IBM Tivoli NetView for z/OS
バージョン 6 リリース 2

カスタマイズ・ガイド

SA88-4388-01
(英文原典：SC27-2849-02)

IBM Tivoli NetView for z/OS
バージョン 6 リリース 2

カスタマイズ・ガイド

SA88-4388-01
(英文原典：SC27-2849-02)

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、 223 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-NV6) の バージョン 6 リリース 2 および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

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

原典： SC27-2849-02
IBM Tivoli NetView for z/OS
Version 6 Release 2
Customization Guide

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

第1刷 2014.2

目次

図

本書について

IBM® Tivoli® NetView® for z/OS® 製品の高度な機能により、マルチプラットフォームおよびマルチベンダーの複合ネットワークとシステムを一元的に管理して、高レベルの可用性を維持することができます。 この資料 (*IBM Tivoli NetView for z/OS カスタマイズ・ガイド*) では、カスタマイズ可能な NetView プログラムの各部分について説明し、関連情報の資料を示しています。

対象読者

この資料の対象読者は、NetView プログラムをカスタマイズするシステム・プログラマーです。

資料

このセクションには、IBM Tivoli NetView for z/OS ライブラリーの資料や関連文書がリストされています。また、Tivoli オンライン資料へのアクセス方法と、Tivoli の資料の注文方法についても説明します。

IBM Tivoli NetView for z/OS ライブラリー

IBM Tivoli NetView for z/OS ライブラリーでは、以下の資料が入手可能です。

- 「 *アドミニストレーション・リファレンス* 」(SA88-4383) では、システム管理で必要とされる NetView プログラム定義ステートメントについて説明しています。
- 「 *アプリケーション・プログラマーズ・ガイド* 」(SA88-4384) では、NetView プログラム間インターフェース (PPI) と、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) の使用方法を説明しています。
- 「*自動操作ガイド*」(SA88-4387) では、自動化された操作を使用してシステムとネットワークの効率性およびオペレーターの生産性を改善する方法を説明しています。
- 「 *Command Reference Volume 1 (A-N)* 」(SC27-2847) および「 *Command Reference Volume 2 (O-Z)* 」(SC27-2848) では、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可能な NetView コマンドについて説明しています。
- 「 *カスタマイズ・ガイド* 」(SA88-4388) では、NetView 製品のカスタマイズ方法を説明し、関連情報の資料を掲載しています。
- 「 *Data Model Reference* 」(SC27-2850) では、Graphic Monitor Facility ホスト・サブシステム (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・モデルに関する情報を提供しています。
- 「 *インストール:追加コンポーネントの構成* 」(GA88-4389) では、基本機能以外の NetView 機能の構成方法を説明しています。

- 「 *インストール:グラフィカル・コンポーネントの構成* 」(GA88-4390) では、NetView グラフィック・コンポーネントのインストールおよび構成方法を説明しています。
- 「 *Installation: Configuring the GDPS Active/Active Continuous Availability Solution* 」(SC14-7477) には、GDPS アクティブ／アクティブ継続的可用性ソリューションと一緒に使用される NetView 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: NetView Enterprise Management Agent の構成* 」(GA88-4401) では、NetView for z/OS Enterprise Management Agent のインストールおよび構成方法を説明しています。
- 「*インストール:概説*」(GI88-4261) では、基本 NetView プログラムをインストールおよび構成する方法について記述しています。
- 「 *インストール:マイグレーション・ガイド* 」(GA88-4391) 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。
- 「*IP 管理*」(SA88-4386) では、NetView 製品を使用して IP ネットワークを管理する方法を説明しています。
- 「 *Messages and Codes Volume 1 (AAU-DSI)* 」(GC27-2856) および「 *Messages and Codes Volume 2 (DUI-IHS)* 」(GC27-2857) では、NetView 製品のメッセージ、NetView 異常終了コード、NetView メッセージに含まれるセンス・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。
- 「 *プログラミング:アセンブラー* 」(SA88-4392) では、アセンブラー言語を使用して NetView 製品の出口ルーチン、コマンド・プロセッサー、およびサブタスクを作成する方法について説明しています。
- 「 *プログラミング:パイプ* 」(SA88-4393) では、NetView パイプラインを使用して NetView のインストールをカスタマイズする方法について説明しています。
- 「 *プログラミング: PL/I および C* 」(SA88-4394) では、PL/I または C を使用して NetView 製品のコマンド・プロセッサーとインストール・システム出口ルーチンを作成する方法について説明しています。
- 「 *プログラミング: REXX および NetView コマンド・リスト言語* 」(SA88-4395) では、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 製品のコマンド・リストを作成する方法について説明しています。
- 「 *Resource Object Data Manager and GMFHS Programmer's Guide* 」(SC27-2862) では、非 SNA ネットワークを RODM に定義する方法や、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 RODM を使用する方法など、NetView リソース・オブジェクト・データ・マネージャー (RODM) について説明しています。
- 「 *セキュリティー・リファレンス* 」(SA88-4397) では、NetView 環境で許可検査を実装する方法について説明しています。
- 「 *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* 」(SA88-4398) では、サブエリア、拡張対等通信ネットワーキング (Advanced Peer-to-Peer Networking)、および TN3270 リソースの管理に使用可能な、NetView SNA トポロジー・マネージャーの計画および実装について説明しています。
- 「 *Troubleshooting Guide* 」(GC27-2865) には、NetView プロダクトで発生する問題の文書化、診断、および解決に関する情報が提供されています。

- 「 *チューニング・ガイド* 」(SA88-4399) では、NetView 製品とネットワーク環境の特定のパフォーマンス目標を達成するのに役立つチューニング情報を提供しています。
- 「 *Automated Operations Network ユーザーズ・ガイド* 」(SA88-4385) では、イベント・ドリブン・ネットワーク自動化によってシステムとネットワークの効率性を向上させる NetView Automated Operations Network (AON) コンポーネントの使用方法について説明しています。また、AON コンポーネントの自動化操作機能の調整方法と拡張方法についても説明しています。
- 「 *ユーザーズ・ガイド: NetView* 」(SA88-4400) では、NetView 製品を使用して複合的なマルチベンダーのネットワークおよびシステムを一元的に管理する方法を説明しています。
- 「 *NetView Enterprise Management Agent ユーザーズ・ガイド* 」(SA88-4402) には、NetView Enterprise Management Agent を使用する方法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* 」(SA88-4396) では、NetView 製品の NetView 管理コンソール・インターフェースについての情報を提供しています。
- 「*Licensed Program Specifications*」(GC31-8848) では、NetView 製品のライセンス情報を提供しています。
- 「*Program Directory for IBM Tivoli NetView for z/OS US English*」(GI11-9444) には、IBM Tivoli NetView for z/OS 製品のインストールに関する資料と手順についての情報を記載しています。
- 「*Program Directory for IBM Tivoli NetView for z/OS Japanese*」(GI11-9445) には、IBM Tivoli NetView for z/OS 製品のインストールに関する資料と手順についての情報を記載しています。
- 「*Program Directory for IBM Tivoli NetView for z/OS Enterprise Management Agent*」(GI11-9446) には、IBM Tivoli NetView for z/OS Enterprise Management Agent のインストールに関する資料と手順についての情報を記載しています。
- 「*IBM Tivoli NetView for z/OS V6R2 Online Library*」(LCD7-4913) には、NetView for z/OS ライブラリーにある資料が含まれています。資料は、PDF および HTML 形式で用意されています。

関連資料

追加の製品情報は、NetView for z/OS Web サイト (http://www.ibm.com/software/tivoli/products/netview-zos/) 上で検索できます。

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。

オンライン用語集へのアクセス

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 http://www.ibm.com/software/globalization/terminology/ でアクセスできます。

NetView for z/OS の用語と定義については、IBM Terminology Web サイトを参照してください。このライブラリーでは、以下の用語が使用されています。

NetView
以下の製品の場合:
- Tivoli NetView for z/OS バージョン 6 リリース 2
- Tivoli NetView for z/OS バージョン 6 リリース 1
- Tivoli NetView for z/OS バージョン 5 リリース 4
- Tivoli NetView for z/OS バージョン 5 リリース 3
- Tivoli NetView for OS/390® バージョン 1 リリース 4
- サポートされなくなった NetView リリース

CNMCMD
CNMCMD メンバーと、%INCLUDE ステートメントを使用してその中に組み込まれるメンバーの場合

CNMSTYLE
CNMSTYLE メンバーと、%INCLUDE ステートメントを使用してその中に組み込まれるメンバーの場合

DSIOPF
DSIOPF メンバーと、%INCLUDE ステートメントを使用してその中に組み込まれるメンバーの場合

PARMLIB
SYS1.PARMLIB および連結シーケンスの他のデータ・セットの場合

MVS™ z/OS オペレーティング・システムの場合

MVS エレメント
z/OS オペレーティング・システムの基本制御プログラム (BCP) エレメントに関する用語

VTAM®
Communications Server - SNA Services に関する用語

IBM Tivoli Network Manager
以下のいずれかのプロダクトに関する用語
- IBM Tivoli Network Manager
- IBM Tivoli OMNIbus and Network Manager

IBM Tivoli Netcool/OMNIbus
以下のいずれかのプロダクトに関する用語
- IBM Tivoli Netcool/OMNIbus
- IBM Tivoli OMNIbus and Network Manager

特に断りのない限り、トピックでプログラムに言及する場合は、そのプログラムの最新のバージョンとリリースを指します。トピックで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指します。

トピックでパーソナル・コンピューターまたはワークステーションの使用に言及する場合は、プログラマブル・ワークステーションであればいずれも使用できます。

NetView for z/OS オンライン・ヘルプの使用

インストール済み環境と構成に応じて、以下の種類の NetView for z/OS メインフレーム・オンライン・ヘルプが用意されています。

- 一般ヘルプおよびコンポーネント情報
- コマンド・ヘルプ
- メッセージ・ヘルプ
- センス・コード情報
- 推奨処置

マニュアルへのオンライン・アクセス

資料 DVD「*IBM Tivoli NetView for z/OS V6R2 Online Library*」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 および HTML 形式で用意されています。ドキュメンテーションの利用方法については、DVD 上の README ファイルを参照してください。

IBM では、この製品の他すべての Tivoli 製品に関する資料が使用可能になった時点および更新された時点で、Tivoli Documentation Central の Web サイト (https://www.ibm.com/developerworks/mydeveloperworks/wikis/home/wiki/Tivoli%20Documentation%20Central) に掲載しています。

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

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「マニュアル・出版物情報」をご覧ください。(URL は、変更になる場合があります)

アクセシビリティ

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製品を快適に使用できるようサポートします。製品では標準のショートカット・キーとアクセラレーター・キーが使用されており、これらはオペレーティング・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。

詳しくは、「*ユーザーズ・ガイド: NetView*」の付録『アクセシビリティー』を参照してください。

Service Management Connect

サービス・マネジメント専門家と情報交換、学習、および共有を行います。これらの専門家は製品サポート技術のエキスパートであり、さまざまな見通しや専門知識を提供します。

Service Management Connect (http://www.ibm.com/developerworks/servicemanagement/z/) にアクセスします。 Service Management Connect は以下の方法で利用できます。

- Tivoli 製品の他のユーザーと IBM 開発者の間の公開された進行中の取り組みである透過的開発に参加する。初期設計、スプリント・デモ、製品ロードマップ、プレリリース・コードにアクセスすることができます。
- 専門家と 1 対 1 でつながり、Tivoli および NetView コミュニティーに関して共同作業およびネットワーキングを行う。
- ブログを読んで、他の人の専門知識や経験を参考にする。
- WiKi やフォーラムを使用して、より広範囲にわたるユーザー・ コミュニティーと共同作業を行う。

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。

Tivoli ユーザー・グループ

Tivoli ユーザー・グループは、独立した、ユーザーにより運営されたメンバーシップ組織であり、Tivoli ユーザーに対して、Tivoli Software ソリューションをインプリメントする際にユーザーを支援する情報を提供します。このユーザー・グループを介して、メンバーは情報を共有することができ、また、他の Tivoli ユーザーの知識や経験を習得することができます。

ダウンロード

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 アプリケーションは、以下の NetView for z/OS サポート Web サイトからダウンロードできます。

http://www.ibm.com/software/sysmgmt/products/support/IBMTivoliNetViewforzOS.html

「サポート・ショートカット」ペインで、「**Tivoli NetView for z/OS**」を展開し、「**Fixes (downloads)**」をクリックして、ダウンロードを検索または選択できるページに移動します。

これらのアプリケーションは、以下のタスクで使用できます。

- カスタマイズ・パラメーターと初期化ステートメントを前のリリースから CNMSTUSR メンバーに、およびコマンド定義を前のリリースから CNMCMDU メンバーにマイグレーションする
- 自動化テーブルの統計情報を取得し、その統計を自動化テーブルのリストとマージする
- ジョブ入力サブシステム (JES) ジョブの状況を表示するか、指定された JES ジョブを取り消す

- プログラム間インターフェース (PPI) を使用して NetView プログラムにアラートを送信する
- PPI を使用した、MVS コマンドの送信および受信
- タイム・シェアリング・オプション (TSO) コマンドを送信し、応答を受信する

サポート情報

IBM ソフトウェアに問題が発生した場合、迅速に解決する必要があります。IBM は、必要なサポートをユーザーに提供するために以下の方法を用意しています。

オンライン

Tivoli Software Support サイト (http://www.ibm.com/software/sysmgmt/products/support/index.html?ibmprd=tivman) にアクセスします。IBM Software Support サイト (http://www.ibm.com/software/support/probsub.html) にアクセスします。

IBM Support Assistant

IBM Support Assistant 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。Support Assistant 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。Support Assistant 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。

トラブルシューティング情報

NetView for z/OS 製品の問題解決について詳しくは、「*IBM Tivoli NetView for z/OS Troubleshooting Guide*」を参照してください。NetView for z/OS 製品の追加サポートは、Yahoo の NetView ユーザー・グループ (http://groups.yahoo.com/group/NetView/) で得られます。このサポートの対象は NetView for z/OS ユーザーに限定されており、登録する必要があります。このフォーラムは、質問に答え、ガイダンスを与える NetView 開発者がモニターしています。コードに関する問題が見つかると、解決策を得るため正式な問題管理レコード (PMR) を開くよう求められます。

本書で使用される規則

このセクションでは、本書で使用される規則について説明します。

書体の規則

本書では、以下のような書体の規則を使用しています。

太字

- 太字にしないと、周囲のテキストと見分けがつけ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、複数列のリスト、コンテナー、メニュー選択、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)

- 本文中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* と呼ばれます)
- 語および文字の強調 (語そのものを取り上げる場合の例: 「制限節を挿入するには、単語 *that* を使用します。」、文字そのものを取り上げる場合の例: 「LUN アドレスは文字 *L* で始まる必要があります。」)
- テキスト中の新規用語 (定義リスト内を除く): ビュー 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。
- 指定する必要がある変数および値: ... 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けにくいファイル名、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のコンポーネントの場合、この資料では、環境変数の指定とディレクトリーの表記において UNIX の規則に従います。

Windows コマンド行を使用する場合、環境変数では **$**変数 を **%**変数**%** に置き換え、ディレクトリーのパスではスラッシュ (**/**) をそれぞれ円記号 (**¥**) に置き換えます。環境変数の名前は、Windows 環境と UNIX 環境とで常に同じとは限りません。例えば、Windows 環境の %TEMP% と UNIX 環境で同等なのは、$TMPDIR 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

構文図には、以下の構文エレメントが示されます。水平線 (メインパス) に従い、左から右、上から下に向かって構文図を見てください。

- 『シンボル』
- xvii ページの『パラメーター』
- xvii ページの『句読点と括弧』
- xviii ページの『省略形』

構文の例については、 xviii ページの『構文例』を参照してください。

シンボル

構文図では、以下のシンボルを使用しています。

►► コマンド構文の先頭をマークします。

► コマンド構文が続くことを示します。

| コマンド構文の断片または一部の開始および終わりをマークします。

►◄ コマンド構文の終わりをマークします。

パラメーター

構文図では、以下のタイプのパラメーターを使用しています。

必須 必須パラメーターはメインパス上に表示します。

オプション
オプション・パラメーターはメインパスの下に表示します。

デフォルト
デフォルト・パラメーターはメインパスの上に表示します。パラメーターの説明では、デフォルト・パラメーターに下線が付けられています。

構文図では、強調表示、大括弧、または中括弧を使用していません。構文図において、主構文線に対する要素の相対位置は、要素が必須なのか、オプションなのか、またはデフォルト値なのかを示します。

コマンドを発行するときには、コンマなど別の区切り文字が構文内で指定されていない限り、パラメーター間にスペースが必要です。

パラメーターは、キーワードまたは変数に分類されます。キーワードは大文字で表記されます。ユーザーが指定する名前または値を表す変数は小文字で示され、斜体かまたは (NetViewヘルプでは) 異なる色で表示されます。

以下の例では、USER コマンドがキーワード、*user_id* パラメーターが必須の変数、そして *password* パラメーターがオプションの変数です。

句読点と括弧

コロン、セミコロン、コンマ、負符号 (-)、および一重と二重の両方の引用符など、構文図で示されているすべての句読点を含める必要があります。

オペランドに複数の値がある場合、一般にそれらの値は、括弧で囲んでコンマで区切ります。単一の値の場合は一般に、括弧を省略できます。詳しくは xix ページの『複数のオペランドまたは値』を参照してください。

キーワードと変数を区切るためにコマンドで定位置コンマが必要な場合、定位置コンマは、キーワードまたは変数の前に置きます。

コマンドの例を示す場合は、定位置オペランドが存在しないことを示すためにもコンマを使用します。例えば、2 番目のコンマはオプションのオペランドが使用されていないことを示します。

```
COMMAND_NAME opt_variable_1,,opt_variable_3
```

末尾の定位置コンマを指定する必要はありません。末尾の定位置コンマと非定位置コンマは、無視されるか、またはコマンドが拒否されます。後ろにコンマがあるかどうかの各コマンド状態の制限により、コマンドは拒否されます。

省略形

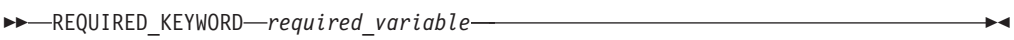
コマンドおよびキーワードの省略形は、各コマンドの説明の後の同義語テーブルにリストしています。

構文例

次の例では、構文エレメントのさまざまな使用法を示します。

- 『必須の構文要素』
- 『オプションの構文要素』
- 『デフォルトのキーワードと値』
- xix ページの『複数のオペランドまたは値』
- xix ページの『1 行より長い構文』
- xix ページの『構文断片』

必須の構文要素:
必須のキーワードと変数は、メイン構文線上に表示します。必須のキーワードと変数をコーディングする必要があります。



必要な選択項目 (2 つ以上の項目) は、メインパスの上側にある垂直スタックに表示されます。項目は英数字順に表示されています。

オプションの構文要素:
オプションのキーワードと変数は、メイン構文線の下に表示します。オプションのキーワードと変数は、コーディングしないことを選択できます。

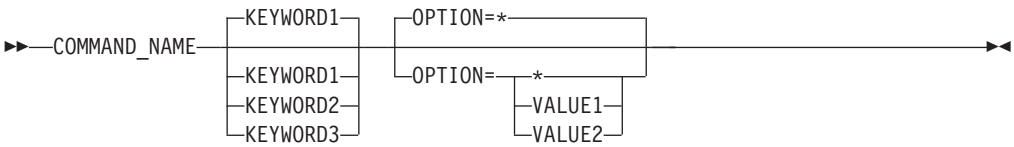
必要な選択項目 (2 つ以上の項目) は、メインパスの下側にある垂直スタックに表示されます。項目は英数字順に表示されています。

デフォルトのキーワードと値:
デフォルトのキーワードと値は、次のいずれかの方法でメイン構文線の上に示します。

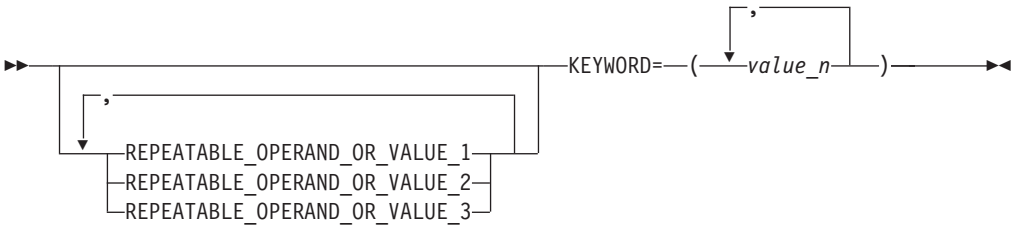
- デフォルト・キーワードは、メイン構文線の上にのみ示します。このキーワードを指定するか、デフォルトにすることができます。以下の構文例では、デフォル

ト・キーワード KEYWORD1 をメイン構文線の上に、オプションのキーワードの残りをメイン構文線の下に示しています。

- オペランドのデフォルト値がある場合は、そのオペランドをメイン構文線の上下両方に示します。メイン構文線の下の値は、オペランドを指定する場合には、デフォルト値または示されている別の値も指定しなければならないことを示します。オペランドを指定しない場合は、メイン構文線の上にあるデフォルト値が使用されます。以下の構文例では、メイン構文線の上下にオペランド OPTION=* のデフォルト値が示されています。



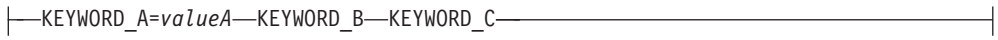
複数のオペランドまたは値:
一群のオペランドまたは値の上にある左に戻る矢印は、複数選択が可能か、または 1 つの値を繰り返すことができることを示しています。



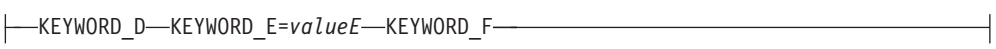
1 行より長い構文:
図が 1 行より長い場合は、続きのある各行が 1 つの矢印で終わり、次の行の先頭が 1 つの矢印で始まります。

構文断片:
構文図によっては、構文の長い、複雑な、または繰り返されるセクションを表すために使用する構文断片が含まれています。構文断片は、メインの構文図に続きます。各構文断片名は大/小文字混合で、メインの構文図と断片の見出しに表示されます。以下の構文例は、Fragment1 と Fragment2 という名前の 2 つの断片が存在する構文図を示しています。

Fragment1



Fragment2



第 1 章 機能の設計

NetView プログラムを使用すると、複雑なマルチベンダー・ネットワークおよびシステムを単一ポイントから管理できます。本章では、NetView プログラムに対する追加または変更を行う前に、知っておく必要のある事項を説明します。また、タスクのカスタマイズに役立つ利用可能な機能を挙げています。

カスタマイズの分野

NetView プログラムのカスタマイズは、ネットワークおよびシステムを構築するときの種々の場面で必要なため、これらのトピックについては別の何冊かの NetView 資料で説明されています。 4 ページの表 1 は、次に挙げる項目についての詳しい情報が載っている NetView の資料を示しています。

別名 は、ネットワーク間の通信に使用されます。別名の使用によって、複数のネットワークに存在するリソース名が重複することによる問題を解決することができます。送信側ネットワークからの論理装置 (LU)、サービス・クラス、送信側 LU (SRCLU)、または LOGON モード・テーブルなどのリソースの名前は、別名により、受信側ネットワーク内で固有な名前に変換されます。別名の定義方法については、「*IBM Tivoli NetView for z/OS インストール:概説*」を参照してください。

フィルター操作 は、オペレーターに提示されるデータの量を制御するものです。また、フィルター操作は、ネットワーク・ログに記録されるデータの量も制御します。NetView 自動化テーブルを使用することにより、ネットワークの各オペレーターが受け取るメッセージのタイプとメッセージ・ログに記録されるデータ量を制御することができます。自動化ステートメントの説明と、メッセージを抑止する (フィルターにかける) 自動化ステートメントの使用方法の説明については、「*IBM Tivoli NetView for z/OS 自動操作ガイド*」を参照してください。

また、ネットワーク・リソースによりハードウェア・モニターに送られたイベント・データもフィルターにかけることができます。***記録フィルター*** は、どの情報をハードウェア・モニターのデータベースに記録するかを制御します。***ビューイング・フィルター*** は、ネットワークの各オペレーター用端末にどのレコードを表示するかを決定します。ハードウェア・モニターのフィルター操作については、「*IBM Tivoli NetView for z/OS ユーザーズ・ガイド: NetView*」に詳しい情報が記載されています。特定のイベントに関して記録フィルターを設定する自動化ステートメントの使用方法の説明については、「*IBM Tivoli NetView for z/OS 自動操作ガイド*」を参照してください。また、SRF および SVF コマンドについては、NetView オンライン・ヘルプも参照できます。

フォーカル・ポイント・サポートを使用することにより、NetView プログラムをフォーカル・ポイント・ノードまたは分散されたエントリー・ポイント・ノードとして定義することができます。フォーカル・ポイントは中央ネットワーク・ノードであり、分散されたエントリー・ポイント・ネットワーク・ノードから情報を受け取ります。エントリー・ポイントからフォーカル・ポイントに送られる情報には、メ

ッセージ、アラート、MSU などがあります。NetView フォーカル・ポイント・サポートについての詳細は、「*IBM Tivoli NetView for z/OS 自動操作ガイド*」を参照してください。

自動操作 をインプリメントすることにより、ネットワークで発生したイベントに対し自動的に応答することができます。システム・オペレーターおよびネットワーク・オペレーターの生産性を向上させるための NetView 自動化ステートメントの定義方法の詳細については、「*IBM Tivoli NetView for z/OS 自動操作ガイド*」を参照してください。NetView プログラムの自動化に関する追加情報については、「*IBM Tivoli NetView for z/OS 自動操作ガイド*」を参照してください。

総称アラート および***コード・ポイント*** を使用すると、NetView プログラムによる自動的な問題判別サポートが行われないネットワーク内のデバイスおよびアプリケーションに対しての問題判別サポートが可能になります。 87 ページの『第 6 章 ハードウェア・モニターの表示データのカスタマイズ』では、NetView プログラム提供のコード・ポイント・テーブル、およびユーザー定義のコード・ポイント・テーブルを使用して、ハードウェア・モニターの Alerts-Dynamic、Alerts-Static、Alerts-History、Event Detail、および Most Recent Events の各パネルを作成する方法が説明されています。

各国語サポート により、オペレーターは英語以外の言語で NetView プログラムと対話できるようになります。その他のサポートされている言語でユーザー独自のメッセージ翻訳を作成する方法については、「*IBM Tivoli NetView for z/OS インストール:追加コンポーネントの構成*」を参照してください。日本語版では、日本語の NetView のパネルとメッセージを提供します。

オペレーター制御とセキュリティー については考慮の必要があります。NetView プログラムにアクセスできる人と、オペレーターがネットワークに与えることのできる効果を制御するためには、あるレベルでのログオン検証、コマンド権限および制御スパンについて考慮してください。NetView プログラムで使用可能なセキュリティー検証のさまざまなレベルをインプリメントする方法、オペレーターが出すことのできるコマンドを制限する (コマンド権限) 方法、およびオペレーターが制御できるネットワーク・リソースの部分 (制御スパン) 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。

NetView コマンド・ファシリティー・パネル のカラーおよび形式を変更することができます。詳しくは、 31 ページの『第 2 章 NetView コマンド・ファシリティー・パネルのカスタマイズ』を参照してください。

オンライン・ヘルプ、オンライン・メッセージ・ヘルプ、NetView のヘルプ・デスク、ハードウェア・モニター、およびユーザー作成のすべてのフルスクリーン・アプリケーションの***パネル*** は作成および変更することができます。これらのコンポーネント用に新しいパネルを作成する方法、または NetView プログラム提供のパネルを変更する方法について詳しくは、 73 ページの『第 4 章 オンライン・ヘルプ情報の変更と作成』または 87 ページの『第 6 章 ハードウェア・モニターの表示データのカスタマイズ』を参照してください。

順次ログ (順次アクセス方式ログ・サポート) により、可変長レコードを複数のユーザー定義ログに書き込むことができます。オペレーティング・システムの機能を使用して、これらのログをブラウズまたは印刷することができます。順次ログ・タス

クの定義方法については、「*IBM Tivoli NetView for z/OS インストール:追加コンポーネントの構成*」、「*IBM Tivoli NetView for z/OS プログラミング:アセンブラー*」、または「*IBM Tivoli NetView for z/OS プログラミング: PL/I および C*」を参照してください。

セッション・モニター・データ は、セッション・モニター・データベース内に収集され保存されます。 収集および保管されるセッション・データの量を制御するには、いくつかのセッション・モニター定義ステートメントをカスタマイズしてください。詳しくは、「*IBM Tivoli NetView for z/OS インストール:追加コンポーネントの構成*」を参照してください。「*IBM Tivoli NetView for z/OS インストール:追加コンポーネントの構成*」には、応答時間モニター (RTM) 機能のパフォーマンス・クラスの定義方法も記載されています。各パフォーマンス・クラスについて目標と境界が設定され、1 つのセッションに対し 1 つのパフォーマンス・クラスが選択されます。

ユーザー作成機能 は、NetView プログラムに新規機能を追加、または既存の機能を変更します。ユーザー独自のコマンド・リストやユーザー作成のコードを作成することができます。「*IBM Tivoli NetView for z/OS プログラミング: REXX および NetView コマンド・リスト言語*」では、 REXX または NetView コマンド・リスト言語を用いて、ネットワークを制御したりオペレーターのジョブを簡単にするためのコマンド・リストを作成する方法を説明しています。コマンド・プロシージャーやインストール・システム出口などのコードの作成方法は、「*IBM Tivoli NetView for z/OS プログラミング: PL/I および C*」で説明されています。アセンブラー言語によるコマンド・プロセッサー、インストール・システム出口ルーチン、およびユーザー・サブタスクの作成方法は、「*IBM Tivoli NetView for z/OS プログラミング: アセンブラー*」で説明されています。

NetView の***リソース・オブジェクト・データ・マネージャー (RODM)*** は、ネットワーク構成とシステム・リソースに関する状況の情報を保管するデータ・キャッシュです。 RODM により、RODM で定義されるリソースに関連するネットワーク管理機能を自動化できます。さらに、RODM アプリケーションを作成して、ほかのネットワーク管理タスクおよび自動化タスクを実行することができます。詳しくは、「*IBM Tivoli NetView for z/OS Resource Object Data Manager and GMFHS Programmer's Guide*」を参照してください。

変更を始める前に考慮すべき機能

NetView の機能をカスタマイズする場合は、ユーザー独自のコマンド・プロシージャーを作成することも、NetView プログラムで提供される既存のコマンド・プロシージャーを変更することもできます。既存の機能を変更する方法には次のものがあります。

- NetView プログラムにより書き込まれるシステム管理機能 (SMF) レコードをフィルター操作するまたは変更する。
- オペレーター・メッセージの経路を定める方針を提供する。
- オペレーター・メッセージの再フォーマット設定、分析、または編集を行う。
- コマンド権限を検査する。

追加できる機能としては、X.25 データ・ネットワーク・コンポーネントや音声ネットワーク・コンポーネントなどの、ネットワーク内の追加コンポーネントの管理な

どがあります。既存の管理機能を使って、ユーザーの要件に合うように新規アプリケーションを作成し統合することができます。ユーザー定義の機能の例として、次のものが挙げられます。

- ネットワーク内の特定のリソース、アプリケーションまたはコンポーネントの実時間モニター
- 傾向分析やその他の必要なデータ整理アプリケーションのための追加の SMF データ収集および記録
- 追加の応答時間問題検出およびアラートの提供
- 異なるクラスの回線問題の検出

カスタマイズ情報の関連資料

表 1 は、カスタマイズ項目をリストし、その項目に関係する情報が記載されている資料名を示しています。

表 1. カスタマイズ項目と参考資料

項目	CGD	GET	OLH	CLS	PLC	ASL	AUT	PIP	ASR	NUG	ADV
別名		X							X		
コマンド・ファシリティー 画面フォーマット	X								X	X	
自動化							X			X	X
総称アラート	X						X				
各国語 サポート											X
オペレーター管理: ログオン・セキュリティー コマンド・セキュリティー 制御スパン									X X X		
パネル: ハードウェア・モニター ヘルプ ヘルプ・デスク ユーザー作成	X X X X									X X X X	

表 1. カスタマイズ項目と参考資料 (続き)

項目	CGD	GET	OLH	CLS	PLC	ASL	AUT	PIP	ASR	NUG	ADV
順次ログ	X				X	X					X
セッション・モニター・ データ: 応答時間モニター モニター・ セッション認識											 X X
抑止: メッセージ ハードウェア・モニター				X			 X X		X		
ユーザー作成機能: コマンド・リスト ユーザー作成 プログラミング (PL/I、C) ユーザー作成 プログラミング (アセンブラー) NetView パイプライン			 X	X	 X	 X		 X			

凡例:

CGD *IBM Tivoli NetView for z/OS カスタマイズ・ガイド*

GET *IBM Tivoli NetView for z/OS インストール:概説*

OLH NetView オンライン・ヘルプ

CLS *IBM Tivoli NetView for z/OS プログラミング: REXX および NetView コマンド・リスト言語*

PLC *IBM Tivoli NetView for z/OS プログラミング: PL/I および C*

ASL *IBM Tivoli NetView for z/OS プログラミング:アセンブラー*

AUT *IBM Tivoli NetView for z/OS 自動操作ガイド*

PIP *IBM Tivoli NetView for z/OS プログラミング:パイプ*

ASR *IBM Tivoli NetView for z/OS アドミニストレーション・リファレンス*

NUG *IBM Tivoli NetView for z/OS ユーザーズ・ガイド: NetView*

ADV *IBM Tivoli NetView for z/OS インストール:追加コンポーネントの構成*

AON のカスタマイズについては、「*IBM Tivoli NetView for z/OS Automated Operations Network ユーザーズ・ガイド*」を参照してください。

NetView 管理コンソールのカスタマイズについては、「*IBM Tivoli NetView for z/OS プログラミング: REXX および NetView コマンド・リスト言語*」を参照してください。

Tivoli NetView for z/OS Enterprise Management Agent のカスタマイズについては、「*IBM Tivoli NetView for z/OS NetView Enterprise Management Agent ユーザーズ・ガイド*」を参照してください。

データの収集

カスタマイズ・プロシージャーに有用なデータ収集源の代表的なものは次のとおりです。

- NetView プログラムに用意されているインストール・システム出口インターフェース
- 状況、構成、処理、または権限の情報を提供するシステム・サービスまたは NetView サービス
- システム・サービスまたは NetView サービスを使ってアクセスされるデータ・ファイルおよびネットワーク・デバイス
- システムまたはアプリケーションに重要なイベントが発生していることをオペレーターに示すメッセージ

インストール・システム出口

NetView インストール・システム出口の中には、ネットワーク管理データへのアクセスを可能にするものがあります。これらのインストール・システム出口およびユーザー作成の機能を使用して、オペレーター・コマンド、メッセージ、およびログオンのテキストを得ることができます。 VTAM 通信ネットワーク管理 (CNM) インターフェースのデータと同様に、NetView プログラムが VSAM ファイルおよび SMF ログに書き込むデータは、他の NetView インストール・システム出口内でアクセスすることができます。

参照: NetView インストール・システム出口の詳細については、「*IBM Tivoli NetView for z/OS 自動操作ガイド*」、「*IBM Tivoli NetView for z/OS プログラミング:アセンブラー*」、および「*IBM Tivoli NetView for z/OS プログラミング: PL/I および C*」を参照してください。

サービス・ルーチン

システムまたは NetView サービスにより、次のような情報にアクセスすることができます。

- システム日付および時刻
- プログラムのアドレス
- 指定ストレージ域のアドレス
- 有効な NetView オペレーター
- オペレーター制御スパン
- コマンド・リスト変数の値

参照: DSIDATIM、DSICES、DSIFIND、DSIQOS、DSIQRS、および DSIKVS などのマクロについては、「*IBM Tivoli NetView for z/OS プログラミング:アセンブラ*

ー」を参照してください。CNMINFC、CNMNAMS、CNMSCOP、および CNMVARS などのサービス・ルーチンについては、「*IBM Tivoli NetView for z/OS プログラミング: PL/I および C*」を参照してください。

データ・ファイル

NetView プログラムには、ネットワーク管理データ・ファイルにアクセスするための特殊なディスク・サービスおよび VSAM データ・サービスがあります。さらに、PL/I や C のような高水準言語 (HLL) で作成された機能により、NetView 区分データ・セットの読み取りおよび VSAM 入出力要求のためのシステム割り振りおよびアクセス方式を呼び出すことができます。通信ネットワーク管理インターフェース・サービスもまた、ネットワーク内のデバイスから送られてきたデータへのアクセスを可能にします。

NetView PIPE コマンドを使用する場合、QSAM および < (ディスクから) ステージを使用して、データ・ファイルを読み取ることができます。パイプ機能により、DSIVSAM と DSIVSMX を使用して VSAM データにアクセスすることもできます。DSIVSAM および DSIVSMX については、「*IBM Tivoli NetView for z/OS プログラミング:パイプ*」を参照してください。

REXX コマンド・リストは、順次データ・セットまたは区分データ・セット・メンバーから読み取るまたは書き込むために EXECIO コマンドを使用することができます。

参照: VSAM および通信ネットワーク管理インターフェース・サービスについては、「*IBM Tivoli NetView for z/OS プログラミング: PL/I および C*」を参照してください。

パイプの詳細については、「*IBM Tivoli NetView for z/OS プログラミング:パイプ*」を参照してください。

REXX ファイルの入出力については、「*IBM Tivoli NetView for z/OS プログラミング: REXX および NetView コマンド・リスト言語*」を参照してください。 NetView データ・セットまたはファイルへの読み取りアクセスを行うための DSIDKS の使用、VSAM 入出力用の DSIZVSMS の使用、および CNM データ・サービス用の DSIZCSMS の使用については、「*IBM Tivoli NetView for z/OS プログラミング:アセンブラー*」を参照してください。

オペレーター・コマンドとメッセージ

コマンド・プロシージャー内でオペレーター・コマンドを出し、状況データを要求することができます。その結果要求された状況データを含む応答メッセージは、コマンド・プロシージャーにトラップされて処理されます。また、ユーザー作成のコマンド・プロシージャーで、他のシステムおよびネットワーク・メッセージのデータを処理することができます。このコマンド・プロシージャーは、NetView 自動操作で呼び出されます。

参照: REXX および NetView コマンド・リスト言語のメッセージ処理については、「*IBM Tivoli NetView for z/OS プログラミング: REXX および NetView コマンド・リスト言語*」を参照してください。PL/I および C 言語のメッセージ処理については、「*IBM Tivoli NetView for z/OS プログラミング: PL/I および C*」を参照し

てください。自動操作オプションの作成についての詳細は、「*IBM Tivoli NetView for z/OS 自動操作ガイド*」を参照してください。

データの保管と記録

NetView コマンド・プロシージャーを使用して、多くのユーザー作成機能のために必要なデータを保管し検索することができます。REXX、NetView コマンド・リスト言語、PL/I、または C で書かれているコマンド・プロシージャーは、グローバル変数およびタスク変数を作成し、設定し、読み取ることができます。

データの永続保管およびデータ・ボリュームの増大を考えると、特定の情報に名前を付けてコマンド・リスト変数として保管させるより、その情報をデータ・ファイルに記録した方がよい場合があります。NetView プログラムにより、このデータをログに記録することが可能になります。例えば、NetView プログラムがログに記録しているシステムまたはネットワークの活動と一緒に、ユーザー・アプリケーションの活動をログに記録することができます。また、自分で収集したデータ用の別のログを作成することができます。

参照: 順次ログの詳細については、「*IBM Tivoli NetView for z/OS インストール:追加コンポーネントの構成*」および本書の 16 ページの『言語の選択』を参照してください。

オペレーター用表示

VIEW コマンドを使用して、またはオペレーターにデータを表示するためにいくつかの NetView システムのコンポーネントが使用するパネルを変更することで、NetView プログラムのオペレーター用表示機能のいくつかをカスタマイズまたは拡張することができます。詳細については、37 ページの『第 3 章 VIEW コマンドの使用』 および 73 ページの『第 4 章 オンライン・ヘルプ情報の変更と作成』 を参照してください。

オペレーターに情報を表示するメッセージも使用することができます。メッセージの使用により、ユーザー作成機能からのデータは、NetView 自動操作処理の対象となり、ユーザー独自の機能の自動操作および手操作の両方が可能になります。

参照: DSIWCS、DSIMBS、DSIMQS、DSIPSS およびその他のメッセージ・サービスについては、「*IBM Tivoli NetView for z/OS プログラミング:アセンブラー*」を参照してください。CNMSMSG については、「*IBM Tivoli NetView for z/OS プログラミング: PL/I および C*」を参照してください。 REXX および NetView コマンド・リスト言語によるオペレーター宛メッセージ (WTO) とその他のメッセージ・サービスについては、「*IBM Tivoli NetView for z/OS プログラミング: REXX および NetView コマンド・リスト言語*」を参照してください。

また、NetView コマンド・ファシリティー・パネルもカスタマイズすることができます。詳しくは、31 ページの『第 2 章 NetView コマンド・ファシリティー・パネルのカスタマイズ』を参照してください。

タスク

NetView プログラムに対して拡張機能を作成する場合には、NetView が z/OS に基づいて設計されていることに留意してください。

参照: z/OS ライブラリーには、*dispatch*、*task* などの用語、および種々のシステム・サービス名がこのセクションでどのように使用されているかが説明されており参考になります。

システム・アプリケーション・プログラムとしての NetView プログラム

NetView プログラムは、複数の並列タスクで構成され、マルチタスキング環境においてそれぞれを別個にディスパッチすることができます。そのうちの 1 つが遊休タスクであれば、その他のタスクが実行される資格を有します。システムのマルチタスキングのディスパッチャーは、NetView プログラムの ATTACH システム・サービスを使用してそれぞれの新しいタスクを作成します。WAIT システム・サービスは、各 NetView タスクが行う処理がなくなって遊休タスクになる場合に、それぞれの NetView タスクによって呼び出されます。POST システム・サービスを使用すると、タスクを遊休状態から取り出すことができ、新しい入力データがそのタスクのために処理できるようになったときに、そのタスクを指名することができるようになります。

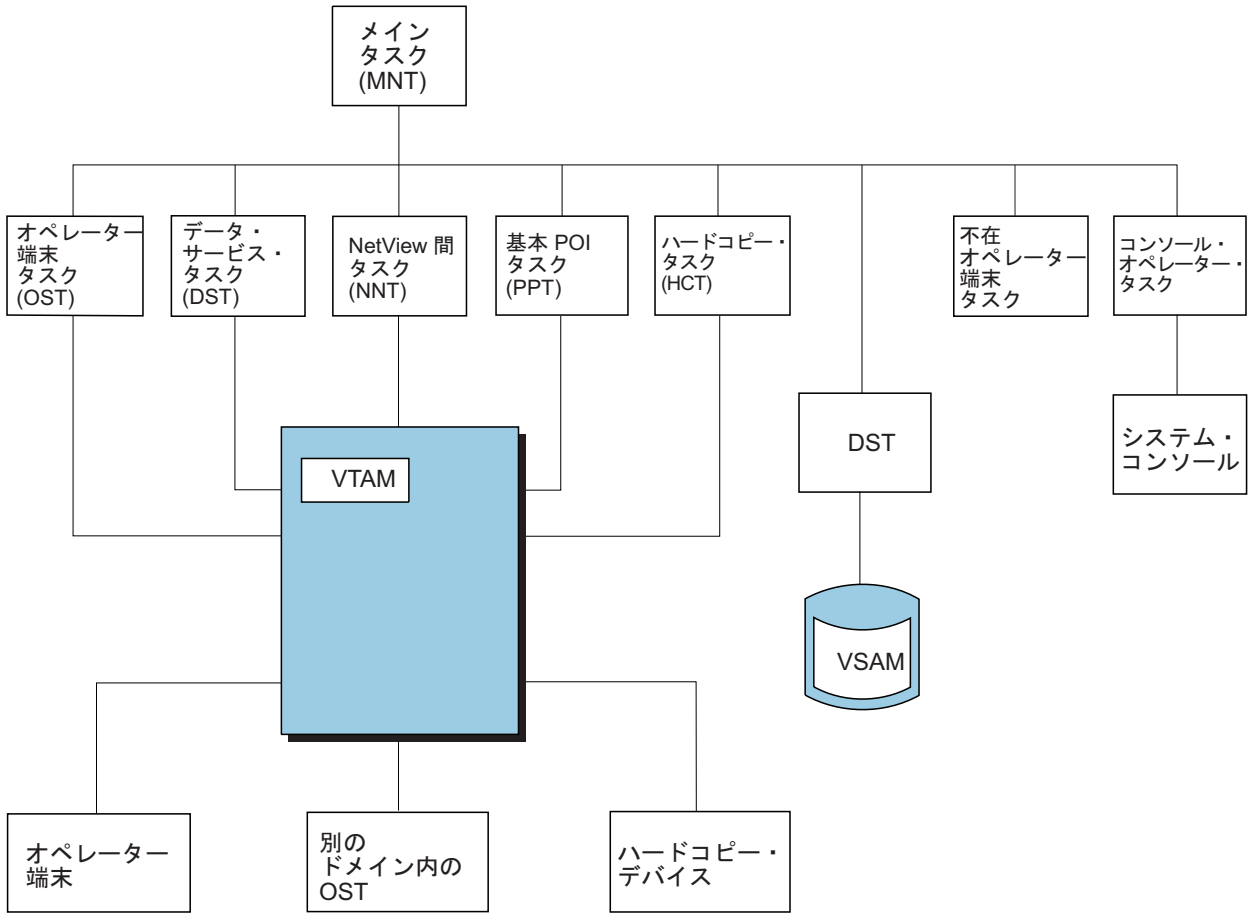
NetView プログラム・タスク

NetView プログラムが始動すると、そのメインタスクは、実行される機能に応じて、異なるタイプのいくつかのサブタスクを生成します。異なるそれぞれのタスク・タイプにより、そのタスクで利用可能な特定のシステム・インターフェースおよびオペレーター・インターフェース、および実行可能なトランザクションのタイプが決まります。

各オペレーター端末タスク (OST) は、固有名で識別される NetView オペレーター 1 人をサポートします。オペレーター ID (OPID) は、NetView パラメーター・ライブラリーに定義されます。自動化操作プログラム (自動タスクと呼ばれる) が AUTOTASK コマンドの使用により活動化されるか、またはオペレーターが VTAM 接続の端末を使ってログオンすると、OPID が OST に割り当てられます。

各 NetView 間タスク (NNT) もオペレーターをサポートします。このタイプのタスクは、オペレーターが NetView プログラムへのログオンを端末からではなく、ほかの NetView プログラムから行う場合に使用します。ほかの NetView プログラムは、別の計算機で実行されているものでもかまいませんが、VTAM を介して接続されていなければなりません。オペレーターは START DOMAIN コマンドを使って、別の NetView プログラムからログオンを行います。

各ハードコピー・タスク (HCT) は、VTAM を介して接続されている 3287 印刷装置をサポートし、オペレーターにハードコピー・ログを提供します。コマンド・ファシリティーの構造の概要およびタスクの構造については、10 ページの図 1 を参照してください。



注: VTAM がアクティブでないときも NetView は実行できます。

図 1. コマンド・ファシリティーの構造の概要

各 NetView プログラムには、基本プログラム式オペレーター・インターフェース・タスク (PPT) が 1 つだけあります。 VTAM の実行中、PPT は特殊な VTAM アプリケーション制御ブロック (ACB) を開いて、 VTAM プログラム式オペレーター・インターフェース (POI) が VTAM からの非送信請求データを受け取れるようにします。

注: 本書で使用する *VTAM* という用語は、z/OS Communications Server の VTAM コンポーネントを意味しています。

各オプショナル・タスク (OPT) は、TASK ステートメントにより、NetView パラメーター・ライブラリーに定義する必要があります。OPT 用に実行されるプログラム・モジュールは、 15 ページの『オプショナル・タスクの NetView プログラムへの追加』で説明されているオプショナル・タスクの条件に適合するものであればどのようなプログラムでもかまいません。

各データ・サービス・タスク (DST) は、オプショナル・タスクの特定なものです。 15 ページの『オプショナル・タスクの NetView プログラムへの追加』を参照してください。DST 用の TASK ステートメントは、NetView パラメーター・ライブラ

リーに初期設定メンバーを指定することができます。このライブラリーからは、指定された DST により実行される機能のパラメーターを定義するためのステートメントが読み出されます。

タスク内のプログラム活動

それぞれのタイプの NetView タスクは、活動化されると、ある要求が特定の作業単位を実行するのを待ちます。その作業単位が完了すると、タスクは通常の待ち状態になります。作業単位を実行する別の要求を受信すると、そのタスクは再び実行されます。各タスクは WAIT を出す場合、イベント制御ブロック (ECB) のリストを使用します。NetView ではカスタマイズのマクロおよびサービスが提供され、暗黙の待機はすべてタスクの ECB リストを介して確実に行われるようになり、これにより、NetView プログラム内のタスク要求インターフェースはすべて使用可能状態が維持されます。

各 NetView タスクには、独自の終了 ECB と独自のメッセージ・キュー ECB があります。タスクのタイプ (例えば、OST、DST など) によっては、タスクの ECB リストにさらに ECB がある場合があります。これらの追加 ECB は、タスクが WAIT 状態から外されてポストされたときにテストを行い、実行する処理を表します。

NetView プログラム・タスクへの作業のキューイング

1 つのタスクが通常の WAIT 状態である場合、NetView プログラム内のもう 1 つのタスクが実行可能です。実行中の NetView タスクは、常に、システムのイベントにより割り込まれたり、より優先順位の高いタスクによって、そのタスクが通常の WAIT を出すまで占有されたりする可能性があります。NetView プログラムの外部のシステム機能も、特定の NetView タスクに関連したスケジュール済みの割り込み出口ルーチンの実行により、NetView 処理に割り込むことができます。

タスクのデータは、タスクのメッセージ・キューまたは他の作業キューに入れることができ、タスクはいつでもその作業を実行するようにポストすることができます。このデータは、ほかの NetView タスクで作成されることもあります。これは、DST がメッセージ・データを OST のキューに入れてオペレーターに表示するときに行われることがあります。このデータは、VTAM RECEIVE 要求の完了などのイベントによってスケジュールされる、割り込み出口ルーチンを介して NetView プログラムに入力することができます。

メッセージおよびコマンドのバッファー

種々のタスク・キューに入れられるデータは、メッセージ・バッファーまたはコマンド・バッファーと呼ばれる特殊なデータ構造にフォーマット設定されます。バッファーの最初にあるヘッダーは、バッファー内に収められているデータのタイプを示し、またそのデータにアクセスするために特殊なフォーマットが必要であればそのフォーマットを示します。コマンドは、NetView プログラムのカスタマイズ・プログラミングで提供するコマンド・プロセッサーと呼ばれるプログラムによって処理されます。メッセージは、NetView タスクに作成された事前定義に応じて、または NetView 自動化コマンド・プロセッサーのいずれかによって処理されます。また、メッセージ・バッファーはインストール・システム出口を介して NetView 処理の種々の時点での自動化が利用可能です。

即時コマンド

即時コマンドは、オペレーターがそのコマンドを入力すると同時に処理を開始します。要求された機能は、そのタスクが長い作業キューの中にあっても即座に実行されます。

即時コマンドは OST および NNT サブタスク環境下で実行されます。他のコマンドと異なり、即時コマンドは TVBINXIT ビットがオンに設定されて制御を受け取ることができます。即時コマンドは、メインラインの処理に割り込みますが、別のコマンドによって割り込まれることはありません。他の非同期活動の出口が即時コマンドに割り込むことができます。

長期実行コマンド

長期実行コマンドは、処理を中断してオペレーター・コマンドやデータ検索などの他の活動ができるようにし、後で処理を再開することができるコマンドです。NetView のコンポーネントは、すべて長期実行コマンドです。NetView コマンド・リスト言語、REXX、PL/I、および C コマンド・プロシージャーも長期実行コマンドです。DSIPUSH マクロにより、アセンブラー・コマンドを長期実行コマンドとして実行できます。

長期実行コマンドは、OST、NNT、PPT、または DST (ログオフ・ルーチンの場合のみ) のもとで実行されます。長期実行コマンドは、次のように呼び出すことができます。

- オペレーターの入力により直接呼び出す。
- コマンド・リストにより呼び出す。
- 他の長期実行コマンドにより呼び出す。

長期実行コマンドは、作業のスケジューリングが終わると、処理の完了前に制御を NetView プログラムに戻します。次に、NetView プログラムは、他に保留されている作業があればそれを処理します。

長期実行コマンド・プロセッサーは、他のタスクや領域のデータの検索時に、呼び出し機能や呼び出しコマンド・リストが検索中に続行されないようにする場合によく使われます。検索が実行されると、そのプロセッサーのタスクは、メッセージの受信およびコマンドの受諾を続行できます。

データ・サービス・コマンド

データ・サービス・コマンド・プロセッサー (DSCP) は、DST サブタスク環境下で実行されます。DSCP は CNM データ・サービスおよび VSAM データ・サービスを行います。また、DSCP は、通信ネットワーク管理インターフェース・サービスも VSAM サービスも使用しない集中ユーザー定義機能または順次ユーザー定義機能のために呼び出すこともできます。

ホストでのユーザー作成プログラムの定義: 出口プログラムおよびコマンド

NetView タスク環境内で作成できるユーザー作成プログラムには、次の 2 種類があります。

- インストール・システム出口
- コマンド・プロセッサー

参照: プログラミング・インターフェースの詳細については、「*IBM Tivoli NetView for z/OS プログラミング: PL/I および C*」および「*IBM Tivoli NetView for z/OS プログラミング:アセンブラー*」を参照してください。ユーザー作成機能を設計する際には、独自のプログラミングを NetView プログラムの全体的な構造に適合させるために、NetView プログラムでインストール・システム出口インターフェースおよびコマンド・プロセッサー・インターフェースを使用することができます。

インストール・システム出口プログラム

インストール・システム出口は、NetView プログラムにおいて、ログオンおよびログオフ・データ、コマンド・バッファー、およびメッセージ・バッファーの処理過程での数個所にあります。さまざまな出口は、バッファーの起点とその出口が存在する NetView 処理の段階に基づいて駆動されます。特殊出口は DST のもとで駆動され、初期設定、入力、および出力中にタスクのデータを処理します。

参照: NetView インストール・システム出口の要約については、「*IBM Tivoli NetView for z/OS 自動操作ガイド*」、「*IBM Tivoli NetView for z/OS プログラミング:アセンブラー*」、および「*IBM Tivoli NetView for z/OS プログラミング: PL/I および C*」を参照してください。

一般のインストール・システム出口は事前割り当てされたモジュール名 (DSIEX*nn*) によって識別されて呼び出され、DST 出口はタスク DSTINIT 初期設定ステートメントで固有に識別されます。

DSIEX21 が、DSITCPRF メンバーへのアクセスに使用されます。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。

コマンド・プロセッサーおよびコマンド・リスト

NetView のコマンド・プロセッサーおよびコマンド・リストは、次のものにより開始されます。

- オペレーターからの要求
- NetView の任意のプログラムによる処理のため、タスクのキューに入れられたコマンド・バッファー
- 別のコマンド・プロセッサーからのコマンドの呼び出し
- NetView 自動化テーブルに指定されたアクション

参照: NetView コマンド・リスト言語または REXX で作成されたコマンド・リストを NetView プログラムに対して定義するには、それらを NetView コマンド・リスト・ライブラリー (*ddname* DSICLD) に入れます。特定のオペレーティング・システム用にコマンド・リストを作成する方法については、「*IBM Tivoli NetView for z/OS プログラミング: REXX および NetView コマンド・リスト言語*」を参照してください。

PL/I、C、およびアセンブラー・コマンド・プロセッサーを NetView ロード・ライブラリー (*ddname* STEPLIB) にリンク・エディットして、それらを NetView プログラムに定義しなければなりません。 NetView プログラムに対して PL/I、C、またはアセンブラーで作成されたコマンド・プロセッサーを定義するには、DSIPARM の CNMCMD メンバーにある CMDDEF ステートメントを使用します。コマンド・プロセッサーは、NetView ロード・ライブラリーにリンク・エディットされます。

複数のインストール・システム出口プログラムおよびコマンド・プロセッサーで、1 つの機能のうちのいくつかの部分をインプリメントすることができます。コマンド・プロセッサー全体に対して機能を分割する一般的な方法としては、処理を OST と DST に分割する方法があります。OST はデータをオペレーター端末から受け取り、データをそこへ戻すため、コマンド・プロセッサーは次のことを行うように作成されます。

