルチシステム・マネージャー LNM リソースの場合、 BLDVIEWS はコマンドに演算子と相関係数を付加することにご注意ください。

```
ADP=ALL,
        DISPLAY='ADP QUERY ADP=%NAME% SEG=%SEGMENT%',
        DEACTIVATE='ADP REMOVE ADP=%NAME% SEG=%SEGMENT%'

BRIDGE=ALL,TYPE=REAL,
```

```
            DISPLAY='BRG QUERY NAME=%NAME%',
            ACTIVATE='BRG LINK NAME=%NAME%',
            DEACTIVATE='BRG UNLINK NAME=%NAME%'

  CAU=ALL,TYPE=REAL,
          DISPLAY='CAU QUERY UNIT=%NAME%',
          RECYCLE='CAU RESTART UNIT=%NAME% CONFIRM=N'
```

BLDVIEWS の例 3:

この例は、マルチシステム・マネージャー TCP/IP ルーター、ハブ、ブリッジ、ホスト、およびアダプターに対して RODM での総称コマンドを設定します。 DisplayStatusCommandText (総称表示コマンド) フィールドは、rping を行うように設定されます。 DisplayResourceUserData (リモート・コンソール) は、TELNETPM を行うように設定されます。

BLDVIEWS は、コマンドを RemoteConsole = # および # で囲みますが、これは DisplayResourceUserData フィールドを適切に設定して、リモート・コンソール・サポートが正常に機能するようにします。

```
IP_ROUTER=ALL,
              DISPLAY='asis rping -n 2 %NAME%',
              CONSOLE='TELNETPM.EXE %NAME%'

IP_HUB=ALL,
              DISPLAY='asis rping -n 2 %NAME%',
              CONSOLE='TELNETPM.EXE %NAME%'

IP_BRIDGE=ALL,
              DISPLAY='asis rping -n 2 %NAME%',
              CONSOLE='TELNETPM.EXE %NAME%'

IP_HOST=ALL,
             DISPLAY='asis rping -n 2 %NAME%',
             CONSOLE='TELNETPM.EXE %NAME%'

INTERFACE=ALL,
           DISPLAY='asis rping -n 2 %NAME%',
           CONSOLE='TELNETPM.EXE %NAME%'
```

BLDVIEWS の例 4:

この例は、非 SNA リソース mercury.raleigh.ibm.com の DisplayResourceName を、Router1 に設定します。

```
NONSNA=mercury.raleigh.ibm.com,
       DISPLAY_NAME='Router1'
```

BLDVIEWS の例 5:

この例は、サービス・ポイント A19SRVCP によって管理されるブリッジ集合リソースすべてを含むビューを作成します。

```
VIEW=GAF_ALLBridgesA,ANNOTATION='All Bridge Aggregates'
LANSPNAME=A19SRVCP
BRIDGE=ALL,TYPE=AGG
```

BLDVIEWS の例 6:

この例は、サービス・ポイント A19SRVCP によって管理される特定のブリッジおよびセグメント・リソースを含むビューを作成します。また、この例は、セグメント集合の集約しきい値を、 20%、60%、および 80% に設定します。

```
VIEW=GAF_BLDG_500,ANNOTATION='Building 500'
LANSPNAME=A19SRVCP

BRIDGE=A085C17,TYPE=AGG
BRIDGE=A082C17,TYPE=AGG
BRIDGE=AC15C17,TYPE=AGG
BRIDGE=A056C17,TYPE=AGG
BRIDGE=AC15C16,TYPE=AGG
BRIDGE=A056C16,TYPE=AGG
BRIDGE=AC16B00,TYPE=AGG
BRIDGE=A032C01,TYPE=AGG
BRIDGE=A03B032,TYPE=AGG

SEGMENT=0C16,TYPE=AGG,AGGTHRESH=(20%,60%,80%)
SEGMENT=0056,TYPE=AGG,AGGTHRESH=(20%,60%,80%)
SEGMENT=0C15,TYPE=AGG,AGGTHRESH=(20%,60%,80%)
SEGMENT=0C17,TYPE=AGG,AGGTHRESH=(20%,60%,80%)
SEGMENT=0082,TYPE=AGG,AGGTHRESH=(20%,60%,80%)
SEGMENT=0085,TYPE=AGG,AGGTHRESH=(20%,60%,80%)
SEGMENT=0C01,TYPE=AGG,AGGTHRESH=(20%,60%,80%)
SEGMENT=0032,TYPE=AGG,AGGTHRESH=(20%,60%,80%)
SEGMENT=003B,TYPE=AGG,AGGTHRESH=(20%,60%,80%)
```

BLDVIEWS の例 7:

この例は、特定のリソースを持つ 2 つの新規集合リソースを含むビューを作成します。

```
VIEW=GAF_Key_Bridges,ANNOTATION='Key Bridges'
LANSPNAME=A19SRVCP

AGG=GAF_B500_Bridges,type=BRIDGE,
                      AGGTHRESH=(40%,60%,75%),CREATE=YES
AGGCHILD=A085C17,TYPE=BRIDGE_AGG
AGGCHILD=A082C17,TYPE=BRIDGE_AGG
AGGCHILD=AC15C17,TYPE=BRIDGE_AGG
AGGCHILD=A056C17,TYPE=BRIDGE_AGG
AGGCHILD=AC15C16,TYPE=BRIDGE_AGG
AGGCHILD=A056C16,TYPE=BRIDGE_AGG
AGGCHILD=AC16B00,TYPE=BRIDGE_AGG
AGGCHILD=A032C01,TYPE=BRIDGE_AGG
AGGCHILD=A03B032,TYPE=BRIDGE_AGG

AGG=GAF_MS_Bridges,type=BRIDGE,
                    AGGTHRESH=(40%,60%,75%),CREATE=YES
AGGCHILD=AC01B00,TYPE=BRIDGE_AGG
AGGCHILD=AB01B00,TYPE=BRIDGE_AGG
AGGCHILD=AC03B00,TYPE=BRIDGE_AGG
AGGCHILD=AC24B00,TYPE=BRIDGE_AGG
AGGCHILD=AC03B01,TYPE=BRIDGE_AGG
AGGCHILD=AC24B01,TYPE=BRIDGE_AGG
AGGCHILD=AC05B00,TYPE=BRIDGE_AGG
AGGCHILD=AC06B01,TYPE=BRIDGE_AGG
AGGCHILD=A059C05,TYPE=BRIDGE_AGG
AGGCHILD=A059C06,TYPE=BRIDGE_AGG
AGGCHILD=A061C05,TYPE=BRIDGE_AGG
AGGCHILD=A062C05,TYPE=BRIDGE_AGG
AGGCHILD=A062C06,TYPE=BRIDGE_AGG
```