- オペレーターによりそのコマンドが入力された場合に呼び出されます。
- コマンド・データを解析しデータ・サービス要求を形成します。
- データ・サービス・コマンドを収めるコマンド・バッファーをキューに入れて、DST により処理されるようにします。
- オペレーターにエラー・メッセージまたはコマンド確認メッセージを戻します。

DST は、コマンド・バッファーが最初のコマンド・プロセッサーにより作成され、キューに入れられたために、呼び出された、別のコマンド・プロセッサーの機能を完了します。DST のもとでは、VSAM の特殊データ・サービス、外部ログ、または VTAM 通信ネットワーク管理インターフェースを必要とする機能が実行され、コマンドをキューに入れたオペレーター・タスクにメッセージが戻されます。図 2 は、通信ネットワーク管理インターフェースおよび VSAM サービスを使用する機能の場合の代表的なプログラム設計を示しています。

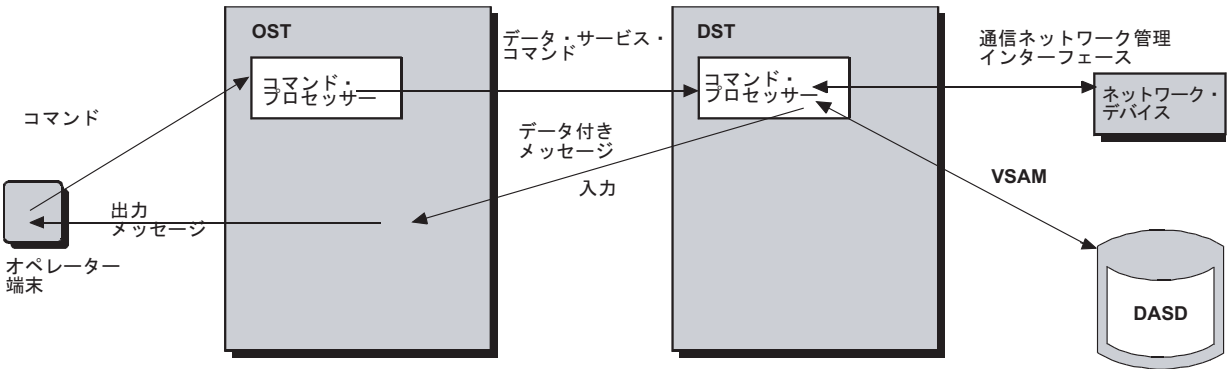


図 2. DST 機能用のプログラム設計例

長期実行コマンドでは、複合機能を別々のトランザクションの順序列に分離することができます。コマンド・プロセッサーは、アンカー・アドレスを保管する名前付きのスタック項目を設定できます。関連するコマンド・プロセッサーは、後でこのアドレスを検索したり、同じ処理の別のフェーズも実行できます。

コマンドに命名する場合は、以下のガイドラインに従ってください。

- 文字 (英字) で開始する。
- コンマやコロンなどの特殊文字は使用しない。

- NetView のコマンド名は使用しない (内部コマンドと CNMCMD に入れて出荷されたコマンドの両方とも)。NetView の内部コマンド名は、CSCFDST、HMSTATS、LOGNMVT、LOGRU、MESSAGE、PIPE、および VIEW です。
- 次の NetView 接頭部は使用しない。
  - AAU
  - BNH
  - BNI
  - BNJ
  - BNK
  - BNT
  - CNM
  - DSI
  - DUI
  - DWO
  - EGV
  - EKG
  - EUY
  - EXQ
  - EYV
  - EZL
  - FKB
  - FKV
  - FKW
  - FKX
  - FLB
  - FLC
  - FMG
  - FNA
  - IHS

注: コマンド・プロセッサーまたはコマンド・リストで異なる方法によって発行されたメッセージは、コマンド・プロセッサーまたはコマンド・リストで要求されたものと同じ順序で宛先に表示されない場合があります。例えば、コマンド・リストが指定文字 (DSIG) を介して制御権を取得し、CONSOLE=*ANY* 自動タスク上で実行されるとします。コマンド・リストが **CONSOLE** ステージを使用して **PIPE** コマンドを発行し、次に **WTO** コマンドを発行した場合、**WTO** コマンドによって発行されたメッセージが、**CONSOLE** を使用して **PIPE** コマンドで発行されたメッセージよりも前にオペレーター・コンソールに表示されることがあります。

オプショナル・タスクの NetView プログラムへの追加

NetView プログラムがオプショナル・タスク (OPT) またはサブタスクとして開始する、まったく新しいサブタスクをアセンブラー言語で作成することができます。

OPT には、サブタスクの初期設定、インストール・システム出口、メッセージ、およびコマンド処理機能および終了のためのコーディングを行う必要があります。既存の DST にはすでにこれらのエレメントの一部が設定されているため、DST を開始点として使用するとより実用的です。

参照: アセンブラー言語における OPT および DST の詳細については、「*IBM Tivoli NetView for z/OS プログラミング:アセンブラー*」を参照してください。

言語の選択

ご使用のシステムにとって必要な特定のカスタマイズには、特定のアプリケーション・プログラム・インターフェースが最も適している場合があります。使用するインターフェースを決めるときは、パフォーマンスへの影響、作成の容易さ、および維持管理の上での効果を考えてください。このセクションでは、利用可能な言語について説明し、言語選択の理由をリストしています。

入出力

REXX、PL/I、C、およびアセンブラーのすべてには、直接アクセス記憶装置 (DASD) からの読み取りまたは書き込みのための機能があります。NetView プログラムには、ネットワーク管理データ・ファイルにアクセスするための特殊なディスク・サービスおよび VSAM データ・サービスがあります。さらに、PL/I や C で作成された機能は、システム割り振り方式およびアクセス方式を呼び出して、データの読み書きを行うことができます。通信ネットワーク管理インターフェース・サービスもまた、ネットワーク内のデバイスから送られてきたデータへのアクセスを提供します。

参照: VSAM および通信ネットワーク管理インターフェース・サービスについては、「*IBM Tivoli NetView for z/OS プログラミング: PL/I および C*」を参照してください。 NetView データ・セットまたはファイルへの読み取りアクセスを行うための DSIDKS の使用、VSAM 入出力用の DSIZVSMS の使用、および CNM データ・サービス用の DSIZCSMS の使用については、「*IBM Tivoli NetView for z/OS プログラミング:アセンブラー*」を参照してください。

REXX ファイルの入出力については、「*IBM Tivoli NetView for z/OS プログラミング: REXX および NetView コマンド・リスト言語*」を参照してください。

パフォーマンス

パフォーマンスを重視するアプリケーションは、コンパイルまたはアセンブルされる言語で作成してください。一般に、コンパイル済みまたはアセンブル済みのコマンド・プロシージャーは、解釈コマンド・リスト (REXX および NetView コマンド・リスト言語) より速く実行されます。

NetView 駆動式インストール・システム出口ルーチンはアセンブラー、PL/I、または C で作成しなければなりません。また、NetView 制御ブロックにアクセスするすべてのコマンド・プロセッサーは、アセンブラーで作成しなければなりません。端末入力またはメッセージによって駆動することができ、NetView 制御ブロックにアクセスする必要のないコマンド・プロシージャーは、通常、REXX または NetView コマンド・リスト言語で作成することができます。通常、REXX で作成されたコマンド・リストは、NetView コマンド・リスト言語で作成されたコマンド・リストよりも実行時のパフォーマンスが若干よくなります。 17 ページの『REXX と NetView コマンド・リスト言語の比較』を参照してください。また、REXX コマンド・リストのパフォーマンスは、REXX コマンド・リストをコンパイルすると向上させることができます。

REXX または NetView コマンド・リストの事前ロード (LOADCL コマンドについては NetView オンライン・ヘルプを参照) により、そのコマンド・リスト全体のパフォーマンスを改善することができます。

参照: REXX コマンド・リストのコンパイルについては、「*IBM Tivoli NetView for z/OS チューニング・ガイド*」を参照してください。

補足のパフォーマンス推奨事項については、「*IBM Tivoli NetView for z/OS インストール:追加コンポーネントの構成*」および「*IBM Tivoli NetView for z/OS インストール:追加コンポーネントの構成*」を参照してください。

安定度

長期間使用していくうちに、あるいは操作環境が変わったときに、ユーザーのプロシージャーを変更する必要が生じた場合は、最初にコマンド・リストを使用して、プロシージャーをインプリメントしてみてください。変更はコマンド・リストで行うとより簡単に行えます。コマンド・リストでは、NetView プログラムを再始動しないで変更内容を組み込み、オンラインでのテストを実施することができるからです。プロシージャーの安定度に確信が持てたときに、プロシージャーをコンパイル済み言語に変換することができます。

テスト方法

コマンド・リストのテストでは、オペレーター・コマンドまたはコマンド・リスト・ステートメントのいずれかを使用したトレースを実行することができます。リモート対話式デバッガー (RID) は、NetView のオペレーター・コンソールに情報を表示し、PL/I および C のユーザー作成コマンド・プロセッサーおよびインストール・システム出口をユーザーがデバッグする際の助けとなります。 NetView プログラムには、アセンブラー・プログラムのデバッグを支援する特別な機能はありません。

インプリメンテーションの速さ

コマンド・リストは作成したり、テストしたり、実行に移したりしやすいため、すぐに操作する必要がある場合にはコマンド・リストを使用するのが適切です。

REXX と NetView コマンド・リスト言語の比較

ユーザーのシステムすべてで REXX を実行できる場合は、コマンド・リストを作成する際に NetView コマンド・リスト言語よりも REXX の方を選択してください。REXX は構造化言語であり、ユーザーがサブルーチンを使用できるようにします。REXX は容易に学べる言語であり、数式機能および改善されたストリング処理のような追加機能を提供します。REXX は EXECIO を使用して、データ・セットからの読み取りと書き込みができます。また、REXX コマンド・リストのパフォーマンスは、その REXX コマンド・リストをコンパイルすると向上させることができます。

REXX 言語のスキルは、NetView プログラム以外の環境でも使用することができます。ただし、NetView プログラム用に作成された REXX プロシージャーは、その機能内容からして、おそらく他の環境へは移すことはできません。複数の環境で

は、REXX はさらに有効です。REXX プログラミング・スキルを移行して、別の言語を習得することなくNetView の問題点を解決することができるからです。

ご使用のシステムが複数のオペレーティング・システムを使用している場合、REXXをサポートするオペレーティング・システムとサポートしないオペレーティング・システムが併用される場合があります。この場合ユーザーは、REXX と NetView コマンド・リストの両方のバージョンの命令を持つバイリンガル・コマンド・リストを作成することができます。REXX が利用可能な場合、コマンド・リストは REXX で実行されます。利用可能でない場合、コマンド・リストは NetView コマンド・リスト言語で実行されます。

参照: REXX コマンド・リストのコンパイルについては、「*IBM Tivoli NetView for z/OS プログラミング: REXX および NetView コマンド・リスト言語*」を参照してください。

バイリンガル・コマンド・リストの詳細については、「*IBM Tivoli NetView for z/OS プログラミング: REXX および NetView コマンド・リスト言語*」を参照してください。

機能による言語の選択

表 2 は、使用する言語を選択するときにさらに考慮すべき機能についてリストしています。

表 2. 機能による言語の選択

機能	REXX または NetView CLIST	PL/I または C	アセンブラー
回線モードで NetView オペレーターにメッセージを送る	有	有	有
NetView オペレーターの画面を介してオペレーターと対話する (PAUSE/GO コマンド)	有	有	NO
NetView コマンドを呼び出す	有	有	困難
オペレーターに宛先指定されたメッセージをトラップし、処理する	有	有	困難
タスクおよび共通グローバル変数にアクセスする	有	有	有
名前付きのストレージ域を作成し、そこにアクセスする	REXX では有。CLIST では無。	有	有
フルスクリーン・パネルを介してオペレーターと対話する	VIEW コマンド使用	VIEW コマンド使用	困難
通信ネットワーク管理インターフェースによって非 SPCI データと通信する	NO	有	有

表 2. 機能による言語の選択 (続き)

機能	REXX または NetView CLIST	PL/I または C	アセンブラー
DASD または VSAM ファイルにアクセスする **注:** PIPE コマンドにはディスクから読み取る機能があります。DSIVSAM および DSIVSMX が VSAM ファイルへのアクセスを提供します。	有	有	有
プログラム・デバッグ・サポートを提供する	有	有	NO
NetView インストール・システム出口をインプリメントする	NO	ほとんど可能	有
NetView 制御ブロックにアクセスする	NO	NO	有

参照: いろいろな言語で作成する場合の考慮事項については、個々のプログラミング言語の資料を参照してください。

ロギング

NetView プログラムには、情報をログに記録する方法が何種類かあります。 表 3 は、一般のログ方式で利用できる機能のリストです。

表 3. NetView でのログ方式の機能

機能	ネットワーク・ログ	外部 SMF ログ	外部ユーザー定義ログ	NetView 順次ログ
アクセス方式	VSAM	VSAM	順次	BSAM
装置独立の機能	NO	NO	有	有
機能の提供	すべてのオペレーター端末活動の記録	サービス・レベルの確認と会計	ユーザー定義の機能	ユーザー定義機能のための基本サービス
API–PL/I および C *	CNMSMSG	CNMSMSG	CNMSMSG	CNMSMSG
API アセンブラー	DSIWLS	DSIWLS	DSIWLS	DSIWLS
記録の開始	START	*IBM Tivoli NetView for z/OS インストール:追加コンポーネントの構成*を参照してください。	*IBM Tivoli NetView for z/OS インストール:追加コンポーネントの構成*を参照してください。	*IBM Tivoli NetView for z/OS インストール:追加コンポーネントの構成*を参照してください。
ブラウズ	NetView の BROWSE	NO	オペレーティング・システムのブラウズ	オペレーティング・システムのブラウズ
複数のログ・タスク	NO	NO	NO	有

表 3. NetView でのログ方式の機能 (続き)

機能	ネットワーク・ログ	外部 SMF ログ	外部ユーザー定義ログ	NetView 順次ログ
可変長のブロックおよびレコード	NO	有	有	有
1 次/2 次データ・セットまたはファイル	有	システム制御	NO	有
SWITCH、RESUME、AUTOFLIP	有	N/A	NO	有
インストール・システム出口	多数	XITXL	XITXL	XITBN、XITBO

参照: ネットワーク・ログについては、「*IBM Tivoli NetView for z/OS 自動操作ガイド*」を参照してください。システム管理機能 (SMF) を使用する外部ログ、ユーザー定義のログ、または順次ログについては、「*IBM Tivoli NetView for z/OS インストール:追加コンポーネントの構成*」を参照してください。

メッセージおよび環境機能の相互参照

21 ページの表 4、 22 ページの表 5、および 23 ページの表 6 は、NetView システム・データ、タスク・データ、およびメッセージ機能の相互参照テーブルです。このマトリックスを参照すると、使用する機能が自動化テーブル、REXX、NetView コマンド・リスト言語、またはアセンブラーで利用可能かどうかを判断することができます。機能の名前も判別することができます。各マトリックスは、REXX 機能の名前で英字順に配列されています。

注:

1. アセンブラー言語コマンド・プロセッサーを作成する場合、正確なフィールド定義のために、DSITIB マッピング・マクロ、DSIIFR マッピング・マクロ、および DSIAIFRO マッピング・マクロ内の BUFHDR マッピングを調べるときは、「*IBM Tivoli NetView for z/OS プログラミング:アセンブラー*」を参照してください。
2. コマンド・リストを作成する場合、NetView コマンド・リスト言語制御変数および REXX 機能の詳細については、「*IBM Tivoli NetView for z/OS プログラミング: REXX および NetView コマンド・リスト言語*」を参照してください。
3. PL/I または C 言語で作成する場合、CNMINFC、CNMINFI、および CNMGETA サービス・ルーチンの詳細については、「*IBM Tivoli NetView for z/OS プログラミング: PL/I および C*」を参照してください。
4. 自動化テーブル・ステートメントを作成する場合、自動化テーブルの条件項目の説明は、「*IBM Tivoli NetView for z/OS 自動操作ガイド*」を参照してください。

表 4. 自動化変数の相互参照テーブル (システム・データ用): 戻されるデータは、システムに関するものです。すべてのタスクのすべてのメッセージに同じデータが戻されます。

REXX 機能	説明	自動化 テーブルの 条件 項目	HLL サービス・ ルーチンおよび オプション	制御 ブロック・ フィールド
ASID()	NetView アドレス・スペース ID	利用不可	CNMINFI ASID	ASCBASID
CURSYS()	現行 z/OS システム名	CURSYS	CNMINFC CURSYS	CVTSNAME (MVS)
Date (USA)	現行日付	利用不可	CNMINFC DATE	
DOMAIN()	現行ドメイン名	DOMAIN	CNMINFC DOMAIN	MVTCURAN
ECVTPSEQ()	製品のシーケンス番号	ECVTPSEQ	CNMINFC ECVTPSEQ	IHAECVT (MVS)
MVSLEVEL()	現行 z/OSシステム・レベル	MVSLEVEL	CNMINFC MVSLEVEL	CVTPRODN (MVS)
NETID()	VTAM ネットワーク ID	NETID	CNMINFC NETID	ACB ベクトル
NETVIEW()	NetView のバージョンおよびリリースの ID	NETVIEW	CNMINFC NVVER	MVTVER
OPSYSTEM()	NetView プログラムのコンパイル対象となったオペレーティング・システム	OPSYSTEM	CNMINFC OPSYSTEM	DSISYS コンパイラー変数
STCKGMT() 8 バイトの値	グリニッジ標準時の保管時間値	利用不可	CNMINFC CLOCK 8 バイトの値	
SUPPCHAR()	コマンド・エコーまたはコマンドのメッセージ出力を抑止する、NetView プログラムの抑止文字	利用不可	CNMINFC SUPPCHAR	MVTSPCHR
SYSPLEX()	コマンド・リストを実行中の z/OS SYSPLEX の 1 から 8 文字の名前	SYSPLEX	CNMINFC SYSPLEX	ECVTSPLX
TIME (オプション)	現行時刻	利用不可	CNMINFC TIME	
VTAM()	アクティブな場合 VTAM レベル	VTAM	CNMINFC VTAM	ACB ベクトル MVTACB ACBOPEN
VTCOMPID()	VTAM コンポーネント ID	VTCOMPID	CNMINFC VTCOMPID	ACB ベクトル MVTACB ACBOPEN
WEEKDAYN()	曜日を表す 10 進数	WEEKDAYN	CNMINFI WEEKDAYN	

表 5. 自動化変数の相互参照テーブル (システム・データ用): 戻されるデータは、システムに関するものです。すべてのタスクのすべてのメッセージに同じデータが戻されます。

REXX 機能	説明	自動化 テーブルの 条件 項目	HLL サービス・ ルーチンおよび オプション	制御 ブロック・ フィールド
	NetView プログラムの終了インディケーター	NVCLOSE	CNMINFI CLOSING	MVTCLOSE
APPLID()	現行タスクのアプリケーション名	利用不可	CNMINFC APPLID	TVBAPID
ARG()	アクティブ・コマンド・リストの入力パラメーター	利用不可	利用不可	
ATTENDED()	タスク情報	ATTENDED	CNMINFI ATTENDED	TVBSYSCN TVBAUTOO TVBDAUT
AUTCONID()	自動タスクと関連のある MVS コンソール名。この MVS コンソールは、この自動タスクのもとで実行する NetView コマンドを発行することができます。	利用不可	CNMINFC AUTCONID	TVBSYSCN TVBCNAME
AUTOTASK()	自動タスク・インディケーター	AUTOTASK	CNMINFI AUTOTASK	TVBAUTOO
COMPNAME()	コマンド・リストが呼び出されたときにアクティブであったコンポーネント名	利用不可	利用不可	
CURCONID()	NetView タスクが MVS コマンドを出したり MVS メッセージを受信したりするときに使用する MVS コンソール名	利用不可	CNMINFC CURCONID	TVBMCSNU TVBMCSNA
DISTAUTO()	分散自動タスク・インディケーター	DISTAUTO	CNMINFI DISTAUTO	TVBDAUT
HCOPY()	このタスクへのハードコピー・タスク	利用不可	CNMINFC HCOPY	TVBHCTVB -> TVBOPID
LU()	現在実行中のタスクの端末名	利用不可	CNMINFC LU	TVBLUNAM
NVCNT()	使用可能なドメインの数	利用不可	利用不可	
NVID(n)	ドメイン ID の配列	利用不可	利用不可	
NVSTAT(名前)	ドメインの状況	利用不可	利用不可	
OPID()	現在実行中のタスクの ID	OPID	CNMINFC OPID、または CNMINFC TASKNAME	TVBOPID
PARMCNT()	アクティブ・コマンド・リストに含まれる入力パラメーターの数	利用不可	利用不可	

表 5. 自動化変数の相互参照テーブル (システム・データ用) (続き): 戻されるデータは、システムに関するものです。すべてのタスクのすべてのメッセージに同じデータが戻されます。

REXX 機能	説明	自動化テーブルの条件項目	HLL サービス・ルーチンおよびオプション	制御ブロック・フィールド
TASK()	タスクのタイプ	TASK	CNMINFC TASK	DSITVB 内の CBHTYPE
WTO.REPLY	WTOR 応答テキスト	利用不可	利用不可	

表 6. 自動化変数の相互参照テーブル (メッセージ・データ用): データは各メッセージまたは MSU ごとに異なります。メッセージ ID とはメッセージ・データのことです。

REXX 関数および変数	説明	自動化テーブルの条件項目	HLL サービス・ルーチンおよびオプション	制御ブロック・フィールド
	1 から 1100 バイトのソース・オブジェクト	利用不可	CNMGETA MSGSRCOB	MSODATA MSOLEN
ACTIONDL()	メッセージ削除の理由	ACTIONDL	CNMCAGA ACTIONDL	IFRAUDLO IFRAUDTO IFRAUNVD IFRAUDFL IFRAUDF2
ACTIONMG()	アクション・メッセージ	ACTIONMG	CNMCAGA ACTIONMG	IFRAUACN
AREAID()	MVS 区域 ID	AREAID	CNMGETA AREAID	IFRAUWMA CPOCAREA MDBCAREA
AUTOTOKE()	MPF 自動化トークン 1 から 8 文字またはヌル	AUTOTOKE	CNMGETA AUTOTOKE	IFRAUTOK CPOCAUTO MDBCAUTO
CART()	8 バイトのコマンドおよび応答トークン	CART	CNMGETA CART	CPOCCART MDBCCART
DESC()	2 バイトの MVS 記述子コード	DESC	CNMGETA DESC	IFRAUWDS CPOCDESC MDBCDESC
GETMLINE コマンド	メッセージ・テキスト	TEXT	CNMGETD GETFIRST または CNMGETD GETNEXT	

表 6. 自動化変数の相互参照テーブル (メッセージ・データ用) (続き): データは各メッセージまたは MSU ごとに異なります。メッセージ ID とはメッセージ・データのことです。

REXX 関数および変数	説明	自動化テーブルの条件項目	HLL サービス・ルーチンおよびオプション	制御ブロック・フィールド
GETMPRES コマンド	4 バイトの表示属性 この情報は、IFRAUTBA からチェーニングされたテキスト・バッファーに含まれる。	LINEPRES LINEPRES はメッセージの最初の行の表示特性を戻す。	利用不可	HDRTMTPA MDBTMTPA
GETMSIZE コマンド	メッセージの行数の 2 バイト・カウント CPOCLCNT 内の値は、メッセージ内のバッファーの実際の数を反映していないことがある。したがって、アセンブラー・コマンド・プロセッサーは、IFRAUTBA チェーンでバッファーの数をカウントする必要がある。	利用不可	利用不可	CPOCLCNT MDBCLCNT
GETMTFLG コマンド	2 バイトのテキスト・オブジェクト・フラグ この情報は、IFRAUTBA からチェーニングされたテキスト・バッファーに含まれる。	LINETFLG LINETFLG はメッセージの最初の行のオブジェクト・タイプ・フラグだけを戻す。	利用不可	HDRTLNTY MDBTLNTY
HDRMTYPE()	NetView メッセージ・タイプ	HDRMTYPE	ORIG_MSG_TYPE ORIG_MSG_TYPE には、CNMGETD が出された後だけにメッセージ・タイプが入る。	HDRMTYPE
IFRAUGMT()	AIFR が作成された場合 8 バイト 16 進の保管時間値	なし	CNMGETA IFRAUGMT	
IFRAUIND()	2 バイトの自動化 IFR インディケーター・フラグ	IFRAUIND(nn)	CNMGETA IFRAUIND	IFRAUIND
IFRAUIN3()	1 バイトのインディケーター・ビット	IFRAUIN3(nn)	CNMGETA IFRAUIN3	IFRAUIN3
IFRAUI3X()	最初の 8 ビットが IFRAUIN3 である 32 ビットのフィールド	IFRAUI3X	CNMCAGA IFRAUI3X	IFRAUI3X
IFRAUNVF	MVS 保存フラグ	MVSRTAIN	CNMGETA MVSRTAIN	IFRAUNVF

表 6. 自動化変数の相互参照テーブル (メッセージ・データ用) (続き): データは各メッセージまたは MSU ごとに異なります。メッセージ ID とはメッセージ・データのことです。

REXX 関数および変数	説明	自動化テーブルの条件項目	HLL サービス・ルーチンおよびオプション	制御ブロック・フィールド
IFRAUSDR()	メッセージまたは MSU の元の送信側。しかし HDRSENDR は信頼できない。	IFRAUSDR	CNMGETA IFRAUSDR	IFRAUSDR
IFRAUSRB() IFRAUSB2()	AIFR からの 2 バイトのユーザー・フィールド。このユーザー・フィールドは、ビットまたは文字として参照できる。	IFRAUSRB(nn), IFRAUSB2(n)	CNMGETA IFRAUSRB, CNMGETA IFRAUSB2	IFRAUSRB
IFRAUSRC() IFRAUSC2()	AIFR からの 16 バイトのユーザー・フィールド。このユーザー・フィールドは、ビットまたは文字として参照できる。	IFRAUSRC, IFRAUSC2	CNMGETA IFRAUSRC、 CNMGETA IFRAUSC2	IFRAUSRC
IFRAUTA1()	6 バイトの制御フラグ	IFRAUTA1(nn)	CNMGETA IFRAUTA1	IFRAUTA1
IFRAUWF1()	4 バイトの MVS 特定 WQE フラグ	IFRAUWF1(nn)	CNMGETA IFRAUWF1	IFRAUWF1
JOBNAME()	8 バイトの MVS ジョブ名	JOBNAME	CNMGETA JOBNAME	IFRAUWJA GOJGJBNM MDBGJBNM
JOBNUM()	8 バイトの MVS ジョブ番号	JOBNUM	CNMGETA JOBNUM	IFRAUWJU CPOCOJID MDBCOJID
KEY()	メッセージと関連のある 8 バイトのキー	KEY	CNMGETA KEY	CPOCKEY MDBCKEY
LINETYPE() GETMTYPE コマンド	メッセージ MLWTO インディケーター	利用不可	ORIG_LINE_TYPE ORIG_LINE_TYPE には、CNMGETD が出された後だけに行タイプが入る。	HDRLNTYP IFRAUWF1(3) HDRTTYPE MDBTTYPE

表 6. 自動化変数の相互参照テーブル (メッセージ・データ用) (続き): データは各メッセージまたは MSU ごとに異なります。メッセージ ID とはメッセージ・データのことです。

REXX 関数および変数	説明	自動化テーブルの条件項目	HLL サービス・ルーチンおよびオプション	制御ブロック・フィールド
MCSFLAG()	2 バイトの MVS MCS フラグ コマンド・リスト、PL/I、および C では、MCSFLAG は 8 つの MCSFLAG ビットの選択値を戻す。自動化テーブルでは、MCSFLAG がアセンブラー制御ブロック・フィールドに一致する 16 ビットを戻す。	MCSFLAG	CNMGETA MCSFLAG	IFRAUMCS
MSGASID()	z/OS システム・アドレス・スペース ID	利用不可	CNMGETA MSGASID	IFRAUASI IFRAUWAS CPOCASID MDBCASID
MSGAUTH()	MVS システム・メッセージが許可プログラムによって発行されたかどうかを示す。	MSGAUTH	CNMGETA MSGAUTH	CPOCAUTH MDBCAUTH
MSGCATTR()	2 バイトの MVS メッセージ属性フラグ	MSGCATTR	CNMGETA MSGCATTR	CPOCATTR MDBCATTR
MSGCMISC()	1 バイトの MVS の種々のルーティング情報フラグ	MSGCMISC	CNMGETA MSGCMISC	CPOCMISC MDBCMISC
MSGCMLVL()	2 バイトの MVS メッセージ・レベル・フラグ	MSGCMLVL	CNMGETA MSGCMLVL	CPOCMLVL MDBCAUTH
MSGCMSGT()	2 バイトのメッセージ・タイプ・フラグ	MSGCMSGT	CNMGETA MSGCMSGT	CPOCMSGT MDBCMSGT
MSGCNT()	メッセージ内のトークンの数	利用不可	利用不可	
MSGCOJBN()	8 文字の元のジョブ名	MSGCOJBN	CNMGETA MSGCOJBN	CPOCOJBN MDBCOJBN
MSGCPROD()	メッセージを出したシステムの MVS システム・プロダクトのレベル	MSGCPROD	CNMGETA MSGCPROD	CPOCPROD MDBCPROD
MSGCSPLX()	受信されたメッセージが発信された MVS SYSPLEX の 1 から 8 文字の名前	MSGCSPLX	CNMGETA MSGCSPLX	CPOCSPLX
MSGCSYID()	10 進のシステム ID (DOM の場合)	利用不可	CNMGETA MSGCSYID	CPOCSYID MDBCSYID

表 6. 自動化変数の相互参照テーブル (メッセージ・データ用) (続き): データは各メッセージまたは MSU ごとに異なります。メッセージ ID とはメッセージ・データのことです。

REXX 関数および変数	説明	自動化テーブルの条件項目	HLL サービス・ルーチンおよびオプション	制御ブロック・フィールド
MSGDOMFL()	1 バイトの DOM フラグ	MSGDOMFL	CNMGETA MSGDOMFL	CPODOMFL MDBDOMFL
MSGGBGPA()	4 バイトの背景表示属性	MSGGBGPA	CNMGETA MSGGBGPA	GOJGBGPA MDBGBGPA
MSGGDATE()	*yyyyddd* 形式の 7 文字の日付	MSGGDATE	CNMGETA MSGGDATE	GOJGDSTP MDBGDSTP
MSGGFGPA()	4 バイトの前景表示属性	MSGGFGPA	CNMGETA MSGGFGPA	GOJGFGPA MDBGFGPA
MSGGMFLG()	2 バイトの MVS 汎用メッセージ・フラグ	MSGGMFLG	CNMGETA MSGGMFLG	GOJGMFLG MDBGMFLG
MSGGMID()	4 バイトの MVS メッセージ ID フィールド	MSGGMID	CNMGETA MSGGMID	GOJGMID MDBGMID
MSGGSEQ()	MVS メッセージの順序番号。順序番号は MSGGSYID とともに使用され、MSGGMID を判別する。	利用不可	CNMGETA MSGGSEQ	GOJGSEQ
MSGGSYID()	メッセージが発行された MVS システムのシステム ID	利用不可	CNMGETA MSGGSYID	GOJGSYID MDBGSYID
MSGGTIME()	11 バイトの時刻 *hh.mm.ss.th* 文字ストリング	MSGGTIME	CNMGETA MSGGTIME	GOJGTIMH MDBGTIMH GOJGTIMT MDBGTIMT
MSGID()	メッセージ ID は、常にメッセージの最初の項目というわけではない。例えば、メッセージが WTOR の場合、REPLYID はメッセージ ID に先行する。	MSGID	ORIG_PROCESS ORIG_PROCESS には、CNMGETD が出された後にのみメッセージが入る。	
MSGORIGN()	メッセージ・ドメイン名 (または TAF セッション名のこともある)。これは、常に AIFR バッファーにドメイン名を戻す。	DOMAINID	ORIG_DOMAIN ORIG_DOMAIN には、CNMGETD が出された後にのみドメイン名が入る。	HDRDOMID
MSGSRCNM()	ソース・オブジェクトからの 1 から 17 文字のソース名	MSGSRCNM	CNMGETA MSGSRCNM	MSOSUBDA MSOSBNIK MSOSBNID MSOSBNAU

表 6. 自動化変数の相互参照テーブル (メッセージ・データ用) (続き): データは各メッセージまたは MSU ごとに異なります。メッセージ ID とはメッセージ・データのことです。

REXX 関数および変数	説明	自動化テーブルの条件項目	HLL サービス・ルーチンおよびオプション	制御ブロック・フィールド
MSGSTR()	メッセージ ID の後のメッセージのテキスト	利用不可	CNMGETD GETFIRST または CNMGETD GETNEXT	
MSGTOKEN()	メッセージと関連のある数値トークン	利用不可	CNMGETA MSGTOKEN	CPOCTOKN MDBCTOKN
MSGTSTMP()	メッセージ・タイム・スタンプ	利用不可	CNMGETA MSGTSTMP	HDRTSTMP
NVDELID()	NetView DOM ID	NVDELID	CNMCAGA NVDELID	IFRAUGMT HDRDOMID
MSGVAR(n)	メッセージのトークン コマンド・リストでは、メッセージ ID の後のトークンが最初のトークンとして戻される。自動化テーブルでは、メッセージ ID が最初のトークンとして戻される。	TOKEN	CNMGETD GETFIRST または CNMGETD GETNEXT	
PARTID()	ある VSE メッセージの場合に VSE 区分 ID を示す、VSE メッセージ接頭語の最初の 2 文字。	PARTID	CNMGETA PARTID	
PRTY()	2 バイトの MVS メッセージ優先順位	利用不可	CNMGETA PRTY	CPOCPRTY MDBCPRTY
REPLYID()	応答 ID	利用不可	CNMGETA REPLYID	CPOCRPYI MDBCRPYI CPOCRPYB MDBCRPYB
ROUTCDE()	16 バイトの MVS 宛先コード (128 ビット)	ROUTCDE	CNMGETA ROUTCDE	IFRAUWRT CPOCERC MDBCERC
SESSID()	TAF セッション名	SESSID	CNMGETA SESSID	IFRAUTAF
SMSGID()	DOM 相関に関する MVS メッセージ ID	利用不可	CNMGETA SMSGID	IFRAUWID
SYSCONID()	メッセージと関連のある MVS コンソール名	SYSCONID	CNMGETA SYSCONID	IFRAUWUC IFRAUCON CPOCCNID MDBCCNID
SYSID()	メッセージと関連のある 8 バイトの z/OS システム名	SYSID	CNMGETA SYSID	IFRAUWSN GOJGOSNM MDBGOSNM

PF キーおよび即時メッセージ行のカスタマイズ

PF キー行で、検索し、置くことができるグローバル変数を、BROWSE、STATMON、および VIEW コマンドにより表示されるパネル上に設定することができます。VIEW パネルでは、即時メッセージ行を PF キー行として使用することもできます。変数名には接頭部 (&)CNMIM が付けられ、その後にアプリケーション名が続きます。有効な変数としては、CNMIMLBROWSE、CNMIMMBROWSE、CNMIMSTATMON、CNMIMVIEW、および CNMIMWINDOW などがあります。

ビュー・パネルでは、VIEW アプリケーションで CNMIMDL に値が指定されなかった場合、VIEW がグローバル辞書 (タスク、次に共通) を検索して、CNMIM*xxx* という変数を探します。ここで、*xxx* は、VIEW が呼び出されたときに指定されたアプリケーション名です。変数 CNMIM*xxx* が見つからなかった場合、VIEW は同じ辞書で CNMIMVIEW を検索します。これは、VIEW アプリケーションに対するキーの設定方法と同様です。最終的にこれらの変数が 1 つもない場合は、メッセージ BNH257I のテキストを使用します。

CNMKEYS の変更

```
------------------- DEFINE TEXT FOR KEY LINES ------------------
*
*  The separator line above is required in any key definition file
*  which defines "key line" texts.  This separator line MUST begin
*  with 9 dashes.  All key definitions must precede this line, and
*  all "key line" definitions must follow it.
*
*  Optionally uncomment and modify the following statements, which
*  assign values to the "key line" area of Statmon, Browse and View
*  panels.  The same rules are followed in this section as above with
*  respect to commas and continuation lines.  Keep the variable name
*  between the delimiters, and PFKDEF will assign the rest of the line
*  (including continuations) to that variable.  Do not use leading
*  blanks.
*
*/CNMIMSTATMON/1=HLP 2=END 3=RET 4=KYS 5=LOG 6=,
*ROL 7=BCK 8=FWD 9=SR  10=SV  11=SC  12=RTV
*/CNMIMLBROWSE/1=HLP 2=END 3=RET 4=KYS 5=RPF 6=,
*ROL 7=BCK 8=FWD 9=TOP 10=LFT 11=RGT 12=RTV
*/CNMIMMBROWSE/1=HLP 2=END 3=RET 4=KYS 5=RPF 6=,
*ROL 7=BCK 8=FWD 9=TOP 10=WIN 11=WHO 12=RTV
*/CNMIMVIEW/1=HLP 2=END 3=RET 4=KYS 5=LOG 6=,
*ROL 7=BCK 8=FWD 9=TOP 10=WIN 11=ENT 12=RTV
*/CNMIMWINDOW/1=HLP 2=RFR 3=RET 4=KYS 5=FIN 6=,
*ROL 7=BCK 8=FWD 9=TOP 10=LFT 11=RGT 12=RTV
```

図 3. PF キーを設定する CNMKEYS サンプルからの抜粋

PFKDEF コマンド・リスト (CNME1010) では、ターゲット・ファイルから得た 1 つ以上のタスク・グローバル変数を割り当て、該当する NetView アプリケーションのキー設定に一致させることができます。図 3 は、ブラウズ、状況モニター、およびビューのパネルに対する PF キーの設定方法を示しています。

第 2 章 NetView コマンド・ファシリティー・パネルのカスタマイズ

NetView コマンド・ファシリティー・パネルはカスタマイズすることができます。カスタマイズ可能な機能は以下のとおりです。

- パネル上のフィールドのカラー
- メッセージ・テキストに先行する情報
- 保留、アクション、通常、および即時クラスのメッセージのデフォルトのカラー
- コマンド域のカラー
- 保留メッセージおよびアクション・メッセージに取っておくパネル区域の量

画面フォーマット定義の使用

画面フォーマット (SCRNFMT) 定義を使用して、コマンド・ファシリティー・パネルの属性およびメッセージのカラーのデフォルト値を指定することができます。画面フォーマット定義を活動化するには、NetView の DEFAULTS および OVERRIDE コマンドを使用します。 DEFAULTS および OVERRIDE の使用方法について詳しくは、NetView オンライン・ヘルプを参照してください。画面フォーマット定義内に指定できる各オプションの短い説明が、 32 ページの『画面フォーマット定義ステートメント』にリストされています。

参照: 画面フォーマット定義ステートメントの詳細記述については、「*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。CNMSCNFT は、「*IBM Tivoli NetView for z/OS インストール:追加コンポーネントの構成*」で提供されているサンプルの画面フォーマット定義です。

注:

1. カラーと強調表示は、ハードウェアとエミュレーターによってサポートされていなければなりません。さらに、照会タイプのログモードで NetView システムにログオンする必要があります。
2. アクティブ画面フォーマット定義を新しい画面フォーマット定義に置き換える場合、すべての定義ステートメントが置き換えられます。新しい画面フォーマット定義で指定されないすべての定義ステートメントには、 NetView プログラム提供の値が使用されます。 NetView プログラム提供の各定義ステートメントの値は、「*IBM Tivoli NetView for z/OS アドミニストレーション・リファレンス*」にリストされています。

   例えば、画面フォーマット定義が DEFAULTS コマンドで活動化されているとします。続いて、オペレーターが OVERRIDE コマンドを使用して、カスタマイズ画面フォーマット定義を活動化します。このオペレーターの画面フォーマット定義で指定されなかったステートメントには、 DEFAULTS コマンドで活動化された画面フォーマット定義の値ではなく、 NetView プログラム提供の値が使用されます。

画面フォーマット定義ステートメント

以下の画面は、NetView メッセージ・パネルでカスタマイズできるフィールドを示しています。

図 4. NetView メッセージ・パネル :

NetView メッセージ・パネル

以下の形式をカスタマイズすることができます。

1 タイトル域

SCRNFMT 定義内の TITLE ステートメントを使用して、画面上の NETVIEW という単語のカラーをカスタマイズすることができます。

2 ドメイン ID

SCRNFMT 定義内の TITLEDOMID ステートメントを使用して、NetView ドメイン名のカラーをカスタマイズします。

3 オペレーター ID

SCRNFMT 定義内の TITLEOPID ステートメントを使用して、オペレーター名のカラーをカスタマイズします。

4 現行日付

SCRNFMT 定義内の TITLEDATE ステートメントを使用して、日付のカラーをカスタマイズします。DEFAULTS コマンドおよび OVERRIDE コマンドを使用して、日付の形式をカスタマイズすることもできます。

5 最後に表示された時刻のデータ

SCRNFMT 定義内の TITLETIME ステートメントを使用して、時刻のカラーをカスタマイズします。 DEFAULTS コマンドおよび OVERRIDE コマンドを使用して、時刻の形式をカスタマイズすることもできます。

6 および 7 システムの状態

SCRNFMT 定義内の TITLESTAT ステートメントを使用して、パネル上の右上隅の状況文字のカラーをカスタマイズすることができます。

8 COLUMNHEAD 行

SCRNFMT 定義内の COLUMNHEAD ステートメントを使用して、接頭部のラベルをもつ行を画面の上部に作成します。この行には、任意の順序で最

大 16 タグ (C1...C16) まで入れることができます。タグの全長は、各タグ間の 1 つずつのスペースを含み、78 文字を超えてはなりません。タグの設定には、SCRNFMT 定義を使用します。 PREFIX および NOPREFIX ステートメントは、現れるタグを制御します。画面に行が表示されないように選択することもできます。

9 出力域

SCRNFMT 定義の HELD、ACTION、NORMAL、および NORMQMAX ステートメントを使用します。

注: HELD、ACTION、および NORMAL ステートメントは、メッセージのデフォルトのカラーを設定します。メッセージのカラーがすでに設定されている場合、デフォルトのメッセージ・カラーは現れません。詳しくは、34 ページの『メッセージのカラーおよび強調表示』を参照してください。

NORMQMAX ステートメントは、後で表示されるように待機される通常メッセージの数を指定します (保留およびアクション・メッセージを除きます)。この場合の例は、他のパネルを作動させている間、またはパネルがロックされている間保持するメッセージの数です。

NORMQMAX を超過した場合、NetView プログラムは (必要なら) 入力メッセージを自動化し、ログに記録して、オペレーターの割り込みなしでそれらを廃棄します。待機メッセージの数が NORMQMAX 値の半分になるまで、最も古いメッセージから廃棄されます。

オペレーターがコマンド・ファシリティーに戻る (つまりパネルがアンロックされた) 場合、メッセージ DSI593A は、廃棄されたメッセージの数を示します。

NORMQMAX の値は、0 から 2147483647 までの範囲です。デフォルト値は 3000 です。使用できるメッセージの最小値は 100 であるため、100 より少ない数を指定した場合、100 になります。 NORMQMAX 値に 0 を指定すると、無限のキューを意味し、基本的に最大値 2147483647 を指定したのと同じになります。

考慮事項 : NORMQMAX の設定値が高すぎると、ストレージ不足の状態になることがあります。反対に、設定値が低すぎると、メッセージ・トラフィック率が低いときでも、オペレーターはすべてのメッセージを見ることができなくなります。

NORMQMAX 値は、ハードコピー・プリンターおよび OST-NNT クロスドメイン・セッションにも適用されます。ハードコピー・プリンターは、低速になったり用紙がなくなったりするとバックログになります。 OST-NNT セッションは、セッションを介するメッセージ・トラフィックがセッションの送信率を超えるとバックログになります。

10 保留およびアクション・メッセージの区域

SCRNFMT 定義内で HOLDPCNT ステートメントを使用します。 NetView プログラムは、タイトル行、即時メッセージ域、コマンド域、および警告保留メッセージ DSI151I 用に、画面の 10 行を使用します。保留メッセージは、この 10 行に表示されません。HOLDPCNT を使用して、保留メッセー

ジに使用したい残りの行 のパーセントを指定できます。例えば、24 行の画面上で、HOLDPCNT に 100% を設定すると 14 行が保留メッセージに与えられます。

HOLDPCNT に 0 を指定すると、画面に表示される保留メッセージがないことを意味します。HOLDPCNT がゼロ以外の場合、保留メッセージに使用される最小行数は 2 行です。

HOLDWARN を使用して、画面に表示されないが保留メッセージがあるという警告メッセージを得ることができます。

注: NetView プログラムは、メッセージのデータ行がない保留メッセージの制御行を表示しません。これは、オペレーターが誤ってテキストを見ないで保留メッセージを消してしまうことがないようにするためです。

11 字下げ
SCRNFMT 定義内の INDENT および MLINDENT ステートメントを使用します。

分離線 SCRNFMT 定義の LASTLINE ステートメントは、画面の新しいメッセージと古いメッセージの間のダッシュ分離線のカラーを変更します。

12 コマンド入力インディケーター
SCRNFMT 定義の CMDLINE ステートメントを使用します。

ロック/アンロック・インディケーター (***)
SCRNFMT 定義内の LOCKIND ステートメントを使用します。

13 即時メッセージ域
SCRNFMT 定義内の IMDAREA ステートメントを使用します。

14 コマンド域
SCRNFMT 定義内の CMDLINE ステートメントを使用して、コマンド入力域に使用するカラーを変更します。 INPUT コマンドでコマンド域のサイズを変更できます。

メッセージのカラーおよび強調表示

メッセージに関して次の 4 つのカラーおよび強調表示属性を設定できます。

- 前景色
- 背景色
- 輝度
- 強調表示

注: 背景色は、ほとんどの 3270 デバイスおよびエミュレーターでサポートされません。この場合、黒が背景色として使用されます。

メッセージのカラーおよび強調表示の属性は複数の場所で設定できます。

- 自動化テーブル内
- MVS MPF テーブル内 (MVS システム・メッセージの場合)
- インストール・システム出口内
- 画面フォーマット定義内

リストされたすべてのオプションの中で、画面フォーマット定義は最も低い優先順位を取ります。次の優先順位の規則が適用されます。

- MVS システム・メッセージに関する MPF テーブルのカラーの輝度と強調表示は、これらの属性に関する画面フォーマット定義をオーバーライドします。
- カラーの輝度と強調表示の自動化テーブルの指定は、次のものをオーバーライドします。
  - MPF テーブルで指定されたカラーの輝度と強調表示
  - カラーの輝度と強調表示の画面フォーマット定義
  - カラーの輝度と強調表示の DSIEX02A および DSIEX17 指定 (これらの出口は自動操作の前に駆動されます。)
- カラーの輝度と強調表示をインストール・システム出口で指定すると、これらの属性に関する MPF および画面フォーマット定義をオーバーライドします。さらに、インストール・システム出口 DSIEX16 (自動化後処理) は、自動化テーブルで指定されたカラーの輝度と強調表示をオーバーライドできます。

これらの各表示属性は、それぞれ独立して処理されます。例えば、カラーのアクションが自動化テーブルと一致する MVS システム・メッセージは、MPF テーブル内で指定された輝度と強調表示で表示されますが、カラーについては自動化テーブル内で指定された値が使用されます。

第 3 章 VIEW コマンドの使用

本章では、汎用プログラミング・インターフェースとそれに関連する情報について説明します。

VIEW コマンド・プロセッサーを用いて、ユーザー作成プログラムからフルスクリーン・パネルを表示することができます。 VIEW コマンドを使用することにより、ユーザーは独自のパネルを設計したり、パネル・テキストのカラーや強調表示を制御したりすることができます。

VIEW コマンドを使用すると、PL/I または C で書かれたコマンド・リストまたはコマンド・プロセッサーとオペレーターとが、フルスクリーン・パネルを使用して対話することができます。コマンド・リストや PL/I または C 変数からのデータは、パネルに置き換えることができます。

フルスクリーン・パネルの作成

オペレーターのためにパネルを作成するには、データ・セットまたはファイルにテキストおよびフォーマットを定義します。パネルのソースは、表示するパネルを定義する、1 つのプロローグと、テキストおよび変数で構成されます。 38 ページの図 5 は、ヘルプ・ソース・ファイル内の情報の例を示しています。図の中の番号の付いたフィールドの説明は、 38 ページの『一般ヘルプ・フィールド』にあります。

ご使用の画面が多くの行またはメッセージで構成されている場合、フルスクリーン・パネルに WINDOW コマンドを使用する方が簡単な場合があります。WINDOW を使用すると、その画面を変更し、サブコマンドを定義または指定変更することができます。詳細については、WINDOW のオンライン・ヘルプを参照してください。

NetView プログラムにより、VIEW コマンドを使用する多数のコマンド・リストが提供され、フルスクリーン・パネルが表示されます。コマンド・リストから VIEW を呼び出して新しいパネルを表示するには、既存のコマンド・リストを変更するか、あるいは新しいコマンド・リストを作成する必要があります。 IBM 提供のコマンド・リストを変更するときは、まずそれをユーザー・データ・セットにコピーしてから、その名前を変更してください。

```
/*******************************************************************************
/*  (C) COPYRIGHT IBM CORP. 2011 1
/*  DESCRIPTION: MENU FOR NCCF INFORMATION
/*  CHANGE ACTIVITY:
/*******************************************************************************
HELP=CNM5H000  help panel title 2
1 CNM1OVER Cmd Facility Overview
2 CNMKTAAF TAF Help
3 CNMKNCSC Using NCCF Screens 3
4 CMD='HELP NCCF COMMANDS'
5 CNMZZZZZ Field Level Help
*** 4
+CNMKNCCF  5      %COMMAND FACILITY HELP MENU 6
$
$
¥Select+ To get information about
$
$  %1   $Operator's overview of the command facility
$  %2   $Using the terminal access facility (TAF)
$                                                          7
$  %3   $The command facility screen
$  %4   $Command facility commands and command lists
$
$  %5   $Field level help
$
$
$
+Type a number (1 through 5) and press ENTER.
$
$
%                      HELP NETVIEW --->$NetView Help Menu
$
$
$
&CNMIMDL 8
%Action===>~&CUR 9
```

図 5. 一般ヘルプ情報のソースの例:

一般ヘルプ情報のソースの例

一般ヘルプ・フィールド

ソース・ファイルにあるドル記号 ($) やパーセント記号 (%) などの特殊文字については、 45 ページの『フィールドのカラーと強調表示の制御』に説明があります。

1 プロローグ

プログラマーのコメントを入れるオプショナル・セクションです。プロローグの各行は、1 桁目と 2 桁目に /* を入れて始めます。コメントを入れることができるのは、このセクションだけです。コメントが Help (ヘルプ) または Option Definitions (オプション定義) のセクションに表示される場合は、83 の戻りコードが戻され、パネルは表示されません。これらのセクションの後に表示されるコメントは、データとして扱われます。

2 ヘルプ

パネルのオプショナル定義です。このフィールドはプロローグに続けて、次のようにコーディングします。

```
Column
1               15
HELP=helppan    comment
```

注: HELP CMD='command_text' を使用することもできます。以下の **3** の説明を参照してください。

3 オプション定義

オペレーターが選択できる選択項目のオプション・リストです。このリストには、パネル名またはコマンドを入れることができます。パネル名またはコマンドの後にオプションでコメントを 1 つ追加することができます。パネル名またはコマンドとコメントの間は、少なくとも 1 つのブランクで区切る必要があります。このリストは 49 エントリーを超えてはなりません。リストは次の形式でコーディングします。

```
Column
1 3
n panel_name or CMD='command_text'  comment
```

ここで *n* は、パネルを呼び出すとき、またはコマンドを発行するときにオペレーターが入力する文字です。

後続パネルを作成するには、次のように *n* をブランクにします。

```
Column
1 3
  panel_name  comment
```

この場合、後続パネルは panel_name で識別されます。

4 テキスト・インディケーター

プロローグ、ヘルプ、およびパネル定義と、表示されるパネル・テキストを分離するために必要な 3 つのアスタリスクです。これらのアスタリスクの後に、以下のオプションを続けることができます。これらのオプションの順序は任意ですが、1 つ以上のブランクで各オプションを区切る必要があります。

- AT1 オプションは、カラーおよび強調表示属性のための属性セット 1 です。詳細については、 40 ページの表 7 および 46 ページの表 11 を参照してください。
- AT2 オプションは、カラーおよび強調表示属性のための属性セット 2 です。詳細については、 40 ページの表 7 および 46 ページの表 11 を参照してください。
- SFD (画面フォーマット・デフォルト) オプションは、VIEW パネルのあるフィールドに対してカラーまたは強調表示のいずれかが指定されているか、あるいはデフォルトの X'00' (3270 用のデフォルト) に設定されている場合、DEFAULTS SCRNFMT コマンドまたは OVERRIDE SCRNFMT コマンドで NCCF 画面に指定されたカラーおよび強調表示が使用されることを意味します。SFD が指定されていない場合、またはアクティブな SCRNFMT メンバーが有効でない場合、X'00' がデバイスに送られます。VIEW パネル・フィールドが入力コマンド行として解釈される場合、SCRNFMT CMDLINE で指定されたカラーおよび強調表示が使用されます。その他のフィールドに対しては SCRNFMT NORMAL 指定が使用されます。サンプルの CNMSCNFT には、追加情報が含まれています。
- XVAR オプションは、ピリオドを含めて 31 文字まで入れることができる変数を提供します。

このオプションを使用しないと、変数は最大 11 文字までしか入れることができず、ピリオドを含むことはできません。XVAR オプションの詳細については、表 7 および 53 ページの『複合シンボル』を参照してください。

- OPTROW=*optchar* オプションを使用すると、*optchar* によって定義された文字で始まる行をオプション行に指定することができます。オプション行の最大数は、端末でサポートされる行数から 24 (ゼロの場合もあります) を引いた数で定義されます。この最大数を超えてパネルに定義されたオプション行は、表示されません。また、端末の行限界を超える行 (通常行またはオプション行) も表示されません。

  オプション行では、すべての文字は *optchar* を補正するために左方に 1 つ位置がずらされ、最後の位置 (80 桁目) はブランクとして扱われます。

  OPTROW の使用方法の例については、WINDOW コマンド・リスト (CNME1505) とそのビュー・パネル CNMKWIND を参照してください。
- WIDE オプションを使用すると、81 桁以上をサポートする端末で全行幅が使用できるようになります。WIDE が指定されると、各行で最後に指定されたブランク以外のパネル変数は、その行幅に収められる限り置き換えられます。この変数は、端末で定義された行末まで切り捨てられません。

  WIDE の使用方法の例については、WINDOW コマンド・リスト (CNME1505) とそのビュー・パネル CNMKWIND を参照してください。

表 7. テキスト・インディケーター・オプションを使用する例

コード	結果
*** AT1	• 属性セット 1 • 英語 • 11 文字変数名、ピリオドなし

3 つのアスタリスクの後に AT2 オプションが続く場合は、属性セット 2 がカラーと強調表示のために使用されます。例えば、次のようにします。

- 英語の場合は、`*** AT2`
- 属性セット 1 の場合は、*** または `*** AT1` を使用します。

属性セット 1 で、変数が 31 文字の場合、英語では `*** AT1 XVAR` を使用します。

属性セット 1 および 2 の詳細については、45 ページの『フィールドのカラーと強調表示の制御』を参照してください。

5 名前
パネルの名前です。

6 ヘッダー
パネルの用途を説明するテキストです。

7 パネル・テキスト

表示されるパネルを構成する最大 24 行のテキストです。『テキスト・インディケーター』で説明されている OPTROW オプションも参照してください。

コマンド・リスト変数は、パネル・テキストの任意の位置に表示することができます。詳しくは、50 ページの『ソース・パネルの変数の表示』を参照してください。

8 メッセージ域

変数 &CNMIMDL を指定すると、NetView のエラー・メッセージがパネルの 23 行目に表示されます。アプリケーションで CNMIMDL 値が指定されていなかった場合、VIEW がグローバル辞書 (タスク、次に共通) を検索して、CNMIM*xxx* という変数を探します。ここで、*xxx* は VIEW が呼び出されたときに指定されたアプリケーション名です。この変数が見つからなかった場合、VIEW は同じ辞書で CNMIMVIEW を検索します。最終的に、これらの変数が 1 つもない場合は、メッセージ BNH257I のテキストが表示されます。BNH257I のデフォルトの英文テキストは、『TO SEE YOUR KEY SETTINGS, ENTER 'DISPFK'』です。このメッセージ BNH257I のテキストは、メッセージ変換テーブルで変更することができます。

66 ページの『PF キーと VIEW サブコマンドの使用』に記載されている PF キーに割り当てられるサブコマンドのリストと、29 ページの『PF キーおよび即時メッセージ行のカスタマイズ』を参照してください。