BLDVIEWS の例 8:

この例は、レイアウト・タイプ 6 (階層) を持つビューを作成し、特定のリソースを指定された行のビューに入れます。

```
VIEW=GAF_View_Hier,ANNOTATION='Resources on specific rows',
     LAYOUT=6
```

```
LANSPNAME=A19SRVCP
NWSPNAME=A19NWAPU

NONSNA=NV6000.CODEBUST.BUDDY,ROW=1

BRIDGE=A059C06,TYPE=AGG,ROW=2

SEGMENT=0C16,TYPE=AGG,ROW=3

CAU=5A982D60,TYPE=AGG,ROW=4

ADP=GARY,ROW=5

NWSERVER=ESP_A86A,TYPE=IBM_AGENT,ROW=5
```

BLDVIEWS の例 9:

この例は、ブリッジ・リソースをビューからリンク解除します。

```
VIEW=GAF_BLDG_500,CREATE=NO
LANSPNAME=A19SRVCP

BRIDGE=A085C17,TYPE=AGG,UNLINK
```

BLDVIEWS の例 10:

この例は、TME 管理リージョン集合リソースすべてを含むビューを作成します。

```
VIEW=TME_MANAGED_REGIONS,ANNOTATION='MANAGED REGION VIEW',
CREATE=YES,LAYOUT=9
TME_TMR=ALL
```

BLDVIEWS の例 11:

この例は、RTP で始まる TME ポリシー・リージョン・リソースすべてを含むビューを作成します。

```
VIEW=TME_POLICY_REGION_RTP,ANNOTATION='POLICY REGION VIEW',
CREATE=YES,LAYOUT=9
TME_POLICYREGION=RTP*
```

ビューの削除

このセクションでは、DELVIEWS を使用して、指定された接頭部で始まるビューまたはビューのグループを削除する方法について説明します。

DELVIEWS の構文

```
DELVIEWS view_name|view_name_prefix
         {TYPE=NETWORK|PEER|EXCP|BACKBONE|LC|PC|MDL|MDP}
         {RODM=rodmname}
```

view_name は、RODM から削除されるビューの名前です。

接頭部で始まるビューのグループを削除するには、ワイルドカード文字 * で接頭部を指定します。

TYPE は、削除するビューのタイプ (以下を参照) を指定します。

NETWORK	ネットワーク・ビュー (デフォルト)
PEER	構成対等機能ビュー
EXCP	例外ビュー
BACKBONE	構成バックボーン・ビュー

LC	論理接続ビュー
PC	物理接続ビュー
MDL	より詳細な論理ビュー
MDP	より詳細な物理ビュー

RODM は、RODM 名を指定します。マルチシステム・マネージャーが初期設定される場合、RODM 名を指定する必要はありません。 DELVIEWS は RODM に関するマルチシステム・マネージャー共通グローバル変数から RODM 名を取り出すからです。

ビューの削除の例

このセクションでは、ビューを削除するための DELVIEWS の使用例を取り上げます。

MY_LAN_VIEW という名のネットワーク・ビューを削除するには、次のようにします。

```
DELVIEWS MY_LAN_VIEW
```

接頭部 RTP_ で始まるネットワーク・ビューのグループを削除するには、次のようにします。

```
DELVIEWS RTP_*
```

MY_PEER_VIEW という名の構成対等機能ビューを削除するには、次のようにします。

```
DELVIEWS MY_PEER_VIEW TYPE=PEER
```

小文字を含む名前のビューを削除するには、 NetView NETVASIS コマンドに接頭部 DELVIEWS REXX clist を付けます。

```
NETVASIS DELVIEWS Raleigh_Site_LAN
```

詳細については、「*IBM Tivoli NetView for z/OS データ・モデル・リファレンス*」を参照してください。

付録 B. ビュー・レイアウト機能

ビュー・レイアウト機能は、ビューをレイアウトするときに NetView 管理コンソールが使用するサービスを提供します。ビュー・レイアウト機能の入力は、RODM に保管されているビュー情報、およびビュー・プリプロセッサーによって作成されてホストからダウンロードされたビューによって構成されます。

この付録では、各レイアウト・タイプに関する以下の情報を提供します。
- 図形の例
- 長所および短所
- 使用される GMFHS フィールドによって各レイアウト・タイプが受ける影響の説明

ビュー・レイアウトの例

ネットワークのさまざまな側面を表現するために、ネットワーク・モデルのビューの中には、他のビューよりも視覚的に解釈しやすくなっているものがあります。したがって、ビュー・レイアウト機能を使用して多数のビューのタイプを作成することができます。
- リンクごとのクラスター化を表す放射状レイアウト (756 ページの図 165 を参照)
- クラスター ID ごとのユーザー定義のクラスターを表す放射状レイアウト (756 ページの図 165 を参照)
- ブロードバンド・ネットワークを表す放射状レイアウト (756 ページの図 165 を参照)
- トークンリング・ネットワークを表す放射状レイアウト (756 ページの図 166 を参照)
- ローカル・エリア・ネットワークを表す放射状レイアウト (757 ページの図 167 を参照)
- 中央バスのあるローカル・エリア・ネットワークを表す放射状レイアウト (757 ページの図 168 を参照)
- 単一楕円形レイアウト (758 ページの図 169 を参照)
- 階層レイアウト (758 ページの図 170 を参照)
- 接続ツリー・レイアウト (759 ページの図 171 を参照)
- 例外、構成、およびネットワーク・ビューを表すグリッド・レイアウト (759 ページの図 172 を参照)