9 コマンド行

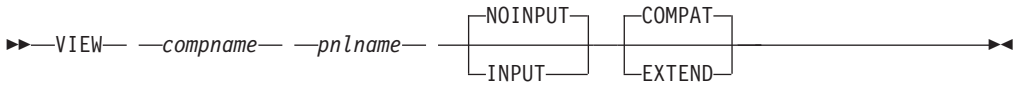
NetView コマンドはコマンド行に入力します。NOINPUT オプションが指定されている VIEW コマンドには、波形 (~) 属性シンボルによりコマンド行が定義されます。&CUR オプションは、コマンド行内のカーソル位置を示します。パネルごとに 1 つの入力フィールドと 1 つの &CUR オプションのみが処理されます。このオプションは、コマンドを入力フィールドに事前定義する場合に便利です。このオプションを使用しない場合は、カーソルのデフォルト値は次の順序になります。

1. 'UY' を指定した最後の属性変数
2. 最初の波形フィールド (存在する場合)
3. 左上隅の最初の位置

VIEW コマンドのコーディング

VIEW コマンドは次のようにコーディングしてください。

VIEW



項目の意味:

compname

NetView プログラムにより PF キー定義と一緒に使用される名前 (1 から 8 文

字) を指定します。最初の文字は英字でなければなりません。ロール可能なアプリケーションごとに異なる名前を使用する必要があります。

pnlname
表示されるパネルの名前 (1 から 8 文字) を指定します。

NOINPUT
これを指定すると、VIEW コマンドは、呼び出したプロシージャーに情報を戻しません。デフォルト値は、NOINPUT です。パネルでコマンド行を定義した場合は、NetView プログラムは入力データをコマンドとして扱います。NOINPUT オプションを使用すれば、UNIQUE コマンドを呼び出すためのコマンド・プロシージャーは不要です。

NOINPUT を指定したときに NetView プログラムが提供する PF キーについては、38 ページの図 5 を参照してください。

INPUT
これを指定すると、入力値および AID 情報が、VIEW コマンドを呼び出しているプロシージャーに戻されます。INPUT を指定すると、さらに、カーソル位置も、VIEW コマンドを呼び出しているプロシージャーから受け取られ、そのプロシージャーに戻されます。INPUT オプションで VIEW コマンドを使用するときは、固有性 (ロール・スタックのコマンドのオカレンスが 1 つだけあること) を維持するため UNIQUE コマンドを使用してください。詳しくは、56 ページの『UNIQUE コマンドの使用』を参照してください。

COMPAT
この VIEW 呼び出しの機能が NetView プログラムのバージョン 5 リリース 1 より前のリリースの VIEW の振る舞いと互換性があることを指定します。機能の詳細については、VIEW を使用するプログラムについて記述されていた、前のリリースの資料を参照してください。COMPAT オプションはデフォルトです。

EXTEND
この VIEW 呼び出しで、Tivoli NetView for z/OS バージョン 5 リリース 1 で導入された拡張機能を使用することを指定します。この機能の例として、以下のものがあります。

- VIEW が、指定されたローカル変数値を取り出し、その値を、指定されたグローバル変数値の代わりに使用する機能。
- メッセージがトラップされたときに、VIEW を RC=2 で割り込みする機能。

EXTEND オプションを使用して、更新処理を実行するために (グローバル変数を使用する) 別々のプログラムを実行することなく、変数の動的更新を行うことができます。

EXTEND オプションは、NetView コマンド・リスト言語ではサポートされません。

使用上の注意

- 以下の表には、EXTEND オプションを指定した VIEW と COMPAT オプションを指定した VIEW の違いを要約しています。

表 8. EXTEND オプションを指定した VIEW と COMPAT オプションを指定した VIEW の比較

機能	VIEW=EXTEND 使用時の振る舞い	VIEW=COMPAT 使用時の振る舞い
変数の探索順序	• ローカルで定義された変数 • 制御変数 • タスク・グローバル変数 • 共通グローバル変数	VIEW 呼び出し CLIST 内での変数定義に応じて、ローカル辞書またはグローバル辞書から検索された値
VIEW は、メッセージがトラップされたとき RC=0 で割り込みされるか？	有	NO
適切なグローバル辞書に保管されたグローバル変数の値を変更するか?	はい。(VIEW の前の GLOBALV で指定された場合)	はい。(VIEW の前の GLOBALV で指定された場合)
ローカル辞書に保管されたグローバル変数の値を変更するか?	有	いいえ。(NetView CLIST 言語には関係なし)

注: 本書では、これ以降の VIEW に関するすべての説明で、 Tivoli NetView for z/OS バージョン 5 リリース 1 で導入された拡張機能を前提としています。ただし、この機能を使用するには、VIEW コマンドに EXTEND オプションを指定する必要があります。

- NOINPUT を指定することにより、オンライン・ヘルプ・パネルを表示するときにコマンド・プロシージャーを使用できるようになります。ヘルプ・パネル階層のコーディング方法については、 73 ページの『第 4 章 オンライン・ヘルプ情報の変更と作成』を参照してください。
- VIEW コマンドでは、TRAP 処理から受信したメッセージのデータを直ちに表示することができます。定期的に、メッセージ以外のソースから更新することも可能です。詳しくは、 68 ページの『動的更新機能』を参照してください。
- VIEW コマンドは、コマンド・プロシージャーからの使用のみを目的として用意されたものです。コマンド・リスト内でパネルを表示するためにビュー・コマンドを使用すると、ビューを出てプロシージャーの終了までの間で最小限の処理が行われます。ビューが終了しプロシージャーの終了までの間は、オペレーターの入力は禁止されます。
- VIEW NOINPUT コマンドが直前の VIEW コマンドと同じ *compname* を指定して実行された場合、直前の VIEW コマンドはキャンセルされ、その VIEW コマンドを実行したコマンド・プロシージャーもキャンセルされます。

VIEW と BROWSE からの戻りコード

表 9 では、VIEW および BROWSE コマンドで受け取られる戻りコードについてリストし、説明しています。また、必要な対応アクションについても簡単に説明します。

表 9. VIEW と BROWSE からの戻りコード

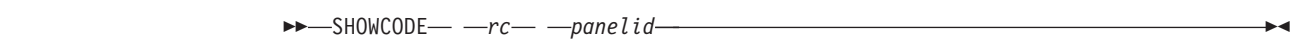
コード	意味	アクション
2	このタスクについてトラップされたメッセージが存在します。	トラップしたメッセージを廃棄または処理してから、 VIEW または BROWSE を呼び出します。
4	• 指定されたパネルが CNMPNL1、CNMMSGF、CNMCMDF データ・セット内にありません。 • 入出力 (I/O) エラーの可能性があります。	パネル定義を正しいデータ・セットまたはファイルに入れてください。
8	パネル定義フォーマットが正しくありません。非コメント行がありません。	パネル定義のフォーマットを訂正してください。
12	メンバーをブラウズすることは許可されていません。	システム・プログラマーに権限の再定義を依頼してください。
16	VIEW コマンド・プロセッサーが無効なパラメーターにより呼び出されました。*Name1* は、1 文字から 8 文字までの値であり、*name2* は、有効なパネル ID でなければなりません。有効なパラメーターは INPUT、NOINPUT、MSG、NOMSG です。	コマンド・リストを訂正して有効なオプションを使用してください。
24	フルスクリーン・コマンド・プロセッサーは OST のみで利用可能です。	OST 以外から VIEW を呼び出さないでください。
28	パネルの論理レコード長が 80 バイトではありません (VM のみ)。	ファイルを 80 バイトの論理レコード長に変更してください。
32	マクロ呼び出しに起因する回復不可能なエラーが発生しました。エラーの原因としては、CNMMSGF または CNMCMDF がオンライン・メッセージまたはコマンド・ヘルプにインストールされていないことが考えられます。 NetView ログのメッセージ DWO050I も参照してください。	CNMMSGF または CNMCMDF をインストールしてください。IBM ソフトウェア・サポートに連絡してください。
36	回復不可能な内部プログラミング・エラーが発生しました。NetView ログのメッセージ DWO050I も参照してください。	IBM ソフトウェア・サポートに連絡してください。
40	ブラウズ・パネル CNMBROWS (メンバーのブラウズに使用する) が見つかりませんでした。	正しいデータ・セットまたはファイルに CNMBROWS を入れてください。
81	パネル定義フォーマットが正しくありません。テキスト・インディケーター行がないか、または 49 個を超えるオプション定義があります。(詳細については、 38 ページの図 5 を参照してください。)	パネル定義のフォーマットを訂正してください。
83	パネル定義フォーマットが正しくありません。コメント行の位置が正しくありません。	パネル定義のフォーマットを訂正してください。

SHOWCODE による VIEW からの戻りコードの表示方法

SHOWCODE コマンド・リストは、コマンド・プロシージャーで使用されて、VIEW コマンドから戻されるゼロ以外の戻りコードの説明を表示します。

SHOWCODE コマンドのコーディングは次のように行ってください。

SHOWCODE



各項目の意味は以下のとおりです。

rc 説明を表示させたい戻りコードをもつ変数の名前です。

panelid
戻りコードを出す前に VIEW コマンドが表示しようとしていたパネルの名前を指定します。このパラメーターは戻りコード 4、8、12、28、81、および 83 のみで必要です。

SHOWCODE は、VIEW からのゼロ以外の戻りコードの説明をメッセージとして表示します。表 10 は、戻りコードとそれに関連したメッセージ ID を示します。

表 10. ゼロ以外の VIEW 戻りコードとそれに関連したメッセージ ID

戻りコード	メッセージ ID
4	CNM335I
8	CNM336I
12	CNM337I
16	CNM338I
24	CNM340I
28	CNM341I
32	CNM342I
36	CNM343I
40	CNM9071
81	CNM388I
83	CNM390I

コマンド・プロシージャーから SHOWCODE を出す前に、戻りコードがゼロでないことを確認してください。SHOWCODE を使用して VIEW からのエラー・メッセージを表示する例については、69 ページの『パネルを更新する REXX コマンド・リストの例』 を参照してください。

フィールドのカラーと強調表示の制御

既存のパネルのカラーおよび強調表示を変更したり追加したりすることができます。表示パネルのテキストのカラーと強調表示は、属性シンボルまたは属性変数によって制御されます。ソース・パネルに属性シンボルがコーディングされると、属性シンボルは表示パネルにブランクとして現れます。

特定の行の属性シンボルまたは属性変数の走査は、1 桁目に属性シンボルまたはパネル変数を含む場合にのみ行われます。そうでなければ、その行は現状のままのデフォルトのカラーで表示され、変数置換は行われません。

注: カラーおよび強調表示は使用している端末によって異なります。

属性シンボル

ソース・パネルに属性シンボルを指定して、テキストをカラー表示または強調表示することができます。ソース・パネルを編集して、テキストの前のブランク・スペースを表 11 または 表 12 の 2 番目の欄から選択した属性シンボルで置き換えてください。

変数は第 1 レベルでのみ解析されます。ネストされた VIEW 変数は、置き換えられますが解析はされません。したがって、ネストされた変数の中にあるカラー属性シンボルは、データとして表示されます。

パネルのヘッダーに指定したオプションにより、そのパネルに使用する 1 組の属性定義が決まります。オプションが指定されていない (***) 場合は、元のセット (属性セット 1) を使用してください。属性セット 2 は、パネル定義のテキスト・インディケーター行にオプションが指定されている (*** AT2) 場合に使用してください。テキスト・インディケーター行についての詳細は、74 ページの『ビュー・ベースのヘルプ』を参照してください。

表 11. セット 1 のカラーおよび強調表示属性

属性セット 1	シンボル	16 進文字	輝度	フィールド
白	%	X'6C'	高輝度	Text
反転の白色	}	X'D0'	高輝度	Text
下線付き白色	!	X'5A'	高輝度	Text
白	~	X'A1'	高輝度	入力
青緑色	$	X'5B'	通常	Text
下線付き青緑色	¥	X'E0'	高輝度	Text
青色	+	X'4E'	通常	Text
反転の青色	{	X'C0'	高輝度	Text
緑色	@	X'7C'	通常	Text
黄色	¬	X'5F'	通常	Text
桃色	¦	X'6A'	通常	Text
赤色	¢	X'4A'	高輝度	Text

表 12. セット 2 のカラーおよび強調表示属性

属性セット 2	シンボル	16 進文字	輝度	フィールド
白	%	X'6C'	高輝度	Text
反転の白色	}	X'D0'	高輝度	Text
反転の赤色	!	X'5A'	高輝度	Text
白	~	X'A1'	高輝度	入力
青緑色	$	X'5B'	通常	Text
反転の緑色	¥	X'E0'	通常	Text
青色	+	X'4E'	通常	Text
反転の青色	{	X'C0'	通常	Text
緑色	@	X'7C'	通常	Text
黄色	¬	X'5F'	高輝度	Text
反転の黄色	¦	X'6A'	高輝度	Text

表 12. セット 2 のカラーおよび強調表示属性 (続き)

属性セット 2	シンボル	16 進文字	輝度	フィールド
明滅する赤色	¢	X'4A'	通常	Text

特殊属性の表示

属性としても使用される特定のシンボルをカラー表示または強調表示された行に表示する場合は、そのシンボルの前に二重引用符 (") を置いてください。例えば、左側の括弧 ({) をテキスト内に表示したい場合は、ソース・パネルに "{ と入力します。二重引用符 (") を表示する場合は、"" と入力します。二重引用符 (") をソース・パネルで使用すると、二重引用符に続くテキストは、表示されたパネル内で左方に移動します。これらのシンボルと同じ 16 進値がシフトアウト制御文字とシフトイン制御文字で囲まれた 2 バイト文字テキストの一部としてコーディングされても、その 16 進値は属性として扱われません。

+ 属性の使用

青色のカラーに使用する正符号 (+) の使い方には注意してください。NetView コマンド・リスト言語に定義される変数に青色のカラーを割り当てる場合は、次のように正符号を引用符で囲んでください。

```
&COLOR = '+'
```

REXX 変数の **A** に青色を割り当てて、その内容である **G** を青に変更する場合は、次のようにしてください。

```
A = '+G'
```

引用符を付けないと、NetView プログラムは正符号を継続文字として解釈します。

$ および @ 属性の使用

文字 $ と @ は、コマンド・リストまたは REXX 変数内でデータとして使用されることが多いため、パネルまたは変数内で定義した場合、VIEW によるこれらの文字の扱い方は、他の文字とは異なります。パネルでは、これらのシンボルは、 46 ページの表 11 および 46 ページの表 12 に説明されているように、属性シンボルとして扱われます。変数内では、データとして扱われます。変数内で関連した属性を必要とする場合は、@ と $ の同義語として、それぞれ、より大記号 (>) と、より小記号 (<) に置き換えます。ユーザーのコマンド・リストの中でそれぞれの同義語を使用するようにしてください。以下の NetView コマンド・リストの例では、AMOUNT フィールドはストリング $1,000 を青緑色で、HEIGHT フィールドはストリング @ 6 feet を緑で表示します。

```
&AMOUNT = '<$1,000'
&HEIGHT = '>@ 6 feet'
```

同じ例が REXX ではどのようになるかを示します。

```
AMOUNT = '<$1,000'
HEIGHT = '>@ 6 feet'
```

より小記号とより大記号が変数内で使用されていない場合は、それらの記号は文字として表示されます。

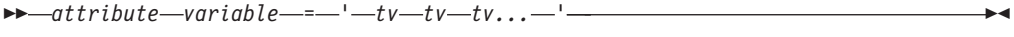
属性変数

属性変数は、ビュー・パネルを駆動するコマンド・プロシージャー内で割り当てられます。属性シンボルをパネルまたは変数データに定義する代りの方法として、パネル変数に関連づけられている属性変数を定義する方法があります。属性変数は、パネル変数とその変数に続く同一行のテキストに関する属性を記述します。属性変数を使うと、属性の選択範囲が広がり、入力フィールドを定義することができるようになります。属性変数を使用したときは、関連づけられているパネルの変数の内容に対して属性シンボルの探索走査はされません。

属性変数名は、パネル変数名の前にドル記号 ($) を連結して形成されます。例えば、NetView コマンド・リスト言語の場合は、パネル変数 &V1 に対する属性は、`&$V1` という変数内で定義されます。

REXX、PL/I、および C では、アンパーサンド (&) は使用されません。PL/I または C プログラムでは、属性変数は、PL/I の CNMVARS、または C の *Cnmvars* を使用して設定する必要があります。

次に属性変数の内容の構文を示します。



`tv` は *タイプ値 (type value)* の対です。1 つの属性変数に同じタイプの対を複数使用することができます。最後の対が受け入れられ、それより前の対は無視されます。

タイプ値 の値は次のとおりです。

tv =
タイプ値

A =
アラーム

AN 音響アラームなし

AY パネルの表示時に音響アラーム (ビープ音)

注: アラーム指定は、即時メッセージ行 ($CNMIMDL) の属性変数に対してのみ適用されます。

C =
カラー

CB 青色

CD カラー値が指定されていないデバイスのデフォルト・カラー。

CG 緑色

CP 桃色

CR 赤色

CT 青緑色

CW 白色または無色

CY 黄色

F =

フィールド

FA 保護。表示されたパネルにデータを入力することはできません。FA がデフォルト値です。

FI 無保護。表示されたパネルにデータを入力することができます。

H =

強調表示

HB 明滅

HD 強調表示値が指定されていない場合の、デフォルトの拡張強調表示。

HR 反転表示

HU 下線表示

I =

輝度

ID 無表示、表示不能

IH 高輝度

IN 通常輝度。輝度値が指定されていない場合のデフォルト値。

U =

カーソル

UN カーソルはこのフィールドの最初に置かれません。UN がデフォルト値です。

UY カーソルはこのフィールドの最初に置かれます。UY を複数の変数に指定すると、最後の変数だけが受け入れられ、それより前の変数は無視されます。

注:

1. カーソルを特定の変数に関連付けたくない場合は、任意の行および桁にカーソルを置くことができます。INPUT オプションを指定して VIEW を呼び出す手順で、VIEWICROW および VIEWICCOL 変数を使用してください。VIEWICROW および VIEWICCOL 変数の詳細については、 58 ページの『フルスクリーン入力機能』を参照してください。
2. VIEWICROW および VIEWICCOL 変数を使用し、かつ属性変数に UY を指定する場合、カーソルはその属性変数によって位置指定されます。
3. VIEWICROW および VIEWICCOL 変数を使用しないか、またはパネルの属性変数のいずれにもカーソルを指定しない場合、カーソルは最初の入力フィールドの先頭に置かれます。

1 つ以上のブランクを使用して*タイプ値* の対を区切ります。次の例は、NetView コマンド・リスト言語の例です。この例では、高輝度で赤色の保護フィールドとして、&V1 を定義しています。また、高輝度で青緑色で、カーソルがそのフィールドにある保護フィールドとして、&V2 が定義されています。

```
&$V1 = 'FA IH    CR'
&$V2 = 'IN IH CT UY IH'
```

次の REXX の例では、V1 がカーソルなしの入力変数 (無保護フィールド) として定義されています。V2 については、すべてデフォルト値が使用されています。

```
$V1 = 'FI  UN'
$V2 = ' '
```

属性変数または属性シンボルによって定義された属性は、以下の中の 1 つが起こるまで適用されます。

- 行の終わり。
- その行内でそれより後ろの位置に属性シンボルが明示的に置かれる。
- その行内のそれより後ろの位置の変数が次のいずれかをもつ。
  - 新しい属性を指定する有効な属性変数。
  - 有効な属性変数はないが、1 つ以上の属性シンボルをもつ。

パネルで定義された定数または変数は、入力フィールドの一部となることができ、その入力フィールドの一部に重ねてタイプ入力した場合のみ更新されます。入力フィールドにデータを入力すると、その入力フィールドの内容全体がパネル変数に割り当てられます。

パネル変数により定義されたフィールドの最初のバイト (&) は、属性の指定に使用され、このバイトの後に変数の内容が続きます。属性変数がパネル変数と対応しているときは、パネル変数が検出されない (したがってブランクに置き換えられている) 場合でも、属性変数はこの最初のバイトで効力を発揮します。

注: 属性変数に構文エラーがあり、NetView ログがアクティブであれば、メッセージ CNM944I がログに書き込まれます。

ソース・パネルの変数の表示

VIEW コマンドがパネル定義にコーディングされた変数名を解決しようとする場合、VIEW コマンドは、値を持つ定義済み変数を見つけるまでそれ以降の環境を次の順序で探索します。

- コマンド・プロシージャーで割り当てられた変数
- 制御変数 (例えば &OPID)
- タスク・グローバル変数
- 共通グローバル変数

パネルに指定された変数名が前述の環境のいずれにも定義されていないと、その変数名はブランクのストリングとして表示されます。制御変数またはグローバル変数として定義された変数は、呼び出しコマンド・プロシージャーでも割り当てられる場合があることに注意してください。パネルには、制御変数値やグローバル変数値の代わりに、変数に割り当てられた値が表示されます。

関連した属性変数が定義されていない場合、変数の置き換えられた値に対して属性シンボルの探索走査が行われます。検出された属性シンボルは、カラーの制御、強調表示、およびデータ・フィールドに使われます。シンボルをシンボルとして表示

し、属性として使用しない場合、変数に関連する属性変数をコーディングします。これで、データ内のシンボルを属性変数ではなくデータとして扱うことができます。

属性シンボルをデータとして表示するときは、特殊な規則に従わなければなりません。この規則についての詳細は、47 ページの『特殊属性の表示』および 48 ページの『属性変数』を参照してください。

注: XVAR オプションがパネル・テキスト・インディケーター行にコーディングされていない場合は、VIEW パネル定義の変数名として 1 から 11 桁の英数字 (A から Z と 0 から 9) のみを使用してください。XVAR オプションがコーディングされている場合は、変数名は最大長 31 文字でピリオドを含むことができます。詳しくは、53 ページの『複合シンボル』を参照してください。英字には大文字を使用します。変数が VIEW を呼び出している言語により参照される場合は、変数名は、その言語によって設定された変数名の命名規則にも従わなければなりません。例えば、PL/I、C、および REXX で使用される変数名は、英字で始まらなければなりません。

グローバル変数は VIEW を使用して検索され、表示されますが、VIEW コマンドを実行する前に、コマンド・プロシージャーで参照される場合もあります。グローバル変数は、NetView コマンド・リスト言語では &TGLOBAL、 &CGLOBAL、または GLOBALV、REXX では GLOBALV、PL/I では CNMVARS または GLOBALV、あるいは C では *Cnmvars* または *GLOBALV* で定義されます。

参照: グローバル変数の詳細については、「*IBM Tivoli NetView for z/OS プログラミング: REXX および NetView コマンド・リスト言語*」または「*IBM Tivoli NetView for z/OS プログラミング: PL/I および C*」を参照してください。

高水準言語プログラムにより呼び出された VIEW コマンドがローカルまたは属性変数を探す場合は、その変数の設定には、PL/I の場合は CNMVARS、C の場合は *Cnmvars* を使用する必要があります。

REXX のユーザーは、VIEW コマンドを開始する前に変数 *varname* 用に以下のタスクを実行することにより、VIEW コマンドを使用してグローバル変数の値を更新することができます。

1. VIEW パネルのグローバル変数が使用するフィールドを、属性変数を使用する入力フィールドとして定義する。
2. GLOBALV DEFT (または DEFC) *varname* コマンドを出して、グローバル変数を定義する。
3. 共通グローバル辞書またはタスク・グローバル辞書に、*varname* が定義されている (ヌル以外の値で) ことを確認する。必要ならば、GLOBALV PUTT (または PUTC) *varname* コマンドを使用して、値を保管します。

上記のすべてのステップを実行すると、グローバル変数 *varname* が更新されます。上記のステップすべてを行わないと、REXX ローカル変数 *varname* が表示され更新されます。 VIEW がこの方法でグローバル変数にアクセスするときは、同じ名前をもつ REXX ローカル変数も VIEW により変更されます。グローバル変数の新規の値にアクセスするには、REXX ユーザーは、例えば GLOBALV GETT (または GETC) コマンドを出して、その値のローカル・コピーを入手する必要があります。

VIEW パネルで NetView 制御変数 (例えば、APPLID または OPID) を指定し、そのフィールドを入力フィールドとして定義すると、更新された値はコマンド・プロシージャー環境にのみ保管されます。制御変数の値を更新することはできません。

次の REXX の例は、VIEW を使ってどのようにグローバル変数を更新するかを示しています。

```
/*   */
'GLOBALV GETT XYZ'
IF LENGTH(XYZ) = 0 THEN
 DO
   XYZ = '    '
   'GLOBALV PUTT XYZ'
 END
$XYZ = 'FI'
'VIEW NAME1 TESTPANL INPUT EXTEND'
SAY 'XYZ IS NOW' XYZ
EXIT
```

変数に割り当てる値の長さがソース・パネルでの変数の長さを超える場合、また、パネル定義で変数に英数字または特殊文字 (!、¢、¥、¦、@、#、$、%、¬、&、"、+ など) が続く場合は、その値は切り捨てられます。変数の後ろに上記以外の文字 (ピリオドまたはダッシュなど) が続く場合は、それらの文字の上に値が上書きされます。

変数に割り当てられた値に 2 バイト・テキストが含まれる場合、この 2 バイト・テキストは DBCS シフトアウト文字とシフトイン文字の間になければなりません。パネルが一組の DBCS シフトアウト文字とシフトイン文字の間のすべての 2 バイト・テキストを表示することができない場合、VIEW は表示できる範囲ですべてのテキストを表示し、さらにピリオド (.) を表示して、表示できずに切り捨てられた文字があることを示します。

例えば、&DBCSTEXT という変数に「NetView ヘルプ・メニュー」という日本語の値を定義した場合、パネル上のフィールドが短すぎる、オペレーターがパネルを右や左にスクロールした、VIEW を使用するアプリケーションによってデータが切り捨てられたなどの理由により、この値が切り捨てられる可能性があります。例えば、NetView WINDOW コマンドは、VIEW を使用して 2 バイト文字の切り捨てを処理します。以下に 2 バイト漢字文字の 16 進表示でテキスト長を示します。

```
....+....1....+....2....+....3..

04945494D4545444A4A4D444A4945450
E39363530343835323F373537373438F
```

パネルの定義で 32 文字未満の値を &DBCSTEXT の値に入れられるようにした場合、あるいは、オペレーターがテキストをスクロールして、パネルに 32 文字未満の値を表示できるようにした場合、VIEW は収められるすべての文字を表示します。VIEW が 2 バイト文字の半分しか表示できない場合は、表示可能な文字部分をピリオド (.) で置き換えます。置き換えるときの方法は、BROWSE がネットログの場合に前後の 2 バイト・テキストを切り捨てるときと同じです。この例で、最初の 2 バイトが切り捨てられた場合は、VIEW は、最初の 2 バイト文字 (X'4399') のうち表示できない後半部分をシフトアウト文字 (X'0E') で置き換えます。最初の 3 バイトが切り捨てられた場合には、VIEW は、2 番目の 2 バイト文字全体 (X'4356') をピリオドとシフトアウト文字 (X'4B0E') で置き換えることになります。

2 バイトのシフトアウトおよびシフトイン文字の組み合わせが正しく定義されていない VIEW パネルをオペレーターが表示しようとすると、無効なデータ・ストリームがデバイスに送信されて、オペレーターがログオフされるなどの、予期できない結果が発生します。2 バイトのシフトアウトおよびシフトイン文字が正しく組み合わされていない DBCS 定義の例には、次のようなものがあります。

- シフトアウト文字またはシフトイン文字のどちらか一方の数が多い (対になっていない)
- 複数の変数の間で一対が分割されている
- 変数とパネル定義の間で一対が分割されている
- パネルの複数行にまたがって一対が分割されている

複合シンボル

複合シンボルは少なくとも 1 つのピリオドと、少なくとも 1 つの他の文字を含みます。数字またはピリオドで開始することはできません。1 つのピリオドのみの場合は、そのピリオドは最後の文字となることはできません。

STEM (最初のピリオドまでおよび最初のピリオドを含むシンボルの一部) で開始される名前は、定数シンボル、単純シンボル、またはヌルの名前 (ピリオドで区切られる) の PART が続きます。定数シンボルは数字 (0 から 9) またはピリオドで開始します。単純シンボルはピリオドを含まず、数字 (0 から 9) では開始しません。

VIEW はパネル内にコード化された複合シンボルで開始します。次に、VIEW は PART をその値で置き換えることで、派生変数名を作成します。さらに、VIEW は、パネルに表示するために、この派生変数の値を要求します。

次の例は REXX プログラムからの部分的な抜粋です。

```
a=3                  /* assigns '3' to the variable 'A'*/
b=4                     /* '4'      to  'B'            */
c='Fred'                /* 'Fred'   to  'C'            */
a.b='Fred'              /* 'Fred'   to  'A.4'          */
a.fred=5                /* '5'      to  'A.FRED'       */
a.c='Bill'              /* 'Bill'   to  'A.Fred'       */
c.c=a.fred              /* '5'      to  'C.Fred'       */
x.a.b='Annie'           /* 'Annie'  to  'X.3.4'        */
d=''                    /* ''       to  'D'            */
e='4'                   /* '4'      to  'E'            */
x.d.e='Annie'           /* 'Annie'  to  'X..4'         */
say  a    b    c    a.a    a.b    a.c    c.a    a.fred   x.a.4   x.d.4
/*                                                              */
/* Rexx will display the following values:                      */
/*   3    4    Fred A.3    Fred   Bill   C.3    5        Annie   Annie*/
/* If these same variables are displayed on a View panel        */
/* (preceded by '&' and in upper case) and if the View panel    */
/* definition includes the XVAR option, View displays the following   */
/* values:                                                      */
/*   3    4    Fred        Fred   5             5        Annie        */
```

図 6. VIEW の変数の値を要求する REXX プログラムの例

インプリメンテーションの最大値

パネル・テキスト・インディケーターに XVAR オプションがある場合は、すべての HLL と REXX 変数は 31 文字までに制限されます。ない場合は、限界は 11 です。NetView コマンド・リスト言語は、複合変数または 12 文字以上の変数名はサポートしません。 REXX がストリングを表示する方法と VIEW がストリングを表示する方法の違いに注意することが重要です。

使用上の注意

1. VIEW は REXX で定義された大/小文字混合のシンボルをサポートしません。例えば、 53 ページの図 6 の `a.c.` は、VIEW では 5 として表示されますが、REXX では Bill として表示されます。
2. VIEW は、複合変数の最終値が未定義、ヌル、または無効の場合は、その値をブランクで表示します。

   53 ページの図 6では `a.a`、`c.a`、および `x.d.4` が VIEW ではブランクとして表示されます。
3. VIEW は不明の複合変数 PART とヌル値をもつ複合変数 PART を区別しません。PART がヌルまたは不明の場合、その NAME は複合変数を作成する際に使用されます。 53 ページの図 6 では、VIEW は `&X..4` ではなく `&X.D.4` を探索するので、Annie を検出することができません。
4. 複合変数を使用するには、パネル定義のテキスト・インディケーター・セクションに ***** XVAR** と入力します。詳しくは、『テキスト・インディケーター』を参照してください。

コマンド・プロシージャーからのコマンド発行

コマンドがコマンド・プロシージャーから直接出されると、そのプロシージャーは、出したコマンドが完了するまで中断されます。呼び出されたコマンドが完了し、戻りコードを利用できるようになると、そのプロシージャーは再開します。呼び出されたコマンドが長期実行コマンドの場合は、そのコマンドと呼び出しプロシージャーが 1 つのグループを形成し、NetView ROLL コマンドにより 1 単位として処理されるようになります (ロール・グループ)。

注: BGNSESS FLSCN コマンドはこの例外で、このコマンドでは、DSIPUSH の MINOR オプションを使用してセッションが開始される前に、呼び出しプロシージャーを完了させることができます。DSIPUSH については、「*IBM Tivoli NetView for z/OS プログラミング:アセンブラー*」を参照してください。

複数のコマンドとプロシージャーのグループ化は、各パネルを別々のプロシージャーで構築して、関連パネルの階層を構築したい場合に有益です。オペレーターから受けとったコマンドのような、無関係のコマンドを実行するときには、コマンドとプロシージャーをグループ化してはいけません。

呼び出しプロシージャーから無関係のコマンドを切り離すには、CMD コマンドを使用します。例えば、変数 *cmdline* に、自分のパネルに入力されたオペレーターのコマンドが入っているものとします。REXX コマンド・プロシージャー内で以下のコマンドのいずれかを出すと、*cmdline* コマンドを非同期的にキューに入れることができます。

```
'CMD HIGH ' cmdline
'CMD LOW ' cmdline
```

CMD コマンドの HIGH または LOW パラメーターは、コマンドをキューに入れる際の優先順位を示します。

注: HIGH パラメーターを指定して CMD コマンドを出すと、必ず他の処理に割り込み、キューに入れられたコマンドが実行されます。

例えば、オペレーターが STATMON コマンドをパネルのコマンド行に入力するものとします。 CMD コマンドを使用すると、STATMON コマンドを直接呼び出すのではなく、STATMON コマンドをキューに入れることができます。これによって、STATMON が完了していなくても、オペレーターは STATMON からコマンド・プロシージャーにロールバックすることができます。ROLL 機能について詳しくは、「*IBM Tivoli NetView for z/OS プログラミング:アセンブラー*」を、また CMD コマンドについて詳しくは、NetView オンライン・ヘルプを参照してください。

コマンドを呼び出すのではなくキューイングすると、プロシージャーが、キューに入れられたコマンドに起こりえるリセット状態になることを避けられます。

VIEW を用いたロール可能コンポーネントの作成

NetView のコンポーネントは端末画面を制御するコマンドまたはコマンド・プロシージャーであり、オペレーターが任意の NetView コマンドを入力できるようにし、このようなコマンドの完了時に再開することができます。 コマンド・プロシージャーでは、VIEW を使って、必要な画面制御を行うロール可能なコンポーネントを作成することができます。

NOINPUT オプションを指定した場合、VIEW はユーザーに代ってオペレーター・コマンド・インターフェースを処理します。VIEW コマンドに INPUT オプションを指定した場合は、VIEW は、名前付きの変数の形式でオペレーターの入力をプロシージャーに返し、そのうちの 1 つまたは複数の変数がコマンドとして扱われる場合があります。

これらの変数に含まれるコマンドは、NetView プログラムでは大文字でなければなりません。PL/I および C コマンド・プロシージャーでは、CNMCMD が出される前にこれらのコマンド・ストリングが大文字であることを検査しなければなりません。 NetView コマンド・リスト言語では、変数の内容を大文字に変換するために UPPER コマンドが提供されます。REXX コマンド・リストは、UPPER 命令を使用して、コマンドが大文字であることを確認することができます。

UPPER コマンドの使用

UPPER コマンドを使用して、指定された変数の内容を大文字に変換します。

UPPER コマンドの形式は次のとおりです。

UPPER

各項目の意味は以下のとおりです。

variable
英大文字に変換すべき、1 文字から 11 文字までの変数の名前を指定します。反復区切り文字のコンマは、1 つの UPPER コマンドに複数の変数名を任意に指定できることを示します。

例:

```
UPPER CMDLINE
CMD HIGH &CMDLINE
```

使用上の注意

1. 変数名の前に先行するアンパーサンド (&) を指定しないでください。
2. 複数の変数が指定された場合は、変数の中の 1 つがエラー状態にあっても (検出されない、長さが無効など)、すべての変数が変換されます。
3. UPPER コマンドは、NetView コマンド・リスト言語でのみ提供されます。類似する機能が、REXX UPPER 命令をもつ REXX コマンド・リストで利用可能です。
4. UPPER コマンドは、コマンド・ストリング内の他のコマンドと連結してはなりません。

戻りコード: このコマンドの戻りコードは次のとおりです。

0 すべての指定変数が正常に完了した。

4 少なくとも 1 つの変数が見つからない、または少なくとも 1 つの変数が無効である。

8 少なくとも 1 つの変数の長さが範囲を超えている。

12 少なくとも 1 つの変数が見つからず、その他の変数の中で少なくとも 1 つの変数の長さが範囲を超えている。

16 コマンド・プロシージャーから呼び出されていない。

20 変数が指定されていない。

UNIQUE コマンドの使用

UNIQUE コマンドにより、発行側のコマンド・プロシージャーと同じメンバー名 (コマンド・リストおよび REXX の場合) またはモジュール名 (PL/I および C の場合) のサブコンポーネントをもつコンポーネントをロール・スタック内で探索することができます。このようなコンポーネントが見つかると、UNIQUE コマンドにより、2 つのコンポーネントのうちの 1 つだけ (発行元のコンポーネントまたは古い方のコンポーネント) がそのロール・スタックに残されます。

UNIQUE コマンドの形式は次のとおりです。

UNIQUE

各項目の意味は以下のとおりです。

CANCEL
ロール・スタックで現在実行中のコンポーネントと一致したエレメントがロール・スタックに入っているロール・グループのリセット (CANCEL) を指定します。CANCEL はデフォルト値です。(コマンドを出したコンポーネントはロール・スタック上に残ります。)

PROMOTE
ロール・スタックで現在実行中のコンポーネントと一致したエレメントがロール・スタックに入っているロール・グループを位置決めする (PROMOTE) ことを指定します。

使用上の注意

1. UNIQUE コマンドはコマンド・リストから出された場合のみ有効です。
2. NetView プログラムでは、オペレーターは同じコマンド・プロセッサーの複数のコピーを開始させることができます。NetView コンポーネントの作成時など、複数のコピーがない方がよい場合もあります。DSIPOP または DSIPUSH を PROMOTE オプションで使用することにより、アセンブラー・プログラマーは長期実行コマンドの固有性を保証することができます。また、UNIQUE コマンドの使用により、コマンド・プロシージャーでの固有性が保証されます。
3. ユーザーのプロシージャーから UNIQUE を出しても、そのプロシージャーの現行のコピーが唯一のアクティブなコピーである場合は効果がありません (0 の戻りコードが出されます)。アクティブ な長期実行コマンドまたはプロシージャーとは、処理中のいずれかの段階にあって、完了していないものをいいます。アクティブなプロシージャーには、他の長期実行コマンドにより中断されている (またはブロックされている) ものも含まれます。同じプロシージャーの別のコピーが同じタスクのもとに存在する場合は、UNIQUE コマンドは、そのコピーを含むロール・グループ全体に影響します。
4. UNIQUE コマンドで CANCEL オプション (デフォルトの形式) を使用すると、古い方のコピーがリセット条件付きで制御権を得て、その間、呼び出しプロシージャーは一時的に中断されます。NetView プログラムは、プロシージャーがリセットされるときに通常出される取り消しメッセージを抑止します。プロシージャーの取り消されたコピーとそのグループ内の他のコピーが完了すると、発行元のコピーは UNIQUE コマンドの次の行から再開します。戻りコード 4 が設定されます。
5. UNIQUE コマンドで PROMOTE オプションを使用すると、呼び出しプロシージャーの前のコピーとそのロール・グループがロール・スタックの最初に移動し、UNIQUE を出したコピーの完了時に再開可能な状態になります。戻りコード 4 が設定されます。UNIQUE の呼び出しプロシージャーを終了し、指示されたプロシージャーが制御を再獲得できるようにしなければなりません。制御を再獲得できるようになったことを呼び出し側に知らせるために、終了コード -5 が使用されます。
6. UNIQUE を NetView コマンド・リスト言語で使用するときは、抑止文字 (&SUPPCHAR) をコーディングして、コマンドにエラーがある場合に起きる不必要なコマンド・エコーを抑止してください。REXX プロシージャーでは、SIGNAL ON HALT をコーディングして REXX の取り消しメッセージを抑止

してください。HALT サブルーチンは、戻りコード 5 を戻します。REXX プロシージャーに SIGNAL ON ERROR をコーディングすると、戻りコード 4 がエラー・ラベルを表します。

7. ROLL コマンドでは特殊な処理は必要ありません。このコマンドは、他の NetView コマンドと同じ方法で出すことができます。他の NetView アプリケーションと整合するように、PF6 と PF18 で ROLL コマンドが出されるように設定してください。
8. パラメーターの同意語がサポートされています。
9. パラメーター許可の制限は UNIQUE コマンドには適用されません。
10. コンポーネントが取り消されると、REXX、PL/I、および C コマンド・プロシージャーは終結処置を実行することができます。

戻りコード: このコマンドの戻りコードは次のとおりです。

0 呼び出しプロシージャーが固有です。

4 一致するプロシージャーが検出されました。アクションは正常に完了しました。

12 無効な環境 (プロシージャーから呼び出されていない)。

16 構文エラー、無効な引数。

フルスクリーン入力機能

VIEW コマンドにより、呼び出しプロシージャーから下記の値を受け取ることができます。

- カーソル行位置
- カーソル桁位置

上記の情報は、INPUT キーワードを用いるとともに、呼び出しプロシージャーに VIEWICROW および VIEWICCOL をコーディングすることにより指定します。パネルが表示されると、カーソルは、VIEWICROW および VIEWICCOL で指定された位置に置かれます。カーソルを変数に関連付けるために属性変数を使用した場合は、その値が、VIEWICROW および VIEWICCOL で指定されたカーソル位置をオーバーライドします。 59 ページの表 13 に、この 2 つの変数の説明があります。

VIEW コマンドにより、下記のものを呼び出しプロシージャーに戻すことができます。

- パネル上の複数入力可能変数の内容
- アテンション ID (AID) 情報
- カーソル位置
- VIEW コマンドによって出力されるパネル行数
- VIEW コマンドによって出力されるパネル桁数

上記の情報は、INPUT キーワードを用いるとともに、属性変数に*タイプ値* の対 FI をコーディングすることにより指定します。

INPUT オプションを使用する場合は、FI を指定して属性変数を定義した場合にのみ、入力フィールドを利用することができます。(*タイプ値* の対の説明は、48 ページの『属性変数』を参照してください。)

パネルが表示されると、パネルには、重ね打ちで変更できる変数値が表示されます。変更した変数は、AID キーを押すと呼び出しプロシージャーに戻されます。 60 ページの表 14 は、AID キー、および呼び出しコマンド・プロシージャーに戻ったときに設定される変数を説明しています。

表 13. 呼び出しコマンド・プロシージャーで指定される変数

REXX、PL/I、および C	**NetView コマンド・リスト言語**	**説明**
VIEWICCOL	&VIEWICCOL	VIEW を呼び出すコマンド・プロシージャーによって設定されるカーソル位置 (桁)。この変数を VIEWICROW とともに使用して、パネル上の任意の場所にカーソルを位置付けます。許容値は、パネル上の桁数以下の正または負の整数です。正の整数は、カーソルをパネルの左側に位置付け、負の整数は右側に位置付けます。パネルの桁数より大きい整数を指定すると、カーソルは最初の入力フィールドの先頭に置かれます。 60 ページの図 7を参照してください。
VIEWICROW	&VIEWICROW	VIEW を呼び出すコマンド・プロシージャーによって設定されるカーソル位置 (行)。この変数を VIEWICCOL とともに使用して、パネル上の任意の場所にカーソルを位置付けます。許容値は、パネル上の行数以下の正または負の整数です。正の整数は、カーソルをパネルの上部に位置付け、負の整数は下部に位置付けます。パネルの行数より大きい整数を指定すると、カーソルは最初の入力フィールドの先頭に置かれます。 60 ページの図 7を参照してください。

パネルのサイズが 80 桁 x 24 行で、呼び出しプロシージャーで次のように指定したとします。

```
VIEWICCOL = 2
VIEWICROW = 2
```

この場合、カーソルは左から 2 桁目、上から 2 行目に置かれます。

```
VIEWICCOL = -2
VIEWICROW = -2
```

この場合、カーソルは右から 2 桁目、下から 2 行目に置かれます。

```
VIEWICCOL = 82
VIEWICROW = 22
```

この場合、カーソルは最初の入力フィールドの先頭に置かれます。これは、変数の 1 つにパネルのサイズより大きい値を指定したためです。

図 7. VIEWICCOL および VIEWICROW の例:

VIEWICCOL および VIEWICROW の例

表 14. 呼び出しコマンド・プロシージャーに戻ったときに設定される変数

REXX、PL/I、および C	**NetView コマンド・リスト言語**	**説明**
VIEWAID	&VIEWAID	入力データを入れるときに使用する AID キー
VIEWCURCOL	&VIEWCURCOL	AID キーが押されたときのカーソル位置 (桁)
VIEWCURROW	&VIEWCURROW	AID キーが押されたときのカーソル位置 (行)
VIEWCOLS	&VIEWCOLS	VIEW コマンドによって出力される桁数。パネル・テキスト・インディケーター行で WIDE または OPTROW のいずれもコーディングされていない場合、あるいは端末で 80 桁しかサポートされない場合は、デフォルト値は 80 桁となります。それ以外の場合、VIEWCOLS の設定は端末のサポート桁数になります。詳しくは、 73 ページの『第 4 章 オンライン・ヘルプ情報の変更と作成』を参照してください。
VIEWROWS	&VIEWROWS	VIEW コマンドによって出力された、指定パネルの行数。ソース・パネルの通常のデータ行数、ソース・パネルのオプションのデータ行数、および出力端末で利用可能な行数によって決定されます。詳しくは、 73 ページの『第 4 章 オンライン・ヘルプ情報の変更と作成』を参照してください。

VIEWAID 変数の内容は PF1 から PF24、PA1、PA2、PA3、または ENTER キーとして定義します。

PA1、PA2、または PA3 を押した場合は、AID (VIEWAID) 情報だけが呼び出しプロシージャーに戻されます。カーソル行、カーソル列などのパネルについて定義された入力フィールドは戻されません。

注: SNA 端末で ATTN キーを押すと、INPUT/NOINPUT をもつ VIEW は終了します。

図 8 から 65 ページの図 11 は、INPUT オプションを指定した VIEW を使用してロール可能コンポーネントを作成するソース・パネルを示しています。図 8 および 62 ページの図 9 は、置き換えられる入力可能変数をもつソース・パネルを示しています。これらのパネルでは、属性セット 2 (46 ページの表 12 を参照) の属性が使用されています。

```
/*********************************************************************/
/*  FIRST PANEL DISPLAYED                                            */
/*********************************************************************/
*** AT2
+PANEL1
$ X=================================================================X
$ |                                                                 |
$ |                                                                 |
$ |%     PPPPPPP    AAAAAAA   NN    NN  EEEEEEEE  LL          111   $ |
$ |%     PP    PP  AA     AA  NNN   NN  EE        LL        11 11   $ |
$ |%     PPPPPPPP  AAAAAAAAA  NN NN NN  EEEEEEEE  LL          11   $ |
$ |%     PP        AA     AA  NN   NNN  EE        LL          11   $ |
$ |%     PP        AA     AA  NN    NN  EEEEEEEE  LLLLLLLL  11111111$ |
$ |-----------------------------------------------------------------|
$ |  INPUT VARIABLE 1 = &VARIN1                                     $|
$ |  INPUT VARIABLE 2 = &VARIN2                                     $|
$ |                                                                 |
$ |  You entered:  &VAROUT1                                         |
$ |  You also entered:  &VAROUT2                                    |
$ X=================================================================X
$
$Enter a command on the command line OR...
$Enter NEXT or press PF8 to view the next panel.
$
%Action==> &COMMAND                                                 %
$                          PF2= End
$          PF6/PF18= Roll     PF8=Next
```

図 8. 第 1 パネル (入力可能な変数およびコマンド行をもつ) のソース :

第 1 パネル (入力可能な変数およびコマンド行をもつ) のソース

```
/*********************************************************************/
/*  SECOND PANEL DISPLAYED                                           */
/*********************************************************************/
*** AT2
+PANEL2
$ X=================================================================X
$ |                                                                 |
$ |                                                                 |
$ |%     PPPPPPP    AAAAAAA   NN    NN  EEEEEEEE  LL        22222222 $ |
$ |%     PP    PP  AA     AA  NNN   NN  EE        LL              22 $ |
$ |%     PPPPPPPP  AAAAAAAAA  NN NN NN  EEEEEEEE  LL        22222222 $ |
$ |%     PP        AA     AA  NN   NNN  EE        LL        22       $ |
$ |%     PP        AA     AA  NN    NN  EEEEEEEE  LLLLLLLL  22222222 $ |
$ |                                                                 |
$ |-----------------------------------------------------------------|
$ |                                                                 |
$ |                                                                 |
$ |                                                                 |
$ |                                                                 |
$ |                                                                 |
$ X=================================================================X
$
$Enter a command on the command line OR...
$Enter BACK or press PF7 to view the previous panel.
$
%Action==> &COMMAND                                                %
$                          PF2= End
$          PF6/PF18= Roll     PF7= Previous
```

図 9. 第 2 パネル (コマンド行のみをもつ) のソース :

第 2 パネル (コマンド行のみをもつ) のソース

『ロール可能コンポーネントを駆動する REXX コマンド・リストの例』 は、INPUT オプションを指定した VIEW を呼び出して PANEL1 を表示する REXX コマンド・リストの例です。このコマンド・リストは、ソース・パネルの入力可能変数 VARIN1 および VARIN2 に初期値を割り当てます。また、このコマンド・リストは、AID 情報とコマンド行入力を呼び出し側に戻します。

ロール可能コンポーネントを駆動する REXX コマンド・リストの例

```
/*********************************************************************************/
/*  EXAMPLE:  NETVIEW COMPONENT USING THE VIEW COMMAND                           */
/*********************************************************************************/
SIGNAL ON HALT
/*********************************************************************************/
/*  RESUME OLD COPY IF ONE EXISTS                                                */
/*********************************************************************************/
 'UNIQUE PROMOTE'
 if rc = 4 then EXIT -5               /* -5 will cancel caller if it exists */
SIGNAL ON ERROR              /* any nonzero rc other than as a result of the */
 /* UNIQUE command is an error                  */
/*********************************************************************************/
/* set up VAR1 and VAR2 as input capable fields                                  */
/*********************************************************************************/
$VARIN1 = 'FI IN CR HB UN'
$VARIN2 = 'FI IH CG HR UN'
/*********************************************************************************/
/* set up COMMAND as an input command line using an attribute                    */
/* variable.  Also define the cursor to stop at this field.                      */
/*********************************************************************************/
$COMMAND = 'FI UY'
  VARIN1 = 'INITIALIZE 1'
  VARIN2 = 'INITIALIZE 2'
Do forever
  COMMAND = '00'X                        /* COMMAND = nullchar (this clears */
                                         /* the command line and provides   */
```

```
                                        /* for insert capability)          */
  'VIEW USERAPPL PANEL1 INPUT'
  UPPER COMMAND
  VAROUT1 = VARIN1
  VAROUT2 = VARIN2
  SELECT
    When viewaid = 'PF2' then exit                      /* Quit if PF2       */
    When viewaid = 'PF6' then CMD HIGH ROLL             /* Roll if PF6       */
    When viewaid = 'PF8' then call PANEL2               /* Next panel if PF8 */
    When viewaid = 'ENTER' then
        SELECT
          when command = NEXT then call PANEL2
          /*******************************************************************/
          /*         Assume any other input given on command line is         */
          /*         to be issued to NCCF                                    */
          /*******************************************************************/
          when COMMAND ¬= ' ' then
               DO
                'CMD HIGH' COMMAND
               END
          otherwise nop
        END
    OTHERWISE nop
  End                                                         /* select */
End                                                       /* Do forever  */
PANEL2:
Do forever
  COMMAND = '00'X                       /* COMMAND = nullchar (this clears */
                                        /* the command line and provides   */
                                        /* for insert capability)          */
  'VIEW USERAPPL PANEL2 INPUT'
  UPPER COMMAND
  SELECT
    When viewaid = 'PF2' then exit                      /* Quit if PF2       */
    When viewaid = 'PF7' then return                    /* Previous panel PF7*/
    When viewaid = 'PF6' then 'CMD HIGH ROLL     '            /* Roll if PF6 */
    When viewaid = 'ENTER' then
        SELECT
          When COMMAND = 'BACK' then return
          /*******************************************************************/
          /*          Assume any other input given on command line is        */
          /*          to be issued to NCCF                                   */
          /*******************************************************************/
          when COMMAND ¬= ' ' then
               DO
                'CMD HIGH' COMMAND
               END
          otherwise nop
        END
    OTHERWISE nop
  End    /* select */
End   /* Do forever  */
RETURN
ERROR:
 EXIT -1                    /*  -1 means "FATAL ERROR IN NESTED PROCEDURE" */
HALT:
 EXIT -5                    /*  -5 means "CANCEL REQUESTED"                */
```

64 ページの図 10 は、このコマンド・リストから作成した第 1 パネルの例です。このパネルのソースについては、 61 ページの図 8 を参照してください。変数 VARIN1 および VARIN2 は、それぞれ、実際の値 INITIALIZE 1 と INITIALIZE 2 で置き換えられています。属性指定は $VARIN1 および $VARIN2 で定義されます。(詳細は 48 ページの『属性変数』を参照してください。)

次に示す属性は、VARIN1 に関するもので、入力フィールドの長さは次の属性シンボルが検出されるまで続きます。ここでは、属性シンボルは % です。

VARIN1 属性は次のとおりです。
- 入力、タブ (無保護)
- 通常輝度
- 赤色
- 明滅
- カーソルの位置指定なし

次の属性は VARIN2 に関するもので、入力フィールドの長さは行の終わりまで続きます。

VARIN2 の属性
- 入力、タブ (無保護)
- 高輝度
- 緑色
- 反転表示
- カーソルの位置指定なし

COMMAND の属性は次のとおりです。
- 入力、タブ (無保護)
- このフィールドの先頭にカーソルを置く

```
PANEL1
 X=====================================================================X
 |                                                                     |
 |                                                                     |
 |      PPPPPPP    AAAAAAA   NN    NN  EEEEEEEE  LL          111       |
 |      PP    PP  AA     AA  NNN   NN  EE        LL        11 11       |
 |      PPPPPPPP  AAAAAAAAA  NN NN NN  EEEEEEEE  LL          11        |
 |      PP        AA     AA  NN   NNN  EE        LL          11        |
 |      PP        AA     AA  NN    NN  EEEEEEEE  LLLLLLLL  11111111   |
 |---------------------------------------------------------------------|
 |  INPUT VARIABLE 1 = INITIALIZE 1                                    |
 |  INPUT VARIABLE 2 = INITIALIZE 2                                    |
 |                                                                     |
 |  You entered:                                                       |
 |  You also entered:                                                  |
 X=====================================================================X

Enter a command on the command line OR...
Enter NEXT or press PF8 to view the next panel.

Action==> _
                           PF2= End
          PF6/PF18= Roll   PF8=Next
```

図 10. REXX コマンド・リストによる変数置き換え後のコンポーネントの表示パネル:

REXX コマンド・リストによる変数置き換え後のコンポーネントの表示パネル

65 ページの図 11 は、コマンド・リストによる第 2 表示パネルです。このパネルのソースについては、 62 ページの図 9 を参照してください。

```
PANEL2
 X=====================================================================X
 |                                                                     |
 |                                                                     |
 |      PPPPPPP    AAAAAAA   NN    NN  EEEEEEEE  LL        22222222   |
 |      PP    PP  AA     AA  NNN   NN  EE        LL              22   |
 |      PPPPPPPP  AAAAAAAAA  NN NN NN  EEEEEEEE  LL        22222222   |
 |      PP        AA     AA  NN   NNN  EE        LL        22         |
 |      PP        AA     AA  NN    NN  EEEEEEEE  LLLLLLLL  22222222   |
 |                                                                     |
 |---------------------------------------------------------------------|
 |                                                                     |
 |                                                                     |
 |                                                                     |
 |                                                                     |
 |                                                                     |
 X=====================================================================X

Enter a command on the command line OR...
Enter BACK or press PF7 to view the previous panel.

Action==> _
                           PF2= End
          PF6/PF18= Roll     PF7= Previous
```

図 11. コンポーネントの表示パネル:

コンポーネントの表示パネル

コマンド行入力の戻し方

NOINPUT を指定して NetView プログラムの処理をコマンド行で開始させるときは、パネル上に波形記号 (~) を定義して表示させる必要があります。

波形記号の定義により、NetView プログラムにコマンドとして戻される入力フィールドが定義されます。同じ行の波形の後に &CUR をコーディングすると、カーソルが置かれる位置が決定されます。

&CUR は部分コマンドの事前定義に使用すると便利です。例えば、次のようにします。

```
 ~ V NET,ACT,ID=&CUR
```

とコーディングすると、次のように表示されます。

```
V NET,ACT,ID=_
```

残りの ID はオペレーターが完成させます。

パネルに複数の &CUR が定義されている場合は、最後の &CUR が処理され、その前のものは無視されます。複数の波形記号 (~) がパネルに定義されている場合は、最初の波形記号が処理され、それ以降のその他のものはパーセント符号 (%) に変えられます。

NetView プログラムに INPUT を指定する場合、コマンド行でのコーディングは、他の入力可能フィールドのコーディングと同じように行ってください。&CUR および波形の定義は使用しないでください。パネルを表示するプロシージャーが、コマ

ンドを出します。CMD HIGH の出し方については、54 ページの『コマンド・プロシージャーからのコマンド発行』を参照してください。

PF キーと VIEW サブコマンドの使用

PF キーおよび VIEW サブコマンドは、INPUT と NOINPUT の 2 つのビュー・オプションにより処理が相違します。以下の 2 つのセクションで、この違いについて説明します。

PF キーと NOINPUT オプションでのサブコマンドの使用

NOINPUT オプションを VIEW で使用する場合、PF キーを PFKDEF コマンドを使用して定義することができます。ユーザーが割り当てる値は、NetView コマンドか VIEW サブコマンドのどちらかになります。以下にリストした VIEW サブコマンドの中には、類似した NetView コマンドと同じ名前をもつものがあります。

ヘルプ 事前にコード化されたヘルプ・パネルを表示する。

```
HELP=helppan
```

End 元のコンポーネントに出る。

戻り 選択を行ったパネル (1 つ前のパネル) に戻る。

Top 複数ページ・パネルの最初のページに戻る。

Bottom
複数ページの最後のページへ進む。

Backward
複数ページ・パネルの前のページに戻る。

BACKWARD サブコマンドの PF キーを割り当てるだけでなく、コマンド行に次のコマンドを入力して、特定のページ分後方スクロールすることができます。

B *n* *n* ページまたは *n* パネルだけ後方スクロールする。

Forward
複数ページ・パネルの次のページへ進む。

FORWARD サブコマンドの PF キーを割り当てるだけでなく、コマンド行に次のコマンドを入力して、特定のページ分前方スクロールすることができます。

F *n* *n* ページまたは *n* パネルだけ前方スクロールする。

Entry Point
オペレーターがヘルプに入ったときに最初に見たパネルを表示する。

参照: 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*」の PFKDEF コマンドを参照してください。

PF キーと INPUT オプションでのサブコマンドの使用

INPUT オプションを指定して VIEW を使用する場合、PFKDEF コマンドを使って定義された設定可能 PF キーを使用したり、あるいは、コマンド・リストの PF キ

ーを解釈することができます。パネル定義およびパラメーターは、選択したオプションによってコーディングを変える必要があります。

設定可能 PF キーの使用

設定可能 PF キーを VIEW で使用するには、次の各ステップを完了させます。

1. パネル定義において、入力フィールドにするための属性変数 ($CNMIMDL) をもたない、CNMIMDL という変数を作成します。行の 1 桁目に &CNMIMDL を入れて、即時メッセージ行を定義します。その行には、ほかになにも入れてはいけません。

   VIEW アプリケーションで CNMIMDL 値が指定されていなかった場合、VIEW がグローバル辞書 (タスク、次に共通) を検索して、CNMIM*xxx* という変数を探します。ここで、*xxx* は VIEW が呼び出されたときに指定されたアプリケーション名です。この変数が見つからなかった場合、VIEW は同じ辞書で CNMIMVIEW を検索します。これは、VIEW アプリケーションに対するキーの設定方法と同様です。最終的にこれらの変数が 1 つもない場合は、メッセージ BNH257I のテキストを使用します。
2. パネル定義において、入力フィールドにするための属性変数 ($CNMCMDL) をもつ、CNMCMDL という変数を作成します。CNMCMDL によってコマンド域が定義されます。
3. オプションにより、CNMDIMD という別の変数を作成して、デフォルトの即時メッセージを定義します。このメッセージは、CNMIMDL メッセージが表示され、ほかに即時メッセージがないときに必ず、NetView プログラムによって表示されます。CNMDIMD を作成しない場合、NetView プログラムは CNMIMDL のときと同じ方法でデフォルト値を使用します。

これらのすべての変数は、属性 ($) 変数をサポートします。

例えば、VIEW の呼び出しで CNMIMDL にエラー・メッセージ、CNMDIMD にデフォルトのメッセージが入り、$CNMIMDL の設定を CR、$CNMDIMD の設定を CG とする場合があります。エラー・メッセージは赤色で表示されますが、例えば、ユーザーが RETRIEVE キーや遅延入力キーを押した場合は、赤色のメッセージでなく緑色のデフォルト・メッセージが表示されます。

REXX コマンド WINDOW は、PF キーを設定するための VIEW パネル・コーディングのよい例です。 BROWSE WINDOW と入力して、このコマンドの REXX ソースを確認します。

注:

1. ステップ 1 および 2 を行う VIEW の入力アプリケーションは、VIEW を呼び出した後、常にその VIEWAID 変数を ENTER に設定します。これは、ユーザーがコマンド・テキストを入力して ENTER キーを押したかのように、その他のキーが変換されるためです。
2. VIEW パネルが出力された (例えば、オペレーターが入力した即時コマンドによって) 後に、NetView プログラムによって即時メッセージ域が上書きされたことを VIEW が検出すると、VIEW からコマンド・リストに制御が戻されるときに &CNMIMDL 変数がヌルにされます。

3. 特殊変数の CNMIMDL および CNMDIMD は、VIEW INPUT だけでなく VIEW NOINPUT でもサポートされています。CNMCMDL が特別な意味をもつのは、VIEW INPUT においてのみです。

動的更新機能

VIEW コマンドを使用して、表示されているパネルの内容を動的に更新することができます。更新は、以下で制御することができます。

- **呼び出しプロシージャー**

  EXTEND モードを使用している場合、VIEW は、メッセージ TRAP が満たされていることを検出すると、呼び出しプロシージャーに制御を戻し、 VIEW パネルに表示されているローカル変数値の更新を認めます。RESUME コマンドを使用して制御が VIEW に戻されると、VIEW は新しい値で画面を最新表示します。

- **同じタスクで自動化またはプロシージャーを実行する**

  VIEW パネルで指定されている変数が呼び出しプロシージャーで定義されていないと、 VIEW はタスク・グローバル変数から値を読み取ることを試みます。詳細については、オンライン・ヘルプの GLOBALV コマンドおよび PIPE VAR ステージを参照してください。タスク・グローバル変数の値は、同じタスク (同じオペレーター ID) から呼び出されるどのプロシージャーでも更新でき、 VIEW は、プロシージャーが完了すると画面を直ちに最新表示します。

- **NetView プログラムの任意のプロシージャー**

  VIEW パネルで指定されている変数が呼び出しプロシージャーで定義されておらず、タスク・グローバル変数としても存在していないと、 VIEW は、共通グローバル変数から値を読み取ることを試みます。詳細については、オンライン・ヘルプの GLOBALV コマンドおよび PIPE VAR ステージを参照してください。NetView プログラムのどのプロシージャーでも、共通グローバル変数の値を更新できます。ただし、VIEW が画面を最新表示するのは、VIEW を開始したタスクでイベント (メッセージの受信など) が発生した場合のみです。

パネルが表示されている間、タイマー、メッセージ、またはアラートからの自動操作によってコマンド・プロシージャーが駆動され、それによって置換されたいくつかの変数が表示されたパネルに更新されます。パネルが表示される OST のもとで処理が行われると、変更された変数の新しい値でパネルを動的に更新します。

パネルの内容を見やすくし、パネルが動的に更新されている間にパネル内容を入力しやすくするには、動的に変更されるパネル上のすべての変数について、属性変数に値を割り当ててください。これによって、VIEW は、各更新ごとに画面全体の書き直しをしないで、更新済み内容だけを画面に送ることができます。

VIEW は、共通変数、タスク変数、およびローカル変数、またはそれに関連した属性変数に対する特定の変更を検出した場合、パネル全体を作成し直さなければなりません。

画面全体が再表示されたとき、オペレーターが入力した変更は再表示された画面に反映されません。以下は、こうした変更のリストです。

- 特定のデータ変数の属性変数が変更され、フィールドが保護フィールドから無保護フィールドへ、あるいは無保護フィールドから保護フィールドへ変更されたことを示している。
- 特定のデータ変数の属性変数に有効な値が入っている。値は以前には存在していなかったかあるいは無効であった。
- 特定のデータ変数の属性変更の値が存在しなくなったか、あるいは無効となった。以前の値は有効であった。
- データ変数の値が変更され、データ変数に関連している有効な属性変数がなくなった。

表示パネルで使用する変数を更新した後で VIEW コマンドの処理を続けるには、RESUME コマンドを使用します。

パネル更新の例

以降に示す図は、パネル内容の動的更新を示しています。

『パネルを更新する REXX コマンド・リストの例』 は、出荷時にサンプル CNMS1101 に組み込まれている、 RESDYN というコマンド・リストの例です。RESDYN は、VIEW コマンドでパネルに表示されるデータとして、RESOURCE コマンド出力を使用します。表示データは、コマンド・リストを呼び出すときに指定する時間間隔で更新されます。デフォルトの時間間隔は 10 秒です。この例にある RESDYN 機能 (オプション 12) 用に出された VIEW では、 NetView for z/OS バージョン 5 リリース 1 の拡張機能を使用するために EXTEND パラメーターを使っています。

パネルを更新する REXX コマンド・リストの例

```
/* -------------- Dynamic Resource Display (option 12) -------------- */
/*  A demonstration of using VIEW and TRAP to dynamically update a    */
/*  full screen display.  We use the SPILL option of pipe's KEEP      */
/*  (new for V5) to create a message after the specified refresh      */
/*  interval.  This message is TRAPped, causing VIEW to return        */
/*  control to this procedure WITHOUT removing the displayed panel.   */
/*  The 'RESUME', below is a REINVOCATION of the original VIEW!!!     */
/*                                                                    */
/*  Note that the first call to "fillVars" passes an extra little     */
/*  bit of pipe to the subroutine. The purpose is to get the first    */
/*  word of the second data line (STC name) for the panel.            */
/*                                                                    */
/* ------------------------------------------------------------------ */
resdyn:
  interval = 10              /* refresh at 10 second intervals              */
  privMsgID  = 'CNMRESDYN'   /* special purpose "msgid" for trapping        */
  getSTC = '% STC:|DROP 1|TAKE 1|EDIT W1|VAR JBN'
  Call fillVars getSTC       /* set local variables with data from RESOURCE */
  'TRAP AND SUPPRESS MESSAGES' privMsgID   /* TRAP our special message      */
  'PIPE VAR privMsgID | KEEP RESDYN' interval 'SPILL'  /* make msg later    */
  'VIEW RESDYN CNMSRESP EXTEND'
  DO WHILE (rc = 2)          /* RC indicates "message trapped"?             */
    'MSGREAD'                /* just getting msg off trap queue  */
    CALL fillVars
    'PIPE VAR privMsgID | KEEP RESDYN' interval 'SPILL'  /* make msg later  */
    'RESUME'                 /* Invoke VIEW, previously suspended           */
     /* NOTE: RC, at this point, is RC from VIEW, which was resumed.        */
  END
 'pipe hole | keep resdyn'    /* empty safe created above           */
```

```
return
/* ----------- Obtain data for RESDYN display (option 12) ----------- */
/*   Notice that the stem variable "out." is in our local variable    */
/*   dictionary.  VIEW could always read these value;                 */
/*   we will have an opportunity to update them while VIEW is active. */
/*   ---------------------------------------------------------------  */
fillVars:
  ARG xtraStg                         /* use extra first time only     */
  'PIPE (NAME RESDYN END %)',
  '| NETVIEW RESOURCE',
  '| SEPARATE DATA',                  /* No use for DSI386I title line */
  '| STC: FANOUT',                    /* MAYBE need extra copies       */
  '| EDIT SKIPTO /=/ 2.* STRIPL 1 ',
  '| COLOR WHITE',
  '| $STEM OUT.',
  xtraStg
  TM = date() time()
  $TM = 'CB HR'
return
```

図 12 は、RESDYN コマンド・リストからの出力の例です。

```
CNMSRESP NetView Resource Utilization       5 Sep 2011 15:26:41


         TOTAL CPU PERCENTAGE    =   100.00
         T510EENV CPU PERCENTAGE =   33.62
         T510EENV CPU TIME USED  =   41,175.45 SEC.
         REAL STORAGE IN USE     =   23360K
         PRIVATE ALLOCATED < 16M =   752K
         PRIVATE ALLOCATED > 16M =   23180K
         PRIVATE REGION < 16M    =   7144K
         PRIVATE REGION > 16M    =   65536K



TO SEE YOUR KEY SETTINGS, ENTER 'DISPFK'
CMD ==>
```

図 12. RESDYN コマンド・リストの出力例

71 ページの図 13 は、前出のパネル (図 12) を表示するソース・パネル・テキストです。

VIEW は、RESDYN コマンド・リストの介入なしに PF キーおよびコマンド行を管理します。

```
*** AT2 XVAR SFD
+CNMSRESP NetView Resource Utilization +    &TM
$
$
$        TOTAL CPU PERCENTAGE    =  &OUT.1
$        &JBN     CPU PERCENTAGE =  &OUT.2
$        &JBN     CPU TIME USED  =  &OUT.3
$        REAL STORAGE IN USE     =  &OUT.4
$        PRIVATE ALLOCATED < 16M =  &OUT.5
$        PRIVATE ALLOCATED > 16M =  &OUT.6
$        PRIVATE REGION < 16M    =  &OUT.7
$        PRIVATE REGION > 16M    =  &OUT.8
$
$
$
$
$
$
$
$     $Display is updated approximately every 10 seconds.
&CNMIMDL
%CMD==>~&CUR
```

図 13. CNMSRESP ソース・パネル・テキスト

ブラウズの色を変更する

『カラー属性を示す BROWSE コマンド・パネル定義』 に表示されているテンプレートは、区分データ・セットのメンバーをブラウズする際に使用されます。 46 ページの表 11 と 46 ページの表 12 に示されているカラー属性のさまざまな応用に注目してください。文字 %、$、¬、および + はそれぞれ、それらの位置のすぐ右の画面区域に特定の色を割り当てます。画面のカラー域を変更する場合は、カラー属性を変更するだけです。ただし、変更できるのは既存の属性フィールドのみです。その他のフィールドを変更すると、ブラウズの際にエラーを生じます。

カラー属性を示す BROWSE コマンド・パネル定義

```
/*******************************************************************************
/*  BROWSE Command Panel for displaying member data                             *
/*******************************************************************************
*** WIDE OPTROW=(
%NETVIEW.BRWS ------ BROWSE &BMEMBER (&BDDNAME) --- LINE &BTOP TO &BBOT OF &BTOT
¬&BMESSAGE                                                    %&BSCL     &BSC +
&BCOL                                                                       $
&BDATALINE
&BDATALINE
&BDATALINE
&BDATALINE
&BDATALINE
&BDATALINE
&BDATALINE
&BDATALINE
&BDATALINE
&BDATALINE
&BDATALINE
&BDATALINE
&BDATALINE
&BDATALINE
&BDATALINE
&BDATALINE
&BDATALINE
&BDATALINE
&BDATALINE
```

```
&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE

(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
(&BDATALINE
%CMD==>~&BCOMMAND                                                                   $
&CNMIMDL
```

第 4 章 オンライン・ヘルプ情報の変更と作成

NetView プログラムにはヘルプ機能が含まれます。これには次の 2 種類のヘルプ情報があります。

ヘルプの 1 番目のタイプは**ビュー・ベースのヘルプ** で、これは VIEW コマンドを使用すると表示されます。2 番目のタイプは**ウィンドウ・ベースのヘルプ** で、これは WINDOW コマンドを使用すると表示されます。

本章では、ヘルプ情報の追加、削除、または変更方法について説明します。説明は、ユーザーがこれを行うために使用する順序で配列されています。次のような順序になります。

1. ヘルプ・ソース・ファイルを見付ける。
2. ソース・ファイルをコピーし、変更する。
3. コピーを保管する。
4. ヘルプを表示して、変更内容をテストする。

ヘルプ・ソース・ファイルの探索

ソース・ファイルとは、表示されるパネルの内容を定義するものです。

各ヘルプ情報は別々のファイルに含まれ、区分データ・セット (PDS) のメンバーとして出荷されます。英語のヘルプ・ソース・ファイルは、NETVIEW.V6R2M0.CNMPNL1 データ・セットに保管されています。

注:

1. 日本語のヘルプ・ソース・ファイルは、NETVIEW.V6R2M0.SCNMPNL2 データ・セットに保管されています。
2. コマンド・ヘルプおよびメッセージ・ヘルプは、Web サーバーに保管されています。 NetView データ・セット・メンバーでコマンド・ヘルプおよびメッセージ・ヘルプをカスタマイズする場合は、Web サーバーのファイルにも同じ変更を加えなければなりません。

ライブラリー名を変更していないことを確認してください。

新しいヘルプ・ソースを作成する前に、作成しようとするヘルプ・ソースに類似した既存のオンライン・ヘルプを探してください。一般に、ヘルプ・ソース・ファイルを表示すると、その左上隅にファイル名が示されます。

コマンド・ヘルプ情報の場合は、HELPMAP をブラウズすることにより、変更したいソース・ファイルを見付けることができます。ウィンドウ・ベースのヘルプ・ファイルには、接頭部として文字 < が付いています。HELPMAP の詳細については、79 ページの『HELPMAP 機能』を参照してください。メッセージ・グループのヘルプ情報は、PDS のメンバーとして保管されます (それぞれのグループごとに 1 つのメンバーが保管される)。メンバー名は、メッセージ ID の最後の数字以降を切り捨てることによって判別できます。例えば、メッセージ DSI001I および DSI002I の

ヘルプは、メンバー DSI00 に保管されます。メッセージ EKGV68001I のヘルプは、メンバー EKGV6800 に保管されます。

メッセージ・ヘルプ・パネルまたはコマンド・ヘルプ・パネルが現在表示されている場合は、SHOWDATA コマンドを使用してソース・ファイルを見付けることができます。図 14 には、コマンド行に SHOWDATA を入力した後で戻される情報が表示されています。

注: 図 14 では、次のことがあてはまります。

1. このパネルは CNMPNL1 データ・セットのメンバー EUYCLIST にあります。
2. SHOWDATA コマンドからの応答にリストされている !+! は、ヘルプ検索手順による特殊な処理で生成されるもので、無視してかまいません。

```
CNMPNL1.EUYCLIST      HELP PIPE STAGES                      LINE 1 OF 41

  <         Read from a PDS member . . . . . . . . . . . . <
  $STEM     Read and set stemmed variables and attributes. $1
  $VAR      Read and set variables and attributes  . . . . $2
  BETWEEN   Divide message streams into sections . . . . . B2
  CASEI     Compare character strings  . . . . . . . . . . C3
  CHANGE    Replace string occurrences . . . . . . . . . . C12
  CHOP      Truncate lines after string  . . . . . . . . . C14
  COLLECT   Create multiline messages  . . . . . . . . . . C31
  CONSOLE   Display messages in a pipeline . . . . . . . . C32
  CORRCMD   Process a command in a pipeline  . . . . . . . C33
  CORRWAIT  Allow asynchronous messages in a pipeline  . . C34
  CONSOLE   Display messages in a pipeline . . . . . . . . C59
  DROP      Drop messages from a pipe  . . . . . . . . . . D34
  ENVDATA   Output environment data  . . . . . . . . . . . E15
  EXPOSE    Exposes messages in a pipe . . . . . . . . . . E23
  FANIN     Read from multiple input streams . . . . . . . F1
  HELDMSG   Place held messages in a pipeline  . . . . . . H18
  HOLE      Discard messages or judge correlation  . . . . H34
  INTERPRT  Build stages from data . . . . . . . . . . . . I10
  JOINCONT  Joins consecutive messages . . . . . . . . . . J1
CNMPNL1.EUYCLIST, for !+! PIPE,STAGES PIPE,COMMANDS STAGES
CMD==> showdata
```

図 14. ヘルプ・ソース・ファイル探索のための SHOWDATA コマンドの使用例

ビュー・ベースのヘルプ

ソース・ファイルには、表示されるパネルのテキストおよびそのパネルに関連した定義ステートメントが収められています。定義ステートメントには、次のものが含まれます。

- プロローグ
- ヘルプ・パネルの名前
- 継続パネルの名前
- 関連ヘルプ・パネルのリスト

ビュー・ベースのヘルプ・パネルのソースを表示するには、次のように入力します。

```
BROWSE CNMPNL1.panelid
```

ここで *panelid* は、そのヘルプのソースの左上隅に表示される名前です。詳しくは、 37 ページの『フルスクリーン・パネルの作成』を参照してください。

ウィンドウ・ベースのヘルプ

76 ページの図 15 は、ウィンドウ・ベースのヘルプ情報のソース・フォーマットの例です。番号のついたフィールドの説明が図の後にあります。

```
*** EUYRET    5697-B82 (C) Copyright IBM Corp. 2011      1
*             All Rights Reserved.
*  CHANGE ACTIVITY:
*
============== REPEAT RFIND      2
 REPEAT (BROWSE)      3

:H2. Syntax     4

 >>--REPEAT--><
:H2. IBM-Defined Synonyms

 +-----------------------------------+-----------------------------------+
 | Command or Operand                | Synonym                           |
 +-----------------------------------+-----------------------------------+
 | REPEAT                            | R or RFIND                        |
 +-----------------------------------+-----------------------------------+

:H2. Purpose of Command

 The REPEAT command reissues the last FIND command while you are browsing
 the network log or a member of a partitioned data set.  Since this
 command is sensitive to the current position of the cursor, it is
 normally entered using a PF key.

 By repeatedly pressing the PF key set to REPEAT, you can find successive
 occurrences of a specified character string.  After the first occurrence
 of a character string has been found, the REPEAT key will find the next
 occurrence.  After the last occurrence of a character string has been
 found, the REPEAT key can be used to continue the search, wrapping
 around from the bottom line to the top line (or from the top line to the
 bottom line if the FIND command included the PREV parameter.)
============== RETURN RET
 2
 RETURN (BROWSE, HELP, HELPDESK, NCCF, NLDM, NPDA, STATMON, TARA, VIEW)

:H2. Syntax

 >>--RETURN--><
:H2. IBM-Defined Synonyms

 +-----------------------------------+-----------------------------------+
 | Command or Operand                | Synonym                           |
 +-----------------------------------+-----------------------------------+
 | RETURN                            | RET (for BROWSE, HELP, HELPDESK,  |
 |                                   |     STATMON, and VIEW)            |
 |                                   |                                   |
 |                                   | R   (for NLDM, NPDA, and TARA)    |
 +-----------------------------------+-----------------------------------+

 Note:  The command facility has no synonym for RETURN.

:H2. Purpose of Command

 The RETURN command returns you to the previous component or the last
 selection panel that you used.

 You should not issue this command from a command list.
:H2. Restrictions

⋮
```

図 15. メッセージおよびコマンド・ヘルプ情報のソースの例:

メッセージおよびコマンド・ヘルプ情報のソースの例

1 プロローグ
プログラマーのコメントを入れるオプショナル・セクションです。

2 メッセージまたはコマンド
テキストが適用されるメッセージまたはコマンドです。ヘルプ情報が複数の

コンポーネントに使用されるコマンド用の情報である場合、コマンド名には接頭部として当該コンポーネントの名前が付けられます。コマンド名の前には、14 の等号 (=) とブランク・スペースを 1 つ置く必要があります。

3 メッセージまたはコマンド・ヘルプの表題

このヘルプ・ソース・ファイルの表題です。

4 タグ

情報の表示の仕方には、さまざまな方法があり、次のようなタグが使用されます。

- :H2. はコマンド名の強調表示に使用されます。
- :XMP. および :EXMP. は例の前後に使用されます。
- :IF DTYPE=PANEL および :ENDIF は、HELP によりフルスクリーン表示が行われる場合に、画面に表示されるセクションを囲むマークです。
- :IF DTYPE=MSGS および :ENDIF は、HELP により行モード表示が行われる場合に、画面に表示されるセクションを囲むマークです。これは、自動タスクで HELP が呼び出された場合、またはフルスクリーン表示がサポートされていない場合に使用されます。
- :LINK. は、あるトピックから別のトピックに移動するときに使用します。:LINK. タグは英大文字で 1 桁目から開始する必要があります。タグは、このタグが関係する表示行の前に置かれます。この行はタブ停止位置となり、WINDOW で強調表示されます。リンクのためにテキストの 2 行以上を強調表示する場合は、:LINK. タグをそれぞれの行の前に置かなければなりません。 78 ページの図 16 のコーディングの例を参照してください。

  オペレーターは、該当の行にカーソルを置くか、行を選択する FIND コマンドを出して、リンクする行を選択します。オプションで、オペレーターがこのコマンドを出すときに入力できるキーワードを、ユーザーが指定することができます。キーワードは括弧で囲み、:LINK. タグの直後に入力します。
- :CMD. は、その行を選択すると即時に実行されるコマンドの前に使用されます。コマンド行には、変数テキスト (例えば、HELP *msgno*) を入れることができます。オペレーターはこの変数テキストを特定のデータでオーバーレイした上で、Enter キーを押して、コマンドを実行することができます。:CMD. タグには終了タグの :ECMD. があり、これをコマンド・テキスト行の後に置かなければなりません。:CMD. とその終了タグは、どちらも大文字で 1 桁目から入力する必要があります。

:IF DTYPE および :LINK ステートメントのコーディング方法を示すため、78 ページの図 16 に EUYSLIST の部分が示してあります。

```
:
============== COLLECT
COLLECT (NLDM,PIPE)

COLLECT is associated with more than one NetView component.

:IF DTYPE=PANEL
Select    To Get Information About
:LINK.(A)HELP NLDM COLLECT
 A        NLDM COLLECT  Use Session Monitor to collect response time data
:LINK.(B)HELP PIPE COLLECT
 B        PIPE COLLECT  A Pipe stage which collects messages in a pipe
:LINK.(C)HELP PIPE STEM
 C  If you use the COLLECT command following a STEM command, see the
:LINK.(C)HELP PIPE STEM
    description of the COLLECT operand of the STEM command.  Enter C.
:ENDIF
:IF DTYPE=MSGS
 Enter HELP NLDM COLLECT for help on the Session Monitor COLLECT command
 Enter HELP PIPE COLLECT for help on the COLLECT pipe stage
:ENDIF
:
:
```

図 16. :IF DTYPE= および :LINK. の使用例 :

:IF DTYPE= および :LINK. の使用例

ヘルプ・ソース・ファイルのコピーおよび変更

新しいヘルプ・ソース・ファイルを作成する前に、作成しようとするヘルプ・ソース・ファイルに類似した既存オンライン・ヘルプ・ファイルを探します。 73 ページの『ヘルプ・ソース・ファイルの探索』を参照してください。

相当するパネルが見つかった場合には、画面エディターを使ってそれをコピーします。既存のテキストに上書きするか、テキストを追加して、パネルを変更してください。類似したオンライン・ヘルプ・ファイルが見付からない場合は、画面エディターを使用して新しいヘルプ・ファイルを作成します。

NetView プログラムの実行中にヘルプ・ソース・ファイルを変更または作成したい場合は、パネル・データ・セットを 2 次エクステントなしで定義してください。そのようにしないと、パネルは新しいエクステントにファイルされる可能性があり、そのパネルを使用するには NetView プログラムをクローズさせて再開する必要があります。

新しいパネルを構成する場合の規則は、既存のパネルを変更する場合と同様です。HELPMAP にリストされている完全修飾データ・セット名を使用しているのでなければ、ヘルプ・ソース・ファイルはすべて固定長のブロック化レコード・フォーマットであって、80 バイトの論理レコード長 (RECFM=FB、LRECL=80) でなければなりません。詳しくは、 79 ページの『HELPMAP 機能』を参照してください。 80 バイトの中にはヌル文字の数も入ります。また、新しく作成したパネルによって影響を受けるコマンド・リストまたは他のパネルを変更する必要もあります。

ご使用のシステムに特定のトピックを組み込むように、HELPDESK をカスタマイズすることができます。NetView プログラムには、これらのトピックを作成するために編集することができるテンプレート・ファイル CNMHDSKU が提供されています。

1. CNMHDSKU に新規トピックを追加する。
2. ファイル CNMHDSK0 の目次に新規トピック ID を追加する。

注: 既存の HELPDESK ファイル (CNMHDSK1 から CNMHDSK9) のいずれかをカスタマイズする場合は、必要な情報を別々のファイルに指定して %INCLUDE ステートメントを使用してください。そうしない場合、その情報はそれぞれのリリースごとに追加する必要があります。

ヘルプ・ファイルを作成または変更した後、そのヘルプ・ファイルは DDNAME CNMPNL1 に連結されたデータ・セットに保管します。これに代わる方法として、SMP USERMOD を使ってパネルの変更を行うことができます。詳しくは、『ヘルプ・ソース・ファイルの保管』を参照してください。

ヘルプ・ソース・ファイルの保管

ユーザーのパネル名に使用する接頭部が、NetView プログラム提供のパネル名の接頭部と同じにならないようにしてください。

作成または変更したヘルプ・ソース・ファイルはすべて保管します。ヘルプ・ファイルの保管方式には、次の 2 つがあります。

- データ・セット NETVIEW.V6R2M0.CNMPNL1 の**前に**、変更済みヘルプ・ファイルをもつユーザー区分データ・セットを NetView の始動プロシージャーの CNMPNL1 DD ステートメントで連結します。Tivoli サポート・センターでパネルを変更した場合、その変更内容はユーザーのヘルプ・ファイルには追加されません。
- 変更したヘルプ・ファイルをシステム修正変更プログラム (SMP) の USERMOD に組み込み、その USERMOD を適用すると、SMP が変更済みパネルを NETVIEW.V6R2M0.CNMPNL1 に保管できます。変更済みパネルに対して Tivoli サポート・センターがその後行う変更については、SMP が自動的にユーザーに知らせます。SMP USERMOD の使用方法について詳しくは、 *システム修正変更プログラム* のライブラリーを参照してください。

注:

1. 当製品の日本語版のデフォルト・データ・セットは、NETVIEW.V6R2M0.SCNMPNL2 です。
2. 英語のヘルプ・ソース・ファイルは、NETVIEW.V6R2M0.CNMPNL1 データ・セットに保管されています。ライブラリー名を変更していないことを確認してください。

HELPMAP 機能

HELP コマンドは、探索目標として引数を使用して、必要なコマンド・ヘルプ・メンバー名を求めるために HELPMAP を走査します。HELP は次のようにして引数を使用します。

- 引数がない場合

  引数なしで HELP を入力すると、現在使用しているコンポーネントに関するコンポーネント・レベルの HELP が得られます。

目標引数がテーブル内に見つからない場合、HELP は括弧 () の対を探索し、関連したパネル名を使います。

- 引数が 1 つの場合

  引数が 1 つだけ与えられた場合、HELP は、可能であれば引数をコマンド同義語として変換しようとします。

- 引数が 2 つまたは 3 つの場合

  引数が 2 つまたは 3 つ与えられた場合、探索目標はコンマで引数を連結して作成されます。例えば、次のようにします。

  ```
  ONE,TWO,THREE
  ```

HELPMAPU は、コマンド用に作成されたユーザー定義のヘルプ・ファイルのための特定の HELPMAP です。HELPMAP に含まれる %INCLUDE ステートメントは HELPMAPU を組み込んでおり、ユーザーが作成したヘルプ・ファイルのマッピングを行うことができます。

注: ユーザー定義のヘルプ・ファイルを HELPMAP にマップしないでください。こうした変更を行うと、IBM が HELPMAP に保守を適用する際に悪影響を及ぼします。

ヘルプの名前がどのようにリストされるかを示すため、81 ページの図 17 に CNMHELPF の部分が示してあります。接頭部に < 文字が付いているものはウィンドウ・ベースのヘルプ・ファイルで、その他のものはビュー・ベースのヘルプ・ファイルです。

```
*************************************************************************
*    5697-B82 (C) COPYRIGHT IBM CORPORATION 2011                        *
*     ALL RIGHTS RESERVED.                                              *
*    NAME(CNMHELPF)  SAMPLE(CNMHELPF)  RELATED-TO(HELPMAP)              *
*    DESCRIPTION: NETVIEW HELP MAPPINGS FOR                             *
*                 FULL BASE FUNCTION.                                   *
*                                                                       *
*************************************************************************
CNMKNEEW  ()
<EUYACQ   ACQ
<EUYACT   ACT
<EUYACION ACTION NPDA,ACTION
.
.
.
<EUYMENU  MENU NLDM,MENU NPDA,MENU TARA,MENU
<EUYMEAGE MESSAGE
<EUYMONIT MONIT STATMON,MONIT
<EUYMOOFF MONOFF STATMON,MONOFF
<EUYMONON MONON STATMON,MONON
<EUYMRENT MRECENT MR NPDA,MRECENT NPDA,MR
<EUYMSG   MSG
<EUYSLIST MVS
<EUYMVS   NCCF,MVS COMMAND,MVS
<EUYSTART MVS,START
CNMKNCCF  NCCF DSINCCF
.
.
.
```

図 17. HELPMAP の例:

HELPMAP の例

HELPMAP には、完全修飾データ・セット名を単一引用符で囲んで追加することができます。次の例を参考にしてください。

```
< 'USER.CNMPNL1(MYCMDHLP)' MYCOMAND
```

新規ヘルプ・パネルの表示

新規ヘルプ・パネルを作成した後、HELP コマンドを使用して、新規パネルとそれに関連したコマンドまたはパネルを表示し、それらが正しく表示されることを確認してください。

第 5 章 セッション・モニター・センス・コード記述のカスタマイズ

NetView プログラムでは、セッション・モニターの SENSE コマンドにより VTAM センス・コードのヘルプを提供します。2 バイトまたは 4 バイトのセンス・コードのヘルプを要求することができます。センス・コードの説明を表すのに使用される情報は、DSIPARM データ・セットにメンバーのセットとして保管されています。これらのメンバーをカスタマイズしたり、あるいは追加メンバーを組み込んで、特定のアプリケーションに意味を持つセンス・コードのヘルプを追加することができます。

セッション・モニター・センス・コード

セッション・モニター・センス・コード記述は、CNMB*nnn* という名前の DSIPARM メンバーに保管されます。ここで、*nnn* は、このメンバーにおいて記述される 2 バイトおよび 4 バイトのセンス・コードの先頭の 3 桁の 16 進数です。例えば、センス・コード 08B2 および 08B60001 のヘルプは、DSIPARM メンバー CNMB08B に保管されます。NetView プロダクトとともに出荷される CNMB08B メンバーを、 84 ページの図 18 に示します。

一般規則として、以下の規則があります。

- 記述はまず、分離文字 $$$KEY *xxxx*???? を使用して、センス・コード左端の 2 バイトによってグループ化されます。ここで、*xxxx* は左端にある 2 バイトの 16 進値です。2 バイトのセンス・コード *xxxx* (または 4 バイトのセンス・コード *xxxx*0000) の記述は、この分離文字の後にきます。
- 拡張センス・コード記述は、4 バイトのセンス・コードの右端 2 バイトで識別され、分離文字 $*nnnn* を使用してグループ化されます。ここで、*nnnn* は右端にある 2 バイトの 16 進値です。拡張された記述は、この分離文字の後にきます。
- テキスト記述は、DSIPARM メンバーの 1 から 57 桁に入れられなければなりません。このテキストでは、DBCS は使用できません。

注: 既存の DSIPARM CNMB*xxx* メンバーに対する変更は、保守または NetView プロダクトの別のリリースによって置き換えられることがあります。 DSIPARM CNMB*xxx* メンバーの始めにあるコメントを更新して変更内容を入れたり、NetView プログラム提供の DSIPARM データ・セットの前に連結したデータ・セットに、作成または変更したメンバーを格納することができます。これは、ユーザーの行った変更が以降の保守や製品変更によってオーバーレイされるのを防ぐのに役立ちます。

```
*********************************************************************
*    5697-B82 (C) COPYRIGHT IBM CORP. 2011                          *
*    DESCRIPTION:  SAMPLE  --  SENSE CODES                          *
*    CNMB08B  CHANGED ACTIVITY:                                     *
*    CHANGE CODE  DATE      DESCRIPTION                             *
*    -----------  --------  ----------------------------------------*
*********************************************************************
$$$KEY 08B2????
Data transmission failure: the data transmission between
an application program in an SNA MS entry point and an
application program in a subentry point was incomplete,
causing abnormal termination of the function.  Bytes 2
and 3 following the sense code contain sense code
specific information.
$0000
No specific code applies.
$0001
A time-out has occurred while waiting for transmission of
data between the two application programs.  For example,
a service processor has timed out while waiting to
receive data from the main processor.
$0002
A time-out has occurred while waiting for transmission of
data between two applications.
$$$KEY 08B5????
Network Node Server Not Required:  Sent by an APPN end
node control point to a network node control point (1) to
deactivate CP-CP sessions with the NNCP, or (2) to reject
a CP-CP session BIND from the NNCP.  The end node no
longer requires network node services from the receiver.
Note:  This sense data value is carried within the X'35'
control vector on an UNBIND(Type = X'01') for case (1)
above, or on an UNBIND(Type = X'FE') for case (2).
VTAM Hint: A possible cause of this error is that the
Network Node Server for the CP-CP session attempt is not
in the Network Node Server List.
$$$KEY 08B6????
CP-CP Sessions Not Supported:  Sent by a network node
control point to reject a CP-CP session BIND from another
APPN control point; support for CP-CP sessions on that TG
was removed since the time when the TG was first
activated.
Note:  This sense data value is carried within the X'35'
control vector on an UNBIND(Type = X'01').  Bytes 2 and
3 following the sense code contain sense-code-specific
information.
$0000
No specific code applies.
$0001
During link activation on a switched link, it
was discovered that the partner node does not
support CP-CP sessions on this TG.
```

図 18. CNMB08B センス・コードのヘルプ

例

以下に示すのは、DSIPARM のセンス・コード記述メンバーの追加および変更の例です。

- センス・コード 08B2 または 08B20000 のヘルプに情報を追加するには、NetView プログラム提供のヘルプを次のように変更します。

```
$$$KEY 08B2????
Data transmission failure: the data transmission between
an application program in an SNA MS entry point and an
application program in a subentry point was incomplete,
causing abnormal termination of the function.  Bytes 2
and 3 following the sense code contain sense code
specific information.
The SNA MS entry points currently defined are SYSTEM1
and SYSTEM2.
```

2 行のヘルプ情報が、ご使用のシステムに固有のセンス・コードに関して追加されました。

- 新規センス・コード 08B3 または 08B30000 のヘルプを追加するには、 NetView プログラム提供のセンス・コード 08B2 の情報の直後に次の情報を追加します。例えば、次のようにします。

```
$$$KEY 08B3????
This sense code is generated by application XYZ when a
failure occurs between components of the application.
```

2 行のヘルプ情報が、ご使用のシステムに固有のセンス・コードに関して追加されました。

- 新規センス・コード 08B60002 のヘルプを追加するには、 NetView プログラム提供のセンス・コード 08B60001 の情報の直後に次の情報を追加します。例えば、次のようにします。

```
$0002
During link activation on a switched link, it
was discovered that the partner node does not
permit sessions with this partner.
```

3 行のヘルプ情報が、ご使用のシステムに固有のセンス・コードに関して追加されました。

- 新規センス・コード 08C1*xxxx* のヘルプを追加するには、CNMB08C という名前の DSIPARM に新しいメンバーを作成し、次のステートメントを入れます。

```
$$$KEY 08C1????
This sense code is generated by application ABC when a
failure occurs in a component of the application.
The third and fourth bytes of the sense code identify
the failing component ID.
```

4 行のヘルプ情報が、ご使用のシステムに固有のセンス・コードに関して追加されました。

第 6 章 ハードウェア・モニターの表示データのカスタマイズ

本章では、総称アラートおよび非総称アラートの表示を変更する方法について説明します。以前のリリースの NetView プログラムでは、Recommended Action (推奨アクション) パネル、Event Detail (イベント詳細) パネル、およびアラート・メッセージはホストに保管されていました。各非総称アラート・パネルは、固有のパネル・セットおよびメッセージ・セットをもっていました。これらの多くは、NetView プログラムの現行リリースに残されています。総称アラートでは、総称アラート・コード・ポイントを使用してハードウェア・モニター・パネルを動的に作成することができます。

この章では、以下の事項を説明します。

- 非総称アラートの Recommended Action パネルおよび Event Detail パネル・テキストの変更方法
- 非総称アラート・メッセージの変更方法
- 総称アラートからの推奨アクション番号の書き換え方法
- ハードウェア・モニター・パネル用カラーおよび強調表示の使用の制御方法
- 総称コード・ポイントの作成と変更またはハードウェア・モニターへのリソース・タイプの追加などの、ユーザー定義のエラーを含める方法

注: ハードウェア・モニター・ヘルプ・パネルおよびコマンド記述パネル用のカラー・マップは、NetView プログラムの以前のリリースでのみ利用可能です。

パネルまたはアラート・メッセージが 2 バイト文字を必要とする言語に変換されている場合は、2 バイト文字セット (DBCS) のストリングの保全性が維持されるよう注意してください。

ハードウェア・モニター非総称パネルの変更

Recommended Action パネルと Event Detail パネルは、総称アラート・レコードに基づいていないイベント状態に関して定義されています。複数のイベント状態が同じ Recommended Action パネルまたは Event Detail パネルを使用する場合、そのパネルは単一の***実際のパネル名*** のもとで物理的に定義されます。実際のパネルが他の名前で表示される場合、その名前は***パネルの別名*** といいます。パネル名が実際の名前であるか別名であるかを判別することは、パネル・テキストの変更の第 1 ステップとなります。

パネルのテキストに変更を行うと、これらの変更はこのパネルのすべての別名に反映されます。また、パネルの別名を変更すると、前の別名をもつ新しいパネルが作成されます。

パネル名の判別

パネル名、およびそれが実際のパネル名であるか別名であるかを判別するには、変更しようとするテキストに関連するイベントを知り、その上でこのイベントがどの

リソースに対してログに記録されているかを識別する必要があります。名前のタイプを判別するための手引きとして、次のステップを使用してください。

1. リソースを識別するため、Alerts-Static、Alerts-History、または Most Recent Events パネルを表示します。
2. `sel# C` を入力します。`sel#` には、変更しようとするテキストに関連するイベントのパネル上の選択番号が入ります。メッセージ BNJ962I により、そのイベントに関連した 5 桁のコードが表示されます。メッセージ BNJ378I が表示される場合は、そのイベントは汎用イベントであり、そのイベントに関連するパネルは保管されていません。

   5 桁のコードではなく、製品 ID またはアラート ID を受け取った場合は、それに伴うレコードは総称アラートです。総称アラートには、ハードウェア・モニターに固有の事前保管パネルはありません。総称アラートの詳細については 103 ページの『ユーザー作成プログラムに対する NMVT サポートの使用』を参照してください。
3. NetView プログラムが戻す 5 桁のコード *xxxyy* を調べます。この変数の説明は次のとおりです。

   xxx リソースに対して NetView が指定した製品コードまたはブロック ID です。

   yy 個々のパネルの ID です。
4. 次のようにして、変更しようとするテキストがどのパネルに含まれているのかを判別します。
   - Recommended Action パネルの場合は、パネル名 (または、パネルの別名) は BNI*xxxyy* であり、*xxx* および *yy* はステップ 3 で識別したコードです。
   - Event Detail パネルの場合は、パネル名 (または、パネルの別名) は BNK*xxxyy* であり、*xxx* および *yy* はステップ 3 で識別したコードです。
   - 次のようにして、BNI*xxxyy* と BNK*xxxyy* のどちらがパネルの実際の名前で、どちらが別名であるのかを判別します。
     - ISPF/PDF などのエディターを使用して、パネル名のディレクトリー・リストを調べます。このリストは、NetView プログラム提供の NETVIEW.V6R2M0.BNJPNL1 という名前の区分データ・セット (PDS) 内にあります。別名の場合は、*alias* という語がパネル名の横に表示されます。
   - 実行したいアクションについては、90 ページの『パネル・テキストの変更』、 90 ページの『別名のパネルから実際のパネルへの変更』、 90 ページの『実際のパネル名または別名の削除』、または 91 ページの『実際のパネルまたは別名のパネルの追加』の該当するセクションを参照してください。

『サンプル BNJBLKID テーブル』 は、BNJBLKID テーブルの例です。

サンプル BNJBLKID テーブル

```
TITLE 'BNJBLKID:  LIST OF ALIAS TABLES BY BLOCK ID'
BNJBLKID CSECT
         EJECT
           DS  0F
NUMENT     DC  AL4((TABEND-TABSTART)/LENG)  NO. OF ENTRIES
TABSTART   EQU *
           DC  CL3'FED'
```

```
          DC  CL3'FEE'
          DC  CL3'FEF'
          DC  CL3'FE1'
          DC  CL3'FE2'
          DC  CL3'FE3'
          DC  CL3'FE4'
          DC  CL3'FFD'
          DC  CL3'FFE'
          DC  CL3'FFF'
          DC  CL3'FF2'
          DC  CL3'FF5'
          DC  CL3'FF6'
          DC  CL3'FF7'
          DC  CL3'FF8'
          DC  CL3'FF9'
          DC  CL3'GA1'
          DC  CL3'GB1'
          DC  CL3'GC1'
          DC  CL3'003'
          DC  CL3'005'
          DC  CL3'017'
          DC  CL3'02D'
          DC  CL3'02F'
          DC  CL3'021'
          DC  CL3'022'
          DC  CL3'023'
          DC  CL3'03E'
          DC  CL3'036'
          DC  CL3'037'
          DC  CL3'038'
          DC  CL3'04A'
          DC  CL3'04B'
          DC  CL3'04C'
          DC  CL3'04D'
          DC  CL3'04E'
          DC  CL3'04F'
          DC  CL3'043'
          DC  CL3'044'
          DC  CL3'047'
          DC  CL3'048'
          DC  CL3'049'
          DC  CL3'057'
          DC  CL3'47C'
TABEND    EQU *
LENG      EQU 3                    ENTRY BYTE LENGTH
          END BNJBLKID
```

『サンプル BNJALxxx テーブル』 は、BNJAL*xxx* テーブルの例です。

サンプル BNJAL*xxx* テーブル

```
      TITLE 'BNJAL036: ALIAS TABLE FOR BLOCKID 036'
BNJAL036  CSECT
        EJECT
          DS 0F
NUMENT    DC  AL4((TABEND-TABSTART)/LENG)   NO. OF PAIRS
*              REAL NAME       ALIAS NAME
TABSTART  EQU *
          DC  CL8'BNI03609',CL8'BNI0366D'
          DC  CL8'BNI03608',CL8'BNI0366C'
          DC  CL8'BNI03607',CL8'BNI0366B'
          DC  CL8'BNI03606',CL8'BNI0366A'
          DC  CL8'BNI03605',CL8'BNI03669'
          DC  CL8'BNI03605',CL8'BNI03671'
          DC  CL8'BNI03605',CL8'BNI0360D'
          DC  CL8'BNI03604',CL8'BNI03668'
          DC  CL8'BNI03604',CL8'BNI03670'
```

```
          DC  CL8'BNI03604',CL8'BNI0360C'
          DC  CL8'BNI03603',CL8'BNI03667'
          DC  CL8'BNI03602',CL8'BNI03666'
          DC  CL8'BNI03601',CL8'BNI03665'
          DC  CL8'BNI0360B',CL8'BNI0366F'
          DC  CL8'BNI0360A',CL8'BNI0366E'
          DC  CL8'BNK03609',CL8'BNK0366D'
          DC  CL8'BNK03608',CL8'BNK0366C'
          DC  CL8'BNK03607',CL8'BNK0366B'
          DC  CL8'BNK03606',CL8'BNK0366A'
          DC  CL8'BNK03605',CL8'BNK03669'
          DC  CL8'BNK03604',CL8'BNK03668'
          DC  CL8'BNK03603',CL8'BNK03667'
          DC  CL8'BNK03602',CL8'BNK03666'
          DC  CL8'BNK03601',CL8'BNK03665'
          DC  CL8'BNK0360D',CL8'BNK03671'
          DC  CL8'BNK0360C',CL8'BNK03670'
          DC  CL8'BNK0360B',CL8'BNK0366F'
          DC  CL8'BNK0360A',CL8'BNK0366E'
TABEND    EQU *
LENG      EQU 16                  ENTRY PAIR BYTE LENGTH
          END BNJAL036
```

パネル・テキストの変更

BNI*xxxyy* または BNK*xxxyy* が実際のパネル名 (別名ではない) である場合は、次のステップを実行してパネルの表現を変更してください。BNI*xxxyy* パネルの非コメント行の行数は 14 行でなければなりません。BNK*xxxyy* パネルの非コメント行の行数は 7 行でなければなりません。コメント行の 1 桁目にはアスタリスク (*) が入っています。

1. ISPF/PDF などのエディターを使用して、そのパネルをもつ PDS メンバーを編集します。PDS 名は NETVIEW.V6R2M0.BNJPNL1 (インストール時に変更されていない場合) であり、メンバー名はパネル名と同じです。
2. 変更したメンバーを保管します。

変更は、そのパネルまたはその別名を使用するすべてのイベント状態に適用されます。

別名のパネルから実際のパネルへの変更

現在別名で表示されているパネルを実際のパネルにしたい場合は、次のステップを行います。

1. ISPF/PDF などのエディターを使用して、別名のパネルを含む PDS メンバーを編集します。PDS 名は NETVIEW.V6R2M0.BNJPNL1 (インストール時に変更されていない場合) であり、メンバー名の別名はパネル名と同じです。
2. 変更したメンバーを保管します。TSO が別名のパネルを実際のパネルに変換します。

新しい実際のパネルは、以前別名であった名前で作成されています。

参照: z/OS ユーティリティーおよび JCL の詳細については、 z/OS ライブラリーを参照してください。

実際のパネル名または別名の削除

実際のパネルまたは別名のパネル名を削除する場合は、以下の 1 つを行ってください。

- 削除する実際のパネル名または別名をもつ PDS メンバーを削除します。PDS 名は NETVIEW.V6R2M0.BNJPNL1 (インストール時に変更されていない場合) であり、メンバー名はパネル名と同じです。
- ユーティリティー IEHPROGM を使用します。例えば、このユーティリティーを使って別名 BNK04B2E および BNK04B2F を削除する場合、次のようにコーディングすることができます。

```
//DELMEBR2 JOB  MSGLEVEL=(1,1)
//STEP1    EXEC  PGM=IEHPROGM
//SYSPRINT  DD  SYSOUT=A
//DS1   DD  VOL=SER=vsnum,DISP=SHR,UNIT=device_type
//SYSIN DD *
 SCRATCH VOL=device_type=vsnum,DSNAME=panel_dsname,
         MEMBER=BNK04B2E
//STEP2    EXEC  PGM=IEHPROGM
//SYSPRINT  DD  SYSOUT=A
//DS1   DD  VOL=SER=vsnum,DISP=SHR,UNIT=device_type
//SYSIN DD *
 SCRATCH VOL=device_type=vsnum,DSNAME=panel_dsname,
         MEMBER=BNK04B2F
/*
```

この例で、*device_type* はデバイス・タイプ、*vsnum* はデータ・セットが常駐するボリュームの通し番号、*panel_dsname* はそのパネルが入っているデータ・セットの名前です。

参照: z/OS ユーティリティーおよび JCL の詳細については、 z/OS ライブラリーを参照してください。

実際のパネルまたは別名のパネルの追加

BNI*xxyy* または BNK*xxyy* を新しい (または代わりの) パネル名または別名としたい場合は、次のステップを行います。

- ISPF/PDF などのエディターを使って新しいパネルを入力し、希望のパネルと似ている既存のパネルをコピーします。その後で、このコピーしたパネルを変更します。
- ユーティリティー IEBUPDTE を使って、新しいパネル名または別名を追加します。

例えば、IEBUPDTE を使って BNK04B2A の別名として BNK04B2E を追加する場合は、次のようにコーディングします。

```
//PANELS JOB MSGLEVEL=1,MSGCLASS=A
//UPDATE1 EXEC PGM=IEBUPDTE,PARM=NEW
//SYSPRINT DD SYSOUT=A
//SYSUT2 DD DSN=panel_dsname,DISP=SHR,UNIT=device_type,
//       VOL=SER=vsnum
//SYSIN DD *
./ ADD NAME=BNK04B2A
 DETAIL DESCRIPTION: THE ERROR ANALYSIS MICROCODE
 HAS DETECTED AN INVALID ERROR LOG ENTRY.



     LOG ENTRY 0-3       4-7               8-11
**************************************************
                                .
                                .
                                .
```

```
*
*
*
***************LAST LINE OF PDS MEMBER************
./  ALIAS  NAME=BNK04B2E
/*
```

この例で、*panel_dsname* はそのパネルが保管されているデータ・セットの名前、*vsnum* はそのデータ・セットが常駐するボリュームの通し番号です。この例では新規別名を 1 つだけ定義していますが、別名は最大 15 個まで有効です。

参照: z/OS ユーティリティーおよび JCL の詳細については、 z/OS ライブラリーを参照してください。

非総称アラート・メッセージ

総称アラートと関連していない Alerts-Static、Alerts-History、Alerts-Dynamic、Event Detail、または Most Recent Events パネルの Event Description: Probable Cause (イベント記述: 推定原因) テキストを変更する場合は、次のステップを行ってください。

1. そのテキストに関連するイベントを判別し、そのイベントがどのリソースに対してログに記録されているかを識別します。
2. ステップ 1 で識別されたリソースの場合は、Alerts-Static、Alerts-History、Alerts-Dynamic、Event Detail、または Most Recent Events パネルを表示します。
3. `sel# C` と入力します。`sel#` には、変更しようとするテキストに関連するイベントの選択番号が入ります。メッセージ BNJ962I により、そのイベントに関連した 5 桁のコードが表示されます。メッセージ BNJ378I が表示される場合、そのイベントは汎用イベントです。

   5 桁のコードではなく、製品 ID またはアラート ID を受け取った場合は、それに伴うレコードは総称アラートです。総称アラートには、ハードウェア・モニターに固有の事前保管された Event Description: Probable Cause (イベント記述: 推定原因) テキスト・メッセージはありません。総称アラートの詳細については 103 ページの『ユーザー作成プログラムに対する NMVT サポートの使用』を参照してください。
4. NetView プログラムが戻す次の 5 桁のコード *xxxyy* を調べます。

   xxx リソースに対して NetView が指定した製品コードまたはブロック ID です。

   yy 個々の 16 進数のパネル ID です。
5. 変更したいテキストを含む CSECT を検索し編集する場合は、ISPF/PDF などのエディターを使用します。CSECT の名前は BNJVM*xxx* (NETVIEW.V6R2.BNJSRC1 の PDS メンバー) であり、ここで、*xxx* はステップ 4 で判明したブロック ID です。
6. BNJVM*xxx* 内でメッセージ・テキストを探します。このテキストに対するメッセージ番号は、*yy* と同等の 10 進数です。*yy* は、ステップ 4 で判別した 16 進数の ID です。
7. アセンブラー言語マクロ DSIMDS を変更します。

参照: 変更したいテキストに関する DSIMDS の構文については、「*IBM Tivoli NetView for z/OS プログラミング:アセンブラー*」を参照してください。

8. 変更された CSECT を保存します。
9. CSECT を再アセンブルし、同じ名前のロード・モジュールに CSECT をリンク・エディットします。

ACTION コマンド・リストの使用

ハードウェア・モニターに表示される推奨アクションの詳細は、ACTION コマンド・リストを使用して取得することができます。ACTION コマンド・リストによって表示される Action Help パネルの変更方法については、73 ページの『第 4 章 オンライン・ヘルプ情報の変更と作成』を参照してください。D*nnn*、E*nnn*、および I*nnn* は、Recommended Action パネルに表示される推奨アクションの番号です。R*nnn* 番号は、解決アクション・パネルに示されるアクションです。ACTION コマンド・リストがこれらの推奨アクション番号とともに表示する記述は、次の説明のとおりです。

ACTION Dnnn
NetView プログラムに付属する推奨アクションの詳細な説明が表示されます。

ACTION Ennn
ユーザー定義の総称アラート・アクション用にご使用のシステムのシステム・プログラマーが作成した推奨アクションの説明が表示されます。

ACTION Innn
NetView プログラムに付属する総称アラート・アクション用に作成された推奨アクションの説明が表示されます。

ACTION Rnnn
NetView プログラムに付属する解決アクション用に作成された実際のアクションの説明が表示されます。

推奨アクション番号の書き換え

特定の総称アラートに対して表示される Recommended Action は送信側製品によって細部が異なるため、考えられるすべての一般的なアクションに対して Action Help パネルを用意することは不可能です。このため、総称アラート用に作成された NetView Action Help パネルでは、各推奨アクションの前に I 番号 (NetView プログラム提供のアクション) または E 番号 (ユーザー提供のアクション) が置かれています。