各レイアウト・タイプの長所と短所のリストについては、760 ページの表 240 を参照してください。

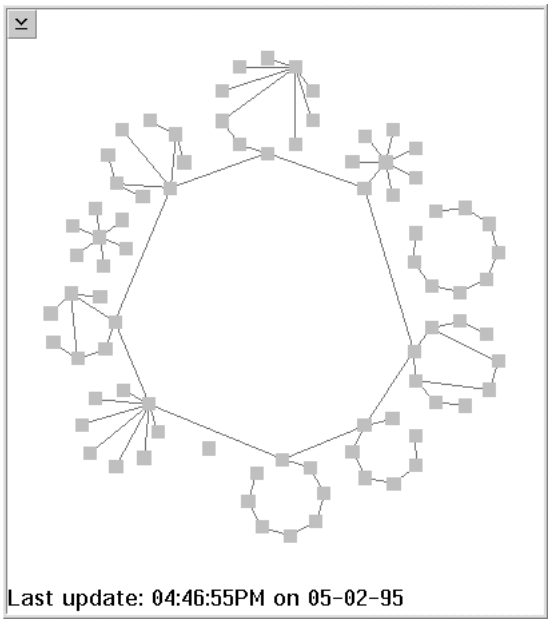


図 165. 放射状レイアウトの例

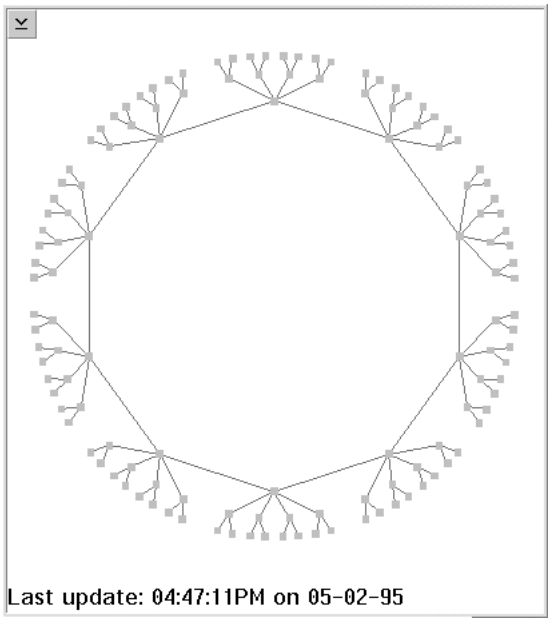


図 166. トークンリング・レイアウトの例

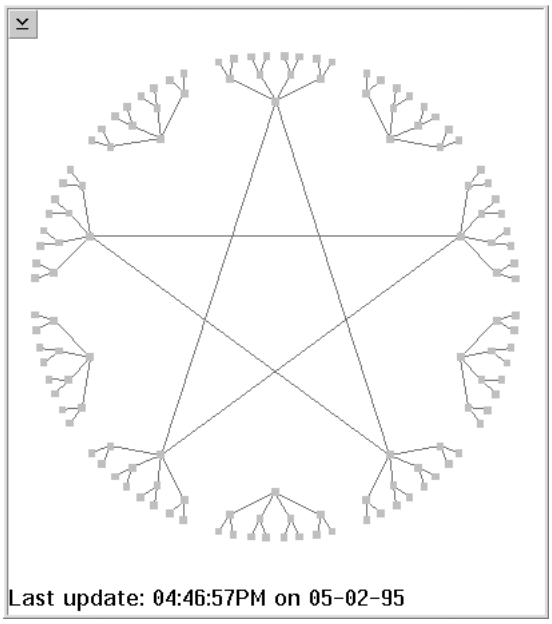


図 167. LAN ネット・レイアウトの例

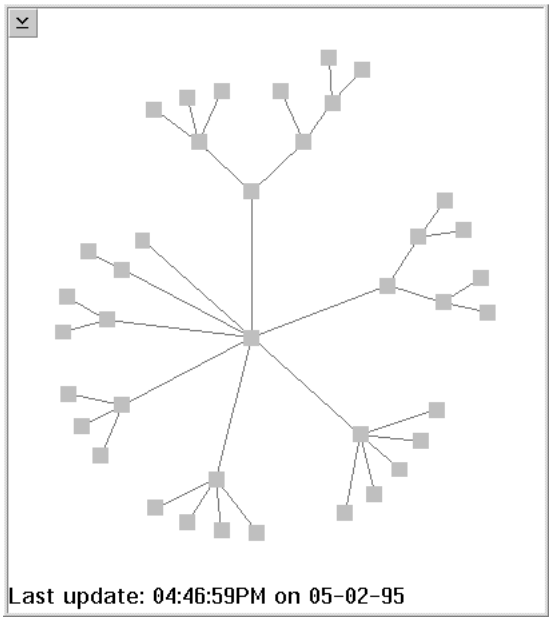


図 168. LAN バス・レイアウトの例

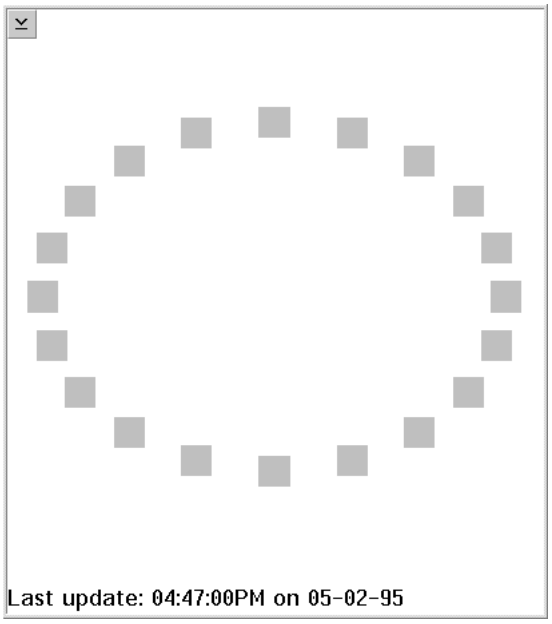


図 169. 楕円形レイアウトの例

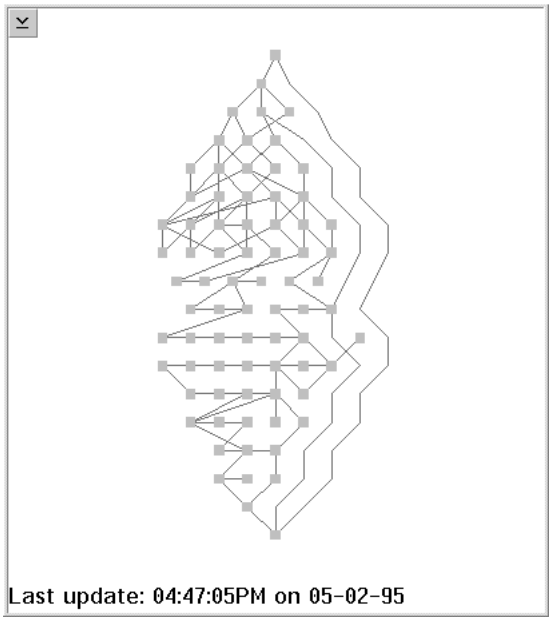


図 170. 階層グラフ・レイアウトの例

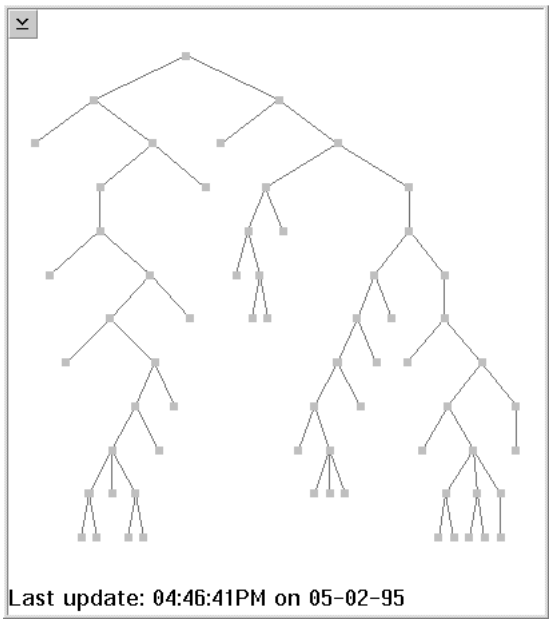


図 171. 接続ツリー・レイアウトの例

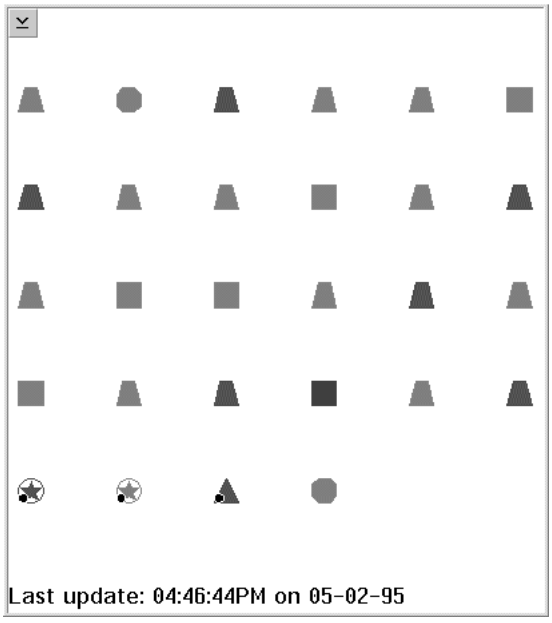


図 172. グリッド・レイアウトの例

ビュー・レイアウト・タイプの選択

表 240 は、各レイアウト・タイプの長所と短所を説明しています。

表 240. ビュー・レイアウト・タイプの長所と短所

ビュー・レイアウト・タイプ	長所	短所
リンク・タイプごとの放射状	ワークステーションの表示スペースを効果的に使用できます。 物理サイトでリソースのグループ化を効果的に表示できます。 接続とは無関係に任意のビューをレイアウトできます。	ユーザーの感覚的イメージがビュー・レイアウトに対応しない可能性があります。 親子の関係が正しく伝わりません。