ハードウェア・モニターの推奨アクション・パネルでは、各種の推奨アクションが特別なアクション番号によって識別されています。94 ページの図 19 は、3 つの推奨アクション (D225、D001、および D238) を示す Recommended Action パネルのサンプルの図です。

```
N E T V I E W          SESSION DOMAIN: CNM01    OPER1     05/17/10 14:40:53
NPDA-45A           * RECOMMENDED ACTION FOR SELECTED EVENT *     PAGE  1 OF  2
 CNM01       CENTRAL    LN08PTP    PU32768
             +--------+           +--------+
 DOMAIN      |  COMC  |----LINE----|  CTRL  |
             +--------+           +--------+


 USER    CAUSED - LSL 2 REMOTE DSU/CSU IN TEST MODE
                  LSL 2 REMOTE DSU/CSU IN CONFIGURATION MODE
                  LINE SWITCHED TO INCORRECT POSITION
        ACTIONS - D001 - CORRECT THEN RETRY

 INSTALL CAUSED - LSL 2 REMOTE DSU/CSU ADDRESS INCORRECT
                  LSL 2 DSU/CSU'S SPEED MISMATCH
                  PHYSICAL LINE CONNECTIONS
        ACTIONS - D225 - CORRECT ADDRESS FROM DSU/CSU CONTROL PANEL
                  D001 - CORRECT THEN RETRY
                  D238 - PERFORM REMOTE DSU/CSU PROBLEM DETERMINATION

 ENTER ST (MOST RECENT STATISTICS), DM (DETAIL MENU), OR D (EVENT DETAIL)

 ???
CMD==>
```

図 19. 選択されたイベントについての Recommended Action (推奨アクション) パネル:

選択されたイベントについての Recommended Action (推奨アクション) パネル

I 番号と E 番号のアクションには、NetView プログラム提供の関連付けられたパネルはありません。しかし、NetView プログラムにより、ユーザーは I 番号と E 番号をアクション番号で書き換えることができ、送信側製品に特有のパネルを作成することができます。

これは、アクション番号をもつ製品設定 ID と相関関係のあるテーブル BNJDNUMB、またはアクション番号をもつ製品共通名と相関関係のあるテーブル BNJDNAME のどちらかを変更して行うことができます。BNJDNUMB は、BNJDNAME の前に探索されます。

NETVIEW.V6R2M0.BNJPNL2 のテーブル BNJDNUMB または BNJDNAME を変更し、PDS メンバー BNJ*wwwww* を作成します。

BNJDNUMB、BNJDNAME、および BNJwwwww の変更

このセクションでは、区分データ・セット (PDS) メンバーを示すために名前 BNJDNUMB と BNJ*wwwww* を使用します。

BNJDNUMB

BNJDNUMB は、プロダクト・セット識別 (PSID) とこの製品用に使用されるアクション番号が入っている固有のファイルまたは PDS メンバー (BNJ*wwwww*) とを関連づけます。BNJDNUMB を変更する場合は、ISPF/PDF などのエディターを使用します。

注: PSID が BNJDNUMB に存在せず、また製品共通名が BNJDNAME に存在しない総称アラートを NetView プログラムが受け取ると、デフォルトの I 番号または E 番号は変更されません。

BNJDNUMB の形式は次のとおりです。

```
xxx
yyyyyyyyy BNJwwwww     comment
    .         .            .
    .         .            .
    .         .            .
```

各項目の意味は以下のとおりです。

xxx BNJDNUMB 内の項目数です。この番号は 1 桁目から始まる 3 文字で指定します (必要な場合は先行ゼロを置きます)。

yyyyyyyyy
PSID を表す文字を指定します (最大 9 文字)。この項目は 1 桁目から始める必要があります。

BNJ*wwwww*
PDS メンバーの名前が 11 桁目から始まる場合は、総称アラート推奨アクション・コード・ポイントおよび関連したアクション番号を含んでいます。例えば、BNJDNUM2、BNJDNUM3 などの名前が推奨される名前です。しかし、固有の名前を使用してもかまいません。BNJDNUM1 はすでにハードウェア・モニターが作成する総称アラートで使われています。

BNJDNUMB 内の項目は昇順で指定しなければなりません。コメント行の 1 桁目にはアスタリスク (*) が入っています。

PSID の決定方法: 送信側製品は、ハードウェア製品であることもソフトウェア製品であることもあるため、PSID の定義は次のように行います。

- ハードウェア製品の場合、PSID は、X'00' サブフィールドにあるマシン・タイプを示す 4 つの数字 (ハードウェア製品 ID。総称アラートの最初の X'10' サブベクトルの最初の X'11' サブベクトルにあります) により定義されます。
- ソフトウェア製品の場合、PSID は、X'02' サブフィールドにある 9 文字の大文字の英数字で構成されるサービス可能コンポーネント ID (ソフトウェア製品サービス可能コンポーネント ID。総称アラートの最初の X'10' サブベクトルの最初の X'11' サブベクトルにあります) により定義されます。

  注: X'02' サブベクトルが存在しない場合は、X'08' サブベクトルにある 7 文字の大文字の英数字で構成されるライセンス・プログラム番号 (ソフトウェア製品プログラム番号。総称アラートの最初の X'10' サブベクトルの最初の X'11' サブベクトルにあります) を使用してください。

ハードウェア・モニター・データベースのログに記録されている総称アラートの PSID を決定するには、次の 2 種類の方法があります。

- Alerts-Static、Alerts-History、または Most Recent Events パネルから `sel# C` を選択して、その PSID をもつメッセージを表示します。
- Event Detail メニューから選択して、PSID パネルの 1 ページ目を表示します。このパネルは送信側 PSID を表示します。

BNJDNAME

BNJDNAME は、製品共通名と、この製品を使用するためのアクション番号を含む固有ファイルまたは PDS (BNJ*wwwww*) を相関させます。BNJDNAME を変更する場合は ISPF/PDF などのエディターを使用します。

BNJDNAME の形式は次のとおりです。

```
xxx
yyyyyyyyyyyyyyyyyyyyyyyyyyyyyy BNJwwwww     comment
```

各項目の意味は以下のとおりです。

xxx BNJDNAME 内の項目数です。この番号は 1 桁目から始まる 3 文字で指定します (必要な場合は先行ゼロを置きます)。

yyy...y ソフトウェア製品共通名を表す 30 文字までを指定するか、あるいはハードウェア共通名を指定する 15 文字までを指定します。

BNJ*wwwww*
PDS メンバーの名前が 32 桁目から始まる場合は、総称アラート推奨アクション・コード・ポイントおよび関連したアクション番号を含んでいます。BNJDNUM2、BNJDNUM3 などの名前が推奨される名前です。しかし、固有の名前を使用してもかまいません。BNJDNUM1 はすでにハードウェア・モニターが作成する総称アラートで使われています。

comment
コメントは 45 桁目から始まらなければなりません。

NetView プログラムは、この PDS メンバーに次のデータを提供します。

サンプル BNJDNAME テーブル

```
001
NETVIEW                          BNJDNUM1     NETVIEW PRODUCT
```

製品共通名の決定方法: 送信側製品は、ハードウェアであることもソフトウェアであることもあるため、製品共通名は次のように定義されます。

- ハードウェア製品の場合、ハードウェア共通名は、X'0E' サブフィールドにある EBCDIC 文字 (ハードウェア製品共通名。総称アラートの最初の X'10' サブベクトルの最初の X'11' サブベクトルにあります) により定義されます。
- ソフトウェア製品の場合、ソフトウェア共通名は、X'06' サブフィールドにある EBCDIC 文字 (ソフトウェア製品共通名。総称アラートの最初の X'10' サブベクトルの最初の X'11' サブベクトルにあります) により定義されます。

ハードウェア・モニター・データベースのログに記録されている総称アラートの製品共通名を判別するには、Event Detail メニューから 2 を選択します。この選択により、送信側製品の共通名 (ハードウェアまたはソフトウェア) が表示されます。

BNJwwwww

各 BNJ*wwwww* メンバーは総称アラート推奨アクション・コード・ポイントおよび関連したアクション番号を含んでいます。BNJDNUMB テーブルで指定された BNJ*wwwww* ファイルまたはメンバーを作成する場合は、ISPF/PDF などのエディターを使用してください。各 BNJ*wwwww* PDS メンバーは、DD ステートメント BNJPNL2 の連結ストリング内の最初のデータ・セットに保管するようにしてください。この DD ステートメントは、NetView 始動プロシージャーに入っています。

NetView プログラムが実行している間は、パネルを変更または作成する際に 2 次エクステントをもつパネル・データ・セットを定義しないようにしてください。NetView プログラムが実行している間に 2 次エクステントを定義すると、2 次エク

ステント障害が発生し、エラー・リカバリーと要求の単一インスタンスの消失の原因となります。要求を実行する 2 番目の試みがなされると、エラー・リカバリーは要求の実行時に成功します。しかし、データ・セット全体について NetView プログラムの再生処理が必要になります。

BNJ*wwwww* の形式は次のとおりです。

```
xxxx      yyyyyyyy  dnum
 .          .       .
 .          .       .
 .          .       .
```

各項目の意味は以下のとおりです。

xxxx 4 文字の総称アラート推奨アクション・コード・ポイント (総称アラート体系により定義された推奨アクション・コード・ポイントの EBCDIC 版) です。このフィールドは 1 桁目から始めてください。

yyyyyyyy
8 文字のアラート ID 番号です (X'92' サブベクトル体系で定義されたアラート ID の EBCDIC 版)。このフィールドはオプションです。このフィールドを指定する場合は、11 桁目から始めてください。

dnum 4 文字の固有アクション番号です。このフィールドは 21 桁目から始めてください。アクション番号は、EBCDIC 文字の任意の組み合わせで指定することができます。指定できるアクション番号はこれらの 4 文字を使用して関連するパネルを表示する ACTION コマンド・リストの機能によって制約されます。

各 BNJ*wwwww* ファイルまたはメンバー内の項目は 16 進数の昇順で指定してください。 16 進数でないものを使用すると、その数はスキップされます。

BNJDNUMB または BNJDNAME に指定された BNJ*wwwww* ファイルまたはメンバーは、一致するものが検出されるまで、またはファイルの最後に達するまで、順次に検索されます。最初の * が 1 桁目で検出されると、順次検索は停止します。

特定のアクション・コード・ポイントに対するアラート ID フィールドには、固有のアラート ID とともに、ブランクを置くことができます。

図 20 は、サンプル BNJ*wwwww* ユーザー定義テーブルを示したものです。

```
1002               D562
1002    93987791   D890
1002    D2556B79   D777
```

図 20. サンプル BNJwwwww ユーザー定義テーブル

アラート D2556B79 では、コード・ポイント 1002 はそのアクション番号として D777 を使用します。アラート 93987791 では、コード・ポイント 1002 はそのアクション番号として D890 を使用します。この送信側製品からのその他のアラートの場合はすべて、コード・ポイント 1002 はそのアクション番号として D562 を使用します。

ハードウェア・モニター・パネルのカラーおよび強調表示の変更

ハードウェア・モニター表示画面では、表示テキストのカラー、強調表示、および輝度を変更することができます。また、表示装置が音響アラームを出すようにすることもできます。 NetView プログラムにより割り当てられるこれら 4 種の属性を変更する場合は、表示画面ユーザーの要望を考慮してください。

注: 属性の長さ、行の位置、桁の位置は変更しないでください。変更結果は予測できません。

表示テキストのストリングの前にブランクがある場合は、そのストリングの属性 (最大で 4 種類) を変更することができます。

カラー テキストは、赤、黄色、青、白、緑、青緑色、または桃色で表示されます。

強調表示
テキストは、下線表示、明滅、または反転表示されて表示されます。

輝度 テキストが高輝度になります (単色の端末の場合のみ)。

アラーム
テキストの表示にともなってユーザー端末の音響アラームが鳴ります。

これらの属性は、特定の表示装置またはすべての表示装置に対して変更することができます。例えば、すべての表示装置のプロンプト行に同じカラーを選択することができます。

これらの属性を変更するプロシージャーは、カラー・マップから始まります。カラー・マップは、種々の属性を表している文字をカラー・バッファーに組み込むためのテーブルです。カラー・バッファーに入れられた文字は、テキストの外観を制御します。

自動化テーブルを使用して、ハードウェア・モニター表示装置の特定のアラートのカラーと強調表示を設定または変更することもできます。

参照: 詳細については、「*IBM Tivoli NetView for z/OS 自動操作ガイド*」を参照してください。

カラー・マップの選択

ハードウェア・モニター表示画面を変更するための第 1 ステップは、変更しようとする表示画面を制御するカラー・マップを決定することです。 213 ページの『付録 A. ハードウェア・モニター・パネルのカラー・マップ』には、パネル名、パネル番号、およびハードウェア・モニター・パネル用のカラー・マップの表が示されています。

必要なカラー・マップを識別した後で、そのマップを ISPF/PDF などのエディターで編集してください。カラー・マップを含んでいる PDS の名前は NETVIEW.V6R2M0.BNJPNL2 です (インストール時にその名前が変更されていない場合)。メンバー名はそのカラー・マップ名と同じです。

注: 特定の属性を各パネルの同じ部分に適用させたい場合は、カラー・マップ BNJOVERW を変更して他のすべてのパネル特有のカラー・マップに上書きしま

す。BNJOVERW の結果は、実動システムでの使用の前に必ず各パネルでテストしてください。このマップにより予期せぬ結果が生じることがあります。

カラー・マップの変更

カラー・マップを選択した後で、それを変更することができます。カラー・マップには、マップ・エレメントと呼ばれる一連のデータ行が収められています。カラー・マップの先頭行には、常に、その後に続くマップ・エレメントの数が表示されます。マップ・エレメントは 1 桁目から始まり、41 桁目から始まるコメントと対になっています。

各マップ・エレメントは、特定の表示画面の行、属性、行内での属性の位置、および文字長に対する指定をします。マップの各項目の後にはコンマが続き、最後の項目の場合はコンマでなくピリオドになります。

注: 属性の長さ、行の位置、桁の位置は変更しないでください。変更結果は予測できません。

『サンプル・カラー・マップ』 は、サンプル・カラー・マップの図です。

サンプル・カラー・マップ

```
13, 1                       NUMBER OF ELEMENTS IN TABLE
1,1,1,79,BLU, 2             NETVIEW HEADER
1,2,1,14,BLU,               SCRN ID
2,2,16,64,HIG,WHI,          SCRN TITLE
1,3,1,7,BLU,                DOMAIN
1,3,9,71,TUR,
1,5,1,79,BLU,               HEADING
99,SIZE-0-7,2, 3            REPETITION
2,6,1,4,HIG,WHI,            SEL #
1,6,6,74,TUR,               DATA
1,SIZE-4,1,50,BLU, 4        PROMPT LINE
2,SIZE-4,52,1,HIG,WHI,      PROMPT LINE
1,SIZE-4,54,26,BLU,         PROMPT LINE
1,SIZE-3,1,79,BLU.          PROMPT LINE
```

1 カラー・マップの最初の項目は、後に続くデータの行、またはマップ・エレメントの数を表しています。1 つのマップには、任意の数のマップ・エレメントを指定することができます。サンプル・マップでは、マップ・エレメント数は 13 です。

2、**3**、および **4** では、次に示す 3 種類のマップ・エレメントを記述しています。

2 このタイプのマップ・エレメントには、次の形式の属性情報が入ります。

- 最初の項目には、そのマップ・エレメント内の属性の数が入ります。属性数は 1 から 4 の値です。1 つのマップ・エレメントは 1 つの属性セットのみもつことができます。例えば、桃色のカラー、または属性の任意の組み合わせ (桃色のカラーと下線表示など) となります。サンプルのマップ・エレメントは、属性を 1 つ (青色 (BLU) のカラー) もっています。
- 2 番目の項目には、その属性を反映させる表示画面上の行番号が入ります。サンプルでは、属性は 1 行目に表示されます。

- 3 番目の項目には、その属性文字をもつ表示画面上の桁の番号が入ります。サンプルでは、属性文字は 1 桁目に配置されるため、表示テキストは 2 桁目から始まります。

  注: 変更しようとする表示テキストの前にブランク・スペースがあることを確認してください。ブランク・スペースがないと、カラー・バッファーで属性を表している文字によって表示テキストの文字の一部が上書きされ、一部の文字がブランクになります。例えば、次のようなストリングでは、コロンをテキストと別のカラーにすることはできません。

  ```
  EVENT DESCRIPTION:PROBABLE CAUSE
  ```

- 4 番目の項目には、その属性の最大文字長が入ります。サンプルでは、指定された属性は、画面上の 79 文字 (つまり 2 桁目から 80 桁目) に適用されます。
- 最後の項目には、属性または複数の属性を並べたものが入ります。サンプルでは、青色のカラーが指定された属性です。属性は 4 つまで指定することができます。ただし、各カテゴリーについては 1 つの属性しか指定できません。同じ文字またはストリングに複数の属性を適用したい場合は、各カテゴリーごとに次の順序で属性を指定しなければなりません。
  1. アラーム。ALM と指定すると音響アラームが鳴ります。
  2. 輝度
     - HIG と指定するとカラーが高輝度になります。
     - NOH と指定するとカラーが通常の輝度に戻ります。
  3. 強調表示
     - UND と指定すると文字またはストリングに下線が引かれます。
     - BLI と指定すると文字またはストリングが明滅表示されます。
  4. カラー
     - RED と指定すると赤色になります。
     - YEL と指定すると黄色になります。
     - BLU と指定すると青色になります。
     - WHI と指定すると白色になります。
     - GRE と指定すると緑色になります。
     - TUR と指定すると青緑色になります。
     - PIN と指定すると桃色になります。

このマップ・エレメントにより、1 行目の 2 桁目から 80 桁目のテキストが青色になります。このマップ・エレメントに関するコメントに書かれているように、この青色のストリングのテキストは表示画面のヘッダーになります。

3 このタイプのマップ・エレメントは、反復因数オプションを使用して、特定の行に対して指定された属性 (1 つまたは複数) を後続の行にコピーします。この反復マップ・エレメントでは、次の形式を使用します。

- 番号 99 は、エレメントを反復することを示します。
- SIZE-*x*-*y* の説明:
  - SIZE はそのパネル内の総行数を表します。SIZE という単語はそのまま使用します。これを数字で置き換えないでください。

– *x* は、パネル・データの最後とプロンプト行の間の未使用のまたはブランク行の数を表します。このサンプルでは、パネル・データの最後とプロンプト行の間にはブランク行または未使用行がないことを表しています。
– *y* は、前の行の属性 (1 つまたは複数) がコピー、すなわち反復されるストリングの最初の行の番号を示します。サンプルでは、6 行目の属性が、それに続く 7 行目以降の行で繰り返されることを表しています。

- 最後の項目 (2) は、反復される 6 行目の属性の数を示します。サンプルでは、6 行目にマップされている 2 つの属性が反復されることになります。

サンプル・カラー・マップでは、このマップ・エレメントにより、6 行目に指定された 2 つの属性が、7 行目からプロンプト行までの後続行にコピーされます。

4 このタイプのマップ・エレメントは、変数行配置オプションを使用してその属性をもつ行を指定します。このオプションには次の形式を使用します。

- 最初の項目 (1) は、マップ・エレメント内の属性の数を示します。属性数は 1 から 4 です。このサンプルでは、このマップ・エレメントは 1 つの属性 (青色 (BLU) のカラー) をもちます。
- 2 番目の項目 (SIZE-*x*) には、その属性を反映する表示画面上の行が示されます。内容は次のとおりです。
  – SIZE はその表示内の総行数を表します。SIZE という単語はそのまま使用します。これを数字で置き換えないでください。
  – *x* には、コマンド行までの行数が入ります。例えば、Alerts-Static (アラート静的) の表示画面の場合は次のようになります。
    - SIZE-4 は最初のプロンプト行を表します。
    - SIZE-3 は 2 番目のプロンプト行を表します。
    - SIZE-2 はメッセージ行を表します。
    - SIZE-1 は NetView 状況表示行を表します。
    - SIZE-0 はコマンド行を表します。

    サンプルでは、属性は最初のプロンプト行に表示されます。

    注: コマンド行が NetView 状況表示行の 80 バイト目に定義されていることを確認してください。80 バイト目に定義されていないと、一部のバイトが上書きされます。
- 3 番目の項目 (1) は、その属性文字をもつ表示画面上の桁番号を示します。サンプルでは、属性文字は 1 桁目に配置されるため、表示テキストは 2 桁目から始まります。

  注: 変更しようとする表示テキストの前にブランク・スペースがあることを確認してください。ブランク・スペースがないと、カラー・バッファーで属性を表している文字によって表示テキストの文字の一部が上書きされ、一部の文字がブランクになります。
- 4 番目の項目 (50) は、属性の最大文字長を示します。サンプルでは、指定された属性は画面上の 50 文字に適用されます。
- 最後の項目 (BLU) には、属性または複数の属性を並べたものが示されます。属性は 4 つまで指定することができます。ただし、各カテゴリーについては 1 つの属性しか指定できません。同じ文字またはストリングに対して複数の属性を設定

したい場合は、 99 ページの『サンプル・カラー・マップ』 ページに示されている順序で属性を指定する必要があります。サンプルでは、青色のカラーが指定された属性です。

このサンプルのマップ・エレメントでは、最初のプロンプト行の 2 桁目から 51 桁目のテキストが青色になります。

プロンプト強調表示トークン

プロンプト強調表示トークン・テーブル BNJPROMP は、NETVIEW.V6R2M0.BNJPNL2 という名前の PDS の中にあります。このテーブルは変更することができます。このテーブルの最大サイズは、15 バイトの文字フィールドで構成されるプロンプト 25 個分です。このテーブルを変更する場合は、そのテーブルに関する注釈用のコメント欄を利用してください。性能上の理由により、このテーブルは Alert Dynamic (アラート動的) パネルを作成中は処理されません。カラーは 20 桁目から始まる 3 バイトの文字フィールドです。選択できるカラーは、カラー・マップで有効なカラーです。表 15 は、プロンプト強調表示トークン・テーブルの形式のサンプルを示します。

表 15. プロンプト強調表示トークン

プロンプト・トークン	カラー	コメント
SEL#	WHI	PROMPT SEL#
LDM	WHI	PROMPT LDM
LSL1	WHI	PROMPT LSL1
LSL2	WHI	PROMPT LSL2
RESNAME	WHI	PROMPT RESNAME
RESNAME1	WHI	PROMPT RESNAME1
RESNAME2	WHI	PROMPT RESNAME2
'A'	WHI	PROMPT A
'B'	WHI	PROMPT B
'P'	WHI	PROMPT P
'EV'	WHI	PROMPT EV
'ST'	WHI	PROMPT ST
'DM'	WHI	PROMPT DM
'M'	WHI	PROMPT M
'DEL'	WHI	PROMPT DEL
'S'	WHI	PROMPT S
'D'	WHI	PROMPT D
'R'	WHI	PROMPT R

このテーブルは、初期設定時にストレージに読み込まれます。プロンプト強調表示トークンは、再定義したり、最大 25 個まで新しく追加したりすることができます。初期設定時にテーブルが正しく読み込まれないと、メッセージが出されます。

ユーザー作成プログラムに対する NMVT サポートの使用

ネットワーク管理ベクトル転送 (NMVT) サポートを使用すると、ユーザー作成プログラムによって総称アラートを使ってハードウェア・モニターにエラーを報告することができます。総称アラートの前に、Recommended Action パネル、Event Detail パネル、およびアラート・メッセージが NetView プログラムのホストに保管されます。各非総称アラート・パネルは、固有のパネル・セットおよびメッセージ・セットをもっていました。

注: 元の NMVT のエンコードにはアラートを含む多くの SNA 主ベクトルを含んでいます。MDS_MU と CP_MSU のような以降のエンコードには同じ主ベクトルが多く含まれ、このセクションの用語 NMVT でカバーされます。

コーディングされた総称アラートは、NMVT に収められています。総称アラート・コード・ポイントは動的にハードウェア・モニター・パネルを作成するために使用されます。非総称アラートは、主としてマイグレーションのために使用されます。新しいユーザー定義のアラートを作成する場合は、総称アラートを使用してください。

参照: NMVT の主ベクトルおよびサブベクトルの詳細は、 *SNA* ライブラリーを参照してください。

このセクションでは、サンプル総称アラート、およびハードウェア・モニターによって作成された関連パネルについて説明します。(106 ページの図 21 から 111 ページの図 25 を参照してください。) このセクションでは、各パネルがどのように作成されるかについても説明します。

ユーザー定義アラート (非総称)

ユーザー定義のアラートを生成するために、NMVT の主ベクトル X'0000' の一部である 16 個のブロック ID (X'F00' から X'F0F') が予約されています。

ハードウェア・モニターは、NMVT の最初の製品 ID (X'11') サブベクトルのソフトウェア製品プログラム番号 (X'08') サブフィールド内の対応する 7 文字のソフトウェア ID として使用するために、USER0xx から USERFxx を予約しています。ここで、xx はブランク・スペース X'40' 文字で、名前を 7 文字にする埋め込みに使用されます。これらは、ブロック ID X'F00' から X'F0F' までにマップされます。

ハードウェア・モニターにより、NMVT の基本アラート (X'91') サブベクトル内で 1 バイトのアラート記述コードを使用することができます。このコードにより、そのアラートをより詳しく記述することができます。アラート記述コードは、2 バイトのアラート記述コード・フィールドの 2 バイト目に置いてください。ハードウェア・モニターはそのフィールドの 1 バイト目を無視します。

NMVT からパネル ID へのマッピング

ソフトウェア製品プログラム番号およびアラート記述コードから得られたブロック ID を使用することにより、ハードウェア・モニターは NMVT を以下のものへマップします。

- 14 行パネル

14 行パネルは、NMVT に関するハードウェア・モニターの Recommended Action パネル上に表示されます。 この 14 行パネルの PDS メンバー名は、BNIF00*xx* から BNIF0F*xx* までの範囲になります。ブロック ID の範囲は X'F00' から X'F0F' で、*xx* は 16 進値のアラート記述コードです。行の長さは最大 80 文字です。

- 7 行パネル

  7 行パネルは、NMVT に関するハードウェア・モニターの Event Detail パネル上に表示されます。7 行パネルの PDS メンバー名は、BNKF00*xx* から BNKF0F*xx* までの範囲です。ブロック ID の範囲は X'F00' から X'F0F' で、*xx* は 16 進値のアラート記述コードです。

  最初の 3 個の X'A0' 修飾子サブベクトルまたは X'A1' 修飾子サブベクトルの最初の 8 個の変換済みの文字が、Event Detail パネルの直後の 8 行目に表示されます。7 行目のタイトルを使って Event Detail メッセージを書き込み、修飾子の説明を記述してください。

- 48 バイト・アラート記述

  48 バイトのアラート記述は、Alerts-Dynamic、Alerts-Static、Alerts-History、Event Detail、および Most Recent Events パネルに表示されます。ブロック ID に関するこの 48 バイト・テキストの記述は、BNJVMF00 から BNJVMF0F までのリンク・エディット・ロード・モジュール名をもつ NetView メッセージ CSECT に収められています。

パネル・フォーマット

それぞれの新しい Recommended Action パネルまたは Event Detail パネルには、既存のパネルと同じ形式を使用し、パネルを NetView パネル・ライブラリーまたは連結されたユーザー・ライブラリーに追加してください。

新しい 48 バイト・アラート記述 CSECT には、既存の BNJVM*xxx* CSECT と同じ形式を使用してください。BNJVM*xxx* CSECT は、マクロ DSIMDS を使用してコーディングされます。48 バイト・アラート記述に対しては、変数置換は利用できません。

ユーザー定義アラート (総称)

総称アラートでは、コーディングされたアラート・データをアラート内に移送することができ、パネルを保管させる必要がありません。コーディングされるデータは、次のいずれかです。

- 事前定義テーブルへの索引。パネルを作成するために使用されている短いテキスト単位をもっています。
- パネルに直接現れるテキストのデータ。

コード化データはカスタマイズすることのできるコード・ポイント・テーブル内で保守されます (コード・ポイント・テーブルをカスタマイズする際の情報は、 113 ページの『総称コード・ポイント・テーブルの変更』を参照してください)。コード・ポイントで索引付けされているテキスト・ストリング、およびアラートに送られたテキストのデータの表示は、どの製品がそのアラートを送っていても同じ形式

になります。また、各製品は Tivoli が定義した用語を使用しているため、異なる製品であっても類似の問題の定義には同じ用語が使われています。

総称アラートが生成する Alerts、Recommended Action、および Detail panels は、ハードウェア・モニターの非総称アラート・サポートと同じですが、保管されたパネルを使用するのではなく、パネルが動的に作成されます。コード・ポイントは、Tivoli およびユーザーにより定義されたテーブルへの索引となります。

アラート記述および推定原因コード・ポイントは、ハードウェア・モニターの Alerts-Dynamic、Alerts-Static、Alerts-History、Event Detail、および Most Recent Events パネルの作成用に使用されます。ユーザー原因、インストール原因、障害原因、および推奨アクション・コード・ポイントは、ハードウェア・モニターの Recommended Action パネルの作成用に使用されます。明細データ・コード・ポイントは、ハードウェア・モニターの Recommended Action または Event Detail パネルに表示できる修飾子を識別するために使用されます。Alerts、Recommended Action、および Detail パネルを定義する場合、製品は、製品に依存しない同じアーキテクチャー準拠の用語セットを使います。NMVT に移送されたテキスト・データは、Event Detail パネルに表示されます。

NetView プログラムはカスタマイズすることのできる総称コード・ポイント・テーブルを用意しています (コード・ポイント・テーブルのカスタマイズの詳細については、113 ページの『総称コード・ポイント・テーブルの変更』を参照してください)。NetView 製品が提供する総称コード・ポイント・テーブルは、次のとおりです。

- BNJ92TBL - アラート記述コード・ポイント
- BNJ93TBL - 推定原因コード・ポイント
- BNJ94TBL - ユーザー原因コード・ポイント
- BNJ95TBL - インストール原因コード・ポイント
- BNJ96TBL - 障害原因コード・ポイント
- BNJ81TBL - 推奨アクション・コード・ポイント
- BNJ82TBL - 明細データ・コード・ポイント
- BNJ85TBL — 詳細データ・コード・ポイント、サブフィールド X'85'
- BNJ86TBL - 実際のアクション・コード・ポイント

GENALERT コマンドの使用方法

GENALERT コマンドを使用して独自のアラートを作成することができます。GENALERT コマンドは NetView オンライン・ヘルプで説明されています。GENALERT コマンドで使用できるコード・ポイントおよびコード・ポイント・フォーマットについて詳しくは、「*IBM Tivoli NetView for z/OS Messages and Codes Volume 2 (DUI-IHS)*」の総称アラート・コード・ポイントに関する付録を参照してください。

総称アラート・パネルの作成方法

106 ページの図 21 は、総称アラート NMVT の例です。総称アラート・レコードに収められている情報を使用して固有のパネルを作成します。

参照: NMVT の詳細については、*SNA* ライブラリーを参照してください。

```
X'41038D5002000000'              Response Header
X'01230000'                      Major Vector Length and Key
X'0A0108105901020A2827'          01 SV - Date/Time
X'0B92000001'                    92 SV - Alert Description
X'1603'                            code point
X'1A2B3C4D'
X'0693'                          93 SV - Probable Cause(s)
X'0403'                            code point
X'2012'                            code point
X'1195'                          95 SV - Install Cause(s) and Action(s)
X'0601'                            01 SF - install cause(s)
X'1502'                              code point
X'13E1'                              code point
X'038391'                          83 SF - qualifier(s)
X'0681'                            81 SF - recommended action(s)
X'0101'                              code point
X'1504'                              code point
X'2796'                          96 SV - Failure Cause(s) and Action(s)
X'0601'                            01 SF - failure cause(s)
X'0503'                              code point
X'33C2'                              code point
X'068200'                          82 SF - qualifier(s)
X'61'                                code point
X'0004'
X'0C8200'                          82 SF - qualifier(s)
X'53'                                code point
X'11F0F0406040F1C6'
X'0A81'                            81 SF - recommended action(s)
X'0611'                              code point
X'0500'                              code point
X'3110'                              code point
X'00E1'                              code point
X'038321'                          83 SF
X'1705'                          05 SV - Resource Hierarchy
X'151000'                          10 SF
X'07D7E4F9F9F9F900F1'                name/type pair
X'07D3C9D5C5F0F440F9'                name/type pair
X'4D1000'                        10 SV - PSID
X'341104'                          11 SV - Product Identifier
X'0E02C1C3C661C9C2D44040F0F0F3'      02 SF - software product serviceable component ID
X'0804F0F1F0F2F0F3'                  04 SF - software product common level
X'0A06C1C3C661C9C2D440'              06 SF - software product common name
X'0A07C6C6C7C1C9E3D9F3'              07 SF - software product customization ID
X'07098603351225'                    09 SF - software product customization date and time
X'161101'                          11 SV - Product Identifier
X'130012'                            00 SF - hardware product identifier
X'F9F9F9F9F1F1C1F0F5'
X'F0C1F0C1F0C1F0'
X'1798'                          98 SV - Detailed Data
X'0782213400'                      82 SF - qualifier
X'0004'
X'0782000911'                      82 SF - qualifier
X'F2F2'
X'0782000E00'                      82 SF - qualifier
X'00DC'
X'2548'                          48 SV - Correlation
X'1060'                            60 SF - correlation for supporting data
X'D7C3C9C4D3E4F0F4'
X'05C3D5D4F0F1'
X'0D82'                            82 SF - qualifier
X'00DA11C3D6D4D460C5D9D9'
X'068200D1010F'                    82 SF - qualifier
X'3631'                          31 SV - Self Defining Text Message
X'060211340500'                  02 SF - Coded Character Set ID
X'0512C5D5E4'                    12 SF - National Language ID
X'032112'                        21 SF - Sender ID
X'2630'                          30 SF - Text Message
X'E3C8C9E240E2E4C2C6C9C5D3C440C9C4C5D5E3C9C6C9C5E240E3C8C540E3C5E7E340D4E2'
```

図 21. 総称アラート・レコードのサンプル

107 ページの図 22 から 111 ページの図 24 で、総称アラートの NMVT に収められている情報を使って、それぞれの固有パネルがどのように作成されているかを説明します。 107 ページの図 22 は、Alerts-Dynamic パネルのサンプルを示しています。図中の参照番号に対する説明は、パネルの後にあります。

Alerts-Dynamic (アラート動的) パネル

```
N E T V I E W          SESSION DOMAIN: CNM01    OPER1     03/01/11 14:41:03
NPDA-30A                      * ALERTS-DYNAMIC *


    DOMAIN RESNAME  TYPE  TIME  ALERT DESCRIPTION:PROBABLE CAUSE
     CNM01 PU9999  *LINE  14:41 COMM SUBSYSTEM FAILURE:COMM SUBSYSTEM  CTRL    +
             1     2                  3                      4           5
```

```
DEPRESS ENTER KEY TO VIEW ALERTS-STATIC

???
CMD==> _
```

図 22. Alerts-Dynamic (アラート動的) パネルのサンプル:

Alerts-Dynamic (アラート動的) パネルのサンプル

Alerts-Dynamic パネルの各項目は、複数のサブベクトル (X'92'、X'93'、および X'05') から作成されています。 106 ページの図 21 の結果が図 22 になります。

1 RESNAME (リソース名) と TYPE (タイプ) は、X'05' サブベクトル内の最後の名前/タイプの組になります。サンプル表示画面では、RESNAME (リソース名) が PU9999 であり、TYPE (タイプ) が LINE (回線) になっています。

2 *は、TYPE (タイプ) の前の RESNAME (リソース名) がその TYPE (タイプ) に属していないことを示します。TYPE (タイプ) は、常にその階層内の最後の名前に関連付けられますが、名前については、X'05' がどのようにコーディングされているかによって異なります。リソース名表示不可インディケーター・ビットは、最後の名前/タイプの組 (サブベクトル X'05'、サブフィールド X'10' の 2 番目の名前/タイプの組の 8 バイト目第 2 ビット) で 1 に設定されます。

3 ALERT DESCRIPTION (アラート記述) は、X'92' サブベクトルのコード・ポイント X'1603' から取得されます。このコード・ポイントは、アラート記述テキスト・メッセージをもつテーブルへの索引として使用されます。サンプルでは、ALERT DESCRIPTION (アラート記述) として COMM SUBSYSTEM FAILURE (コマンド・サブシステム障害) が表示されています。

4 PROBABLE CAUSE (推定原因) は、X'93' サブベクトルのコード・ポイント X'0403' から取得されます。このコード・ポイントは、推定原因テキスト・メッセージをもつテーブルへの索引として使用されます。サンプルでは、PROBABLE CAUSE (推定原因) として COMM SUBSYSTEM CTRL (コマンド・サブシステム制御) が表示されています。

5 + は、 107 ページの図 22 の X'93' サブベクトルに複数の推定原因コード・ポイントが存在するために表示されています。+ は、イベント詳細パネルで他の推定原因を見ることができることを示しています。

図 23 は、Recommended Action パネルのサンプルを示しています。図中の参照番号に対する説明は、パネルの後にあります。

Recommended Action for Selected Event (イベントの推奨アクション) パネル

```
N E T V I E W          SESSION DOMAIN: CNM01    OPER1     03/01/11 14:41:17
NPDA-45A           * RECOMMENDED ACTION FOR SELECTED EVENT *       PAGE 1 OF 1
CNM01        PU9999     LINE04  1
            +--------+
 DOMAIN     |  PU    |----LINE----  2
            +--------+

USER    CAUSED - NONE  3

INSTALL CAUSED - INCORRECT MICROCODE FIX  4
                 INCORRECT SOFTWARE GENERATION: ACF/IBM  5
       ACTIONS - I013 - VERIFY X.25 SUBSCRIPTION NUMBER  6
                 I085 - APPLY CORRECT SOFTWARE LEVEL

FAILURE CAUSED - COMMUNICATIONS SUBSYSTEM  7
                 LINE ADAPTER MICROCODE
                      ADAPTER NUMBER 04  8
                      LINE ADDRESS RANGE 00 - 1F  9
       ACTIONS - I032 - DUMP CHANNEL ADAPTER MICROCODE  10
                 I026 - RUN APPROPRIATE TRACE
                 I136 - CONTACT COMMUNICATIONS SYSTEMS PROGRAMMER
                 I010 - PERFORM 9999 PROBLEM DETERMINATION PROCEDURES
                                11

ENTER DM (DETAIL MENU) OR D (EVENT DETAIL)

 ???
CMD==> _
```

図 23. Recommended Action for Selected Event (選択イベントの推奨アクション) パネルのサンプル :

Recommended Action for Selected Event (選択イベントの推奨アクション) パネルのサンプル

Recommended Action パネルは、いくつかのサブベクトル (X'94'、X'95'、および X'96') とサブフィールド (X'01'、X'81'、X'82'、および X'83') から作成されています。

1 リソース名 (PU9999 と LINE04) は、X'05' の階層名リストのサブベクトルから取得されます。 106 ページの図 21 では、インディケーター・ビット X'05' サブベクトルの階層完了インディケーター・ビット (バイト 2、ビット 0) が X'0' に設定されているため、X'05' サブベクトルから取り出した名前だけが使用されています。このビットが 1 に設定されている場合は、NetView プログラムにより、VTAM が指定した名前に X'05' サブベクトル内の名前が連結されます。

2 リソース・タイプ (PU および LINE) は、X'05' サブベクトルの X'10' サブフィールド内のタイプ・コード (X'F1' および X'F9') が、表示可能なリソース・タイプに変換されることによって得られます。リソース・タイプの変更について詳しくは、 117 ページの『リソース・タイプの追加または変更』を参照してください。

3 X'94' サブベクトル (NONE) により、ユーザー原因の情報が表示されます。 106 ページの図 21 には、X'94' サブベクトルがないため、ユーザー原因の情報は表示されません。

4 インストール起因の推定原因としては次の 2 つが表示されます。

```
INCORRECT MICROCODE FIX
INCORRECT SOFTWARE GENERATION:
```

これらは、X'95' サブベクトルの X'01' サブフィールド内のコード・ポイント (X'1502' および X'13E1') から作成されています。 X'13E1' コード・ポイントの E は、インストール原因を完結するために X'83' サブフィールドが必要であることを示しています。

5 インストール原因に関する修飾子 (ACF/IBM) は、X'95' サブベクトルの X'83' サブフィールドにより表示されます。X'83' サブフィールドには、修飾子が最初の製品 ID サブベクトル (X'11') の製品 ID サブフィールド (X'06' ソフトウェア製品共通名) から得られたことを示す値 X'91' が入っています。

6 インストール原因のアクションとしては次の 2 つが表示されます。

```
I013 - VERIFY X.25 SUBSCRIPTION NUMBER
I085 - APPLY CORRECT SOFTWARE LEVEL
```

これらは、X'95' サブベクトルの X'81' サブフィールド内のコード・ポイント (X'0101' および X'1504') から得られます。

7 障害原因の推定原因として次の 2 つが表示されます。

```
COMMUNICATIONS SUBSYSTEM
LINE ADAPTER MICROCODE
```

これらは、X'96' サブベクトルの X'01' サブフィールド内のコード・ポイント (X'0503' および X'33C2') から得られます。X'33C2' コード・ポイントの C は、障害原因を完結するために 2 つの明細データ・サブフィールド X'82' または X'85' のいずれかのサブフィールドが必要であることを示しています。この例では X'82' サブフィールドが使用されています。 X'82' または X'85' サブフィールドのいずれかが使用できますが、この 2 つの組み合わせは無効です。サブベクトル内では、すべての詳細修飾子が X'82' サブフィールドまたは X'85' サブフィールドでなければなりません。

8 ADAPTER NUMBER 04 は、X'96' サブベクトルの最初の X'82' サブフィールドから分割されていることを示しています。この番号は次のようにして示されます。

00 PSID サブベクトルからはなんの情報も得られていないことを示しています。

61 アダプター番号に対応するコード・ポイント。

00 16 進データが続くことを示しています。

04 表示される 16 進データ。

9 LINE ADDRESS RANGE 00 - 1F は、X'96' サブベクトルの 2 番目の X'82' サブフィールドから分割されています。この範囲は次のようにして示されます。

00 PSID サブベクトルからはなんの情報も得られていないことを示しています。

53 回線アドレス範囲に対するコード・ポイント。

11 EBCDIC データが続くことを示しています。

F0F0406D40F1C6
表示される EBCDIC データ。

10 障害原因のアクションとして次のものが表示されます。

```
I032 - DUMP CHANNEL ADAPTER MICROCODE
I026 - RUN APPROPRIATE TRACE
I136 - CONTACT COMMUNICATIONS SYSTEMS PROGRAMMER
I010 - PERFORM 9999 PROBLEM DETERMINATION PROCEDURES
```

これらは、X'96' サブベクトルの X'81' サブフィールド内のコード・ポイント (X'0611'、X'0500'、X'3110'、および X'00E1') から得られます。 X'00E1' コード・ポイントの E は、障害原因を完結するために X'83' サブフィールドが必要であることを示しています。

11 障害原因に関する修飾子 (9999) は、X'96' サブベクトルの X'83' サブフィールドが存在することにより表示されます。X'83' サブフィールドには、修飾子が PSID サブベクトル (X'11') の最初のハードウェア PSID サブフィールド (X'00') から得られたことを示す値 X'21' が入っています。

111 ページの図 24 および 111 ページの図 25 は、Event Detail (イベント詳細) パネルのサンプルを示しています。図中の参照番号に対する説明は、図の後にあります。

Event Detail (イベント詳細) パネル

```
N E T V I E W          SESSION DOMAIN: CNM01    OPER1     03/20/11 14:41:32
NPDA-43S                      * EVENT DETAIL *                   PAGE 1 OF 2

 CNM01       PU9999     LINE04 1
            +--------+
 DOMAIN     |  PU    |----LINE---- 2
            +--------+

 DATE/TIME: RECORDED - 01/02 10:41    CREATED - 03/20/11 10:40:39 3

 EVENT TYPE: PERMANENT 4

 DESCRIPTION: COMMUNICATIONS SUBSYSTEM FAILURE 5
 PROBABLE CAUSES:
    COMMUNICATIONS SUBSYSTEM CONTROLLER 6
    TOKEN-RING LAN

 QUALIFIERS:
    1) 9999 COMMUNICATION CONTROL UNIT 0004 7

 ENTER A (ACTION) OR DM (DETAIL MENU)

 ???
CMD==> _
```

図 24. Event Detail (イベント詳細) パネルのサンプル (1 ページ目):

Event Detail (イベント詳細) パネルのサンプル (1 ページ目)

```
N E T V I E W          SESSION DOMAIN: CNM01    OPER1     03/20/11 14:41:49
NPDA-43S                      * EVENT DETAIL *                   PAGE 2 OF 2

 CNM01       PU9999    LINE04
            +--------+
 DOMAIN     | PU     |----LINE----
            +--------+

 QUALIFIERS (CONTINUED):
    2) EVENT CODE 22
    3) REASON CODE 00DC

 CONTROL PROGRAM TEXT: 8
    THIS SUBFIELD IDENTIFIES THE TEXT MS

 CORRELATION FOR SUPPORTING DATA 9
   PCID: PCIDLU01          NETWORK QUALIFIED NAME: CNM01
    1) LOG ID COMM_ERR
    2) LOG RECORD NUMBER 15

 UNIQUE ALERT IDENTIFIER: PRODUCT ID - ACF/IBM            ALERT ID - 1A2B3C4D
                                          10                        11
 ENTER A (ACTION) OR DM (DETAIL MENU)

 ???
CMD==> _
```

図 25. Event Detail (イベント詳細) パネルのサンプル (2 ページ目):

Event Detail (イベント詳細) パネルのサンプル (2 ページ目)

Event Detail (イベント詳細) パネルは、サブベクトルの
X'92'、X'93'、X'98'、X'01'、X'31'、X'48' と、サブフィールドの X'82' から作成されています。

1 リソース名 (PU9999 と LINE04) は、X'05' の階層名リストのサブベクトルから取得されます。 106 ページの図 21 では、X'05' サブベクトルの階層完了インディケーター・ビット (バイト 2、ビット 0) が X'0' に設定されているため、X'05' サブベクトルから取得された名前だけが使用されます。このビットが 1 に設定されている場合は、NetView プログラムにより、VTAM が指定した名前に X'05' サブベクトル内の名前が連結されます。

2 リソース・タイプ (PU と LINE) は、X'05' サブベクトルの X'10' サブフィールド内のタイプ・コード (X'F1' と X'F9') を表示可能なリソース・タイプに変換することによって取得されます。リソース・タイプの変更について詳しくは、 117 ページの『リソース・タイプの追加または変更』を参照してください。

3 DATE/TIME RECORDED (日付/時刻記録) は、そのレコードがハードウェア・モニターのデータベースにログされた時刻です。CREATED フィールドは、送信側製品がそのレコードを作成した時刻を示します。これは X'01' サブベクトルの X'10' サブフィールドから取り出されます。

4 EVENT TYPE (イベント・タイプ) は、X'92' サブベクトルの 4 バイト目 (アラート・タイプ) から得られます。

5 DESCRIPTION (記述) は、アラート・パネルの記述と同様に、X'92' サブベクトルのコード・ポイント (X'1603') から得られます。ただし、この画面ではより長いテキストが表示されます。

6 PROBABLE CAUSES (推定原因) は、X'93' サブベクトルのコード・ポイント (X'0403' および X'2012') から取り出されます。この画面では、前出のアラート・パネルの場合より長いテキストが表示されます。また、すべての推定原因が表示されます。

7 QUALIFIERS (修飾子) は X'82' または X'85' サブフィールドから得られます。NetView プログラムは、X'98' サブベクトル内の X'01' サブフィールドとそれに関連したサブフィールド (X'82' および X'85' を含む) を無視します。

X'82' または X'85' サブフィールドのいずれかが使用できますが、この 2 つの組み合わせは無効です。サブベクトル内では、すべての詳細修飾子が X'82' サブフィールドまたは X'85' サブフィールドでなければなりません。

この例は X'82' サブフィールドを使用し、修飾子は次のような意味をもちます。

X'98' サブベクトルの最初のサブフィールド

21 データは PSID サブベクトル (X'11') の最初のハードウェア PSID サブフィールド (X'00') から取り出されることを示しています。

34 通信制御装置を示すコード・ポイント。

00 16 進データが続くことを示しています。

0004 表示される 16 進データ。

X'98' サブベクトルの 2 番目のサブフィールド

00 PSID サブベクトルからなんのデータも取り出されないことを示しています。

09 イベント・コードを示すコード・ポイント。

11 EBCDIC データが続くことを示しています。

F2F2 表示される EBCDIC データ。

X'98' サブベクトルの 3 番目のサブフィールド

00 PSID サブベクトルからなんのデータも取り出されないことを示しています。

0E 理由コードを示すコード・ポイント。

00 16 進データが続くことを示しています。

00DC 表示される 16 進データ。

イベント詳細パネルの 2 ページ目 (111 ページの図 24 参照) には、次の情報が収められています。

8 CONTROL PROGRAM TEXT (コントロール・プログラム・テキスト) は、テキストのタイトルであり、サブベクトル X'31' のサブフィールド X'21' の存在により表示されます。テキスト自体は X'31' サブベクトルのサブフィールド X'30' から直接取り出されて画面に表示されます。

9 CORRELATION FOR SUPPORTING DATA (サポートするデータとの相関) は、X'48' サブベクトルから表示されます。サブフィールド X'60' は、ネットワーク修飾の手順相関 ID を使って 1 つのセッションを固有に識別することを指定します。

X'82' または X'85' のいずれかのサブフィールドがデータのサポートのために使用されます。この例では、2 つの X'82' サブフィールドを使用してサポートするデータを識別しています。

X'82' または X'85' のいずれかのサブフィールドが使用できますが、この 2 つの組み合わせは無効です。サブベクトル内では、すべての詳細修飾子が X'82' サブフィールドまたは X'85' サブフィールドでなければなりません。

10 製品 ID (ACF/IBM) は、最初の PSID (X'10') サブベクトル内の最初の製品 ID (X'11') サブベクトルから直接取り出されます。 106 ページの図 21では、ソフトウェア製品サービス可能コンポーネント ID (X'02') サブフィールドが使用されています。

11 アラート ID 番号 (1A2B3C4D) は、サブベクトル X'92' の 7 から 10 バイト目から取り出されます。

総称コード・ポイント・テーブルの変更

このセクションでは、NetView プログラムと一緒に出荷される総称アラート・コード・ポイント・テーブルを変更する方法を説明しています。NetView 初期設定の前または後で、テーブルを変更することができます。後の場合は、変更を動的に活動化するために CPTBL コマンドを使用します。CPTBL コマンドは NetView オンライン・ヘルプで説明されています。

テーブルの形式

各テーブルには異なるタイプのコード・ポイントが入っています。テーブルは次のとおりです。

- BNJ92TBL: アラート記述コード・ポイント
- BNJ93TBL: 推定原因コード・ポイント
- BNJ94TBL: ユーザー原因コード・ポイント
- BNJ95TBL: インストール原因コード・ポイント
- BNJ96TBL: 障害原因コード・ポイント
- BNJ81TBL: 推奨アクション・コード・ポイント
- BNJ82TBL: 明細データ・コード・ポイント
- BNJ85TBL: 明細データ・コード・ポイント、X'85' サブフィールド
- BNJ86TBL: 実際のアクション・コード・ポイント

テーブル名の 4 文字目と 5 文字目は、コード・ポイントをもつサブベクトルまたはサブフィールドを識別するものです。

コード・ポイント・テーブルの最初の項目は制御項目です。1 桁と 2 桁は、どのコード・ポイント・テーブルが作成または更新されるかを指定するサブベクトル番号を表します。可能な値は 92、93、94、95、96、81、82、85、または 86 です。初期設定の間は、この番号はテーブル名に一致しなければなりません。3 桁目はブランクでなければならず、残りのすべての桁は未使用で無視されます。(この部分をコメントのために使用しないでください。この部分は今後他の目的のために使用される予定です。) CPTBL コマンドを使用すると、コード・ポイント定義を含むファイルの名前は事前定義された名前の 1 つでなくともかまいません。NetView プログラムはこの制御項目を使用して、テーブル・タイプを判別します。

コード・ポイント・テーブルにおける、後続の各項目の形式は次のとおりです。

- 1 から 4 桁目には、4 文字の 16 進コード・ポイント番号が含まれます。有効な文字は 0 から 9 と A から F です。ユーザー用にコード・ポイント X'E000' から X'EFFF' までが予約されています。この範囲以外のコード・ポイントを使用する場合は、Tivoli サポート・センターに相談してください。

  所定のテーブルの中でコード・ポイントが複数回定義されると、最初の項目が使用され、後続の項目は無視され、通知メッセージが生成されます。
- 6 桁目には、埋め込みフラグ (Y) が含まれます。そのフラグは、X'82'、X'83'、または X'85' サブフィールドと関連した修飾子データが、コード・ポイント・テキストの前に置かれているか、コード・ポイント・テキスト内に埋め込まれている、または同じ行のコード・ポイント・テキストの後に続くことを示します。Y 以外のすべての文字は、埋め込みフラグがオフであることを示します。埋め込みフラグがオンになると、総称アラートに含まれる埋め込み情報が、ドル記号 ($) がマークされている場所に埋め込まれます。埋め込みテキストは、BNJ81TBL、BNJ86TBL、BNJ94TBL、 BNJ95TBL、および BNJ96TBL でのみサポートされます。推定原因およびアラート記述には変数置換は利用できないため、BNJ92TBL および BNJ93TBL では埋め込みフラグは無視されます。
- 8 から 72 桁目には、このコード・ポイントのテキスト記述が含まれます。テキスト変数の最大長は次のとおりです。

– 推定原因 - 所定のコード・ポイントの最初の項目は 40 文字、2 番目の項目は 20 文字。(2 番目の項目の説明は、 116 ページの『BNJ92TBL コード・ポイント・テーブルの例』の 4 を参照してください。)
– アラート記述 - 所定のコード・ポイントの最初の項目は 40 文字、2 番目の項目は 25 文字。(2 番目の項目の説明は、 116 ページの『BNJ92TBL コード・ポイント・テーブルの例』の 4 を参照してください。)
– 明細データ - 40 文字
– その他 - 108 文字

テキストを次の行に続ける場合は 2 桁目からコーディングしてください。

- 73 から 80 桁目は無視されるため、オプションのシーケンス番号のために使用することができます。

注:

1. テーブル BNJ82TBL のコード・ポイントは左寄せして 0 (ゼロ) で埋めてください。例えば、コード・ポイント 12 を入力する場合は 1200 と入力します。
2. NetView BNJ81TBL コード・ポイント・テーブルに追加するコード・ポイント項目のテキストは、E*nnn* で始めてください。 NetView BNJ86TBL コード・ポイント・テーブルに追加するコード・ポイント項目のテキストは、R*nnn* で始めてください。 E*nnn* および R*nnn* を使用すると、そのコード・ポイントは ACTION コマンド・リストでサポートすることができます。(ACTION コマンド・リストの詳細については、NetView オンライン・ヘルプを参照してください。)BNJ81TBL と BNJ86TBL のアクション・テキストはこの方法で開始してください。この方法で開始されずに、BNJDNUMB が推奨アクション番号を生成するために使用されると、BNJDNUMB は推奨アクション・テキストの最初の 4 バイトをオーバーレイします。
3. ハードウェア・モニターは特定のコード・ポイントのためのテーブルを探索します。一致するものが見つからない場合は、ハードウェア・モニターは汎用コード・ポイントのためのいくつかのテーブルを探索します。

   汎用コード・ポイントは最後の 2 バイトを 0 (ゼロ) に設定しているコード・ポイントです。例えば、特定のコード・ポイントが 1620 の場合、汎用コード・ポイントは 1600 です。汎用コード・ポイントが見つからない場合は、そのテキストが元のコード・ポイントと一致したものとして返されます。汎用コード・ポイントは適用されるすべての特定コード・ポイントに有効なテキストを含んでいます。汎用コード・ポイントは BNJ82TBL と BNJ85TBL では使用できません。(汎用コード・ポイントについては、 *SNA* ライブラリーを参照してください。)
4. コード・ポイント・テーブルはすべて大文字で表示されます。ただし、ユーザー独自のコードを小文字または大/小文字混合で入力する場合には、NetView プログラムがそのテキストを大文字に変換することはありません。

%INCLUDE ステートメントの使用

コード・ポイント・テーブルで %INCLUDE ステートメントを使用すると、ユーザーのコード・ポイント情報をより容易に保守できるように編成することができます。

各コード・ポイント・タイプごとに 1 つのメイン・テーブルをもつようにすることができます。このテーブルには、NetView プログラムと一緒に出荷されるコード・ポイント、および、ユーザー定義のサブテーブルと他のプロダクトで定義するサブテーブル用の %INCLUDE ステートメントを含めることができます。

Tivoli では、BNJ*xx*TBL テーブル (*xx* はテーブル番号) の変更をお勧めしません。これらのテーブルは各コード・ポイントのメイン・テーブルとして使用してください。 これらのテーブルのカスタマイズが必要な場合は、この目的のためにメイン・テーブル (BNJ*xx*TBL) によって組み込まれている BNJ*xx*UTB (*xx* はテーブル番号) を使用してください。

BNJ92TBL コード・ポイント・テーブルの例

コード・ポイント・テーブルの例は『BNJ92TBL コード・ポイント・テーブルの例』 に示されています。図中の参照番号に対する説明は、図の後にあります。

BNJ92TBL コード・ポイント・テーブルの例

```
   * An asterisk in column 1 indicates a comment line.
   * The following line is the control entry indicating table type.
1  92
   * Blank lines are allowed for readability.

2  %INCLUDE BNJ92UTB
    3     4
   0100   SIMPLE CODE POINT TEXT;
5  E123   THIS TEXT IS EXACTLY FORTY CHARS LONG XX;
   E123   THIS IS THE SAME IN 25 XX;
6  FFFF
```

1 最初の非コメント行は制御項目です。

2 コード・ポイント・テーブルは %INCLUDE ステートメントを使用して、他のファイルをコード・ポイント・テーブルに埋め込むことができます。

3 コード・ポイント (0100) は、4 文字の 16 進数で、1 桁目から始めます。

4 8 桁目から 72 桁目のテキスト記述は、ハードウェア・モニター画面に表示されます。

5 ハードウェア・モニターはさまざまなパネル・フォーマットを持ち、アラート記述 (92) と推定原因 (93) のためのさまざまな長さのテキストを使用することができます。おのおのの項目のテキストの最大長は 40 文字です。テキストがアラート記述で 25 文字を超えるか、または推定原因で 20 文字を超える場合は、簡略化されたテキストが必要です。テキスト項目が 40 文字を超える場合は、エラーが生じることがあります。

6 テーブル内の項目で、コード・ポイント FFFF をもち、テキストがないものはすべて無視されます。(マイグレーションは可能です。) コード・ポイント FFFF とテキストをもつ項目は、その他のコード・ポイントとして扱われます。

BNJ94TBL コード・ポイント・テーブルの例

コード・ポイント・テーブルのもう 1 つの例を 117 ページの『BNJ94TBL コード・ポイント・テーブルの例』 に示します。

BNJ94TBL コード・ポイント・テーブルの例

```
    * An asterisk in column 1 indicates a comment line.
    * The following line is the control entry indicating table type.
    94
 1  %INCLUDE BNJ94UTB
 2  0100 Y CODE POINTS TEXT WITH DETAIL INSERTS $ AND $
 3  0200   CODE POINTS TEXT ILLUSTRATING CONTINUATION OF THE TEXT TO A SECON
     D LINE
 4  0100   DUPLICATE TEXT
```

1 コード・ポイント・テーブルは %INCLUDE ステートメントを使用して、他のファイルをコード・ポイント・テーブルに埋め込むことができます。

2 埋め込みフラグ (6 桁目の Y) は、ドル記号 ($) がマークされている場所に修飾子データが埋め込まれることを示します。

3 次の行テキストが継続する場合は、2 桁目から開始します。最初の行のテキストは 8 桁目から開始され、72 桁まで続きます。

4 このコード・ポイントはすでにテーブルで定義されているため、この項目は無視され、通知メッセージが生成されます。

変更済みコード・ポイント・テーブルの活動化

CPTBL コマンドは AUTOTBL コマンドに非常に類似しており、NetView プログラムが初期設定された後でコード・ポイント・テーブルに行われた変更を動的に活動化するために使用されます (CPTBL コマンドの説明は、NetView オンライン・ヘルプを参照してください)。CPTBL コマンドにテスト・オプションを使用して、活動化の前にコード・ポイント・テーブルの構文を検査してください。

リソース・タイプの追加または変更

メンバー BNJRESTY を変更することによって、ハードウェア・モニターの階層表示画面に新しいリソース・タイプを追加することができます。

BNJRESTY はデータ・セット NETVIEW.V6R2M0.BNJPNL2 のメンバーであり、NetView 始動プロシージャーの定義ステートメント BNJPNL2 によって定義されます。

『BNJRESTY の内容のサンプル』 に、BNJRESTY の形式が示されています。図中の参照番号に対する説明は、図の後にあります。

BNJRESTY の内容のサンプル

```
 1   2 3
10 DISK your comments
```

1 2 文字の 16 進数が、1 桁目から NetView プログラムの X'05' サブベクトルに渡されます。有効な文字は、0 から 9 と A から F です。重複する 16 進コードを指定すると、システムは重複したコードのうち先に入力されたものを使用します。X'E0' から X'EF' までは、ユーザー定義のリソース・タイプ用に予約されています。

2 4 桁目から 7 桁目の 4 文字は、リソース・タイプとして扱われます。有効な文字は、0 から 9、A から Z、および印刷可能なすべての特殊文字です。4 文字に

満たないリソース・タイプは、4 桁目から始めて右側をブランクで埋めなければなりません。リソース・タイプには、コンマ (,)、ピリオド (.)、等号 (=) などの区切り文字を使用しないでください。

3 オプションのコメントはリソース・タイプの後のどこからでも開始することができます。

ハードウェア・モニター・タスク BNJDSERV がアクティブである間に BNJRESTY が変更されると、新規のリソース・タイプは認識されません。その場合は、STOP TASK=BNJDSERV に続けて STARTCNM NPDA を使用して NetView プログラムがすべての新規リソース・タイプを認識できるようにするか、変更された BNJRESTY メンバーを活動化するために RTTBL コマンドを使用してください。

NetView プログラムは、NetView プログラムのアクティブ化時または RTTBL コマンドの呼び出し時に BNJRESTY で無効な項目を検出すると、エラー・メッセージをコマンド・ファシリティー・コンソールに表示します。NetView プログラムは、IBM 提供のリソース・タイプを使用します。

第 7 章 ネットワーク資産管理コマンド・リストの変更

ネットワーク資産管理プログラムにより、ハードウェアおよびソフトウェアのデバイスのサブセットから、インベントリー・データを自動的に集めることができます。ネットワーク資産管理プログラムを使用して、ハードウェア製品の製造番号、機械タイプ、型式番号やソフトウェア情報などの重要プロダクト・データ (VPD) を収集することができます。この情報には、バージョンやリリース・レベルなども含まれます。ただし、NetView プログラムは、ネットワーク資産管理プログラムをサポートするデバイスが戻したデータの検査は行いません。単にデータを収集してログに記録するための方法を提供するだけです。

参照: レコード・フォーマットについては、「*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。NetView プログラム提供のコマンド・リストについては、 NetView オンライン・ヘルプを参照してください。

REQUEST/REPLY PSID 体系をサポートするデバイスはすべて、VPD を NetView プログラムに報告することができます。これらの体系をサポートしないデバイスの VPD を送信請求しようとすると、キーボードがロックされたり、無関係なデータが画面に現れたりします。RESET キーを押したり、画面を消去したりする必要はありますが、このようなアクションが NetView プログラムの VPD の収集に影響を及ぼすことはありません。

参照: REQUEST/REPLY PSID 体系については、 *SNA* ライブラリーを参照してください。

次に示すのは、REQUEST/REPLY PSID 体系をサポートする物理装置 (PU) の例です。

- 3720/NCP
- 3725/NCP
- 3745/NCP
- 3174 このデバイス自体およびこれに接続された (3191、3192、および 3194 ディスプレイ装置の各種モデルなど) 種々のデバイスについてデータを報告します。

OS/2 で実行されるパーソナル・コンピューターが、これらのプロダクトに必須です。

参照: デバイスの VPD を入力するための命令は、そのデバイスのユーザーズ・ガイドに解説されています。

ネットワーク資産管理は、VPDCMD コマンドを使用して VPD を所定のデバイスから収集し、VPDLOG コマンドを使用してレコードを作成し、外部ログ記録機能 (SMF など) に記録します。サービス水準報告プログラム (SLR) を使用して対話式でデータを表示させたり、報告書を生成したり、VPDALL コマンドを使用して NetView ドメイン内のすべてのデバイスに対して VPDPU および VPDDCE コマン

ド項目を生成したりすることもできます。交換回線を必要とするリソースがある場合は、VPD を収集する前に交換回線がアクティブになっているか確認してください。

ネットワーク資産管理プログラムは次のコマンド・リストを提供します。

VPDPU

このコマンド・リストは、1 台の PU およびそれに接続されたデバイスの VPD を収集し、ログに記録します。このコマンド・リストは、オペレーターのコンソールまたは別のコマンド・リストから入力することができます。

VPDDCE

このコマンド・リストは、指定された NCP と指定された PU との間の直接パス内にある DCE からの VPD の送信請求およびログに記録を行います。このコマンド・リストは、オペレーターのコンソールまたは別のコマンド・リストから入力することができます。

VPDACT

これは、VPDALL コマンドが CREATE オプションを指定されて出されたときに生成するコマンド・リストのデフォルト名です。VPDALL は、VTAMLST の VTAM 構成メンバーを入力データとして読み取り、VPDACT (デフォルト名) と呼ばれるコマンド・リストを生成します。 VPDACT には、ドメイン内のデバイスについての VPDPU および VPDDCE 項目のリストが入っています。後で VPDACT を出して、NetView ドメイン内のサポートされるデバイスから VPD を収集してログに記録することができます。

VPDLOGC

これは、START および END レコードを作成し、ログに記録するコマンド・リストです。START レコードは、VPD の送信請求の始めに VPDACT コマンド・リストに対して生成されます。END レコードは、VPD の送信請求の終わりに VPDACT コマンド・リストに対して生成されます。このコマンド・リストは、オペレーターのコンソールまたはユーザー作成のコマンド・リストからは出さないでください。

VPDXDOM

これは、クロスドメイン・リソースから VPD を送信請求するために使用されるサービス・コマンド・リストです。このコマンド・リストは、NetView 自動化テーブルを介して実行されます。このコマンド・リストは、オペレーターのコンソールまたはユーザー作成のコマンド・リストからは出さないでください。

参照: レコード・フォーマットについては「*IBM Tivoli NetView for zOS アドミニストレーション・リファレンス*」を、また VPD コマンド・リストの説明については NetView オンライン・ヘルプを参照してください。補足情報については、「*IBM Tivoli NetView for zOS 自動操作ガイド*」を参照してください。

1 台の物理装置 (PU) からの重要プロダクト・データ (VPD) の収集

以下のリストは、1 台の物理装置 (PU) とそれに接続されたデバイスからの重要プロダクト・データ (VPD) の収集手順を説明しています。

1. リソース名を指定し、VPDPU または VPDDCE コマンド・リストを出します。

2. このコマンド・リストは、指定したリソースからのデータを送信請求する VPDCMD コマンドを出し、応答メッセージを待ちます。
3. PU が、その PU の VPD、または PU と PU に接続するデバイスの VPD で応答します。
4. コマンド・リストは応答メッセージをトラップし、機械タイプ、型式番号、製造番号などの VPD をコマンド・リスト変数に保存します。
5. 完了メッセージを受け取ると、コマンド・リストはレコードを作成し、それを外部ログ機能に書き込みます。
6. 完了前に異常なイベントが発生すると、コマンド・リスト・エラー・メッセージが出され、コマンド・リストは終了します。異常イベントは、ログ障害、非アクティブの VPDTASK、あるいは異常終了の可能性があります。

単一 NetView ドメインからの重要プロダクト・データ (VPD) の収集

以下のリストは、単一 NetView ドメインからの VPD の収集手順を説明しています。

1. NetView オペレーターは次のコマンドを入力します。

   ```
   VPDALL CONFIG(ATCCON01),CREATE,CLIST(VPDACT),ADD
   ```

2. VPDALL コマンド・リストは VTAMLST の構成メンバー (この例では ATCCON01) から指定されたノードを読み取ります。VPDALL は、VTAMLST ノードからすべてのリソース名を抽出し、VPD を収集できるようにします。その後、VPDALL は、VPDACT というコマンド・リストに、VPDPU および VPDDCE 項目を作成します。VPDALL は、交換回線上の DCE も動的再構成デック (DRD) もサポートしません。

   注: ドメイン全体からのデータを収集するには、構成メンバーにそのドメイン内のすべてのリソースの定義がなければなりません。
3. VPDACT は、リソース名の追加または削除により変更することができます。
4. VPDACT コマンド・リストが実行されると、VPDLOGC が呼び出されて START レコードを生成します。その後、VPDACT は、VPDPU および VPDDCE コマンド・リストを呼び出し、それらが完了すると、VPDLOGC を呼び出して END レコードを生成します。

フォーカル・ポイントの重要プロダクト・データ (VPD) 収集

122 ページの図 26 は、VPD 収集のためのフォーカル・ポイント NetView プログラムを示しています。

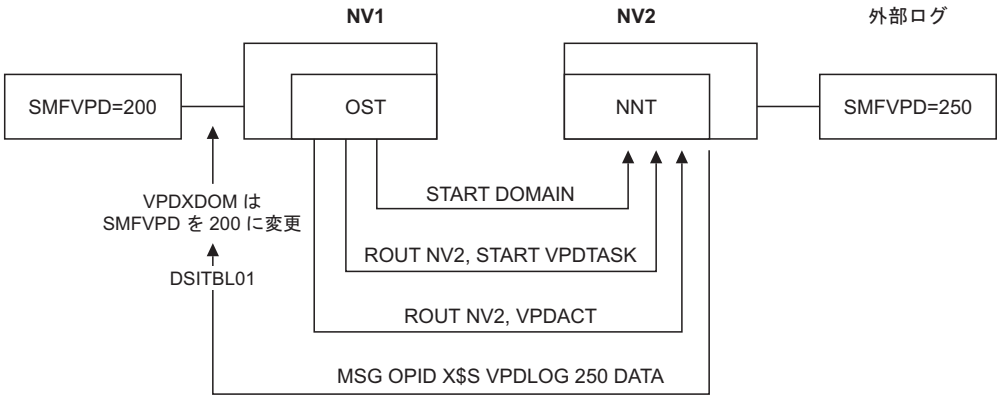


図 26. VPD のフォーカル・ポイント NetView プログラム

以下のステップは、図 26 に示されているサンプル・フォーカル・ポイント NetView プログラムの VPD の収集手順を説明しています。

1. NV1 が、インストール時に共通グローバル変数 SMFVPD を 200 に設定します。NV2 が共通グローバル変数を 250 に設定します。

   注: CNMSTYLE が共通グローバル変数 SMFVPD を 37 に設定します。
2. NV1 が、VPD 収集のためのフォーカル・ポイント NetView プログラムとして指定されます。NV1 の場合のみ、NetView 自動化テーブル (DSITBL01) では、VPDXDOM コマンド・リストを実行するように指定されたステートメントをアンコメントしてください。

   参照: 詳細については、「*IBM Tivoli NetView for z/OS インストール:追加コンポーネントの構成*」を参照してください。
3. フォーカル・ポイント NetView NV1 から DSIELTSK を開始させます。
4. NV1 が START DOMAIN コマンドを使って、NV2 との OST-NNT 直接セッションを確立します。
5. NV1 が START VPDTASK を出します。
6. NV1 が ROUTE NV2、START VPDTASK を出します。
7. NV1 が ROUTE NV2、VPDACT を出します。これで、NV2 の VPDACT コマンド・リストが NNT のもとで実行されます。
8. NV2 では、VPDACT それ自体が NNT のもとで実行されていることを確認し、次のメッセージを生成します。

   ```
   MSG OPID X$S VPDLOG 250 '1 STRING1 10 STRING2...'
   ```

   ここで、X$S は、NetView 自動化テーブルにより認識される特別なストリングです。
9. NV2 の VPDACT コマンド・リストが生成されたメッセージを NV1 のオペレーターに書き出すと、そのメッセージにより NetView 自動化テーブルが起動されて、NV1 の VPDXDOM コマンド・リストが実行されます。

   参照: VPXDOM コマンド・リストの詳細については、「*IBM Tivoli NetView for z/OS 自動操作ガイド*」を参照してください。
10. VPDXDOM が入力されると、メッセージ・ストリングは次のようになります。

```
DSI039I MSG FROM OPID : X$S VPDLOG 250 1 STRING1...
```

11. VPDXDOM は、NV1 が SMFVPD を共通グローバル変数として設定したことを確認し、SMFVPD を 250 (NV2) から 200 (NV1) に変更します。
12. VPDLOGC が、NV1 の SMF レコード番号 200 にデータ・レコードをログに記録します。
13. VPD 送信請求が完了するまでクロスドメイン・セッションがアクティブになっていることを確認します。

カスタマイズに関する考慮事項

NetView プログラム提供の VPD コマンド・リストは、ユーザーの要件に合わせてカスタマイズできます。

ネットワーク資産管理プログラム・コマンド・リストを変更して、違うレコード・フォーマットを作成するときは、1 レコードあたり 256 バイトを超えないようにしてください。NetView プログラムのコマンド・ストリングの上限は 240 文字です。VPD コマンドを最大限に活用するために、コマンド・プロセッサーを作成することも可能です。

参照: コマンド・プロセッサーについては、「*IBM Tivoli NetView for z/OS プログラミング:アセンブラー*」を参照してください。

SMF のレコード・フォーマットを変更する場合、レコード番号 37 を使うことはできません。SMF レコード番号の定義は、ユーザー定義された 128 から 255 までの範囲でグローバルに行わなければなりません。SLR を使用している場合は、SLR テーブルは各自で修正した SMF レコード・フォーマットに合わせて作成しなければなりません。

参照: &WAIT および RESET の使用上の制限について、また、直前のネットワーク資産管理コマンド・リストの実行中に 2 番目のネットワーク資産管理コマンド・リストやネットワーク資産管理コマンドを出す際の考慮事項については、NetView オンライン・ヘルプ、および「*IBM Tivoli NetView for z/OS プログラミング: REXX および NetView コマンド・リスト言語*」を参照してください。

性能を向上させるためには、次のことを行います。

- VPDACT を読み込むコマンド・リストを作成し、ワークロードを複数の自動タスクに分散させます。複数の OST または自動タスクに負荷を分けることにより、複数の VPDPU または VPDDCE 項目を同時に実行できるようになります。このようにしない場合は、VPDPU および VPDDCE 項目は順次に実行されます。
- いくつかの構成メンバーを作成します (例えば、1 つのメジャー・ノードごとに 1 つのメンバー)。または VPDALL を使用して、いくつかのコマンド・リストを作成します。
- 各コマンド・リストは複数のタスク (OST と自動タスクなど) で実行することができます。

第 8 章 イベント自動化サービスのカスタマイズ

イベント自動化サービス (E/AS) により、すべてのネットワーク・イベントを好みのプラットフォームから管理できます。 イベント・サーバー (Tivoli Netcool®/OMNIbus または Tivoli Enterprise Console® などの) あるいは NetView for z/OS プログラムのいずれを使用しても、ユーザーのネットワークにおける包括的なイベント・リストを表示することができます。

イベント自動化サービス: 概説

イベント自動化サービスは以下のサービスで構成されます。

- アラート・アダプター・サービス

  アラート・アダプター・サービスは、イベント・アダプターであり、NetView for z/OS アラートを Event Integration Facility (EIF) イベントに変換し、それらのイベントを の指定されたイベント・サーバーに転送します。アラート・アダプター・サービスは、フィルターに掛けられた SNA アラートを NetView ハードウェア・モニターから直接収集して、適切なイベント・クラスまたはサブクラスのインスタンスに変換します。アラートをNetView プログラムから受信するために、イベント自動化サービスは NetView PPI に登録します。フィルターに掛けられたアラートは、NetView ハードウェア・モニターから収集され、PPI 経由でアラート・アダプター・サービスに送信されます。変換されるすべてのアラートは、「 *IBM Systems Network Architecture Management Services Reference* 」に記述されている形式に一致します。

- 確認済みアラート・アダプター・サービス

  確認済みアラート・アダプター・サービスは、NetView for z/OS アラートを EIF イベントに変換するイベント・アダプターです。この結果得られたイベントは、指定のイベント・サーバーに転送されます。 その後、イベント・サーバーは、EIF イベントの受け入れを示す確認で応答します。

  確認済みアラート・アダプター・サービスは、フィルターに掛けられた SNA アラートを NetView ハードウェア・モニターから直接収集して、適切なイベント・クラスまたはサブクラスのインスタンスに変換します。アラートをNetView プログラムから受信するために、イベント自動化サービスは NetView PPI に登録します。フィルターに掛けられたアラートは、NetView ハードウェア・モニターから収集され、 PPI 経由で確認済みアラート・アダプター・サービスに送信されます。アダプターが待ち受ける確認は、確認済みアラート・アダプターのサンプル・クラス定義ステートメント・ファイル IHSABCDS の注に記述されています。変換されるすべてのアラートは、「 *IBM Systems Network Architecture Management Services Reference* 」に記述されている形式に一致します。

- メッセージ・アダプター・サービス

  メッセージ・アダプター・サービスは、イベント・アダプターであり、NetView メッセージ自動化から転送されるすべてのメッセージを EIF イベントに変換します。この結果得られたイベントは、指定のイベント・サーバーに転送されます。

メッセージ・アダプターは、フィルターに掛けられたメッセージを、NetView 自動化テーブルから直接収集して、適切なイベント・クラスまたはサブクラスのインスタンスに変換します。メッセージをNetView プログラムから受信するために、イベント自動化サービスは NetView PPI に登録します。フィルターに掛けられたメッセージは、NetView メッセージ自動化テーブルから収集され、PPI 経由でメッセージ・アダプターに送信されます。

- 確認済みメッセージ・アダプター・サービス

  確認済みメッセージ・アダプター・サービスは、イベント・アダプターであり、NetView メッセージ自動化から転送されるすべてのメッセージを EIF イベントに変換します。この結果得られたイベントは、指定のイベント・サーバーに転送されます。 その後、イベント・サーバーは、EIF イベントの受け入れを示す確認で応答します。アダプターが待ち受ける確認は、確認済みメッセージ・アダプターのサンプル・メッセージ・フォーマット・ファイル IHSANFMT の注に記述されています。

  確認済みメッセージ・アダプターは、フィルターに掛けられたメッセージを、NetView 自動化テーブルから直接収集して、適切なイベント・クラスまたはサブクラスのインスタンスに変換します。メッセージをNetView プログラムから受信するために、イベント自動化サービスは NetView PPI に登録します。フィルターに掛けられたメッセージは、NetView メッセージ自動化テーブルから収集され、PPI 経由で確認済みメッセージ・アダプターに送信されます。

- イベント受信側サービス

  イベント受信側サービスは、イベント・サーバーからイベントを受信し、それらを SNA アラートに変換します。 変換されたアラートは、次に、NetView ハードウェア・モニターに転送され、そこでフィルターに掛けられて NetView 自動化テーブルに経路指定されます。

- Alert-to-trap サービス

  alert-to-trap サービスは、SNMP サブエージェントであり、NetView for z/OS アラートを SNMP トラップに変換し、そのトラップを SNMP エージェントに転送します。 alert-to-trap サービスは、フィルターに掛けられた SNA アラートを NetView ハードウェア・モニターから直接収集して、適切な SNMP トラップ・インスタンスに変換します。アラートをNetView プログラムから受信するために、イベント自動化サービスは NetView PPI に登録します。フィルターに掛けられたアラートは、NetView ハードウェア・モニターから収集され、PPI 経由で alert-to-trap サービスに送信されます。変換されるすべてのアラートは、「 *IBM Systems Network Architecture Management Services Reference* 」に記述されている形式に一致します。

- Trap-to-alert サービス

  trap-to-alert サービスは、SNMP マネージャーからイベントを受信し、それらを SNA アラートに変換します。変換されたアラートは、次に、NetView ハードウェア・モニターに転送され、そこでフィルターに掛けられて NetView 自動化テーブルに経路指定されます。

イベント自動化サービスの開始

イベント自動化サービス (E/AS) は、MVS システム・コンソールから始動プロシージャーを使用するか、または UNIX システム・サービス・コマンド・シェルからコマンド・ファイルを使用して開始できます。E/AS にインストールされているサンプルの始動プロシージャーは IHSAEVNT です。UNIX システム・サービス・コマンド・シェルから E/AS を開始するために使用されるコマンド・ファイルは IHSAC000 です。

E/AS が開始される環境 (つまり MVS システム・コンソールであるか、または UNIX システム・サービス・コマンド・シェルであるか) によって、以下のように E/AS の特定の操作特性が決まります。

- デフォルトの構成ファイルの位置。
- 特定の始動パラメーターを指定できるかどうか。
- トレース/エラー・データのデフォルトの出力ログ。

E/AS の他のすべての操作特性は、始動環境に関係なく同一です。

E/AS のインストール方法と開始方法については、「*IBM Tivoli NetView for z/OS インストール:追加コンポーネントの構成*」を参照してください。

イベント自動化サービスの初期化のカスタマイズ

イベント自動化サービス (E/AS) には、構成可能な設定値が多数あります。いくつかのものは、E/AS を正常に初期化するために、E/AS 管理者によって設定されなければなりません。詳細については、*IBM Tivoli NetView for z/OS インストール:追加コンポーネントの構成* を参照してください。

構成可能な設定値は、E/AS 管理者が構成ファイル、始動パラメーター、および E/AS 変更コマンドを使用して設定することができます。構成可能な設定値のいくつかは、これらの方式のうちの複数のものを使用して設定することができます。構成可能な設定値は、優先順位の高いものから順に、以下の順序で設定されます。

- E/AS 変更コマンドは、初期化後に E/AS に対して発行されます。構成可能な設定値に影響を与える E/AS 変更コマンドは、E/AS の現行の実行中に限りその値を変更します。
- E/AS 始動パラメーターとして指定された構成可能な設定値。
- 構成ファイルで指定された構成可能な設定値。
- 構成可能な設定値のデフォルト値。

E/AS 変更コマンドについては、「*IBM Tivoli NetView for z/OS Command Reference Volume 1 (A-N)*」で詳しく説明されています。

構成可能な設定値のデフォルト値

下記の表に、構成可能な設定値とそのデフォルト値がすべてリストされています。

値	デフォルト	以下のものでオーバーライド
E/AS PPI 名	IHSATEC	PPI 始動パラメーター、グローバル初期設定ファイル PPI ステートメント

値	デフォルト	以下のものでオーバーライド
グローバル初期設定ファイル名	IHSAEVNT - IHSAINIT で開始 IHSAC000 --/etc/netview/global_init.conf で開始	IHSAINIT 始動パラメーター
アラート・アダプター構成ファイル名	IHSAEVNT - IHSAACFG で開始 IHSAC000 --/etc/netview/alert_adpt.conf で開始	ALRTCFG 始動パラメーター、グローバル初期設定ファイル ALRTCFG ステートメント
確認済みアラート・アダプター構成ファイル名	IHSAEVNT - IHSABCFG で開始 IHSAC000 --/etc/netview/confirm_alert_adpt.conf で開始	CALRTCFG または -b 始動パラメーター、グローバル初期設定ファイル CALRTCFG ステートメント
Alert-to-trap 構成ファイル名	IHSAEVNT - IHSAATCF で開始 IHSAC000 --/etc/netview/alert_trap.conf で開始	ALRTTCFG 始動パラメーター、グローバル初期設定ファイル ALRTTCFG ステートメント
Trap-to-alert 構成ファイル名	IHSAEVNT - IHSATCFG で開始 IHSAC000 --/etc/netview/trap_alert.conf で開始	TALRTCFG 始動パラメーター、グローバル初期設定ファイル TALRTCFG ステートメント
メッセージ・アダプター構成ファイル名	IHSAEVNT - IHSAMCFG で開始 IHSAC000 --/etc/netview/message_adpt.conf で開始	MSGCFG 始動パラメーター、グローバル初期設定ファイル MSGCFG ステートメント
確認済みメッセージ・アダプター構成ファイル名	IHSAEVNT - IHSANCFG で開始 IHSAC000 --/etc/netview/confirm_message_adpt.conf で開始	CMSGCFG または -n 始動パラメーター、グローバル初期設定ファイル CMSGCFG ステートメント
イベント受信側構成ファイル名	IHSAEVNT -- IHSAECFG で開始 IHSAC000 --/etc/netview/event_rcv.conf で開始	ERCVCFG 始動パラメーター、グローバル初期設定ファイル ERCVCFG ステートメント
出力ログ折り返し	0	OUTSIZE 始動パラメーター
OpenEdition シェルへのコンソール・メッセージの転送禁止	使用可能。	-P 始動オプション
コンソール・メッセージのファイル名	IHSAEVNT -- IHSAMSG1 で開始 IHSAC000 -- /usr/lpp/netview/msg/C/ihsamsg1 で開始	-M 始動オプション
トレース/エラー HFS パス	/tmp	-E 始動オプション
トレース設定値	すべてのタスクでオフ	グローバル初期設定ファイル TRACE ステートメント、TRACE コマンド
サービスの開始	すべてのサービスが開始される	グローバル初期設定ファイル NOSTART ステートメント
トレース/エラー・データの論理宛先	SYSOUT	グローバル初期設定ファイル OUTPUT ステートメント、OUTPUT コマンド

値	デフォルト	以下のものでオーバーライド
イベント・サーバーのロケーション	デフォルト値なし	アラート・アダプター、確認済みアラート・アダプター、メッセージ・アダプター、および確認済みメッセージ・アダプターの構成ファイル ServerPort ステートメント
イベント・サーバーのポート番号	確認済みアラート・アダプターおよび確認済みメッセージ・アダプターの場合、デフォルトは 5539 です。アラート・アダプターおよびメッセージ・アダプターの場合、デフォルト値は 0 です。	アラート・アダプター、確認済みアラート・アダプター、メッセージ・アダプター、および確認済みメッセージ・アダプターの構成ファイル ServerPort ステートメント
アラート・アダプターのクラス定義ステートメント (CDS) ファイル名	IHSAEVNT -- IHSAACDS で開始 IHSAC000 --/etc/netview/alert_adpt.cds で開始	アラート・アダプター構成ファイル AdapterCdsFile ステートメント
確認済みアラート・アダプターのクラス定義ステートメント (CDS) ファイル名	IHSAEVNT -- IHSABCDS で開始 IHSAC000 --/etc/netview/confirm_alert_adpt.cds で開始	確認済みアラート・アダプター構成ファイル AdapterCdsFile ステートメント
Alert-to-trap アダプターのクラス定義ステートメント (CDS) ファイル名	IHSAEVNT -- IHSALCDS で開始 IHSAC000 -- /etc/netview/alert_trap.cds で開始	Alert-to-trap 構成ファイル AdapterCdsFile ステートメント
Trap-to-alert アダプターのクラス定義ステートメント (CDS) ファイル名	IHSAEVNT -- IHSATCDS で開始 IHSAC000 -- /etc/netview/trap_alert.cds で開始	Trap-to-alert 構成ファイル AdapterCdsFile ステートメント
イベント受信側のクラス定義ステートメント・ファイル名	IHSAEVNT -- IHSAECDS で開始 IHSAC000 --/etc/netview/event_rcv.cds で開始	イベント受信側構成ファイル AdapterCdsFile ステートメント
メッセージ・アダプター・フォーマット・ファイル名	IHSAEVNT -- IHSAMFMT で開始 IHSAC000 -- /etc/netview/message_adpt.fmt で開始	メッセージ・アダプター構成ファイル AdapterFmtFile ステートメント
確認済みメッセージ・アダプター・フォーマット・ファイル名	IHSAEVNT -- IHSANFMT で開始 IHSAC000 -- /etc/netview/confirm_message_adpt.fmt で開始	確認済みメッセージ・アダプター構成ファイル AdapterFmtFile ステートメント
イベント・キャッシュの最大サイズ	64KB	アラート・アダプター、確認済みアラート・アダプター、メッセージ・アダプター、および確認済みメッセージ・アダプターの構成ファイル BufEvtMaxSize ステートメント
イベント・キャッシュ HFS パス	/etc/Tivoli/tec/cache	アラート・アダプター、確認済みアラート・アダプター、メッセージ・アダプター、および確認済みメッセージ・アダプターの構成ファイル BufEvtPath ステートメント

値	デフォルト	以下のものでオーバーライド
最大のイベント・キャッシュ検索バッファー・サイズ	64KB	アラート・アダプター、確認済みアラート・アダプター、メッセージ・アダプター、および確認済みメッセージ・アダプターの構成ファイル BufEvtRdblkLen ステートメント
イベント・キャッシュ縮小量	8KB	アラート・アダプター、確認済みアラート・アダプター、メッセージ・アダプター、および確認済みメッセージ・アダプターの構成ファイル BufEvtShrinkSize ステートメント
イベント・バッファリングを使用可能にする	YES	アラート・アダプター、確認済みアラート・アダプター、メッセージ・アダプター、および確認済みメッセージ・アダプターの構成ファイル BufferEvents ステートメント
イベント・キャッシュのフラッシュ率	0	アラート・アダプター、確認済みアラート・アダプター、メッセージ・アダプター、および確認済みメッセージ・アダプターの構成ファイル BufferFlushRate ステートメント
イベント・キャッシュで許可された最大イベント数	0	アラート・アダプター、確認済みアラート・アダプター、メッセージ・アダプター、および確認済みメッセージ・アダプターの構成ファイル BufferEventsLimit ステートメント
イベント・サーバーの接続モード	connection_less (コネクションレス型)	アラート・アダプター、確認済みアラート・アダプター、メッセージ・アダプター、および確認済みメッセージ・アダプターの構成ファイル ConnectionMode ステートメント
EIF イベントの最大サイズ	4096 バイト	アラート・アダプター、確認済みアラート・アダプター、メッセージ・アダプター、および確認済みメッセージ・アダプターの構成ファイル EventMaxSize ステートメント
EIF イベント・フィルター操作の定義	フィルター定義なし	アラート・アダプター、確認済みアラート・アダプター、メッセージ・アダプター、および確認済みメッセージ・アダプターの構成ファイル Filter ステートメント
イベント・キャッシュ定義からの EIF イベント・フィルター操作	フィルター定義なし	アラート・アダプター、確認済みアラート・アダプター、メッセージ・アダプター、および確認済みメッセージ・アダプターの構成ファイル FilterCache ステートメント

値	デフォルト	以下のものでオーバーライド
EIF イベント・フィルター操作のモード	OUT	アラート・アダプター、確認済みアラート・アダプター、メッセージ・アダプター、および確認済みメッセージ・アダプターの構成ファイル FilterMode ステートメント
接続切断時の再試行間隔、および応答待機時間	120 秒	アラート・アダプター、確認済みアラート・アダプター、メッセージ・アダプター、および確認済みメッセージ・アダプターの構成ファイル RetryInterval ステートメント。確認済みアラート・アダプターおよび確認済みメッセージ・アダプターのみ、応答を待機します。
否定応答の限度	0	確認済みアラート・アダプター、または確認済みメッセージ・アダプターの構成ファイル BufEvtNegRespLimit ステートメント
EIF イベント転送デバッグ・モード	NO	アラート・アダプター、確認済みアラート・アダプター、メッセージ・アダプター、および確認済みメッセージ・アダプターの構成ファイル TestMode ステートメント
イベント受信側 PPI 名	NETVALRT	イベント受信側構成ファイル NetViewAlertReceiver ステートメント
イベント受信側ポート番号	0	イベント受信側構成ファイル PortNumber ステートメント
イベント受信側の PortMapper を使用可能にする	YES	イベント受信側構成ファイル UsePortmapper ステートメント
着信イベント・データから単一の SV31 を作成する。必要に応じて切り捨てを実行	YES	イベント受信側サービスおよび trap-to-alert サービスの構成ファイル TruncateSV31s ステートメント
Alert-to-trap SNMP エージェント IP ロケーション	ループバック	Alert-to-trap サービス構成ファイル Hostname ステートメント
Alert-to-trap コミュニティー名	public	Alert-to-trap サービス構成ファイル Community ステートメント
Alert-to-trap Enterprise オブジェクト ID	1.3.6.1.4.1.1.1588.1.3	Alert-to-trap サービス構成ファイル Enterpriseoid ステートメント
Trap-to-alert PPI 名	NETVALRT	Trap-to-alert サービス構成ファイル NetViewAlertReceiver ステートメント
Trap-to-alert ポート番号	162	Trap-to-alert サービス構成ファイル PortNumber ステートメント

イベント自動化始動パラメーターのカスタマイズ

MVS システム・コンソールから E/AS を開始する場合は IHSAEVNT 始動プロシージャーに、または IHSAC000 コマンドの場合は UNIX システム・サービス・コマンド行に、始動パラメーターを指定することができます。始動パラメーターの一般形式は次の 2 つです。

- `parameter=value`
- `-option [value]`

以下で特に記載されていない限り、どちらの形式もいずれの始動環境からでも使用することができます。ただし、option/value 形式を IHSAEVNT 始動プロシージャーに渡すには、オプション (option) と値 (value) のリストを単一の parameter/value 形式でエンコードする必要があります。IHSAEVNT 始動プロシージャーには、これを行うための下記のパラメーターがあります。

```
OELINE
```

option/value 形式のパラメーターを IHSAEVNT 始動プロシージャーに渡すための OELINE パラメーターの使用例を以下に示します。

```
s IHSAEVNT,OELINE='-opt1 value1 -opt2 value2...'
```

OELINE パラメーターを用いて渡すオプション (option) と値 (value) は、単一引用符で囲みます。

option/value 形式は大/小文字の区別があります。以下のオプションは、示されているとおりに必ず正確に指定してください。値は英大文字に変換されません。オプションによっては、そのオプションだけを指定します。そのオプションには値がありません。

始動パラメーターは以下のものです。

INITFILE=*file* **または -i** *file*

この始動パラメーターでは、グローバル初期設定ファイルの名前を *file* で指定します。**INITFILE=***file* 形式を使用する場合、IHSAEVNT 始動プロシージャーの IHSSMP3 データ・セット定義に関連する 1 文字から 8 文字の PDS メンバー名を *file* に指定します。E/AS を UNIX システム・サービス・コマンド行から開始している場合は、この形式は無効です。 **-i** *file* 形式を使用する場合、指定するファイルは、完全修飾 MVS データ・セット、または HFS パスとファイル名です。MVS データ・セット名は単一引用符で囲み、完全修飾データ・セットにします。例えば、次のようにします。

```
INITFILE=IHSAINIT
-i 'NETVIEW.V6R2M0.SCNMUXCL(IHSAINIT)'
-i /etc/netview/global_init.conf
```

MSGCFG=*file* **または -m** *file*

この始動パラメーターでは、メッセージ・アダプター構成ファイルの名前を *file* で指定します。**MSGCFG=***file* 形式を使用する場合、IHSAEVNT 始動プロシージャーの IHSSMP3 データ・セット定義に関連する 1 文字から 8 文字の PDS メンバー名を *file* に指定します。 E/AS を UNIX システム・サービス・コマンド行から開始する場合、この形式は使用できません。**-m** *file* 形式を使用する場合、完全修飾 MVS データ・セット、または HFS パスと

ファイル名を *file* に指定します。MVS データ・セット名は単一引用符で囲み、完全修飾データ・セットにします。例えば、次のようにします。

```
MSGCFG=IHSAMCFG
-m 'NETVIEW.V6R2M0.SCNMUXCL(IHSAMCFG)'
-m /etc/netview/message_adpt.conf
```

CMSGCFG=*file* **または -n** *file*

この始動パラメーターでは、確認済みメッセージ・アダプター構成ファイルの名前を *file* で指定します。 **CMSGCFG=***file* 形式を使用する場合、指定するファイルは IHSAEVNT 始動プロシージャーの IHSSMP3 データ・セット定義に関連した 1 から 8 文字の PDS メンバー名です。イベント自動化サービスを UNIX システム・サービス・コマンド行から開始する場合、この形式は使用できません。**-n** *file* 形式を使用する場合、指定するファイルは、完全修飾 MVS データ・セット、または HFS パスとファイル名です。MVS データ・セット名は単一引用符で囲み、完全修飾データ・セットにします。例えば、次のようにします。

```
CMSGCFG=IHSANCFG
-n 'NETVIEW.V6R2M0.SCNMUXCL(IHSANCFG)'
-n /etc/netview/confirm_message_adpt.conf
```

ALRTCFG=*file* **または -a** *file*

この始動パラメーターでは、アラート・アダプター構成ファイルの名前を *file* で指定します。**ALRTCFG=***file* 形式を使用する場合、IHSAEVNT 始動プロシージャーの IHSSMP3 データ・セット定義に関連する 1 文字から 8 文字の PDS メンバー名を *file* に指定します。E/AS を UNIX システム・サービス・コマンド行から開始する場合、この形式は使用できません。**-a** *file* 形式を使用する場合、指定するファイルは、完全修飾 MVS データ・セット、または HFS パスとファイル名です。MVS データ・セット名は単一引用符で囲み、完全修飾データ・セットにします。例えば、次のようにします。

```
ALRTCFG=IHSAACFG
-a 'NETVIEW.V6R2M0.SCNMUXCL(IHSAACFG)'
-a /etc/netview/alert_adpt.conf
```

CALRTCFG=*file* **または -b** *file*

この始動パラメーターでは、確認済みアラート・アダプター構成ファイルの名前を *file* で指定します。**CALRTCFG=***file* 形式を使用する場合、指定するファイルは IHSAEVNT 始動プロシージャーの IHSSMP3 データ・セット定義に関連した 1 から 8 文字の PDS メンバー名です。イベント自動化サービスを UNIX システム・サービス・コマンド行から開始する場合、この形式は使用できません。**-b** *file* 形式を使用する場合、指定するファイルは、完全修飾 MVS データ・セット、または HFS パスとファイル名です。MVS データ・セット名は単一引用符で囲み、完全修飾データ・セットにします。例えば、次のようにします。

```
CALRTCFG=IHSABCFG
-b 'NETVIEW.V6R2M0.SCNMUXCL(IHSABCFG)'
-b /etc/netview/confirm_alert_adpt.conf
```

ALRTTCFG=*file* **または -a** *file*

この始動パラメーターでは、alert-to-trap サービス構成ファイルの名前を *file* で指定します。**ALRTTCFG=***file* 形式を使用する場合、指定するファイルは IHSAEVNT 始動プロシージャーの IHSSMP3 データ・セット定義に関連す

る 1 文字から 8 文字の PDS メンバー名を *file* に指定します。E/AS を UNIX システム・サービス・コマンド行から開始する場合、この形式は使用できません。**-a** *file* 形式を使用する場合、指定するファイルは、完全修飾 MVS データ・セット、または HFS パスとファイル名です。MVS データ・セット名は単一引用符で囲み、完全修飾データ・セットにします。例えば、次のようにします。

```
ALRTTCFG=IHSAATCF
-l 'NETVIEW.V6R2M0.SCNMUXCL(IHSAATCF)'
-l /etc/netview/alert_trap.conf
```

TALRTCFG=*file* **または -t** *file*

この始動パラメーターでは、trap-to-alert サービス構成ファイルの名前を *file* で指定します。**TALRTCFG=***file* 形式を使用する場合、IHSAEVNT 始動プロシージャーの IHSSMP3 データ・セット定義に関連する 1 文字から 8 文字の PDS メンバー名を *file* に指定します。E/AS を UNIX システム・サービス・コマンド行から開始する場合、この形式は使用できません。**-t** *file* 形式を使用する場合、指定するファイルは、完全修飾 MVS データ・セット、または HFS パスとファイル名です。MVS データ・セット名は単一引用符で囲み、完全修飾データ・セットにします。例えば、次のようにします。

```
TALRTCFG=IHSATCFG
-t 'NETVIEW.V6R2M0.SCNMUXCL(IHSATCFG)'
-t /etc/netview/trap_alert.conf
```

ERCVCFG=*file* **または -e** *file*

この始動パラメーターでは、イベント受信側構成ファイルの名前を *file* で指定します。**ERCVCFG=***file* 形式を使用する場合、IHSAEVNT 始動プロシージャーの IHSSMP3 データ・セット定義に関連する 1 文字から 8 文字の PDS メンバー名を *file* に指定します。E/AS を UNIX システム・サービス・コマンド行から開始する場合、この形式は使用できません。**-e** *file* 形式を使用する場合、指定するファイルは、完全修飾 MVS データ・セット、または HFS パスとファイル名です。MVS データ・セット名は単一引用符で囲み、完全修飾データ・セットにします。例えば、次のようにします。

```
ERCVCFG=IHSAECFG
-e 'NETVIEW.V6R2M0.SCNMUXCL(IHSAECFG)'
-e /etc/netview/event_rcv.conf
```

PPI=*ppiname* **または -p** *ppiname*

この始動パラメーターでは、E/AS PPI メールボックスの名前を *ppiname* で指定します。例えば、次のようにします。

```
PPI=IHSATEC
-p IHSATEC
```

OUTSIZE=*size* **または -O** *size*

この始動パラメーターは、出力ログ折り返しを可能にするもので、出力ログ・ファイルの最大サイズを K バイトで指定します。*size* に 0 が指定されていると、出力ログ折り返しを使用できません。E/AS 出力について詳しくは、 136 ページの『イベント自動化サービス出力』 を参照してください。

```
OUTSIZE=0
-O 0
```

-M *msgfile*

この始動パラメーターでは、E/AS メッセージ・ファイルの位置を指定します。 *msgfile* には完全修飾 MVS データ・セット、または HFS パスとファイル名を指定します。MVS データ・セット名は単一引用符で囲み、完全修飾データ・セットにします。例えば、次のようにします。

```
-M 'NETVIEW.V6R2M0.SDUIMSG1(IHSAMSG1)'
-M /usr/lpp/netview/msg/C/ihsamsg1
```

-P E/AS を IHSAEVNT 始動プロシージャーから開始する場合、この始動パラメーターは使用できません。これは、UNIX システム・サービス・コマンド・シェルのもとで E/AS が開始された場合に、MVS システム・コンソール・メッセージを UNIX ステム・サービス・コマンド・シェルに転送しないようにするためのものです。デフォルトで、MVS システム・コンソールに対して発行されたメッセージは、UNIX システム・サービス・コマンド・シェルでも発行されます。

-E *path*

E/AS を IHSAEVNT 始動プロシージャーから開始する場合、この始動パラメーターは使用できません。この始動パラメーターでは、トレース/エラー・ログ・ファイルの HFS パスを指定します。*path* には HFS パスを指定します。例えば、次のようにします。

```
-E /tmp
```

イベント自動化サービス構成ファイルのカスタマイズ

E/AS は構成ファイルを使用します。これらのファイルとそのデフォルトの名前は以下のとおりです。

- グローバル初期設定ファイル

  IHSAINIT または /etc/netview/global_init.conf
- アラート・アダプター構成ファイル

  IHSAACFG または /etc/netview/alert_adpt.conf
- 確認済みアラート・アダプター構成ファイル

  IHSABCFG または /etc/netview/confirm_alert_adpt.conf
- alert-to-trap サービス構成ファイル

  IHSAATCF または /etc/netview/alert_trap.conf
- trap-to-alert サービス構成ファイル

  IHSATCFG または /etc/netview/trap_alert_trap.conf
- メッセージ・アダプター構成ファイル

  IHSAMCFG または /etc/netview/message_adpt.conf
- 確認済みメッセージ・アダプター構成ファイル

  IHSANCFG または /etc/netview/confirm_message_adpt.conf
- イベント受信側構成ファイル

  IHSAECFG または /etc/netview/event_rcv.conf

グローバル初期設定ファイルは、これらのサービス全部に必要な、構成可能な設定値を変更するために使用されます。その他の構成ファイルそれぞれは、そのサービス固有の構成可能な設定値を変更するために使用されます。これらのファイル内のステートメントはすべて、1 行に収まっていなければなりません。これらのファイルのそれぞれにコメントを入れることができます。コメント・ステートメントは番号記号 (#) で開始します。

E/AS が IHSAEVNT 始動プロシージャーから開始される場合は、デフォルトで、指定された 8 文字の PDS 名を用いてファイルが探し出されます。ファイルは、IHSAEVNT 始動プロシージャーの IHSSMP3 データ・セット定義ステートメントで指定されたデータ・セット内に存在していなければなりません。E/AS が UNIX システム・サービス・コマンド・シェルから開始される場合は、デフォルトで、指定された HFS 名を用いてファイルが探し出されます。

構成ファイル内のすべてのステートメントをコメントにすることができます。構成ファイル内のすべてのステートメントがコメントであれば、その構成ファイルは、構成可能な設定値を変更しません。構成ファイル内にコメントしか入っていなくとも、E/AS を正しく初期化するためには、4 個の構成ファイルすべてが**存在していなければなりません**。E/AS は、構成ファイルが見つからない場合、初期化を行いません。

構成ファイル・ステートメントの詳細については、「*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。

イベント自動化サービス出力

イベント自動化サービス (E/AS) のすべての出力は、2 つの宛先 (汎用トレース機能 (GTF) と E/AS 出力ログ) のうちの一方または両方に送信されます。デフォルトで、データは E/AS 出力ログに送信されます。E/AS 出力の宛先は、OUTPUT コマンドまたはグローバル初期設定ファイルの OUTPUT ステートメントを用いて変更することができます。詳しくは、「*IBM Tivoli NetView for z/OS Command Reference Volume 1 (A-N)*」および「*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。

3 つのサービスのそれぞれに関連する出力ログと、E/AS アドレス・スペース全体に関連する出力ログがあります。出力ログ折り返しが使用不可であれば、これらの出力ログは物理的に 1 個のシステム・ファイルで表されます。出力ログ折り返しが使用可能であれば、これらの出力ログは物理的に 2 個のシステム・ファイル (1 次ファイルと 2 次ファイル) で表されます。

折り返しが使用不可のときは、すべての出力ログ・データが 1 次ファイルに書き込まれます。

折り返しが使用可能のときは、1 次ファイルまたは 2 次ファイルのいずれかに書き込むことができる合計バイト数を制限するために折り返しサイズが使用されます。この折り返しサイズを超えると、出力ログが出力されていた現行ファイル (1 次ファイルまたは 2 次ファイルのいずれか) はクローズされ、それまで使用中でなかったもう一方のファイル (1 次ファイルまたは 2 次ファイルのいずれか) が開かれて、ログが継続されます。出力ログが開かれると、それまでログに記録されていたすべてのデータは破棄されます。したがって、使用可能な出力ログ・データの最大

量は、折り返しサイズの 2 倍であり (1 次ファイルと 2 次ファイルの両方が満杯の場合)、使用可能な出力ログ・データの最小量は折り返しサイズです (1 次ファイルまたは 2 次ファイルのいずれかに切り替わったばかりで、切り替わったファイルに入っていた以前のデータがすべて破棄された場合)。

出力ログ折り返しの設定方法について詳しくは、132 ページの『イベント自動化始動パラメーターのカスタマイズ』の OUTSIZE パラメーターを参照してください。

イベント自動化サービス出力ログの名前

E/AS が IHSAEVNT 始動プロシージャーにより開始されると、出力ログの名前は、IHSAEVNT プロシージャー内の以下のデータ・セット定義ステートメントによって定義されます。

- IHSA (1 次ファイル) と IHSAS (2 次ファイル): アラート・アダプター・サービス用の出力ログ・ファイルを定義します。
- IHSB (1 次ファイル) と IHSBS (2 次ファイル): 確認済みアラート・アダプター・サービス用の出力ログ・ファイルを定義します。
- IHSC (1 次ファイル) と IHSCS (2 次ファイル): E/AS アドレス・スペース用の出力ログ・ファイルを定義します。
- IHSE (1 次ファイル) と IHSES (2 次ファイル): イベント受信側サービス用の出力ログ・ファイルを定義します。
- IHSL (1 次ファイル) と IHSLS (2 次ファイル): alert-to-trap サービス用の出力ログ・ファイルを定義します。
- IHSM (1 次ファイル) と IHSMS (2 次ファイル): メッセージ・アダプター・サービス用の出力ログ・ファイルを定義します。
- IHSN (1 次ファイル) と IHSNS (2 次ファイル): 確認済みメッセージ・アダプター・サービス用の出力ログ・ファイルを定義します。
- IHST (1 次ファイル) と IHSTS (2 次ファイル): trap-to-alert サービス用の出力ログ・ファイルを定義します。

出力ログ折り返しが使用不可の場合、2 次ファイル用のデータ・セット定義を IHSAEVNT 始動プロシージャーに入れておく必要はありませんが、そのまま残しておく方がよいでしょう。1 次ファイル用のデータ・セット定義は常に入れておく必要があります。

デフォルトで、出力ログ・ファイルは IHSAEVNT ジョブの SYSOUT データ・セットに設定されます。SYSOUT データ・セットが出力ログ・ファイル用に使用されている場合は、出力ログ折り返しは使用不可です。出力ログ折り返しを使用可能にしたいときは、MVS 順次データ・セットまたは HFS ファイルを参照するように、これらのデータ・セット定義を変更する必要があります。

注: IHSAEVNT 始動プロシージャーのデータ・セット定義ステートメントで指定するファイル・タイプに制限は設けられていません。ただし、PDS メンバーは出力ログ・ファイルとして定義しないでください。これは、PDS メンバーにデータを書き込もうとする際に同期の問題が起こる可能性があるからです。また、それぞれのデータ・セット定義ステートメントごとに別のファイルを使用してください。

トレースを使用可能にして実行するよう Tivoli サービス技術員から指示されていない限り、サンプルの IHSAEVNT 始動プロシージャーで指定されているデフォルト SYSOUT データ・セットを使用すること、および出力ログ折り返しを使用可能にはしないことをお勧めします。

UNIX システム・サービス・コマンド・シェルの IHSAC000 を使用して E/AS が開始される場合は、出力ログ・ファイルの名前が以下のように定義されます。

- これらのファイルは HFS ファイルでなければなりません。デフォルトのファイル・パス名は /tmp です。このパスは -E 始動オプションを使用して変更可能です。 135 ページの『-E path』 ページのこのオプションを参照してください。
- controlp.log (1 次ファイル) と controls.log (2 次ファイル) は E/AS アドレス・スペース用の出力ログ・ファイルの名前です。これらの名前は変更できません。
- alertp.log (1 次ファイル) と alerts.log (2 次ファイル) はアラート・アダプター・サービス用の出力ログ・ファイルの名前です。これらの名前は変更できません。
- calertp.err (1 次ファイル) と calerts.err (2 次ファイル) は、確認済みアラート・アダプター・サービス用の出力ログ・ファイルの名前です。これらの名前は変更できません。
- alrttrpp.log (1 次ファイル) と alrttrps.log (2 次ファイル) は、alert-to-trap アダプター・サービス用の出力エラー・ログ・ファイルです。
- trapalrtp.log (1 次ファイル) と trapalrts.log (2 次ファイル) は、trap-to-alert サービス用の出力エラー・ログ・ファイルです。
- messagep.log (1 次ファイル) と messages.log (2 次ファイル) はメッセージ・アダプター・サービス用の出力ログ・ファイルの名前です。これらの名前は変更できません。
- cmessagep.err (1 次ファイル) と cmessages.err (2 次ファイル) は、確認済みメッセージ・アダプター・サービス用の出力ログ・ファイルの名前です。これらの名前は変更できません。
- eventrcvp.log (1 次ファイル) と eventrcvs.log (2 次ファイル) はイベント受信側サービス用の出力ログ・ファイルの名前です。これらの名前は変更できません。

これらの出力ログ・ファイルが存在していなければ E/AS が作成します。

注: トレースを使用可能にして実行するよう Tivoli サービス技術員から指示されていない限り、出力ログ折り返しを使用可能にはしないことをお勧めします。

イベント自動化サービス出力データのタイプ

E/AS は 2 種類の出力データ (トレース・データとエラー・データ) を生成します。

トレース・データが生成されるのは、トレースが使用可能にされている場合だけです。デフォルト値では、トレースは使用不可にされています。トレースの設定値を変更する際、TRACE コマンドに関する情報については「*IBM Tivoli NetView for z/OS Command Reference Volume 2 (O-Z)*」を、またグローバル初期設定ファイルの TRACE ステートメントに関する情報については「*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。

一般に、Tivoli サービス技術員から要請された場合にのみトレースを使用します。

エラー・データは、MVS システム・コンソール・メッセージと出力ログ専用メッセージから構成されます。通常、エラー条件が E/AS によって検出されると、MVS コンソール・メッセージが作成されます。このコンソール・メッセージは E/AS 出力にも書き込まれます。問題判別に役立つように、追加メッセージも E/AS 出力に書き込まれることがあります。MVS システム・コンソールに対して発行されなかったこれらの出力ログ専用メッセージにより、問題点の詳細が得られることがあります。

システム・コンソール・メッセージと出力ログ専用メッセージを組み合わせることにより、Tivoli サービス技術員の援助を受けなくとも、たいていの E/AS の問題を解決することができます。

必ずしもすべての MVS コンソール・メッセージがエラー条件を指摘するわけではありません。E/AS が発行し、E/AS 出力ログに送信される通知メッセージもたくさんあります。