クラスター ID ごとの放射状	リンク・タイプごとの放射状レイアウトとおなじ長所があります。 ノードのグループ化をユーザーが完全に制御できます。	ビュー内の各ノードにクラスター ID を割り当てる必要があります。
単一の楕円	表示スペースの使用が最適化されます。	1 つのサイトまたはグループ化だけしか表現できません。 リンク交差を少なくするために順序番号を設定する必要があります。
LAN ネットワーク・レイアウト	ブロードバンド LAN を含むビューをレイアウトするのに適しています。	ビュー・レイアウト機能によって定義された LAN ビューの接続性要件をビューが満たしている必要があります。
LAN トークンリング・レイアウト	トークンリング LAN を含むビューをレイアウトするのに適しています。	ビュー・レイアウト機能によって定義されたトークンリング・ビューの接続性要件をビューが満たしている必要があります。
LAN バス・レイアウト	中央バスのある LAN を含むビューをレイアウトするのに適しています。	ビュー・レイアウト機能によって定義された LAN バス・ビューの接続性要件をビューが満たしている必要があります。
接続ツリー・レイアウト	レイアウトが速く行えます。 リソース間の親子関係が表示されます。	ビュー・レイアウト機能によって定義された接続ツリー・ビューの接続要件をビューが満たしている必要があります。

表 240. ビュー・レイアウト・タイプの長所と短所 (続き)

ビュー・レイアウト・タイプ	長所	短所
ノード優先順位ごとの階層グラフ	ネットワーク・リソース間の親子関係が表示されます。 接続とは無関係に任意のビューをレイアウトできます。	ビュー内の各ノードに階層的な優先順位を割り当てる必要があります。
グリッド・レイアウト	レイアウトが速く行えます。 関連するネットワーク・オブジェクトまたは無関係なネットワーク・オブジェクトのリストを表示するのに適しています。	行および列を指定しなければ、ネットワーク・トポロジーが表示されません。 接続が示されません。