イベント自動化サービス出力データの形式

出力ログ・ファイルが初めて開かれるとき、そのファイルの最初の項目は、出力ファイル名の後に以下の形式で日付/時刻のストリングが続きます。

```
day month date time year
```

以下に、E/AS が IHSAEVNT 始動プロシージャーから開始された場合のメッセージ・アダプター・サービス 1 次出力ログ・ファイルのヘッダー例を示します。

```
IHSM Fri Feb 20 10:45:55 2011
```

その他の E/AS 出力データはすべて、ヘッダーに特定データが続きます。

ヘッダーは以下のもので構成されます。

- 下記の形式の日付/時刻ストリング

  ```
  day month date time year
  ```
- メッセージが発行されたモジュールのモジュール名
- メッセージが発行されたモジュール内の行番号
- 以下のいずれかのメッセージ・タイプ
- LOW - LOW または高水準トレースが使用可能にされている場合にこのメッセージが発行されることを示します。
- NORMAL - NORMAL または高水準トレースが使用可能にされている場合にこのメッセージが発行されることを示します。
- VERBOSE - VERBOSE トレース・レベルが使用可能にされている場合にこのメッセージが発行されることを示します。
- CONSMSG - これが MVS コンソール・メッセージであることを示します。
- LOGONLY - これは MVS コンソール・メッセージに伴うメッセージであることを示します。ただし、これは E/AS 出力にのみ発行されます。
- IP - IP トレースが使用可能にされている場合にこのメッセージが発行されることを示します。

E/AS 出力項目の例を以下に示します。

```
 -----date/time--------    module line
|                      | |        |  |
Fri Feb 20 10:45:55 2011 IHSAEASO:2016
                        msgtype ->specific data
                        |     |
                        CONSMSG: IHS0075I Event/AutomationService started
Subtask initialization is in progress for IHSATEC
```

この例のコンソール・メッセージ IHS0075I は、示されている E/AS モジュールから、示されている日時に発行されました。

注: モジュールと行番号は、追加の問題判別が必要な場合に、Tivoli サービス技術員によって使用されます。

NetView プログラムからのアラートおよびメッセージ経路指定のカスタマイズ

NetView プログラムがインストールされるとき、イベント自動化サービスへのアラートおよびメッセージ・データの経路指定は、デフォルトで使用不可にされます。NetView 自動化テーブルのステートメントやハードウェア・モニターのフィルター・コマンドを使用して、イベント自動化サービスへのアラートおよびメッセージ・データの経路指定を使用可能にします。アラートおよびメッセージの NetView プログラムから E/AS への経路指定を使用可能にし、カスタマイズする場合の詳細情報は、「*IBM Tivoli NetView for z/OS 自動操作ガイド*」を参照してください。

複数のイベント自動化サービスの実行

複数の E/AS アドレス・スペースを同時にアクティブにすることができます。多くの場合は 1 つの E/AS で十分ですが、以下に挙げたいずれかの理由で複数個が必要になることがあります。それは、以下のことを行う場合です。

- アラートまたはメッセージのサブセットを変換し、別のイベント・サーバーに送信する。
- アラートまたはメッセージを変換し、複数のイベント・サーバーに送信する。
- EIF イベントのサブセットを変換し、別の NetView アラート受信側に送信する。
- EIF イベントを変換し、複数の NetView アラート受信側に送信する。

複数の E/AS を実行する場合は、E/AS PPI メールボックス名を E/AS ごとに固有の名前にする必要があります。その他の構成可能な設定値はすべて、E/AS 呼び出し間で共用することができます。ただし、以下の構成可能な設定値は E/AS 呼び出し間で変更することを考慮に入れてください。

- 複数のイベント受信側サービスを使用するときは、そのうちの 1 つだけを PortMapper に登録します。その他のものにはポート番号を指定し、PortMapper の使用を使用不可にします。複数のイベント受信側が PortMapper を使用しようとすると、PortMapper を最後にアクセスしたイベント受信側だけが実際に登録され、それ以外のイベント受信側の登録はすべて失われます。イベント受信側 PortMapper の登録が上書きされると、警告メッセージが MVS システム・コンソールに書き込まれます。
- E/AS 出力ログ・ファイルはそれぞれの E/AS 呼び出しごとに固有でなければなりません。そうしないと、ある E/AS のデータが、別の E/AS のデータと同じ出力ログ・ファイルにインターリーブされます。IHSAEVNT 始動プロシージャーを

使用して E/AS を実行しており、しかも出力ログ・ファイルが SYSOUT データ・セットである場合は、それぞれの E/AS 呼び出しごとにこれらのデータ・セットは自動的に固有にされます。

拡張カスタマイズ - データの変換

操作特性を定義するために E/AS が使用する構成ファイルのほかに、各 E/AS サービスは、着信データを EIF イベントまたは SNMP トラップに変換する方法をサービスに伝えるルール一式が入った変換ファイルを使用します。それぞれの変換ファイルは、カスタマイズ可能なテキスト可読ファイルです。

E/AS のサービスが使用する変換ファイルは 2 種類の形式があります。アラート・アダプター、確認済みアラート・アダプター、alert-to-trap サービス、trap-to-alert サービス、およびイベント受信側サービスは、クラス定義ステートメント (CDS) 変換ファイルを使用します。メッセージ・アダプター・サービスと確認済みメッセージ・アダプター・サービスは、メッセージ・フォーマット変換ファイルを使用します。

これらの変換ファイルをカスタマイズするには、EIF イベント、SNMP トラップ、またはその両方の形式を理解している必要があります。

SNMP トラップの詳細については、該当する SNMP エージェント用の z/OS 資料を参照してください。

クラス定義ステートメント・ファイル

クラス定義ステートメント (CDS) ファイルは、データ送信側から送信される情報を基に EIF イベントを構成する方法を定義します。アラート・アダプター・サービス、確認済みアラート・アダプター・サービス、および alert-to-trap サービスの場合、データ送信側は NetView プログラムです。イベント受信側サービスの場合、データ送信側はイベント・サーバーです。trap-to-alert サービスの場合、データ送信側は SNMP トラップ・マネージャーです。このファイル内のステートメントを、クラス定義ステートメントと呼びます。クラス定義ステートメントとは、サービスが受信する着信データをコンソール・イベントにマップできるようにするためのルールです。

注: イベント受信側サービス、alert-to-trap サービス、および trap-to-alert サービスは、これらのクラス定義ステートメントを用いて作成される EIF イベントをさらに処理し、それをアラートまたは SNMP トラップに変えます。イベント・サーバーからのアラートの作成について詳しくは、161 ページの『イベント受信側の CDS 後処理』を参照してください。アラートからトラップを作成する方法の詳細については、187 ページの『Alert-to-Trap の CDS 後処理』を参照してください。SNMP トラップからアラートを作成する方法の詳細については、179 ページの『Trap-to-Alert の CDS 後処理』を参照してください。

CDS ファイルは 1 つ以上の CDS で構成されます。各 CDS にはデータを EIF イベントにマップするためのルールを指定する **SELECT**、**FETCH** および **MAP** セグメントを含めることができます。これらのルールを用いることにより、着信データ

を基にイベント・クラスを**選択し**、コンソール・イベントを作成するために追加データを**取り出し**、着信イベントから収集した情報を発信 EIF イベントのイベント属性に**マップする**ことができます。

CDS の一般形式は以下のとおりです。

```
CLASS <class_name> SELECT <select_statements> FETCH <fetch_statements>
    MAP <map_statements> END
```

CDS ファイルはコメント記号 (#) で始まるコメント行もサポートします。

CDS のキーワードは次のような情報を提供します。

CLASS

<*class_name*> では、着信データがこの CDS に一致する場合に発信コンソール・イベントで使用されるクラス名を定義します。

SELECT

この CDS と一致する、つまりこの CDS を選択するために着信データが満たしていなければならない 1 つ以上の <*select_statement*> 項目からなります。select_statement (選択ステートメント) は、SELECT セグメントに現れている順に評価されます。特定 CDS のすべての <*select_statements*> を満たしていれば、着信データはその CDS と一致します。そうでなければ、アダプターは着信データを次の CDS と突き合わせます。どの CDS とも一致しない着信データは廃棄されます。

FETCH

map セグメントのイベント属性を構築するために着信データから追加データを検索する際に使用されるゼロ個またはそれ以上の <*fetch_statement*> 項目からなります。 FETCH セグメントは、SELECT セグメントによって検索されなかったデータを検索するため、または SELECT セグメントによって検索されたデータを変更するために使用されます。

MAP サービスのデフォルト・データ、ユーザー定義の定数データ、および SELECT や FETCH セグメントで取得したデータを使用して、 EIF イベント・インスタンスのイベント属性の構築方法を指定する、ゼロ個以上の <*map_statement*> 項目からなります。

アラート・アダプター・サービスの場合、イベント・サーバー上にあるサービスの **.baroc** ファイルで定義されたイベント・クラスのそれぞれは、CDS ファイルの 1 つ以上の CDS と一致しなければなりません。 CDS では、着信データを発信 EIF イベント・インスタンスのクラスおよびイベント属性にマップする方法を指定します。CDS ファイルのクラスやイベント属性を変更または追加するときは、イベント・サーバー上の **.baroc** ファイルにも相応の変更を行ってください。

イベント受信側サービスの場合、発信 EIF イベントがイベント・サーバーに送信されることはありません。これは疑似イベントであり、さらに処理されてアラートが作成されます。したがって、イベント受信側の CDS ファイルから作成されるどの EIF イベントについても、イベント・サーバー上には対応する **.baroc** ファイルはありません。

CDS はそれぞれ、CDS ファイル内に現れている順に評価されます。着信イベントは、SELECT セグメントが一致する最初の CDS によって指定されているクラスに

マップされます。特定のイベント・クラスに対して複数の CDS を用意するときは、最も制約が多い SELECT セグメントを持つ CDS を CDS ファイルの初めに置いてください。

<*class_name*> が ***DISCARD*** に等しい場合は、SELECT セグメントに一致する着信データであっても廃棄されます。どの CDS とも一致しないデータも廃棄されることに注意してください。ただし、特定タイプの着信データを常に廃棄する必要があれば、***DISCARD*** ステートメントを定義して CDS ファイルの先頭に置く方が、アダプターにすべての CDS を突き合わさせてから最終的にそのイベントを破棄するよりも効率的です。

着信イベント・データのエンコード

着信イベント・データはサービスによって名前/値 (name/value) の組にエンコードされます。名前/値の組は、***属性*** とも呼ばれます。どの着信イベントの場合も、属性はすべて 1 つのリストに入れられ、そのリストが SELECT、FETCH および MAP セグメントで使用されます。サービスがどの着信データを選択して (全部選択ではない場合) 名前/値の組にエンコードするかについては、この後の特定のサービスでのエンコード方式の説明を参照してください。

属性のうちの名前 (name) 部分はテキスト・ストリングです。名前は***総称*** と***キーワード*** の 2 種類があります。

総称名は、サービスによって作成されるテキスト・ストリングです。サービスはこれらの名前を内部で作成することもありますし、着信生データで提供される情報から作成することもあります。いずれの場合も、サービスが属性名を作成する際の方式については、この章の後ろの具体的なサービスのエンコードのところで説明されています。

キーワードの形式は *$keyword* です。普通、着信データ・ストリームでサービスに提供されるデータは、通常、総称名ではなくキーワードでコード化されます。実際のキーワード名は着信データから取り出されるのではなく、サービスによって定義されます。

キーワードと総称名の主な違いは、CDS ファイル処理における名前の使われ方です。キーワードの方が CDS ファイル処理中のデータ・ルックアップは速くなります。その他の点では、キーワードも総称名もデータ・タグにすぎず、キーワードは先頭に「$」が付いているだけのことです。

属性のうちの値 (value) 部分もテキスト・ストリングです。この場合も、サービスは生イベントのデータに基づいてテキスト・ストリングを割り当てます。

アラート・アダプター・サービス、確認済みアラート・アダプター・サービス、および Alert-to-Trap サービスのデータ・エンコード

アラート・アダプター・サービス、確認済みアラート・アダプター・サービス、および alert-to-trap サービスは、キーワード属性のみを使用してデータをエンコードします。以下の表に、使用される個々のキーワード属性名と、着信アラート・データから値フィールドへの割り当て方法を示します。

属性名	説明
$ALERT_CDPT	総称アラート・データ・サブベクトルのアラート記述コード・フィールド、またはレゾリューション・データ・サブベクトルのレゾリューション記述コード・フィールドからの 2 バイトの 16 進値。
$ORIGIN	階層名リストまたは階層/リソース・リスト・サブベクトルからの階層の名前/タイプ (name/type) の組からなる 1 つの文字ストリング。このストリングは以下の形式の階層を含みます。 `resnam1/typ1,resnam2/typ2,resnam3/typ3, resnam4/typ4,resnam5/typ5` 実際のサブベクトル内にある組の数だけが使用されます。
$SUB_ORIGIN	階層名リストまたは階層/リソース・リスト・サブベクトルからの階層の名前/タイプ (name/type) の組のリストのうちの、最後の組の文字ストリング。このストリングの形式は以下のとおりです。 `resnam`*x*`/typ`*x* ここに、*x* はリスト内の最後の組の番号です。
$HOSTNAME	アラートが発信された SNA ノードの *netid.nau* ノード名。NetView/390 ノード、AS/400® ノードなど。
$ADAPTER_HOST	NetView アラート・アダプター、または確認済みアラート・アダプターが常駐しているホストの IP 名。
$DATE	NetView アラート・アダプター、または確認済みアラート・アダプターがアラートを受信した日付。形式は `MMM HH:MM:SS`、例 `OCT 10 12:08:30`。
$SEVERITY	FATAL、CRITICAL、など。総称アラート・データ・サブベクトルからのアラート・タイプ・フィールド、またはイベント・タイプを使用して重大度が決定されます。「146 ページの表 16」を参照してください。
$MSG	問題を記述している *Long Error Description:Long Probable Cause* メッセージ。このメッセージは、NPDA ALERTS-DYNAMIC パネルに表示される `ALERT DESCRIPTION:PROBABLE CAUSE` メッセージに似ています。

属性名	説明
$ADAPTER_HOST_SNANODE	アラートを NetView アラート・アダプター、または確認済みアラート・アダプターに送信した NetView システムの *netid.domainid* ノード名。
$EVENT_TYPE	例えば、PERMANENT または TEMPORARY。総称アラートでは、総称アラート・データ・サブベクトルのアラート・タイプ・バイトを検査することによってこれが得られます。これは、NPDA EVENT DETAIL (NPDA イベント詳細) パネルに表示される EVENT TYPE (イベント・タイプ) と一致します。
$ARCH_TYPE	GENERIC_ALERT、GENERIC_RESOLUTION、または NONGENERIC_ALERT。総称アラート・データ・サブベクトルを含む NMVT アラート主ベクトルは、GENERIC_ALERT です。NMVT レゾリューション主ベクトルは GENERIC_RESOLUTION です。他のすべてのアラートは NONGENERIC_ALERT です。
$PRODUCT_ID	アラートまたはイベント送信側の、ハードウェアまたはソフトウェアの PSID (プロダクト・セット識別)。これは 4、5、7、または 9 文字です。すべての総称アラートと、一部の非総称アラートに関係します。
$ALERT_ID	個別のアラート条件を指定するために送信側が割り当てた 8 文字の 16 進値。レゾリューション・アラートの場合、この値は常に 00000000。総称アラート (レゾリューションを含む) にのみ関係します。
$BLOCK_ID	アラートに関連する IBM ハードウェアまたはソフトウェアを識別するためのコード。*NetView 資源アラート解説書* の資料を参照してください。非総称アラートにのみ関係します。
$ACTION_CODE	事前定義画面の索引となるコード。非総称アラートにのみ関係します。非総称アラートでは、ブロック ID とアクション・コードの組み合わせで、送信側製品を固有に識別します。
$SELF_DEF_MSG	自己定義テキスト・メッセージ Sv31 から抽出されたテキスト。

属性名	説明
$EVENT_CORREL	MSU 相関 Sv47 から抽出された関連子。これらの関連子はアラートを他のアラートと関連付けます。つまり、原因となる同じ問題に関して 2 つ以上のアラートを受け取る可能性があり、そのようなアラートは Sv47 によって関連付けられています。イベント・マネージャー・サーバー上の tecad_snaevent.rls ファイルに、すでに報告済みのアラートを廃棄するルールが含まれています。
$INCIDENT_CORREL	機能不良識別サブベクトルから抽出された関連子。これらの関連子はアラートをレゾリューションと関連付けます。イベント・マネージャー・サーバー上の tecad_snaevent.rls ファイルに、レゾリューションを受信したときにすべての関連アラートをクローズする (CLOSE) ルールが含まれています。
$ADAPTER_CORREL	アラート・アダプターにのみ有効な関連子。
$DETAILED_DATA	常にストリング "[N/A]"。
$CAUSES	常にストリング "[N/A]"。
$ACTIONS	常にストリング "[N/A]"。

非キーワード属性は、NetView アドレス・スペースでユーザーが割り当てることもできます。NetView プログラムから転送されるアラートのカスタマイズ方法の詳細については、「*IBM Tivoli NetView for z/OS 自動操作ガイド*」を参照してください。この方式を用いて、任意の属性の名前/値 (name/value) の組を作成し、CDS ファイル・プロセスで使用することができます。アラート・アダプター・サービスと trap-to-alert サービスは、総称属性が NetView プログラム内で割り当てられた場合以外は、総称属性を使用しません。

重大度イベント属性の値は、アラート・タイプ (またはイベント・タイプ) を重大度に対応させることによって決定されます。 以下の表に、この対応関係を示します。16 進数バイトは、総称アラート・データ・サブベクトルのアラート・タイプ・フィールドです。

表 16. アラート・タイプと重大度

アラート・タイプ	重大度
0x01, PERMANENT	CRITICAL
0x02, TEMPORARY	HARMLESS
0x03, PERFORMANCE	WARNING
0x04, INTERVENTION REQ'D	CRITICAL
0xNN, CUSTOMER APPLICATION	MINOR
0xNN, END USER GENERATED	MINOR
0xNN, SUMMARY	HARMLESS
0xNN, INTENSIVE MODE REC	HARMLESS
0x09, AVAILABILITY	CRITICAL

表 16. アラート・タイプと重大度 (続き)

アラート・タイプ	重大度
0x0A, NOTIFICATION	WARNING
0x0B, ENVIRONMENT	CRITICAL
0x0C, INSTALLATION	WARNING
0x0D, OPERATION/PROCEDURE	WARNING
0x0E, SECURITY	CRITICAL
0x0F, DELAYED RECOVERED	WARNING
0x10, PERMANENT AFFECTED	MINOR
0x11, IMPENDING PROBLEM	WARNING
0x12, UNKNOWN	UNKNOWN
0xNN, HELD	MINOR
0x14, BYPASSED	WARNING
0x15, REDUNDANCY LOST	WARNING
0x16, SITUATION	WARNING
0xNN, RESENT ALERT	MINOR
0xNN, RESOLVED PROBLEM	HARMLESS
0xNN, UNSUPPORTED TYPE	UNKNOWN

Alert-to-Trap サービスのデータ・エンコード

alert-to-trap サービスは、エンタープライズ・トラップ (タイプ 6) を構成します。CDS ファイルにより、トラップ内の特定コード・フィールドのカスタマイズが可能になります。これを行うには、CDS ファイルの MAP セクション内の SPECIFIC キーワードに値を指定します。

alert-to-trap サービスの基本的な方法は、アラートから EIF イベントのキーワード/値の組を構成し、そのキーワード/値の組 (SPECIFIC 以外) を SNMP OCTET ストリングにマップし、結果のトラップに変数結合データとして組み込みます。キーワードと値の両方が、作成される OCTET ストリングに組み込まれます。

alert-to-trap サービスでは、アラート・アダプター・キーワード属性にアクセスすることができ、これらの属性は、SELECT、MAP、および FETCH ステートメントで使用できます。ただし、すべてのアラート・アダプター属性が SNMP トラップに適用できるわけではありません。

クラス定義ステートメント内の CLASS 名は、alert-to-trap サービスによって作成されるトラップでは使用されません。しかし、CLASS 名は、CDS 構文規則に従うために必要であり、ユーザーが構成しようとしているトラップを文書化する際に便利です。

Trap-to-Alert サービスのデータ・エンコード

trap-to-alert サービスは、その着信データとして SNMP トラップを受信します。このデータは、キーワード属性と総称属性の両方にエンコードされます。

次の表は、trap-to-alert サービスによって作成されるキーワード属性をリストしています。

属性名	説明
$ORIGIN_ADDR	この値は、トラップの作成元の IP アドレスが含まれるストリングです。サンプルのデータグラム転送デーモンが使用される場合、その値は、デーモンが実行されるホストの IP アドレスであることに注意してください。
$ORIGIN_PORT	この値は、トラップの作成元の起点アドレスのポート番号 (10 進数) が入っているストリングです。サンプルのデータグラム転送デーモンが使用される場合、その値は、デーモンがトラップの転送に使用したポートの番号です。
$SNMP_VERSION	この値は、トラップを送信したエージェントでどの SNMP バージョンがインプリメントされたかを示す数値 (10 進数) が入っているストリングです。これにより、トラップのフォーマット方法が判別されます。SNMPv1 の値は "0" です。

次の表は、変数結合でない SNMP トラップ・データから、trap-to-alert サービスによって作成される総称属性をリストしています。すべてのデータが文字ストリングに変換されてから、それに総称属性名が割り当てられます。

属性名	説明
community	SNMP トラップ・コミュニティー・フィールドの値。
enterpriseOID	SNMP トラップ enterpriseOID フィールドの値。
agent_address	SNMP トラップ・エージェント・アドレス・フィールドの値。
generic_trap	SNMP トラップ総称トラップ・フィールドの値。
specific_trap	SNMP トラップ固有トラップ・フィールドの値。
timestamp	SNMP トラップ・タイム・スタンプ・フィールドの値。

変数結合データは、変数結合データから直接作成されます。変数結合名は総称属性の名前になり、変数結合データは、まだ文字ストリングでない場合は文字ストリングに変換され、総称属性に割り当てられます。1 つの SNMP トラップ内にある複数の変数結合の名前が同じである場合、その名前とインデックスが名前に追加され、総称属性名を作成します。例えば、変数結合名が次のとおりであるとします。

```
1.3.6.1.4.1.2.2.1.3.1.0
```

この名前が、同じ SNMP トラップ内で 3 回発生した場合、その結果作成される総称属性名は次のようになります。

```
1.3.6.1.4.1.2.2.1.3.1.0
1.3.6.1.4.1.2.2.1.3.1.0<1>
1.3.6.1.4.1.2.2.1.3.1.0<2>
```

イベント受信側サービスのデータ・エンコード

イベント受信側サービスは EIF イベントを着信データとして受信します。このデータは、キーワード属性と総称属性の両方にエンコードされます。データはすでに属性の名前/値形式になっているので、このエンコードは非常に単純です。着信コンソール・イベント内のすべてのイベント属性名は、属性リストにおける総称属性の名前になり、対応するイベント属性値はその属性の値になります。 イベントの className は、$CLASSNAME キーワード属性の値としてエンコードされます。したがって、イベント受信側は、1 つのキーワード属性 $CLASSNAME と、着信 Tivoli Enterprise Console イベント内にあるイベント属性/値の組と同じ数の総称属性を作成します。

クラス定義ステートメントの SELECT セグメント

CDS の SELECT セグメントは、1 つ以上の *<select_statement>* 項目からなります。各 *<select_statement>* 項目の形式は次のとおりです。

```
<n>: ATTR(<a_op>, <a_op_value>),
     VALUE(<v_op>, <v_op_value>);
```

<select_statement> の条件が満たされるのは、*<select_statement>* の **ATTR** および **VALUE** の式で指定された条件を満たす属性が、サービスにより提供される属性リスト内に検出された場合です。SELECT セグメントが満たされるためには、それぞれの *<select_statement>* ごとに、属性が検出されなければなりません。SELECT セグメントが満たされない場合はその CDS 全体が無視され、CDS ファイルの次の CDS から処理が継続されます。

<n> これは、*<select_statement>* の識別番号です。*n* は任意の有効な整数です。*<select_statement>* には、固有の識別番号を付ける必要があります。この識別番号は、CDS のそれ以降の処理で使用されます。

ATTR 属性の名前を *<a_op_value>* で指定し、さらにその属性名に関する変更条件を *<a_op>* で指定します。**ATTR** 式は select_statement には必須です。着信データを基にサービスによって作成される属性リストは、名前フィールドが **ATTR** 式で示される条件に一致する属性が見つかるまで検索されます。

<a_op>

ATTR 名に条件を加えるもので、以下の値のうちのいずれか 1 つを指定できます。

= *<a_op_value>* で指定する属性名が、属性リストの属性名と一致しなければならないことを指定します。

PREFIX

<a_op_value> で指定する属性名が、属性リストの属性名の接頭部でなければならないことを指定します。

SUFFIX

<*a_op_value*> で指定する属性名が、属性リストの属性名の接尾部でなければならないことを指定します。

<*a_op_value*>

属性の名前を指定します。属性リストが順に検索され、一致する属性が見つかるまで **ATTR** <*a_op*> 式が各属性名フィールドに適用されます。

デフォルトでは、<*a_op_value*> はストリングです。ただし、<*a_op_value*> は変数でもかまいません。変数については、以下のリストで説明します。

<*a_op_value*> をストリングとして指定する際に、そのストリング内にブランク文字が含まれている場合や、そのストリングがすべて数字 (0 から 9) である場合は、<*a_op_value*> を二重引用符 (") で囲んでください。次の例は、使用可能な <*a_op_value*> ストリングを示しています。

```
hello
$ORIGIN
"hello, world"
"12"
```

<*a_op_value*> を変数として指定する場合は、下記の変数タイプのいずれでも指定することができます。

キーワード

イベント・アダプターによって提供されるキーワード。例えば $ORIGIN。

名前 名前変数には、前の <*select_statement*>**ATTR** 式を満たす属性の名前フィールドの値が割り当てられます。名前変数は、$N*n* として指定されます。ここで、*n* は、所要の属性と一致した <*select_statement*> の番号です (例: $N2)。

値 値変数には、前の <*select_statement*> **VALUE** 式を満たす属性の値フィールドの値が割り当てられます。値変数は、$V*n* として指定されます。ここで、*n* は、所要の属性と一致した <*select_statement*> の番号です (例: $V5)。

下記の **ATTR** 式の例は、**user1** に等しい総称名を探します。サービスが **user1** という属性名を提供していれば、この **ATTR** 式は満たされます。

```
ATTR(=,"user1")
```

下記の **ATTR** 式の例は、**$ORIGIN** に等しいキーワードを探します。サービスが **$ORIGIN** という属性名を提供していれば、この **ATTR** 式は満たされます。

```
ATTR(=,$ORIGIN)
```

VALUE

この式はオプションです。対応する **ATTR** 式に一致した属性リスト内の属性について、その属性値が **VALUE** 式の情報を基に突き合わされます。

<*v_op*>

VALUE 式に条件を加えるもので、以下の値のうちのいずれか 1 つを指定できます。

= **VALUE** 式の <*v_op_value*> が、属性リストの属性値と一致しなければならないことを指定します。

PREFIX

VALUE 式の <*v_op_value*> が、属性リストの属性値の接頭部でなければならないことを指定します。

SUFFIX

VALUE 式の <*v_op_value*> が、属性リストの属性値の接尾部でなければならないことを指定します。

!= **VALUE** 式の <*v_op_value*> が、属性リストの属性値と**同じであってはならない**ことを指定します。

<*v_op_value***>**

属性の値を指定します。デフォルトでは、<*v_op_value*> はストリングです。ただし、<*v_op_value*> は変数でもかまいません。

<*v_op_value*> をストリングとして指定する際に、そのストリング内にブランク文字が含まれている場合や、そのストリングがすべて数字 (0 から 9) である場合は、<*v_op_value*> を二重引用符 (") で囲んでください。次の例は、使用可能な <*v_op_value*> ストリングを示しています。

```
hello
$ORIGIN
"hello, world"
"12"
```

<*v_op_value*> を変数として指定する場合は、下記の変数タイプのいずれでも指定することができます。

キーワード

キーワードには定数値 (ストリングまたは数値) が割り当てられ、キーワードを用いてその値を参照することができます。

名前 名前変数には、前の <*select_statement*> **ATTR** 式を満たす属性の名前フィールドの値が割り当てられます。名前変数は、`$N`*n* として指定されます。ここで、*n* は、所要の属性と一致した <*select_statement*> の番号です (例: $N2)。

値 値変数には、前の <*select_statement*> **VALUE** 式を満たす属性の値フィールドの値が割り当てられます。値変数は、`$V`*n* として指定されます。ここで、*n* は、所要の属性と一致した <*select_statement*> の番号です (例: $V5)。

下記の **VALUE** 式の例は、値の接頭部が **Serial** である属性を探します。

```
VALUE(PREFIX,"Serial")
```

この **VALUE** 式に一致するものとしては **Serial1** があります。

SELECT セグメントの評価:

1. 1 つの SELECT セグメント全体が一致するためには、SELECT セグメント内の <*select_statement*> 式それぞれの属性が一致しなければなりません。属性リストの複数の属性が <*select_statement*> を満たす場合があります。属性リスト内でステートメントを満たす最初の属性が、それ以降の CDS 処理に使用されます。
2. SELECT セグメントが満たされると、その SELECT セグメントのクラス名が、発信 EIF イベントに使用されます。イベント処理は FETCH セグメントで続行されますが、クラスが ***DISCARD*** の場合は例外です。その場合、イベントは

破棄されます。着信データが CDS ファイル内の CDS のどの SELECT セグメントにも一致しなければ、着信データは廃棄されます。

3. *<select_statement>* が一致するつど、2 個の変数 ($N*n* と $V*n*) が作成されます。これらの変数は、アダプター提供のキーワードと共に、後続の SELECT、FETCH または MAP セグメントで使用することができます。

クラス定義ステートメントの FETCH セグメント

CDS の SELECT セグメントは着信データから属性名と値を検索するだけで、選択された情報を変更することはできません。状況によっては、属性値のサブストリングを抽出することや、ユーザー定義変数を用意することが必要になります。CDS の FETCH セグメントにより、これが可能になります。

FETCH セグメントは、1 つ以上の *<fetch_statement>* 式からなります。各 *<fetch_statement>* の形式は次のとおりです。

```
<n>:  <expression>
```

内容は次のとおりです。

<n>
: これは *<fetch_statement>* の識別番号です。*<n>* は任意の有効な整数です。各 *<fetch_statement>* には固有の識別番号を付ける必要があります。*<fetch_statement>* では *<expression>* の値が新規変数 $F*n* に割り当てられます。ここで、*n* は *<fetch_statement>* の識別番号です。

<expression>
: 下記のうちのいずれか 1 つです。
  - ストリング
  - SELECT セグメントからの出力値 (アダプター提供のキーワードおよび SELECT セグメント変数など)。
  - 前の *<fetch_statement>* からの出力
  - ストリング、SELECT セグメント出力、および *<fetch_statement>* 出力を任意に組み合わせたサブストリング

サブストリングを使用する FETCH セグメントの例を以下に示します。

```
1: SUBST ($V2, 1, 5);
```

このステートメントは、変数 $V2 の値 (*<select_statement>* 番号 2 から割り当てられたもの) を使用し、$V2 の初めの 4 文字で表されるサブストリングを変数 $F1 に割り当てます。

FETCH セグメントの出力は、フェッチ変数 $F*n* のセットです。

クラス定義ステートメントの MAP セグメント

CDS の MAP セグメントにより、発信 EIF イベントに入れるイベント属性および関連した値が作成されます。

MAP セグメントは、1 つ以上の *<map_statement>* 式から構成されます。それぞれの *<map_statement>* の形式は、以下のいずれかになります。

```
<slot name> = <string>;
<slot name> = <variable>;
<slot name> = PRINTF(<format_string>, <var1>, ..., <varn>);
```

<*slot_name*>
任意のイベント属性の名前。 アラート・アダプター・サービスの場合、これは、イベント・サーバー上にあるこのサービスの **.baroc** ファイルのイベント属性に対応するイベント属性でなければなりません。 イベント受信側サービスの場合、これは、イベント受信側の CDS ファイル後処理に使用可能なイベント属性です。

<*string*>
任意の文字ストリング。

<*variable*>
SELECT セグメントまたは FETCH セグメントから MAP セグメントに渡される任意の変数。例えば、アダプター定義キーワードまたはセグメント変数など。

PRINTF
イベント属性の値を C 言語形式の **printf()** 書式制御ストリングを用いてフォーマット設定できるようにする書式を指定します。 この書式制御ストリングは現在 **%s** 書式指定子のみをサポートします。

<*var*> <*string*> または <*variable*> のいずれかを含めることができます。

下記は MAP セグメントの例です。

```
MAP
  origin = $V2;
  hostname = $HOSTNAME;
  msg = PRINTF("The origin is %s", $V2);
```

この例の **origin** イベント属性には、SELECT セグメントの変数 $V2 の値が入ります。 **hostname** イベント属性には、$HOSTNAME キーワードの値が入ります。変数 $V2 の値が **NV390SP/SP** であるものと仮定すると、**msg** イベント属性の値は "**The origin is NV390SP/SP** " になります (二重引用符は値に含まれません)。

マップ・プロセスの出力は、発信 EIF イベントの生成に使用されるイベント属性の名前/値の組のリストです。発信 Tivoli Enterprise Console イベントはイベント・サーバーに送信されるか、 CDS ファイルの後処理に使用されます。

クラス定義ステートメント・ファイルの MAP_DEFAULT セクション

一部のイベント属性 (source や hostname など) は、特定のサービスによって生成されるすべての EIF イベントに対し定数値を取る場合があります。多くの CDS で同一の map_statement を繰り返さなくてもすむように、CDS ファイルでは MAP_DEFAULT セクションがサポートされます。このセクションで、CDS ファイルのすべての CDS に対するイベント属性の名前/値の組を定義します。 このグローバル定義セクションで定義されているイベント属性は、CDS 内の特定の定義でオーバーライドすることができます。

下記は MAP_DEFAULT セクションの例です。

```
MAP_DEFAULT
  origin = $ORIGIN;
  sub_origin = $SUB_ORIGIN;
  msg = $MSG;
END
```

場合によっては、CDS を複数の CDS ファイルに入れ、それらすべてをサービスに使用させたいことがあります。これを可能にするために、通常の CDS ファイル処理の拡張版が E/AS サービスに追加されています。**%INCLUDE** ステートメントにより、追加の CDS ファイルを現行 CDS ファイルに組み込むことができます。%INCLUDE キーワードの前にブランク文字を付けてはならず、その後ろには区切り記号として 1 個のブランク文字を続ける必要があります。区切り記号の後ろに、開かれる CDS ファイルのファイル名を続けます。このファイル名は、IHSSMP3 データ・セット定義に関連した 1 文字から 8 文字の PDS メンバー名であるか、円記号 (¥) 文字が先頭にある完全なファイル名のいずれかです。同時に開くことのできる CDS ファイル・メンバーの最大数は 20 です。これは、%INCLUDE ステートメントのネスト可能な最大数でもあります。

%INCLUDE ステートメントの構文例を以下に示します。ファイル IHSAACD1 には以下の単一ステートメントが含まれているものとします。

```
sub_origin = $SUB_ORIGIN;
```

例:

```
MAP_DEFAULT                             //Statements from IHSAACDS
  source = NV390ALT;
  origin = $ORIGIN;
%INCLUDE IHSAACD1                       //New file with sub_origin statement
  hostname = $HOSTNAME;                 //Continuation of IHSAACDS
  adapter_host = $ADAPTER_HOST;
END
```

CDS の使用例については、イベント自動化サービスに付属しているサンプルの IHSAACDS、IHSABCDS、または IHSAECDS を参照してください。これらは、アラート・アダプター・サービス、確認済みアラート・アダプター・サービス、イベント受信側サービスのそれぞれに使用されるデフォルトの変換ファイルです。

メッセージ・フォーマット・ファイル

FMT ファイルでは、メッセージ・アダプター・サービスと確認済みメッセージ・アダプター・サービスが、NetView プログラムから送信されたメッセージ情報を基に EIF イベントを構成する方法を定義します。 このファイル内のステートメントを、*書式仕様ステートメント* (FSS) と呼びます。 書式仕様ステートメントは、サービスが NetView プログラムから収集する着信メッセージ・データを、発信コンソール・イベントにマップできるようにするためのルールです。

以下のセクションでは、メッセージ・アダプター・サービスと確認済みメッセージ・アダプター・サービスにおける書式仕様の構文と、書式仕様がイベントにマップされる方法を説明します。

着信イベント・データのエンコード

メッセージ・アダプター・サービス、および確認済みメッセージ・アダプター・サービスにとっての着信データは、メッセージ・ストリングです。メッセージ・テキ

スト・ストリングは FMT ファイルの***書式仕様 (format specifications)*** と突き合わせされます。したがって、1 次情報はメッセージ・ストリングそのものです。

CDS ファイルの場合と同様に、FMT ファイルのジョブにより、ユーザーは着信メッセージ・データを基に発信コンソール・イベントをカスタマイズできます。 この方式ではデータを属性にエンコードせずに、特定のイベント属性名で着信メッセージ・データからデフォルト情報を受け取ります。

デフォルトの各イベント属性名と、それに対応するデフォルト値を、以下の表にリストします。イベント属性の値が着信データ内に実際に存在していない場合、デフォルトのイベント属性値はヌル・ストリングです。書式仕様ステートメントのマップ・ルール部分にリストされるすべてのイベント属性にデフォルト値があります。着信データで値が提供されない場合、そのデフォルト値はヌル・ストリング ("") です。

イベント属性名	**説明**
origin	メッセージ発信元 NetView システムの *netid.domainid* ノード名。
sub_origin	メッセージに関連したジョブ番号。メッセージのジョブ番号が利用不能なときは、この値のデフォルト値はヌル・ストリング ("")。
hostname	origin イベント属性と同じ。
adapter_host	イベント自動化サービスを実行中のホストの IP 名。
date	NetView 自動化テーブルからメッセージが送信された日時。形式は `MMM HH:MM:SS`、例 `OCT 10 12:08:30`。
msg_id	メッセージの最初のトークン。たいていの場合、このトークンは実際のメッセージ ID。

severity	msg_id の最後の文字から判断されます。この文字は次のように変換されて、このイベント属性の値になります。
	`A, E, S CRITICAL`
	`T FATAL`
	`anything else WARNING`

msg	メッセージ・テキスト。その最初のトークンとして msg_id を組み込みます。
adapter_host_snanode	メッセージをメッセージ・アダプター・サービス、または確認済みメッセージ・アダプター・サービスに送信した NetView システムの *netid.domainid* ノード名。
multiline_msg	メッセージの 2 番目以降のメッセージ行。メッセージが 1 行だけからなる場合、multiline_msg の値は **N/A**。
jobname	メッセージに関連したジョブ名。メッセージでジョブ名が利用不能なときは、jobname の値のデフォルト値はヌル・ストリング ("")。

デフォルトのイベント属性と値は、NetView アドレス・スペースでユーザーが割り当てることもできます。NetView プログラムから転送されるメッセージのカスタマイズについて詳しくは、「*IBM Tivoli NetView for z/OS 自動操作ガイド*」を参照してください。この方式を用いて、任意の属性の名前/値 (name/value) の組を作成し、FMT ファイル・プロセスで使用することができます。

書式仕様

FMT ファイルは 1 つ以上の FSS で構成されます。 1 つの FSS は以下の 3 つの部分からなります。

- 書式ヘッダーは、キーワード **FORMAT** とその後にクラス名が続きます。その後にオプションとして、**FOLLOWS** キーワードおよび直前に定義された **FORMAT** クラス名が続きます。着信メッセージがこの FSS に一致すると、**FORMAT** キーワードに続くクラス名が発信 EIF イベントで使用されます。
- 書式内容は書式制御ストリングをもち、その後にオプションとしてマップ・ルール・リストが続きます。書式制御ストリングは CDS ファイルの **SELECT** セグメントと同様の機能を実行します。つまり、着信メッセージを特定の FSS と突き合わせます。マップ・ルールは、CDS ファイルの **MAP** セグメントと同様の機能を実行します。つまり、イベント属性に値を割り当てます。
- **END** キーワードは FSS を終了します。

書式ヘッダー、書式制御ストリング、個々のマップ・ルール、および **END** キーワードは、改行して開始する必要があります。

FOLLOWS という関係を用いて、より一般的な FSS から特定の FSS を構築できるようにします。 書式 B が書式 A に続いていると、B は A の (書式制御ストリング以外の) マップ・ルールをすべて継承します。書式 B には任意に追加のマップ・ルールを定義することができますが、B で再定義されたマップ・ルールは、A からの継承ではなくなります。書式 B では、継承したマップ・ルールを再定義することによってオーバーライドできます。

NetView プログラムによって転送されるメッセージは、一般的に共通の形式をしており、メッセージ ID とメッセージ固有のテキストからなります。 書式制御ストリングでは、これらのメッセージ・コンポーネントを、C 言語形式の **printf()** 表記に非常によく似ているコンポーネント指定子表記を用いて表すことができます。この **printf()** 表記は CDS ファイルで使用される表記と同じです。

以下の書式制御ストリングは、NetView 自動化テーブルにより作成されるメッセージ・クラス全体を記述します。

```
%s*
```

入力メッセージは定数とブランクにトークン化されています。定数とは、非ブランク文字からなる任意の連続ストリングです。FSS を特定メッセージと突き合わせようとするときには、コンポーネント指定子により、定数とブランクをグループ化してさらに複雑な「トークン」にすることができます。現在使用できるコンポーネント指定子は以下のものです。

```
%lengths Matches one constant in the input message
%lengths* Matches zero or more constants in the input message
%lengths+ Matches one or more constants in the input message
```

オプションの ***length*** は任意の大きさの 10 進数で、定数の実際の長さが指定子の長さより大きい場合にこの長さで定数を切り捨てます。複数の定数と突き合わせる指定子の場合は、累積されたストリング内の各定数が切り捨てられます。また、指定子の長さより短い定数の場合は、ストリング自体が実際の長さで終了します。

書式制御ストリング **DSI%s %s*** は、E/AS と共に出荷されるデフォルト・メッセージ・アダプター FMT ファイルから得られます。以下の説明ではこれを用いて書式制御ストリングの使用方法を解説します。

メッセージを **DSI%s %s*** 書式仕様と突き合わせる例として、下記の NetView メッセージを取り上げます。

```
DSI002I INVALID COMMAND: 'BADCOMMAND'
```

コンポーネント指定子および突き合わせは以下のようになります。

```
DSI DSI
%s 002I
%s* INVALID COMMAND:'BADCOMMAND'
```

DSI002I メッセージには、いくつかの定数部分といくつかの可変部があります。つまり、生成されるどの DSI002I メッセージについても、メッセージの特定部分 (定数部分) は同じです。このメッセージの定数部分は以下のとおりです。

```
DSI002I INVALID COMMAND: '  '
```

このメッセージの可変部は以下のとおりです。

```
BADCOMMAND
```

メッセージの最初の単一引用符 (') まではメッセージの最初の定数部分です。2 番目の単一引用符は、メッセージの 2 番目の定数部分の開始になりますが、これはたまたまメッセージの最後の文字でもあります。単一引用符で囲まれているデータはすべて変数です。

下記のメッセージは可変部が異なる別の DSI002I メッセージの例です。

```
DSI002I INVALID COMMAND: 'WORSE COMMAND'
```

この例の可変部は、2 個のワードとスペース (WORSE COMMAND) で構成されています。

書式制御ストリング **DSI%s %s*** を以下のようにして、DSI002I メッセージ専用にすることができます。

```
DSI %s INVALID COMMAND: '%s*'
```

前述の DSI002I メッセージを使用すると、コンポーネント指定子と突き合わせは以下のようになります。

```
DSI DSI
%s 002I
INVALID COMMAND: 'INVALID COMMAND: '
%s* WORSE COMMAND
' '
```

メッセージのワードを区切るブランク文字も書式制御ストリングに入れる必要があります。書式制御ストリング内の 1 個のスペース文字は、メッセージ内の任意の個数のブランク文字に相当します。

ここで、前述の DSI002I 専用書式制御ストリングのコロン (:) と引用符 (') との間のスペースを削除してみます。

```
DSI %s INVALID COMMAND:'%s*'
```

この例の書式制御ストリングはもはや DSI002I メッセージと一致しません。ただし、以下の例では NetView メッセージは書式仕様に**一致します**。それは、入力メッセージでも書式仕様でも、連続するすべてのブランクは単一のブランク文字に要約されるからです。

```
DSI %s INVALID COMMAND:       '%s*'
```

任意の長さのリピーター・コンポーネント指定子 (**%s*** および **%s+**) を使用するときには注意が必要です。下記の書式制御ストリングは賢明とはいえません。

```
This is not a good format %s* %s*
```

最初の **%s*** は、メッセージの終わりまでのすべての文字と一致しますが、2 番目の **%s*** はどの文字にも一致しません。これで問題はないように思えますが、『マップ・ルール』に記載されているマップ・ルールの説明を参照すると、その重要性がわかります。

ただし、下記の書式制御ストリングは意味があります。

```
This is a good format %s* : %s*
```

最初の **%s*** は、最初のコロン (:) までのすべての文字と一致し、2 番目の **%s*** は、メッセージの終わりまでのすべての文字と一致します。

この例でわかるように、コンポーネント指定子を定数に置き換えるか、任意の長さのリピーター指定子を定数で終了させて固定長に制限することにより、より限定的なイベントに一致するように一般形式を特殊化することができます。

マップ・ルール

サービスは、着信メッセージ・データをイベント属性の名前/値の組を持つイベント・クラスに変換し、この情報をイベント・サーバーに送信します。アラート・アダプター・サービスの場合同様、メッセージ・アダプター・サービスにより作成された発信 EIF イベントと突き合わせるための **.baroc** ファイルがイベント・サーバーに存在していなければなりません。これは、確認済みメッセージ・アダプター・サービスの場合は不要です。

イベント・クラスは、すでに述べたように、入力メッセージを FSS と突き合わせることにより決定されます。ただし、いったんクラスが決定されると、イベント属性名に値が割り当てられなければなりません。これらの値は、例えば、メッセージ自体、サービスによって提供されるデフォルト・イベント属性、FMT ファイル内の指定情報など、さまざまな場所から入手できます。マップ・ルールにより、イベント属性に値を割り当てる方法を定義します。

書式制御ストリングのマップ・ルール部分は、**.baroc** ファイル・イベント属性名とそれに続く値指定子を含む、ゼロ行以上の行から成り立ちます。値指定子は、下記の 4 種類のうちの 1 つです。

- $*i*。ここに *i* は、書式制御ストリングにおけるコンポーネント指定子の位置を示します。コンポーネント指定子それぞれには、1 から書式制御ストリング内にあるコンポーネント指定子の最大数までの番号が付きます。例えば、前述の

DSI002I メッセージ専用書式仕様における **%s*** コンポーネント指定子は、マップ・ルールでは **$2** として参照されます。$*i* 値指定子の値 (これもまた 変数 値指定子と呼ぶ) は、コンポーネント指定子によって使われた入力メッセージ部分です。これらの変数は、CDS ファイルの SELECT セグメントおよび FETCH セグメントから出力される変数に非常によく似ています。

- 定数ストリング。イベント属性の値は指定されたストリングです。このストリングが単一定数ならば、二重引用符 (") で囲まずに指定することができます。それ以外の場合は、二重引用符で囲む必要があります。
- 1 つの **PRINTF** ステートメント。この方法により、他のイベント属性からさらに複雑なイベント属性を組み立てることができます。**PRINTF** ステートメントはキーワード **PRINTF**、その後に C 言語形式の **printf()** 書式制御ストリングおよびイベント属性名のリストが続きます。**printf()** 書式制御ストリングは現在 **%s** 変換指定子のみをサポートします。**PRINTF** ステートメントで使用されるイベント属性の値もまた、$*i* 値仕様または定数ストリング値仕様のいずれかから得られたものでなければなりません。これらを別の **PRINTF** 値仕様から得ることはできません。引数イベント属性の値は、**printf()** 書式制御ストリングに従って新規定数ストリングを構成する場合に使用します。この定数ストリングがイベント属性の値になります。この値指定子は CDS ファイルの **PRINTF** MAP セグメントの形式に非常に似ています。
- **DEFAULT**。このキーワードは、アダプター自身の内部論理を用いて、指示されたイベント属性の値を得ることを表します。例えば、着信メッセージ・データにメッセージ発信元の hostname (netid.nau) が含まれているものとします。そのため、hostname イベント属性が値 DEFAULT に設定されていれば、netid.nau は hostname イベント属性の値になります。これはアラート・アダプター・サービスにおけるキーワードの使用方法とほぼ同じです。

  着信メッセージにスロットの具体的な値がない場合は、 DEFAULT 値はヌル・ストリング ("") です。値が指定されていないスロット名の DEFAULT 値はオーバーライドできます。DEFAULT の値指定子の後に、追加の値指定子 (コロン (:) で区切られている) が続きます。スロット値が着信メッセージで提供されない場合にのみ、この値指定子が使用され、そのスロットの DEFAULT 値を提供します。

  DEFAULT の指定変更をするには、定数ストリングおよび $i 変数指定子のみが使用できます。

  例えば、以下の場合、着信メッセージからスロット *numericslot* に DEFAULT 値を割り当てます。

  ```
  numericslot DEFAULT : 0
  ```

  着信メッセージに *numericslot* の値が含まれていない場合、ヌル・ストリングではなく、値 0 が割り当てられます。

  DEFAULT はキーワードであるため、値がストリング DEFAULT である定数マップは、二重引用符 ("") で囲んで指定する必要があります。

1 つの書式仕様では、**.baroc** ファイルのそれぞれのイベント属性につき 1 つだけマップ・ルールを指定します。マップ・ルールはもっと一般的な書式仕様から (**FOLLOWS** キーワードを使用して) 継承することができますし、または入力メッセ

ージにほとんど一致する書式仕様で明示的に定義することができます。サービスは、イベント・サーバー上に存在する **.baroc** ファイルをアクセスすることはないので、書式仕様とそれに対応する **.baroc** ファイル定義とが確実に一致するよう注意を払わねばなりません。例えば、マップ・ルールのイベント属性名のつづりが誤っていると、サービスはエラーを何も報告せずに、イベント・サーバーには通常どおりイベントを送信します。ところが、このイベントはイベント・サーバーにとって無意味です。

着信メッセージには、**.baroc** ファイルのどのイベント属性にも直接対応しない属性が含まれていることがあります。 ところが、サービスではこれらの値を使用して **PRINTF** 形式の定数ストリングを組み立てなければならないことがあります。このデータは一時イベント属性に割り当てる必要があります。これで、**PRINTF** 値の仕様で使用することができますが、独立したイベント属性名/イベント属性値の組としてそのイベント属性がイベント・サーバーに送信されることはありません。一時イベント属性は、マップ・ルール内でイベント属性名の直前に負符号 (-) を付けて指定します。これらの一時イベント属性は、 **.baroc** ファイル・イベント属性ではありません。 PRINTF 仕様で一時イベント属性を参照する際は、負符号 (-) を使用しないでください。

%INCLUDE ステートメント

%INCLUDE ステートメントにより、追加の FMT ファイルを現行 FMT ファイルに組み込むことができます。%INCLUDE キーワードの前にブランク文字を付けてはならず、その後ろには区切り記号として 1 個のブランク文字を続ける必要があります。区切り記号の後ろに、開かれる FMT ファイルのファイル名を続けます。このファイル名は、IHSSMP3 データ・セット定義に関連した 1 文字から 8 文字の PDS メンバー名であるか、円記号 (¥) 文字が先頭にある完全なファイル名のいずれかです。同時に開くことのできる FMT ファイル・メンバーの最大数は 20 です。これは、%INCLUDE ステートメントのネスト可能な最大数でもあります。

フォーマット・ファイルの例: 以下のサンプルを使用して、これまでに説明した概念を具体的に示します。このサンプルは、メッセージ・アダプター・サービスのデフォルトのメッセージ・フォーマット・ファイル (IHSAMFMT) 内のサンプルを一部変更したものです。

```
FORMAT NV390MSG_Event
%s*
source NV390MSG
origin DEFAULT
desctext "This string will be overridden"
END
FORMAT NV390MSG_NetView_NCCF FOLLOWS NV390MSG_Event
DSI%s %s*
sub_source "NetView NCCF"
msgnumber $1
temp1 $2
desctext PRINTF("Got a DSI message: %s", temp1)
END

%INCLUDE MOREFMTS
```

このフォーマット・ファイルを使用し、下記のメッセージをサービスが受信したものとします。

```
DSI002I INVALID COMMAND: 'A BAD COMMAND'
```

このメッセージに一致する別の書式仕様が、MOREFMTS 内の追加の書式仕様で指定されていなければ、このメッセージは上記で定義された NV390MSG_NetView_NCCF 書式仕様に一致します。FMT ファイルの FSS の突き合わせは、ファイル内の**最後の** FSS から開始され、一致するまで最初の FSS に向かって処理が進められることに注意してください。

この突き合わせにより、source イベント属性にストリング値 NV390MSG が割り当てられます。 origin イベント属性には、イベント・アダプターによりこのイベント属性に対応付けられているデフォルト値が割り当てられます。 desctext イベント属性には最初はストリング This string will be overridden が割り当てられます。 これらのイベント属性はすべて、より一般的な NV390MSG_Event FSS によって割り当てられ、その後で NV390MSG_NetView_NCCF FSS に引き継がれます。

sub_source イベント属性には NetView NCCF の値が割り当てられます。 msgnumber イベント属性には値 002I (最初の *%s** 仕様で入力メッセージから分析されたもの) が割り当てられます。 -temp1 一時イベント属性には、ストリング INVALID COMMAND: 'A BAD COMMAND' (2 番目の *%s** 仕様で入力メッセージから分析されたもの) が割り当てられます。 その後この一時変数は **PRINTF** 値指定子で使用され、desctext イベント属性をストリング Got a DSI message: INVALID COMMAND: 'A BAD COMMAND' にオーバーライドします。

-temp1 イベント属性以外のすべてのイベント属性は、発信 EIF イベントを構築するために使用されます。イベントのクラス名は、最も具体的に一致した FSS の名前である NV390MSG_NetView_NCCF になります。

FSS の使用例については、イベント自動化サービスに付属しているサンプルの IHSAMFMT (メッセージ・アダプター・サービス) または IHSANFMT (確認済みメッセージ・アダプター・サービス) を参照してください。

イベント受信側の CDS 後処理

アラート・アダプター・サービス、確認済みアラート・アダプター・サービス、メッセージ・アダプター・サービス、および確認済みメッセージ・アダプター・サービスでは、変換ファイルを用いて着信サービス特有のデータを EIF イベントに変換します。イベント受信側では、CDS ファイルを用いてこれと逆のこと (イベントを NetView アラートに変換する) を行います。

これを行うため、イベント受信側による CDS ファイル処理は、アラート・アダプター・サービス、または確認済みアラート・アダプター・サービスによる CDS ファイル処理から多少変更されます。構文上は、141 ページの『クラス定義ステートメント・ファイル』で記述されているすべての内容がイベント受信側の CDS ファイルにもそのまま当てはまります。イベント受信側は、CDS ファイル・プロセスの出力であるイベントを疑似イベントとみなします。つまり、このイベントはイベント・サーバーには送信されず、NMVT にエンコードされる特定のイベント属性用に解析されることになります。

入力属性リスト

着信 EIF イベントは属性リストにエンコードされます。これについては本章のサービス特有のエンコードのセクションで説明しています。着信イベントが解析される

ときに作成される **$CLASSNAME** キーワードのほかに、イベント受信側によって入力属性リスト用に作成される追加キーワードがあります。 以下で追加キーワードを説明します。

キーワード	説明	デフォルト
$NMVT_TYPE	作成される NMVT のタイプ (アラートまたはレゾリューション)。このキーワードは NMVT_TYPE イベント属性によって変更されます。NMVT_TYPE イベント属性の値は ALERT または RESOLVE です。	ALERT
$CDS_GROUP	このキーワードには、GROUP001、GROUP002、 ... GROUP999 のセット内の値が格納されます。このキーワードの値は、CONTINUE イベント属性の値を使用して設定されます。$CDS_GROUP キーワードと CONTINUE イベント属性について詳しくは、167 ページの『複数の CDS 突き合わせによる疑似イベントの作成』を参照してください。	GROUP001
$BUILD_SV31LIST	BUILD_SV31LIST イベント属性の値が割り当てられます。このイベント属性の値は NO または YES です。 アラートが構築されるとき、サブベクトル 31 を元の EIF イベントのそれぞれのイベント属性/値の組に追加するかどうかを決定する際にこのキーワードの値を使用します。$BUILD_SV31LIST キーワードと BUILD_SV31LIST イベント属性の詳細については、172 ページの『元のイベントを含む SV 31 の構築』を参照してください。	YES