ビュー・レイアウト機能によって使用される GMFHS フィールド

次の GMFHS フィールドで、ビュー・レイアウト機能が使用するデータが提供されます。

- BinPackingFlag
- BusNode
- ClusterIDValue
- DefaultRowSpacing
- EllipseAspectRatioHeight
- EllipseAspectRatioWidth
- FirstNode
- HierarchicalPriority
- LayoutOrientation
- LayoutSequence
- LayoutType
- LayoutWidth
- LinkCrossOptionValue
- ResourceLayoutCharacteristics
- RootNode
- SecondNode

ビュー・レイアウト機能によるこれらのフィールドの使用方法については、以降のセクションを参照してください。

レイアウト・タイプの説明

このセクションでは、ビュー・レイアウト・タイプについて説明します。それぞれのビュー・レイアウト・タイプの説明と、各ビュー・レイアウト・タイプで使用されるフィールドの説明が記載されています。

注: RODM の SymbolRadiusValue フィールドで設定した値がビューの外観に影響を与えることはなくなりました。ビューの外観の制御は NetView 管理コンソール

によって行われるようになったため、ユーザーがビューの外観を変更できるようになりました。 NMC については、オンライン・ヘルプを参照してください。

リンク・タイプごとの放射状レイアウト

リンク・タイプごとの放射状レイアウトは、リンク・タイプに基づいてクラスター化された放射状のレイアウトです。 ResourceLayoutCharacteristics ビット 3 がオンになっているリンクによって接続されたノードは、同じクラスター (円) に配置されます。

フィールドの説明

以下のフィールドはビューに関連するもので、リンク・タイプごとの放射状レイアウト・ビュー機能によるビューのレイアウト方法に影響を与えます。

LayoutType
このタイプのビューを指定するには、LayoutType フィールドの値を 1 に設定してください。

BinPackingFlag
BinPackingFlag フィールドを 1 に設定すると、リンク・タイプごとの放射状レイアウト・ビュー機能は、ノードを均等に分布させるために同じレベルおよび重みのサイトを再配置します。

LinkCrossOptionValue
このフィールドはリンク交差の最適化レベルを制御します。この数値を大きくすると、ビュー内のリンク交差の数を減らすためにビュー・レイアウト機能が費やす時間が長くなります。値の範囲は 0 から 6 までです。

以下のフィールドはビュー内の各ノードに関連するもので、リンク・タイプごとの放射状レイアウト・ビュー機能によるビューのレイアウト方法に影響を与えます。

ResourceLayoutCharacteristics
あるノードに関してこのフィールドのビット 2 をオンにしたときに、そのノードがクラスター (円) 内の 1 つのノードに接続されていて他のどのノードにも接続されていない単一ノードである場合、そのノードは接続先のクラスター (円) に組み込まれます。

以下のフィールドはビュー内の各リンクに関連するもので、リンク・タイプごとの放射状レイアウト・ビュー機能によるビューのレイアウト方法に影響を与えます。

ResourceLayoutCharacteristics
ResourceLayoutCharacteristics ビット 3 がオンになっているリンクによって接続されたノードは、同じクラスター (円) に配置されます。このビットは、ユーザーに適した任意の方法で使用できます。例えば、高速リンクであることが示されているリンク・タイプでは、すべてのリンクについてこのビットをオンにすることができます。高速リンクで接続された装置は、同じサイトに設置されていることが多いため、このようにすると、同じサイトにある可能性の高い装置が同じ円に配置されます。

クラスター ID ごとの放射状レイアウト・ビュー

クラスター ID ごとの放射状レイアウト・ビューは、ビュー内のノードの ClusterIDValue フィールドに基づいてクラスター化された放射状レイアウトです。同じクラスター ID のノードは、同じサイトの円で一緒にクラスター化されます。

フィールドの説明

以下のフィールドはビューに関連するもので、クラスター ID ごとの放射状レイアウト・ビュー機能によるビューのレイアウト方法に影響を与えます。

LayoutType
このタイプのビューを指定するには、LayoutType フィールドの値を 2 に設定してください。

BinPackingFlag
BinPackingFlag フィールドを 1 に設定すると、クラスター ID ごとの放射状レイアウト・ビュー機能は、ノードを均等に分布させるために同じレベルおよび重みのサイトを再配置します。

LinkCrossOptionValue
このフィールドはリンク交差の最適化レベルを制御します。この数値を大きくすると、ビュー内のリンク交差の数を減らすためにビュー・レイアウト機能が費やす時間が長くなります。有効な値の範囲は 0 から 6 までです。

以下のフィールドはビュー内の各ノードに関連するもので、クラスター ID ごとの放射状レイアウト・ビュー機能によるビューのレイアウト方法に影響を与えます。

ResourceLayoutCharacteristics
あるノードに関してこのフィールドのビット 2 をオンにしたときに、そのノードがクラスター (円) 内の 1 つのノードに接続されていて他のどのノードにも接続されていない単一ノードである場合、そのノードは接続先のクラスター (円) に組み込まれます。

ClusterIDValue
このフィールドを使用すると、ノードをどのようにグループ化 (クラスター化) するのかを指示することができます。 ClusterIDValue の値が同じになっているノードは、同じ円で一緒にグループ化 (クラスター化) されます。

ローカル・エリア・ネットワーク・レイアウト・ビュー

ローカル・エリア・ネットワーク・レイアウトは、放射状レイアウトをローカル・エリア・ネットワーク・ビューに合わせて調整したものです。

フィールドの説明

以下のフィールドはビューに関連するもので、ローカル・エリア・ネットワーク・レイアウト機能によるビューのレイアウト方法に影響を与えます。

LayoutType
このタイプのビューを指定するには、LayoutType フィールドの値を 3 に設定してください。

BinPackingFlag
BinPackingFlag フィールドを 1 に設定すると、ローカル・エリア・ネットワーク・レイアウト機能は、ノードを均等に分布させるために同じレベルおよび重みのサイトを再配置します。

LinkCrossOptionValue
このフィールドはリンク交差の最適化レベルを制御します。この数値を大きくすると、ビュー内のリンク交差の数を減らすためにビュー・レイアウト機能が費やす時間が長くなります。有効な値の範囲は 0 から 6 までです。

以下のフィールドはビュー内の各ノードに関連するもので、ローカル・エリア・ネットワーク・レイアウト機能によるビューのレイアウト方法に影響を与えます。

LayoutSequence
同じ親の複数の子がサブサイトおよびサブ・サブサイトの円に存在しているビューでは、子の配列は各ノードの LayoutSequence フィールドの値に基づいて行われます。それぞれの子は、円に沿って右回り方向に移動したときに LayoutSequence フィールドの値が昇順になるように配列されます。ノードを配置する順序を制御する必要がない場合には、ビュー内の各ノードの LayoutSequence フィールドの値をデフォルトの 0 に設定してください。

トークンリング・ネットワーク・レイアウト・ビュー・インターフェース

トークンリング・ネットワーク・レイアウトは、放射状レイアウトをトークンリング・ネットワーク・ビューに合わせて調整したものです。

フィールドの説明

以下のフィールドはビューに関連するもので、トークンリング・ネットワーク・レイアウト機能によるビューのレイアウト方法に影響を与えます。

LayoutType
このタイプのビューを指定するには、LayoutType フィールドの値を 4 に設定してください。

FirstNode
メイン・サイト円にあるノードのうちで、その円の最上部 (12 時の位置) に配置されるノードの ID です。

SecondNode
メイン・サイト円にあるノードのうちで、 ID が FirstNode になっているノードの (右回り方向で) 直後に配置されるノードの ID です。

以下のフィールドはビュー内の各ノードに関連するもので、トークンリング・ネットワーク・レイアウト機能によるビューのレイアウト方法に影響を与えます。

LayoutSequence
同じ親の複数の子がサブサイトおよびサブ・サブサイトの円に存在しているビューでは、子の配列は各ノードの LayoutSequence フィールドの値に基づいて行われます。それぞれの子は、円に沿って右回り方向に移動したときに LayoutSequence フィールドの値が昇順になるように配列されます。ノードを配置する順序を制御する必要がない場合には、ビュー内の各ノードの LayoutSequence フィールドの値をデフォルトの 0 に設定してください。

バス・ネットワーク・レイアウト・ビュー・インターフェース

バス・ネットワーク・レイアウトは、放射状レイアウトをバス・ネットワーク・ビューに合わせて調整したものです。

フィールドの説明

以下のフィールドはビューに関連するもので、バス・ネットワーク・レイアウト機能によるビューのレイアウト方法に影響を与えます。

LayoutType
このタイプのビューを指定するには、LayoutType フィールドの値を 5 に設定してください。

BusNode
ビューの中央バス・ノードのオブジェクト ID です。このノードは、ビューのメイン・サイト円にあるすべてのノードの親ノードになります。

以下のフィールドはビュー内の各ノードに関連するもので、バス・ネットワーク・レイアウト機能によるビューのレイアウト方法に影響を与えます。

LayoutSequence
同じ親の複数の子がサブサイトおよびサブ・サブサイトの円に存在しているビューでは、子の配列は各ノードの LayoutSequence フィールドの値に基づいて行われます。それぞれの子は、円に沿って右回り方向に移動したときに LayoutSequence フィールドの値が昇順になるように配列されます。ノードを配置する順序を制御する必要がない場合には、ビュー内の各ノードの LayoutSequence フィールドの値をデフォルトの 0 に設定してください。

階層グラフ・レイアウト・ビュー

階層グラフ・レイアウト機能は、階層の各レベルが、同じ優先順位を指定されたノードによって占められるレイアウトです。

このタイプのレイアウトでは、どのノードも、 1 レベルを超えたレベル差のノードまたはタック点に接続されていないことが必要です。しかし、ユーザーが作成したビューがこの要件を満たしていないこともあります。このような場合、ビュー・レイアウト機能はこの要件を満たすために必要な数のタック点とリンクを追加します。

フィールドの説明

以下のフィールドはビューに関連するもので、階層グラフ・レイアウト機能によるビューのレイアウト方法に影響を与えます。

LayoutType
このタイプのビューを指定するには、LayoutType フィールドの値を 6 に設定してください。

LayoutOrientation
このフィールドを 0 に設定すると、ビュー・レイアウト機能はグラフを上から下に向かってレイアウトします。このフィールドを 1 に設定すると、ビュー・レイアウト機能はグラフを左から右に向かってレイアウトします。

DefaultRowSpacing
この値は、接続ツリーの行と行の間のデフォルトの間隔を表します。このフ

ィールドを 0 に設定するか、あるいは 1 から 50 までの範囲外の値に設定すると、ビューを正方形にするために必要な間隔だけ、行と行の間が離されます。行と行の間隔を明示的に制御したい場合には、このフィールドを 1 から 50 までの範囲の値に設定してください。この値は、シンボルの半径の倍数を表します。例えば、この値が 3 の場合には、行と行の間隔はシンボルの半径の 3 倍になります。

以下のフィールドはビュー内の各ノードに関連するもので、階層グラフ・レイアウト機能によるビューのレイアウト方法に影響を与えます。

HierarchicalPriority

このフィールドは、ノードの階層優先順位を指定するために使用します。ノードは、その優先順位値がグラフの上から下に向かって、あるいは左から右への方向を指定したビューの場合には左から右に向かって昇順になるように、階層グラフのさまざまなレベルに配置されます。同じ階層優先順位のノードはすべて、ビュー内の同じ行に配置されます。各ノードの階層優先順位フィールドは、必要に応じた任意の方法で割り当てることができます。例えば、ノードのオブジェクト・タイプに対応して階層優先順位を設定して、あるタイプのすべてのノードが同じ行に配置されるようにすることができます。

このタイプのレイアウトでは、階層優先順位が相対値として使用されることにご注意ください。例えば、ビュー内のすべてのノードに階層優先順位値として 1、2、または 12 が割り当てられている場合、行 1 と行 2 の間の間隔は、行 2 と行 12 の間の間隔と同じになります。また、0 がこのフィールドの値としては無効であることにもご注意ください。

楕円形レイアウト・ビュー

楕円形レイアウト機能は、ビューを単一の楕円形としてレイアウトします。

フィールドの説明

以下のフィールドはビューに関連するもので、楕円形レイアウト機能によるビューのレイアウト方法に影響を与えます。

LayoutType

このタイプのビューを指定するには、LayoutType フィールドの値を 7 に設定してください。

EllipseAspectRatioHeight

EllipseAspectRatioHeight および EllipseAspectRatioWidth は、楕円形の縦横比として使用されます。 EllipseAspectRatioHeight に 1 を指定し、EllipseAspectRatioWidth にも 1 を指定すると、円になります。EllipseAspectRatioWidth に 640 を指定し、EllipseAspectRatioHeight に 480 を指定すると、 640 X 480 モードの標準 VGA モニターの高さと幅の比率に適した楕円形になります。

EllipseAspectRatioWidth

EllipseAspectRatioHeight の定義を参照してください。

以下のフィールドはビュー内の各ノードに関連するもので、楕円形レイアウト機能によるビューのレイアウト方法に影響を与えます。

LayoutSequence
ノードは、各ノードの LayoutSequence 値が昇順になるように、楕円形の最上部から順番に右回りで配置されます。 ノードを配置する順序を制御する必要がない場合には、ビュー内の各ノードの LayoutSequence フィールドの値をデフォルトの 0 に設定してください。

接続ツリー・レイアウト・ビュー

接続ツリー・レイアウト機能は、ビューを単純な接続ツリーとしてレイアウトします。このビューは、1 つまたは複数の真のツリーから構成されていなければなりません。ルート・ノードを除き、各ノードは必ず 1 つの親に接続されていなければなりません。ノードは複数の子ノードに接続することができます。子ノードは接続することができません。

フィールドの説明

以下のフィールドはビューに関連するもので、接続ツリー・レイアウト機能によるビューのレイアウト方法に影響を与えます。

LayoutType
このタイプのビューを指定するには、LayoutType フィールドの値を 8 に設定してください。

LayoutOrientation
このフィールドを 0 に設定すると、ビュー・レイアウト機能はグラフを上から下に向かってレイアウトします。このフィールドを 1 に設定すると、ビュー・レイアウト機能はグラフを左から右に向かってレイアウトします。

DefaultRowSpacing
この値は、接続ツリーの行と行の間のデフォルトの間隔を表します。このフィールドを 0 に設定するか、あるいは 1 から 50 までの範囲外の値に設定すると、ビューを正方形にするために必要な間隔だけ、行と行の間が離されます。行と行の間隔を明示的に制御したい場合には、このフィールドを 1 から 50 までの範囲の値に設定してください。この値は、シンボルの半径の倍数を表します。例えば、この値が 3 の場合には、行と行の間隔はシンボルの半径の 3 倍になります。

以下のフィールドはビュー内の各ノードに関連するもので、接続ツリー・レイアウト機能によるビューのレイアウト方法に影響を与えます。

RootNode
このフィールドを 0x80 に設定すると、そのノードがルート・ノードであることがビュー・レイアウト機能に対して示されます。ルート・ノード以外のすべてのノードには、祖先としてルート・ノードが必要です。ルート・ノード以外のノードに親元としてルート・ノードが存在していない場合、そのノードはビューの最下部で長方形の格子状にレイアウトされます。

LayoutSequence
共通の親ノードに接続されたノードは、 LayoutSequence フィールドの値がビューの方向付けに応じて左から右または上から下に向かって昇順になるように配置されます。ノードの配置順序を制御する必要がない場合には、ビュー内の各ノードの LayoutSequence フィールドをデフォルトの 0 に設定することができます。

グリッド・レイアウト

グリッド・レイアウト機能は、ビュー・オブジェクトを行と列の格子に整列させます。オブジェクトの位置は、行番号、列番号、またはその両方を使用して指定することができます。座標を指定しなかった場合、ノードはランダムに格子に配置されます。

グリッド・レイアウトは、以下のタイプのビューで使用することができます。
- 例外
- ネットワーク
- 構成

例外ビューの場合に使用できるレイアウトはグリッド・レイアウトだけであり、また行と列のパラメーターを指定することはできません。

ネットワークまたは構成対等ビューの場合、そのビューの中のすべてのオブジェクトについて行と列の値を指定することをお勧めします。行と列の値によって、ビュー内でのオブジェクトの配置が決まります。

フィールドの説明

以下のフィールドはビューに関連するもので、グリッド・レイアウト機能によるビューのレイアウト方法に影響を与えます。

LayoutType
このタイプのビューを指定するには、LayoutType フィールドの値を 9 に設定してください。

LayoutOrientation
このフィールドを 0 に設定すると、ビュー・レイアウト機能は格子を上から下に向かってレイアウトします。つまり、左上隅が行 1 列 1 で、上から下に移動するにつれて行番号が大きくなり、左から右に移動するにつれて列番号が大きくなります。このフィールドを 1 に設定すると、ビュー・レイアウト機能は格子を左から右に向かってレイアウトします。つまり、左下隅が行 1 列 1 で、左から右に移動するにつれて行番号が大きくなり、下から上に移動するにつれて列番号が大きくなります。

LayoutWidth
ノードを列に割り当てるときにビュー・レイアウト機能によって使用される最大列番号です。ビュー・レイアウト機能は、列番号がゼロになっていたノードについてだけ列を割り当てます。 LayoutWidth フィールドがゼロの場合、ビュー・レイアウト機能はビューが正方形になるような値に LayoutWidth を設定します。

以下のフィールドはビュー内の各ノードに関連するもので、グリッド・レイアウト機能によるビューのレイアウト方法に影響を与えます。

HierarchicalPriority
このフィールドは、ノードに絶対行番号を割り当てるために使用されます。「絶対」ということは、異なる 3 つのノードに行番号 1、2、および 12 を割り当てる場合、ノード 1 とノード 2 が配置される行の間隔は、ノード 2 とノード 3 が配置される行の間隔の 10 分の 1 になることを意味しています。ノードを配置する行を制御する必要がない場合には、このフィールドを

0 に設定してください。そうすると、ビュー・レイアウト機能は、次に利用可能な未割り当て行を割り当てます。デフォルトは 0 です。

LayoutSequence
このフィールドは、ノードに絶対列番号を割り当てるために使用されます。この場合の「絶対」の意味は、HierarchicalPriority フィールドの場合と同じです。ノードを配置する列を制御する必要がない場合には、このフィールドを 0 に設定してください。ビュー・レイアウト機能が、次に利用可能な列を割り当てます。 これはデフォルトです。 LayoutWidth フィールドの値は、ノードが割り当てられる最大桁番号を表します。このフィールドが影響を与えるのは、ビュー・レイアウト機能によって割り当てられた値だけであるため、 LayoutWidth よりも大きな列番号を明示的に割り当てることは有効であることにご注意ください。

以下のフィールドはビュー内の各リンクに関連するもので、グリッド・レイアウト機能によるビューのレイアウト方法に影響を与えます。

HierarchicalPriority
このフィールドは、リンクに絶対行番号を割り当てるために使用されます。リンクは、ビュー・レイアウト機能によってエンドポイント・ノード間に設定されます。これらのエンドポイント・ノードが行に割り当てられていない場合、つまり、これらのノードの HierarchicalPriority フィールドが 0 に設定されている場合には、リンクの行の値がエンドポイント・ノードによって継承されます。リンクを配置する行を制御したくない場合は、このフィールドを 0 に設定すると、ビュー・レイアウト機能がこれを次に使用可能な未割り当て行に割り当てます。デフォルトは 0 です。

LayoutSequence
このフィールドは、リンクに絶対列番号を割り当てるために使用されます。リンクは、ビュー・レイアウト機能によってエンドポイント・ノード間に設定されます。これらのエンドポイント・ノードが列に割り当てられていない場合、つまり、これらのノードの LayoutSequence フィールドが 0 に設定されている場合には、リンクの列の値がエンドポイント・ノードによって継承されます。ノードを配置する列を制御したくない場合は、このフィールドを 0 に設定すると、ビュー・レイアウト機能がこれを次に使用可能な列に割り当てます。デフォルトは 0 です。

グリッド・レイアウトの注意事項

エンドポイントを指定しないでリンクを定義した場合、そのリンクに関してヌルのエンドポイントが作成され、ビュー内にそのリンクを配置できるようになります。グリッド・レイアウトの場合、リンクのエンドポイントとしてヌル・ノードが作成されると、それらのノードはそのリンクの行および列フィールドを継承します。そのリンクについてこれらのフィールドが指定されていない場合には、リンクとそのヌル・ノードはビュー内のランダムな位置に設定されます。

770 ページの表 241 は、さまざまに定義されたリンクの例と、各定義の結果を示しています。

表 241. リンク定義とその結果

行および列レイアウト・パラメーターを指定してリンクが定義されている。このリンクに関するエンドポイントは定義されていない。	リンクは、そのリンクにより指定された座標にある 2 つのヌル・ノード間に設定されます。この場合、リンクのレイアウト・パラメーターが両方のノードのレイアウト・パラメーターに転送されます。
行および列レイアウト・パラメーターを指定しないでリンクが定義されている。このリンクに関するエンドポイントは定義されていない。	リンクは、ランダム位置にある 2 つのヌル・ノード間に設定されます。ノードの位置を制御したい場合には、リンクで座標を指定してください。
行および列レイアウト・パラメーターを指定してリンクが定義されている。行および列レイアウト・パラメーターを指定して 1 つのエンドポイントだけが定義されている。	定義されたエンドポイントが、指定された座標に設定されます。リンクの座標を使用してヌル・ノードが作成されます。定義されたエンドポイントと新しく作成されたヌル・ノードの間にリンクが設定されます。
行および列レイアウト・パラメーターを指定してリンクが定義されている。 1 つのエンドポイントだけが定義されているが、行および列レイアウト・パラメーターは指定されていない。	リンクの座標を使用してヌル・ノードが作成されます。定義されたエンドポイントがランダム位置に設定され、その定義されたエンドポイントと新しく作成されたヌル・ノードの間にリンクが設定されます。
行および列レイアウト・パラメーターを指定してリンクが定義されている。 2 つのエンドポイントが定義され、その両方で行および列レイアウト・パラメーターが指定されている。	両方のエンドポイントが指定された座標位置に設定されます。 2 つのエンドポイント間でリンクが設定されます。リンクの行および列レイアウト・パラメーターは使用されません。

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あり、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。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-8711
東京都港区六本木 3-2-12
IBM World Trade Asia Corporation
Intellectual Property Law & Licensing

以下の保証は、国または地域の法律に沿わない場合は、適用されません。

IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。

国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様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 Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758 U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

プログラミング・インターフェース

本書には、プログラムを作成するユーザーが Tivoli NetView for z/OS のサービスを使用するためのプログラミング・インターフェースが記述されています。

商標

IBM、 IBM ロゴ、 Advanced Peer-to-Peer Networking、 AIX、 BookManager、 Candle、 Language Environment、 MVS、 NetView、 OMEGAMON、 OS/2、 OS/390、 RACF、 REXX、 RISC System/6000、 RS/6000、 SystemView、 Tivoli、 Tivoli Enterprise、 TME、 VSE/ESA、 VTAM、 z/Architecture、 z/OS、 および z/VM は、 International Business Machines Corporation の米国およびその他の国における商標です。

Linux は、Linus Torvalds の米国およびその他の国における商標です。

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。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

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]

[カ行]

［サ行］

[タ行]

［ナ行］

［ハ行］

［マ行］

[ヤ行]

[ラ行]

[ワ行]

[数字]

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

Z

［特殊文字］

ファイル番号: S370/4300/30XX-50
プログラム番号: 5697-ENV

Printed in Japan

SC88-9313-02

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12