出力疑似イベント

他の EIF イベント同様、この疑似イベントもクラス名と、それに続くイベント属性/値の組から成り立ちます。このイベントは Tivoli Enterprise Console に送信されないので、どの Tivoli Enterprise Console サーバーにもこれらのイベントに対応する **.baroc** ファイルは存在しないことに注意してください。通常、CDS ファイルでは任意のイベント属性/値の組、および任意のクラス名を疑似イベントに書き込むことが

できます。疑似イベントに任意のクラス名およびイベント属性/値の組を書き込むことができても、イベント受信側は、事前定義された特定のイベント属性名だけを使用してイベントをアラートに変換します。それ以外のイベント属性は無視されます。

疑似イベント・クラス名

イベント受信側は EIF イベントを変換する際に疑似イベント・クラス名を使用しません。イベント受信側 CDS ファイルのすべての CDS に同じ名前を付けることもできます が、さまざまな CDS の編成およびデバッグを容易にするために、CDS ファイルのそれぞれの CDS ごとに異なるクラス名を使用することをお勧めします。 E/AS と共に出荷されるサンプルの CDS ファイルで使用されている規則では、NMVT 内の特定サブベクトルの作成に関連する CDS はグループ化されており、それらのクラス名は共通の文字ストリングで開始されています。さらに、クラス名の最後になんらかの固有の指定を追加して、クラス名を固有に区別できるようにしています。

例:

```
CLASS SV05_1
...
END
CLASS SV05_2
...
END

CLASS SV05_3
...
END

...
```

この例では、CDS ステートメントそれぞれの SELECT セグメント (表示されていません) により、別のサブベクトル 05 が構築されます。最終的に構築される SV 05 のクラス名は、これを SV 05 として識別する固有な名前になります。この場合も、この情報は編成とデバッグを目で見て分かりやすくするために使われるにすぎません。

NMVT_TYPE イベント属性

CDS の MAP セグメントで NMVT_TYPE イベント属性をコーディングすることにより、NMVT のタイプがアラートであるかレゾリューションであるかを指定することができます。 このイベント属性に有効な値は **RESOLVE** と **ALERT** の 2 つです。このイベント属性の値は、**$NMVT_TYPE** キーワードにコピーされます。

SV イベント属性

このイベント属性は NMVT に置かれるサブベクトル作成の主な手段です。

イベント属性名の接頭部は SV でなければなりません。イベント属性名の残り部分は任意の文字ストリングで構いません。 SV05、SVAA および SVNONSENSE はすべて **SV** イベント属性として認識されます。 この場合も、分かりやすさとデバッグを考慮して、 SV05、SV92、SV05_1 など、作成されるサブベクトルの番号をイベント属性名に含めるようにしてください。

SV イベント属性値には全サブベクトル (長さおよびサブベクトル・キーを含む) が含まれます。 CDS の MAP セグメントで SV イベント属性に割り当てられる値は、文字ストリングと解釈されます。イベント受信側は数字ストリングを、アラートで使用される 16 進値にデコードします。 サンプル CDS ファイルのサブベクトル・イベント属性の例を以下に示します。

```
SV05 = "0B0509100004E3C5C30040";
```

SV05 の値は 16 進文字からなる文字ストリングです。イベント受信側は、この文字ストリングを NMVT に組み込めるように本当の数値形式に変換します。イベント受信側は、このサブベクトルについての検証を行いません。NMVT に置かれるサブベクトルは以下のようになります。

```
0B0509100004E3C5C30040
```

CDS ファイルの一般構文に従い、イベント属性値に数字の 0 から 9 だけが含まれている場合は、ストリングとして解釈されるように値を二重引用符で囲む必要があります。上記の例では英字 (16 進値 A から F を表す) が使われているので、イベント属性値を引用符で囲む必要はありません。 ただし、SV イベント属性を二重引用符で囲むことを習慣にするとよいでしょう。

16 進数ストリングの変換を禁止する

場合によっては、16 進値では**ない**文字ストリングをサブベクトル・ストリングに追加することができます。上記で述べたように、イベント受信側は、デフォルトで、イベント属性値のストリングが 16 進文字 (0 から 9、A から F) 列であるものと見なして、16 進数ストリングを数値形式に変換しようとします。 上記の例では、16 進数ストリング E3C5C3 は、EBCDIC では TEC です。

ストリング TEC をイベント属性値内に直接指定するには、ストリングを <> 括弧で囲んでください。 この括弧の前にはエスケープ文字 # を付ける必要があります。例えば、この規則を使用すると、ストリングは次のようになります。

```
SV05 = "0B0509100004#<TEC#>0040"
```

このイベント属性値は、次のように、最初の例とまったく同じ NMVT サブベクトルを作成します。

```
0B0509100004E3C5C30040
```

括弧が存在していると、括弧で囲まれたデータは変換が必要な 16 進数ストリングでは**なく**、NMVT に直接入れるストリングであることがイベント受信側に指示されます。

出力サブベクトルにおける属性リスト・データの使用

イベント属性には、CDS 変数 ($V、$N、$F 変数) の値、またはキーワードの値、または属性リストの総称属性を割り当てることができます。これらの変数を使用すれば、変数値は変換されません。また、これらの変数にはコード化サブベクトル全体が含まれていないこともあります。これを処理するには、**PRINTF** 形式の MAP ステートメントで値を割り当てると便利です。

ここでは、前述した SV 05 の例を拡張し、ストリング TEC は SELECT セグメントによって生成された $V2 変数の値であると仮定します。NMVT に対して同一の SV 05 を作成するには、次のように入力します。

```
SV05 = PRINTF("0B0509100004#<%s#>0040", $V2);
```

PRINTF 構文を使用しているので、%s 書式指定子は $V2 変数の値 (これは TEC) で置換されます。# 付き括弧があるので、イベント受信側は TEC ストリングを数値形式に変換しません。この場合も作成される次のサブベクトルは、最初の 2 例で作成されたものと同一です。

```
0B0509100004E3C5C30040
```

出力サブベクトルに元の EIF イベントのデータを割り当てるときはいつも、ストリング変換を使用不可にして PRINTF 構文を使用する必要があります。ただし、着信イベントのイベント属性値がストリング E3C5C3 であって、ストリング TEC ではないことがあります。この場合、次のストリングを使用して、必要な NMVT サブベクトルを作成します。

```
SV05 = PRINTF("0B0509100004%s0040", $V2);
```

16 進数ストリングの変換を禁止し続けた場合、出力サブベクトルは以下のようになります。

```
0B0509100004C5F3C3F5C3F30040
```

このようになるのは、E、3、C、5、C および 3 の 6 文字それぞれが文字の状態 (C5、F3、C3、F5、C3 および F3) で残されるからです。

サブベクトル/サブフィールドの長さの自動計算

最初の SV 05 の例を取り上げます。

```
SV05 = "0B0509100004E3C5C30040";
```

サブベクトルの長さはストリング内に直接コーディングされています。サブベクトル内には可変情報がないので、この長さは CDS MAP セグメントのイベント属性値に直接コード化されます。変数データが使用されていると、CDS ファイルの作成時にサブベクトルの長さが分からないことがあります。

サブベクトルに属性リスト・データを挿入する次の例を考えてみましょう。

```
SV05 = PRINTF("0B0509100004#<%s#>0040", $V2);
```

この例の $V2 変数の値は TEC であるため、長さは 3 でした。この長さは、サブベクトルの合計長 (0B)、サブフィールド 10 の長さ (09)、およびリソース名の長さ (04) の算出に使用されました。実際は、$V2 変数の値の長さは、イベントが到着するまで不明です。

イベント受信側がサブベクトル・ストリング部分の長さを計算できるように、ストリングのその部分を中括弧 {} で囲んでください。中括弧にはエスケープ文字 # を付けて拡張する必要があります。中括弧は、長さが計算されると、ストリングから除去されます。ただし、左中括弧はサブベクトル・ストリングにおける長さフィールドのプレースホルダーです。

次のように上記の例を変更します。

```
SV05 = PRINTF("#{05#{1000#{#<%s#>#}0040#}#}", $V2);
```

このイベント属性の変換を段階的に追ってみます。 まず **PRINTF** の置換が起こります。

```
SV05 = "#{05#{1000#{TEC#}0040#}#}";
```

この段階での出力サブベクトルは以下のようになります。

```
...E3C5C3...
```

ここで、省略符号は、これからサブベクトルに変換されるすべてのデータを表します。次に、セグメント #{TEC#} を使用してリソース名項目の長さが計算されます。

出力サブベクトルは次のようになります。

```
...04E3C5C3...
```

最初の #{ はセグメントの長さで置換され、それに対応する #} は除去されます。次に、セグメント #{100004TEC0040#} を使用してサブフィールド 10 項目の長さが計算されます。

出力サブベクトルは次のようになります。

```
...09100004E3C5C30040
```

このときも、#{ はセグメントの長さで置換され、それに対応する #} は除去されます。最後に、セグメント #{050910001000004TEC0040#} を使用してサブベクトル 05 全体の長さが計算されます。

最終の出力サブベクトルは次のようになります。

```
0B0509100004E3C5C30040
```

BUILD_SV31LIST イベント属性

元の EIF イベント全体は、デフォルトで SV 31 にコード化されて、NMVT に付加されます。クラス名、イベント属性/値のそれぞれの組、および END 指定機能は別々の SV 31 にコード化されます。**BUILD_SV31LIST** イベント属性により、この SV 31 のリストを NMVT に追加するかどうかをユーザーが制御することができるようになります。 疑似イベントの完了時に、イベント内に **BUILD_SV31LIST** イベント属性が存在し、しかもその値が **NO** であれば、SV 31 リストは除外されます。それ以外の場合、SV 31 リストは組み込まれます。

1 つのスロット/値のペアが、SV 31 が許可するものを超える場合、そのスロット/値ストリングは追加の SV 31 に引き続き組み込まれます。継続する SV 31 の最後の文字には + (正符号) が組み込まれ、次の SV 31 に続くことを示します。+ (正符号) は、継続を示すために SV 31 の文字位置 255 である必要があります。それ以外の文字位置の場合、+ (正符号) はテキスト・メッセージの一部として解釈されます。

必要に応じてスロット/値のペアを継続するために、複数の SV 31 が作成されます。継続する各 SV 31 には、最後の文字として + (正符号) が組み込まれます。最初の非継続 SV 31 は、スロット/値のペアの終端を示します。

CONTINUE スロット

このイベント属性を使用することにより、複数の CDS を突き合わせて単一の疑似イベントを作成することができるようになります。CDS ファイルのこのマルチパス・プロセスの詳細については、 167 ページの『複数の CDS 突き合わせによる疑

似イベントの作成』で説明します。このイベント属性の値は **NEXT** または **GROUP*xxx*** です。ここで、*xxx* は 000 から 999 までの値です。

このイベント属性の値で、**$CDS_GROUP** キーワードの値が更新されます。 このキーワードのデフォルト値は **GROUP001** です。 CONTINUE イベント属性の値が **NEXT** であれば、$CDS_GROUP の値は、最後にある 3 桁の数値に 1 が加算されて更新されます。 つまり、 $CDS_GROUP の現行値が **GROUP001** で、値が **NEXT** である CONTINUE イベント属性が MAP セグメント内に検出されると、$CDS_GROUP キーワードの値は新たに **GROUP001** になります。

CONTINUE イベント属性の値が **GROUP*xxx*** のとき、この値で $CDS_GROUP 値が置換されるのは、イベント属性値の数値桁が $CDS_GROUP の現在値の数値桁よりも**大きい場合に限られます**。

SF21 スロット

このイベント属性は、元の EIF イベントを送信するために使用される SV 31 内にある、すべてのサブフィールド 21 のコード・ポイントをオーバーライドするために使用されます。このイベント属性の値は以下のものでなければなりません。

```
attributeName=codepoint
```

ここで、attributename は、入力属性リストにおける任意の総称属性の名前であり、codepoint は 2 桁の 16 進数ストリングで、指定された総称属性の SV31 に関連する SF 21 に置かれる値を定義します。

SV イベント属性同様、SF21 の接頭部はストリング SF21 だけにする必要があります。この接頭部以降の文字は無視されます。

複数の CDS 突き合わせによる疑似イベントの作成

イベント・アダプターによる CDS ファイルの処理方法とイベント受信側による CDS ファイルの処理方法の主な違いは、単一の EIF イベント (イベント受信側の場合は疑似イベント) を作成するための突き合わせ可能な CDS の個数です。

1 パス方式

イベント・アダプターでは、一致するステートメントが見つかるか、または一致するものがなくファイル終わりになるまで、1 つの CDS ファイル内にあるすべてのステートメントを一とおり調べます。次に、一致した単一 CDS の MAP セグメントを用いて、発信 EIF イベントに入れるイベント属性/値の組が作成されます。

この同じ 1 パス・プロセスを用いてアラートに変換される疑似イベントをいくつでも作成することができる 反面、非常に複雑な CDS ファイルを作成する羽目になります。以下に例を挙げてこれを説明します。

着信イベントのイベント属性/値の組に基づいて SV 05 と SV 92 をさまざまに組み合わせたアラートを作成するものとします。SV 05 の作成では、 resource1 および resource2 という 2 つのイベント属性が存在するか調べます。下記の 4 つの CDS で SV 05 がマップされます。

```
CLASS SV05_1
  SELECT
    1: ATTR(=,resource1);
    2: ATTR(=,resource2);
  MAP
```

```
    SV05 = PRINTF("#{05#{1000#{#<%s#>#}0084#{#<%s#>#}0040#}#}", $V1, $V2);
END

CLASS SV05_2
  SELECT
    1: ATTR(=,resource1);
  MAP
    SV05 = PRINTF("#{05#{1000#{#<%s#>#}0084#}#}", $V1);
END
CLASS SV05_3
  SELECT
    1: ATTR(=,resource2);
  MAP
    SV05 = PRINTF("#{05#{1000#{#<%s#>#}0040#}#}", $V1);
END

CLASS SV05_4
  SELECT
    1: ATTR(=,$CLASSNAME);
  MAP
    SV05 = "#{05#{1000#{#<NONE#>#}0084#}#}";
END
```

異なる 4 個のイベント属性を作成するためには、それぞれ別の SELECT セグメントを使用してこれらのイベント属性の有無を検査する必要があります。したがって、CDS ファイルには 4 つの異なる CDS があります。これらの SV 05 のうちの 1 つだけが疑似イベントです。最後の CDS では、デフォルト値として $CLASSNAME キーワードを使用しています。このキーワードは必ず存在するので、他の CDS のどれとも一致しなければ、最後の CDS が選択されます。

SV 92 サブベクトルは別のイベント属性 severity の値に依存します。severity イベント属性の値は 3 種類あり、それぞれ異なる SV 92 が作成される可能性があります。また、severity イベント属性にこれらの値のいずれも含まれていない場合、4 番目の SV 92 が作成されます。これらの CDS は、次のとおりです。

```
CLASS SV92_1
  SELECT
    1: ATTR(=,severity), VALUE(=,FATAL);
  MAP
    SV92 = "0B92010001FE0300000000"
END
CLASS SV92_2
  SELECT
    1: ATTR(=,severity), VALUE(=,WARNING);
  MAP
    SV92 = "0B92010011FE0300000000"
END

CLASS SV92_3
  SELECT
    1: ATTR(=,severity), VALUE(=,HARMLESS);
  MAP
    SV92 = "0B92010002FE0300000000"
END

CLASS SV92_4
  SELECT
    1: ATTR(=,$CLASSNAME);
  MAP
    SV92 = "0B92010012FE0300000000"
END
```

この場合も、これら 4 つの異なるイベント属性の 1 つだけを作成するために、4 つの異なる CDS が必要になります。

上記の SV 05 と SV 92 の任意の組み合わせを使用する 1 つの疑似イベントを、CDS ファイルを 1 回パススルーするだけで作成するには、16 の異なる CDS ステートメントが必要になります。つまり、唯一の SV05 の作成に必要な 4 ステートメントと唯一の SV 92 の作成に必要な 4 ステートメントを掛け算した結果です。16 の MAP セグメントには、それぞれ SV 05 と SV 92 が 1 回ずつ記述され、考えられるすべての組み合わせが表現されています。さまざまな SV 92 と組み合わせて両方のリソースを表す 4 つの CDS は以下のとおりです。

```
CLASS SVBOTH_1
  SELECT
    1: ATTR(=,resource1);
    2: ATTR(=,resource2);
    3: ATTR(=,severity), VALUE(=,FATAL);
  MAP
    SV05 = PRINTF("#{05#{1000#{#<%s#>#}0084#{#<%s#>#}0040#}#}", $V1, $V2);
    SV92 = "0B92010001FE0300000000"
END
CLASS SVBOTH_2
  SELECT
    1: ATTR(=,resource1);
    2: ATTR(=,resource2);
    3: ATTR(=,severity), VALUE(=,WARNING);
  MAP
    SV05 = PRINTF("#{05#{1000#{#<%s#>#}0084#{#<%s#>#}0040#}#}", $V1, $V2);
    SV92 = "0B92010011FE0300000000"
END

CLASS SVBOTH_3
  SELECT
    1: ATTR(=,resource1);
    2: ATTR(=,resource2);
    3: ATTR(=,severity), VALUE(=,HARMLESS);
  MAP
    SV05 = PRINTF("#{05#{1000#{#<%s#>#}0084#{#<%s#>#}0040#}#}", $V1, $V2);
    SV92 = "0B92010002FE0300000000"
END

CLASS SVBOTH_4
  SELECT
    1: ATTR(=,resource1);
    2: ATTR(=,resource2);
  MAP
    SV05 = PRINTF("#{05#{1000#{#<%s#>#}0084#{#<%s#>#}0040#}#}", $V1, $V2);
    SV92 = "0B92010012FE0300000000"
END
```

同じ出力 NMVT に入れる必要がある他のサブベクトルをさらに追加するとすれば、必要となる CDS の数と MAP セグメントによるイベント属性マッピングの重複はかなり増えます。

マルチパス方式

この負担を軽減するために、イベント受信側では CDS ファイルのパススルーを複数回行い、作成する 1 つの疑似イベントに関して一致する各セグメントから別個のマッピングを収集します。$CDS_GROUP キーワードと CONTINUE イベント属性は、マルチパス方式を制御するために使用されます。

それぞれのパスは、CDS ファイルの先頭から開始されます。一致した CDS が有効な CONTINUE イベント属性を含んでいると、CDS ファイルのパススルーが少なくともももう 1 つ作成されます。 一致した CDS に CONTINUE ステートメントがないか、どの CDS とも一致しなければ、そのパスが CDS ファイルの最後のパススルーになり、その時点までに収集されたすべてのイベント属性を基に疑似イベントが作成されます。

どの CDS SELECT セグメントにも、GROUP001 から GROUP999 までのストリングと同じになるように、$CDS_GROUP キーワードを検索するステートメントを 1 つ**含める必要があります**。デフォルトで、$CDS_GROUP キーワードの初期値は GROUP001 であるため、突き合わせされる最初の CDS ステートメントでは GROUP001 に等しいこのキーワードを探す必要があります。

一致する CDS があると、その CDS の MAP セグメントにある CONTINUE イベント属性定義により、別の CDS と突き合わせるために別のパスを作成するかどうかが決まります。CONTINUE イベント属性によって $CDS_GROUP キーワードの値は特定の値に変更されるか (CONTINUE = GROUP004) またはその次の数値に変更されます (CONTINUE = NEXT)。 特定の値を与える場合、その値は $CDS_GROUP キーワードの現行値よりも大きい値でなければなりません。

上記の例を使用して $CDS_GROUP キーワードと CONTINUE イベント属性の使用法を解説するために、次のようにキーワードとイベント属性を入力してください。

```
CLASS SV05_1
  SELECT
    1: ATTR(=,$CDS_GROUP), VALUE(=,GROUP001);
    2: ATTR(=,resource1);
    3: ATTR(=,resource2);
  MAP
    SV05 = PRINTF("#{05#{1000#{#<%s#>#}0084#{#<%s#>#}0040#}#}", $V2, $V3);
    CONTINUE = NEXT;
END

CLASS SV05_2
  SELECT
    1: ATTR(=,$CDS_GROUP), VALUE(=,GROUP001);
    2: ATTR(=,resource1);
  MAP
    SV05 = PRINTF("#{05#{1000#{#<%s#>#}0084#}#}", $V2);
    CONTINUE = NEXT;
END
CLASS SV05_3
  SELECT
    1: ATTR(=,$CDS_GROUP), VALUE(=,GROUP001);
    2: ATTR(=,resource2);
  MAP
    SV05 = PRINTF("#{05#{1000#{#<%s#>#}0040#}#}", $V2);
    CONTINUE = NEXT;
END

CLASS SV05_4
  SELECT
    1: ATTR(=,$CDS_GROUP), VALUE(=,GROUP001);
  MAP
    SV05_4 = "#{05#{1000#{#<NONE#>#}0084#}#}";
    CONTINUE = NEXT;
END

CLASS SV92_1
  SELECT
```

```
    1: ATTR(=,$CDS_GROUP), VALUE(=,GROUP002);
    2: ATTR(=,severity), VALUE(=,FATAL);
  MAP
    SV92 = "0B92010001FE0300000000"
END

CLASS SV92_2
  SELECT
    1: ATTR(=,$CDS_GROUP), VALUE(=,GROUP002);
    2: ATTR(=,severity), VALUE(=,WARNING);
  MAP
    SV92 = "0B92010011FE0300000000"
END

CLASS SV92_3
  SELECT
    1: ATTR(=,$CDS_GROUP), VALUE(=,GROUP002);
    2: ATTR(=,severity), VALUE(=,HARMLESS);
  MAP
    SV92 = "0B92010002FE0300000000"
END

CLASS SV92_4
  SELECT
    1: ATTR(=,$CDS_GROUP), VALUE(=,GROUP002);
  MAP
    SV92 = "0B92010012FE0300000000"
END
```

EIF イベントが到着して変換されると、最初に作成されるサブベクトルは SV 05 サブベクトルです。$CDS_GROUP キーワードの初期値は GROUP001 であるため、SV 05 を作成するすべての CDS の SELECT セグメントではこの値を探します。このグループの最初の 3 つの CDS のうちのいずれも選択されないときは、デフォルトで 4 番目のものが選択されます。これらの CDS では、 CONTINUE イベント属性に NEXT の値を定義しているので、$CDS_GROUP キーワードの値は GROUP002 に更新され、別の CDS と突き合わせるために CDS がまたパススルーされます。

$CDS_GROUP キーワードは別の値であるため、SV 05 CDS はすべて無視されます。このゲートがないと、同じ SV 05 CDS の突き合わせがいつまでも続きます。次は SV 92 CDS が突き合わせされます。これは $CDS_GROUP キーワードの値 GROUP002 で決まります。どの SV 92 CDS にも CONTINUE イベント属性はないので、これが CDS ファイルでの最後のパススルーになります。

上記の CDS を使用して、下記のイベント属性をもつイベントが到着したものとします。

```
resource1=FIRSTRES
resource2=SECNDRES
severity=WARNING
```

以下の 2 つのサブベクトルが作成されます。

```
1B0519100009C6C9D9E2E3D9C5E2008409E2C5C3D5C4D9C5E20040
0B92010011FE0300000000
```

NMVT の構築

疑似イベントが作成されると、イベント属性とキーワードのデータから NMVT が構築されます。

元のイベントを含む SV 31 の構築

$BUILD_SV31LIST キーワードによって、元の EIF イベント・データを含む SV 31 が構築されるかどうかが決まります。これらの SV 31 はまず NMVT に追加されます。このキーワードの値は、BUILD_SV31LIST イベント属性の内容で変更されます。

各 SV 31 には元のイベントのエレメントである、クラス名、イベント属性/値の組、または END 指定機能が含まれます。NPDA 画面上にフォーマット設定されたこの単純な CDS の例は下記のようになります (元のイベントに含まれていたクラス名は SAMPLE であるものとします)。

```
ORIGINAL T/EC EVENT:
   SAMPLE;
   resource1=FIRSTRES;
   resource2=SECNDRES;
   severity=WARNING;
   END
```

SF21 コード・ポイントのオーバーライド

どの SV 31 にも SF 21 サブフィールドがあります。デフォルトでは、このサブフィールドに関連付けられているコード・ポイントは X'00' です。2 つのコード・ポイントにより、SV 31 をアラート記述と推定原因に関連付けることができます (推定原因にはコード・ポイント X'21'、アラート記述にはコード・ポイント X'22')。デフォルトでは、***severity*** イベント属性に関連付けられている SV 31 には X'21' のコード・ポイントが割り当てられ、***msg*** イベント属性に関連付けられている SV 31 には X'22' のコード・ポイントが割り当てられます。

アラート記述または推定原因に関連付けられる SV 31 は、**SF21** イベント属性を用いて変更することができます。 このイベント属性には、入力属性リストにある属性の名前 (これは着信 EIF イベントにあるイベント属性値でなければなりません)、それに続いて等号 (=)、さらに、それに続いて 1 バイトの 16 進コード・ポイントが含まれます。例えば、着信イベントにある eventdetail というイベント属性をアラート記述に関連付けるには、以下の CDS をコーディングします。

```
CLASS SF21_1
  SELECT
    1: ATTR(=,$CDS_GROUP), VALUE(=,"GROUP001");
    2: ATTR(=,eventdetail);
  MAP
    SF21_1 = PRINTF("%s=21",$N2);
END
```

SF21_1 イベント属性値は、次のとおりです。

```
eventdetail=21
```

SV 31 リストが構築されるときには、eventdetail によって指名されたイベント属性/値の組にあるデータがアラート記述に関連付けられます。

この SF 21 によるオーバーライドが有効なのは、$BUILD_SV31LIST キーワードで SV 31 リストの構築が指示されている場合だけです。このリストが構築されない場合は、このイベント属性は無視されます。

アラートまたは解決

$NMVT_TYPE キーワードの値は、NMVT がアラート NMVT (タイプ 0000) であるか解決 NMVT (タイプ 0002) であるかを示します。このキーワードのデフォルト値はアラート NMVT です。一致した CDS 内で NMVT_TYPE イベント属性が設定されると、 $NMVT_TYPE キーワードの値がこのイベント属性に設定されます。

ユーザー・サブベクトルの追加

SV 31 が追加されて NMVT タイプが決まると、CDS MAP セグメントから作成されたユーザー・サブベクトルが NMVT に追加されます。すでに述べたように、CDS ステートメントの MAP セグメントでは任意のイベント属性に値を割り当てることができます。ただし、ユーザー・サブベクトルの構築に使用されるイベント属性にのみ接頭部 SV を付ける必要があります。

同一のイベント属性名が複数回使用されると、最後の値がそのイベント属性の値として使用されます。したがって、同一タイプの複数のサブベクトルが必要であれば、サブベクトル・データを固有に識別する名前をイベント属性につける必要があります。 複数の SV 10 のイベント属性名として **SV10** を使用しても、無効です。その理由は、先行するすべてのイベント属性はイベント属性リストで上書きされるからです。**SV10_1**、 **SV10_2** などの固有の名前を使用してください。

サブベクトル・イベント属性につける名前は、実際のサブベクトルに対応している必要はありません。**SV10_1** と名付けたイベント属性の値には、まったく別のサブベクトルのデータが入ることがあります。 サブベクトル・イベント属性の値で決まるのはサブベクトル・タイプであって、イベント属性名ではありません。

サブベクトル・イベント属性の値は、上記で説明したようにデコードされます。サブベクトルは、サブベクトルで定義しているイベント属性が MAP セグメントで検出された順に、NMVT に追加されます。

SV 92 のアラート ID の計算

NMVT の構築時にサブベクトルのアラート ID フィールドを計算する必要があるので、イベント受信側は SV 92 のこのフィールドの値を計算します。ただし、 CDS ファイルにコーディングする SV 92 のすべてのイベント属性にアラート ID のプレースホルダーを指定しておく必要があります。任意の 4 バイトを入れることができます。これらはイベント受信側によって上書きされます。プレースホルダーとして 4 バイトのゼロ (00000000) をコーディングすることをお勧めします。

イベント受信側はアラート ID を計算します。これについては、***SNA Formats*** に説明があります。

例

下記の例では、イベント自動化サービスで提供されるデフォルトのイベント受信側サービスの CDS ファイル (IHSAECDS) を使用します。

以下の EIF イベントが、イベント受信側によって受信されたと仮定します。

```
SNA_Performance_Degraded;source=NV390ALT;origin=B3088P2;
sub_origin=TX12/DEV;hostname=USIBMNT.NTVED;adapter_host=NMPIPL06;
date=OCT 29 16:32:52;severity=WARNING;msg=PERFORMANCE DEGRADED:
CONTROLLER;adapter_host_snanode=USIBMNT.NTVED;
event_type=NOTIFICATION;arch_type=GENERIC_ALERT;
```

```
product_id=3745;alert_id=00000009;
block_id='';action_code='';alert_cdpt=4000;
self_def_msg=[ALRTTXT2];event_correl=[N/A];
incident_correl=[N/A];adapter_correl=E7735930A;END
```

上記イベントは、アラート・アダプターによってイベントに変更されたアラートです。まず、イベント属性/値の組のすべてが入力属性リストの総称属性にコード化されます。$CLASSNAME キーワード属性には、値 SNA_Performance_Degraded が割り当てられます。

CDS ファイルにおける最初のグループは GROUP001 です。これらの CDS により NMVT タイプが決まります。着信 EIF イベントには status イベント属性が含まれていないので、NMVT_TYPE イベント属性および $NMVT_TYPE キーワードは、値 ALERT に設定されます。MAP セグメントで CONTINUE=NEXT が指定されているので、$CDS_GROUP キーワードは GROUP002 に設定されます。

CDS ファイルにおける次のグループでは SV 93 を定義します。元のイベント内に SV 93 の値を決定する情報はありません。このサブベクトルの値は下記のようになります。

```
0493FE03
```

MAP セグメントで CONTINUE=NEXT が指定されます。$CDS_GROUP キーワードは GROUP003 に設定されます。

CDS ファイルにおける次のグループでは SV 05 を定義します。例に挙げたイベントはクラス SV05_4 に一致します。これには hostname、origin、および source のイベント属性はありますが、 probe イベント属性はありません。PRINTF で変換すると、このサブベクトルの値は以下のようになります。

```
2A052810000EE4E2C9C2D4D5E34BD5E3E5C5C4008408C2F3F0F8F8D7F200F509D5E5F3F9F0C1D3E30040
```

MAP セグメントで CONTINUE=NEXT が指定されます。$CDS_GROUP キーワードは GROUP004 に設定されます。

CDS ファイルにおける次のグループでは SV 10 を定義します。元のイベント内に SV 10 の値を決定する情報はありません。このサブベクトルの値は下記のようになります。

```
1C10001911040506C7C5D40908F5F6F9F7C2F8F3080FE3C9E5D6D3C9
```

MAP セグメントで CONTINUE=NEXT が指定されます。$CDS_GROUP キーワードは GROUP005 に設定されます。

CDS ファイルにおける次のグループでは SV 92 を定義します。例に挙げたイベントはクラス SV92_4 に一致します。これには severity=WARNING があるので $NMVT_TYPE は ALERT に設定されます。 このサブベクトルの値は以下のようになります。

```
0B92010011FE0300000000
```

このサブベクトルのアラート ID 部分 (最後の 4 バイト) は、イベント受信側によって計算され充てんされます。MAP セグメントで CONTINUE=NEXT が指定されます。$CDS_GROUP キーワードは GROUP006 に設定されます。

CDS ファイルにおける次のグループでは SV 97 を定義します。例に挙げたイベントはクラス SV97_1 に一致します。$NMVT_TYPE は ALERT に設定されます。 このサブベクトルの値は以下のようになります。

```
0A970881200035003000
```

MAP セグメントで CONTINUE=NEXT が指定されます。$CDS_GROUP キーワードは GROUP007 に設定されます。

CDS ファイルにおける次のグループでは SF 21 を定義します。例に挙げたイベントはこのグループの唯一の CDS に一致します。このイベントには msg イベント属性があります。このサブフィールドをオーバーライドする値は以下のようになります。

```
msg=21
```

MAP セグメントで CONTINUE=NEXT が指定されます。$CDS_GROUP キーワードは GROUP008 に設定されます。

CDS ファイルにおける最後のグループでは別の SF 21 を定義します。例に挙げたイベントはこの最後の CDS に一致します。このイベントには severity イベント属性があります。このサブフィールドをオーバーライドする値は以下のようになります。

```
severity=22
```

$BUILD_SV31LIST キーワードは YES に設定されたままです。前のプロセスから構築される実際の NMVT は以下のようになります。

```
03D800002B31060202800000512C5D5E40321001B30E2D5C16DD7859986969994819583
856DC4858799818485845E2231060202800000512C5D5E40321001230A296A49983857E
D5E5F3F9F0C1D3E35E4A31060202800000512C5D5E40321003A3096998987899957EC2F3
F0F8F8D7F261E2D76BD5C1D761E3D76BC4C5C3D5C5E361E3C5D9D46BD9C1D3E5F461C4C5
E56BE3E7F1F261C4C5E55E263106020280000000512C5D5E40321001630A2A4826D969989
8789957EE3E7F1F261C4C5E55E2931060202800000512C5D5E403210019308896A2A395
8194857EE4E2C9C2D4D5E34BD5E3E5C5C45E283106020280000000512C5D5E40321001830
81848197A385996D8896A2A37ED5D4D7C9D7D3F0F65E2731060202800000512C5D5E403
210017308481A3857ED6C3E340F2F940F1F67AF3F27AF5F25E2331060202800000512C5
D5E40321221330A285A5859989A3A87EE6C1D9D5C9D5C75E3631060202800000512C5D5
E4032121263094A2877ED7C5D9C6D6D9D4C1D5C3C540C4C5C7D9C1C4C5C47AC3D6D5E3D9
D6D3D3C5D95E3531060202800000512C5D5E4032100253081848197A385996D8896A2A3
6DA29581959684857EE4E2C9C2D4D5E34BD5E3E5C5C45E2A31060202800000512C5D5E4
0321001A3085A58595A36DA3A897857ED5D6E3C9C6C9C3C1E3C9D6D55E2A310602028000
000512C5D5E40321001A30819983886DA3A897857EC7C5D5C5D9C9C36DC1D3C5D9E35E22
310602028000000512C5D5E4032100123097999684A483A36D89847EF3F7F4F55E243106
02028000000512C5D5E403210014308193859A36D89847EF0F0F0F0F0F0F95E1E3106
02028000000512C5D5E40321000E3082939683926D89847E7D7D5E213106020280000005
12C5D5E403210011308183A38996956D839684857E7D7D5E223106020280000000512C5D5
E4032100123081938599A36D838497A37EF4F0F0F05E2A31060202800000512C5D5E403
21001A30A28593866D8485866D94A2877EADC1D3D9E3E3E7E3F2BD5E2531060202800000
0512C5D5E4032100153085A58595A36D8396999985937EADD561C1BD5E28310602028000
000512C5D5E403210018308995838984859 5A36D8396999985937EADD561C1BD5E2B3106
02028000000512C5D5E40321001B3081848197A385996D8396999985937EC5F7F7F3F5F9
F3F0C15E153106020280000000512C5D5E40321000530C5D5C40493FE032A052810000EE4
E2C9C2D4D5E34BD5E3E5C5C4008408C2F3F0F8F8D7F200F509D5E5F3F9F0C1D3E300401C
10001911040506C7C5D40908F5F6F9F7C2F8F3080FE3C9E5D6D3C90B92010011FE030000
00000A970881200035003000
```

ASCII テキスト・データの変換

SNMP エージェントは、本来 ASCII テキスト・データであるデータ (変数結合であるか、トラップの他の部分にあるかにかかわらず) を送信しますが、エンコード・トラップのデータ・タイプはオクテット・ストリングを示します。データ・タイプがオクテット・ストリングであるため、trap-to-alert データ・エンコード・プロセスでは、データの各バイトをエンコード文字ではなく、生の 16 進データとして扱います。その結果、trap-to-alert 変換タスクによって行われる解析は、CDS ファイル内で SELECT 基準によりこのデータを 16 進データ・バイトの文字表現に変えるだけです。例えば、文字ストリング **ABC** が、タイプがオクテット・ストリングの変数結合値に表示されるとします。データはオクテット・ストリングであるので、データは、文字ストリング **414243** に変換され、変数結合名と関連した総称キーワードに割り当てられます。

発信アラートで総称キーワードの元の ASCII ストリング値を使用したい場合、ASCII ストリング **414243** を、文字ストリング **ABC** に戻し、EBCDIC に変更する必要があります。ASCII ストリング **414243** を EBCDIC 文字ストリング **ABC** に戻すために、$[および $] のエスケープ・シーケンスが提供されています。

値のエンコードでは、サブベクトル・イベント属性の値 (PRINTF であるかどうかにかかわらず) の二重引用符内で、このエスケープ・セットを使用して、16 進データの文字表現であると見なされるデータが区切られ、ASCII 文字データになります。このように区切られたデータは、 EBCDIC 文字データに変換され、サブベクトル・イベント属性の値の中に入れられます。 例えば、クラス定義ステートメント内で次のイベント属性を割り当てたものとします。

```
SV05 = "0B0509100004#[414243#]0040"
```

このイベント属性値を実際の 16 進アラート・サブベクトルにエンコードすると、次のようになります。

```
0B0509100004C1C2C30040
```

エスケープ・シーケンスによって区切られる範囲内にあるデータが、ASCII 文字である 16 進データの文字表現でないことが分かる場合、EBCDIC への変換は失敗し、トラップの変換 (したがって、アラート/解決の構築) が終了し、トラップが廃棄されます。他のエスケープ・シーケンスが "#[" の後、"#]" の前に生じる場合、これらのエスケープ・シーケンスは、単にサブベクトルに書き込まれる文字として扱われることに注意してください。これらは 16 進数の文字表現でないので、後で 16 進数に変換してから、EBCDIC に変換しようとしても失敗します。また、"#[" または "#]" が "#<" エスケープ・シーケンスの後で生じ、それによってサブベクトル内の 16 進数の文字表現から 16 進データへの変換が「オフ」になる場合と、"#>" の前に生じ、その変換モードが「復元」される場合、"#[" と "#]" は、エスケープ・シーケンスではなく、単に変換されていない文字データとして扱われます。

SNMP 非ストリング・データ・タイプの変換

CDS 選択で使用される一部の属性は、値が抜き出される元のトラップ内の場所に基づいて名前が割り当てられますが、他の名前はトラップから直接適応されます (例えば、変数結合内のオブジェクト ID である変数名)。エンコードされる値はすべ

て、トラップ内のデータの表示可能な書式であるストリング・データであり、これらのストリングの形式は、トラップ内のこれらのデータに割り当てられるデータ・タイプに応じて決まります。

例えば、トラップ内の値のデータ・タイプが、IP アドレスのデータ・タイプであることが分かっているものとします。このデータ・タイプは、trap-to-alert 変換タスクにより、IP アドレスであるストリングに変換されます。次のデータ・タイプを、SNMP トラップ内のデータ、およびそのデータが変換される対応したストリングに割り当てることができます。

integer 符号付き 10 進数ストリング。整数 30 が EBCDIC ストリング "30" になります。

ヌル EBCDIC での単一引用符の組。これは EBCDIC ストリング "''" になります。

オクテット・ストリング
16 進データ・ストリング。16 進ストリング 313233 は、EBCDIC ストリング "313233" になります。

オブジェクト ID
小数点付き 10 進表記形式の ASN.1 データ。オブジェクト 2C010306 は EBCDIC ストリング "1.4.1.3.6" になります。

印刷可能ストリング
EBCDIC ストリング

可視ストリング
EBCDIC ストリング

汎用ストリング
EBCDIC ストリング

IP アドレス
IP アドレス。例えば、小数点付き 10 進表記の形式を使用している場合、アドレス 09080706 は EBCDIC ストリング "9.8.7.6" になります。

カウンター
符号なし 10 進数ストリング。数値 05 は EBCDIC ストリング "5" になります。

ゲージ 符号なし 10 進数ストリング。数値 50 は EBCDIC ストリング "50" になります。

ティック
符号なし 10 進数ストリング。数値 132 は EBCDIC ストリング "132" になります。

値が、リストされているデータ・タイプのいずれでもない場合、データ・タイプがオクテット・ストリングの値として処理されます。また、結合内の値のデータ・タイプが、SEQUENCE OF のような複合構造 (発生しないはずのもの) である場合も、値はヌル・データ・タイプであるものとして扱われます。

下記の例では、イベント自動化サービスとともに提供されるデフォルト trap-to-alert サービスの CDS ファイル (IHSATCDS) を使用します。次のトラップ・データが trap-to-alert 変換タスクによって受信されるものと想定します (読みやすくするために語が分離されています)。

```
303B0201 00040670 75626C69 63A42E06
0C2B0601 14011203 01020101 03400449
B5203F02 01050201 00430100 300F300D
06082B06 01120108 07000201 30
```

また、トラップを発信するエージェントに関連した IP アドレスとポートが、それぞれ 9.50.20.8 と 161 であることも前提としています。

まず、トラップ・データは、対応するキーワードと、入力属性リストの総称属性にコード化されます。エンコードされたストリング属性は次のとおりです。

```
  $ORIGIN_ADDR        9.50.20.8
    $ORIGIN_PORT         161
    $SNMP_VERSION     0
    community                      public
    enterpriseOID
1.3.6.1.20.1.18.3.1.3.1.1.3
    agent_address                73.181.32.63
    generic_trap                   5
    specific_trap                  0
    timestamp                      0
    1.3.6.1.18.1.8.7.0         30
```

CDS ファイルにおける最初のグループは GROUP001 です。この CDS により NMVT タイプと BUILD_SV31LIST 設定が決まります。このトラップは、Multi-System Manager トラップではないので、CDS ファイル IHSATALL によって実行される汎用フォーマットが使用されます。NMVT_TYPE イベント属性 (したがって $NMVT_TYPE キーワード) は値 ALERT に設定されます。BUILD_SV31LIST イベント属性 (したがって $BUILD_SV31LIST キーワード) は値 YES に設定されます。MAP セグメントで CONTINUE=NEXT が指定されているので、$CDS_GROUP キーワードは GROUP002 に設定されます。

CDS ファイルにおける次のグループでは SV 92 を定義します。このサブベクトルの値は以下のようになります。

```
0B92080012FE0000000000
```

このサブベクトルのアラート ID 部分 (最後の 4 バイト) は、イベント受信側によって計算され充てんされます。MAP セグメントで CONTINUE=NEXT が指定されています。$CDS_GROUP キーワードは GROUP003 に設定されます。

CDS ファイルにおける次のグループでは SV 05 を定義します。PRINTF で変換すると、このサブベクトルの値は以下のようになります。

```
22050E100009F7F34BF1F8F14BF300811211000DF7F34BF1F8F14BF3F24BF6F30081
```

MAP セグメントで CONTINUE=NEXT が指定されているので、$CDS_GROUP キーワードは GROUP004 に設定されます。

CDS ファイルにおける次のグループでは SV 10 を定義します。このサブベクトルの値は以下のようになります。

```
5A10002811030300000220EE261F3F9F040D78199819393859340C595A3859997
9989A28540E28599A585992F11040804F0F1F0F3F0F01B06E389A596938940D5
85A3E58985A64086969940D6E261F3F9F00908F5F6F9F7C2F8F2
```

MAP セグメントで CONTINUE=NEXT が指定されているので、$CDS_GROUP キーワードは GROUP005 に設定されます。

CDS ファイルにおける次のグループでは別の SV 10 を定義し、この SV 10 には、トラップを報告するリソースについての情報が入っています。このサブベクトルの値は以下のようになります。

```
2C10000F1109030000090EA495929596A6951A110C0E02F0F0F0F0F0F0F0F0F0
F0F0F00906A495929596A695
```

MAP セグメントで CONTINUE=NEXT が指定されているので、$CDS_GROUP キーワードは GROUP006 に設定されます。

CDS ファイルにおける次のグループでは SV 93 と SV 97 を定義します。これらのサブベクトルの値は、以下のようになります。

```
0493FE000
A970401210004810000
```

MAP セグメントで CONTINUE=NEXT が指定されているので、$CDS_GROUP キーワードは GROUP007 に設定されます。

CDS ファイルにおける最後のグループでは SV 98 を定義します。このサブベクトル内の情報として、enterpriseOID、特定のトラップ、および総称トラップの値が追加されます。このサブベクトルの値は以下のようになります。

```
severity=22
```

$BUILD_SV31LIST キーワードは YES に設定されたままですが、前のプロセスから構築される実際の NMVT は以下のようになります。

```
027B0000293106020280000000512C5D5E40321001930D6D9C9C7C9D56DC1C4C4D97EF94B
F6F74BF5F04BF1F85E2331060202800000000512C5D5E40321001330D6D9C9C7C9D56DD7D6
D9E37EF1F0F3F45E2131060202800000000512C5D5E40321001130E2D5D4D76DE5C5D9E2C9
D6D57EF05E2931060202800000000512C5D5E4032100193083969494A49589A3A87EF7F0F7
F5F6F2F6C3F6F9F6F35E3C31060202800000000512C5D5E40321002C308595A38599979989
A285D6C9C47EF14BF34BF64BF14BF2F04BF14BF1F84BF34BF14BF24BF14BF14BF35E2D31
0602028000000512C5D5E40321001D3081878595A36D8184849985A2A27EF7F34BF1F8F1
4BF3F24BF6F35E2131060202800000000512C5D5E4032100113087859585998 36DA39981
977EF55E2231060202800000000512C5D5E40321001230A2978583898689836DA39981977E
F05E1E31060202800000000512C5D5E40321000E30A3899485A2A38194977EF05E28310602
028000000512C5D5E40321001830F14BF34BF64BF14BF1F84BF14BF84BF74BF07EF4F85E
0B92080012FE00331AA4A122050E100009F7F34BF1F8F14BF300811211000DF7F34BF1F8
F14BF3F24BF6F300815A1000281103030000220EE261F3F9F040D78199819393859340C5
95A38599979989A28540E28599A585992F11040804F0F1F0F3F0F01B06E389A596938940
D585A3E58985A64086969940D6E261F3F9F00908F5F6F9F7C2F8F22C10000F1109030000
090EA495929596A6951A110C0E02F0F0F0F0F0F0F0F0F0F0F00906A495929596A69504
93FE000A9704012100048100002E98208229F811F14BF34BF64BF14BF2F04BF14BF1F84B
F34BF14BF24BF14BF14BF3068229FA11F5068229FB11F0
```

Trap-to-Alert の CDS 後処理

trap-to-alert サービスの CDS 後処理は、イベント受信側の CDS 後処理によって使用されるものとほぼ同じです。相違点は次のとおりです。

- 着信データが EIF イベントではないので、trap-to-alert サービスにより作成される $CLASSNAME キーワードがない。
- 追加のエスケープ・シーケンス・セット $[および $] が、ASCII オクテット・ストリングである変数結合データの変換に役立つように使用可能である。
- EIF イベント・データの場合とは異なり、SNMP トラップ・データのデータ・タイプが文字ストリングではない場合がある。

拡張カスタマイズ - Trap-to-Alert 転送デーモン

イベント自動化サービスの trap-to-alert 変換タスクにおけるトラップの受信は、ユーザーが構成ファイル (サンプル・メンバー名 IHSATCFG) で定義するポートに結合されるデータグラム・ソケットを通じて行われます。標準的なトラップ・マネージャー・データ・ポート番号 162 が、デフォルトのポートです。

ポート 162 は SNMP マネージャー用の「既知の」ポートであり、トラップ・データを使用する複数の SNMP マネージャーがあるので、こうしたポート割り当てにより矛盾が生じる可能性があります。矛盾を解決するために、イベント自動化サービスに付属のサンプル・データグラム転送デーモン IHSAUFWD および関連するサンプル構成ファイル IHSAUCFG もあります。このデーモンは、データグラム・ソケット上でデータを受信し、そのデータを、構成ファイル内で指定された宛先に転送します。

大部分の SNMP エージェントは、ポート 162 でトラップ・マネージャーにトラップを転送するように設定されます。IHSAUFWD はこのポートを使用して、関係するすべてのマネージャー用のトラップ・データを受信してから、このデータをマネージャーに転送します。これらのマネージャーは、ローカル・システム上にあっても、ネットワーク上の任意の IP アドレスにあってもかまいません。

IHSAUFWD デーモンはサンプル構成ファイル (IHSAUCFG) を使用して、データを受信する SNMP マネージャーを指定します。この構成ファイルの内容の説明は、次のとおりです。

コメント

コメントは、ポンド記号 (番号記号) (#)、または感嘆符 (!) で始まる行で形成されます。

ホスト IP アドレスとポート

データグラム転送デーモンの宛先をコード化するには、ファイル内の行に次を書き込みます。

- IP アドレス
- 空白 (1 つ以上のブランク)
- 10 進数のポート番号

このようにコード化される行の例は、次のとおりです。

```
137.45.110.2       6001
```

転送デーモンの使用法とカスタマイズ方法の詳細については、IHSAUFWD サンプル内のコメントを参照してください。

Trap-to-Alert 変換の詳細例

SNMP トラップが問題のある管理対象エンティティーに対して出され、その問題が発生したときに NetView プログラムに何らかのアクションを行わせたい場合を想定します。それを行うには、イベント自動化サービスでトラップを受信し、そのトラップをアラート NMVT に変換し、さらに、 NetView 自動化を使用してアラート NMVT を処理してアクション (コマンドの実行) を行います。

一般的にユーザーは、トラップを解析するために、また、アラート NMVT の処理が最も効果的に行われるよう、最も有用な情報をアラート NMVT に転送するために、SNMP トラップに含まれる情報についてある程度知っている必要があります。SNMP トラップを出すエンティティーに関連した資料には、この種の情報が含まれています。また、このような情報は SNMP トラップが実行されたときにアクティブなトレース (イベント自動化サービスの IP データ・トレースまたは z/OS Communications Server のパケット・トレースなど) からも得ることができます。

SNMP トラップにどのような情報が必要かがわかっていれば、トラップから必要な情報を抽出するのに必要なクラス定義ステートメントを作成し、アラート NMVT を構成します。ユーザーがマルチシステム・マネージャーの IP 管理機能も使用している場合は、もちろん、そのマルチシステム・マネージャーの IP 管理機能が引き続き動作できるようにユーザーの新規定義が組み込まれていることを確認する必要があります。この例のクラス定義ステートメント (CDS) は、サンプル・メンバー IHSATUSR に指定でき、NetView プログラムが提供するサンプル定義を IHSATCDS、および NetView に組み込まれた他のメンバー内で処理できるように設計されています。

この例は、無停電電源装置に問題がある場合に出される SNMP トラップから始まっています。データは 16 進数で表示され、トラップ内容をより明確にするために分離し、注釈を付けてあります。

```
*
* Outermost constructor for the trap (tag and length)
*
30820127
* SNMP version (00 = SNMPv1)
020100
* Community name (public)
04067075626C6963
* Trap PDU
A4820118
*   Enterprise object ID (1.3.6.1.4.1.12270)
06072B06010401DF6E
*   Agent address (10.71.225.20)
40040A47E114
*   Generic trap code (6 = enterprise specific)
020106
*   Specific trap code (32 in decimal)
020120
*   Timeticks
430402A2D49D
*   Variable bindings "container"
308200F9
*     Variable binding 1
3015
*       Variable 1 (1.3.6.1.4.1.12270.200.2.1.1.1)
060D2B06010401DF6E814802010101
*       Value 1 (octet string "1493")
040431343933
```

```
*     Variable binding 2
3019
*       Variable 2 (1.3.6.1.4.1.12270.200.2.1.1.2)
060D2B06010401DF6E814802010102
*       Value 2 (octet string "/L20/O50")
04082F4C32302F4F3530
*     Variable binding 3
3024
*       Variable 3 (1.3.6.1.4.1.12270.200.2.1.1.3)
060D2B06010401DF6E814802010103
*       Value 3 (octet string "2005-01-10T16:13:00")
0413323030352D30312D31305431363A31333A3030
*     Variable binding 4
3014
*       Variable 4 (1.3.6.1.4.1.12270.200.2.1.1.4)
060D2B06010401DF6E814802010104
*       Value 4 (octet string "I14")
0403493134
*     Variable binding 5
3025
*       Variable 5 (1.3.6.1.4.1.12270.200.2.1.1.5)
060D2B06010401DF6E814802010105
*       Value 5 (octet string "DIGIN ON    OCCURRED")
0414444947494E204F4E202020204F43435552524544
*     Variable binding 6
3015
*       Variable 6 (1.3.6.1.4.1.12270.200.2.1.1.6)
060D2B06010401DF6E814802010106
*       Value 6 (octet string "DI=1")
040444493D31
*     Variable binding 7
3025
*       Variable 7 (1.3.6.1.4.1.12270.200.2.1.1.7)
060D2B06010401DF6E814802010107
*       Value 7 (octet string "RC2 Gas Status Man. ")
0414524332204761732053746174757320 4D616E2E20
*     Variable binding 8
3011
*       Variable 8 (1.3.6.1.4.1.12270.200.2.1.1.8)
060D2B06010401DF6E814802010108
*       Value 8 (NULL)
0500
*     Variable binding 9
3011
*       Variable 9 (1.3.6.1.4.1.12270.200.2.1.1.9)
060D2B06010401DF6E814802010109
*       Value 9 (NULL)
0500
```

このタイプの SNMP トラップには常にこれらの変数結合が含まれ、さらにその値 (少なくとも興味のある変数の) が常に同じ種類のデータとなることをご存じである場合、これらのクラス定義ステートメント (CDS) を使用し、SNMP トラップをアラート NMVT に変換する方法を提供できます。これらのサンプル・ステートメントには、注釈が追加されており、NMVT へ転送されるトラップ・データについて説明を加えています。

注: 下記の例では、以下の点にご注意ください。

- 印刷上の制約により、コマンド行によってはページに収まるようにするために「分割」する必要がありました。
- コード・ページ 1047 X'AD' を使用して左大括弧 ([ ) をコーディングし、コード・ページ 1047 X'BD' を使用して右大括弧 ( ]) をコーディングしています。

```
#*********************************************************************
#
# Definitions for catching an SNMP trap indicating a UPS problem
# and turning it into an alert NMVT.
#
#   First pass, build subvectors X'92' (generic alert), X'10'
#   product set ID (one each for alert sender and reported resource),
#   X'93' (probable cause), and X'96' (failure cause).
#
#   The first pass looks for GROUP001 and a specific trap value
#   of 32 (the specific trap value in the trap was converted to
#   a string representing the value in decimal).
#*********************************************************************
CLASS IHSATUSR_UPS1

  SELECT
    1: ATTR(=,$CDS_GROUP), VALUE(=,"GROUP001");
    2: ATTR(=,specific_trap), VALUE(=,"32");
  MAP
#
#               |-- First pass sets desire for alert NMVT
    NMVT_TYPE = ALERT;
#
#                   |-- For this, we don't want SV x'31' set
#                   |   We'll build our own SV x'31' later
    BUILD_SV31LIST = NO;
#
#                    |-- Alert description code-point
#                    |   I chose X'1501' LOSS OF EQUIPMENT COOLING
#                    |   to illustrate.
    SV92 = "#{9208000115010000000#}";
#
#     Hardware and software information for alert builder
#     (basically hard-coded and uses our software product name,
#      because E/AS is building the alert NMVT)
#
#  Note that the line beginning SV10_1 and the line beginning SV10_2 should be
#  coded on continuous lines up to and including the semicolon character
     SV10_1 = "#{1000#{1103#{0000#}#{0E#<S/390 Parallel Enterprise Server#>#}#}#
{1104#{02#<5697-ENV0000#>#}#{04#<050200#>#}#{06#<Tivoli NetView for z/OS#}#}#}";
    SV10_2 = "#{1000#{1109#{0000#}#{0E#<UPS system#}#}#{110C#{02#<000000000000#>#}#
{06#<unknown#}#}#}";
#
#               |-- Probable cause code-point
#               |   x'0301' COOLING FAN chosen to illustrate.
#               |
    SV93 = "#{930301#}";
#
#                  |-- Failure cause code-point
#                  |   X'0301' COOLING FAN chosen to illustrate
#                  |
#                  |         |-- Recommended action code-point
#                  |         |   X'0300' CHECK FOR DAMAGE and
#                  |         |   X'1800' REPLACE DEFECTIVE EQUIPMENT
#                  |         |   to illustrate
    SV96 = "#{96#{010301#}#{8103001800#}#}";
#
#              |-- Keep going to next pass (GROUP002, for example)
    CONTINUE = NEXT;
END
#
#   Second pass for our UPS trap - defer to third pass, where we will
#   use the generic subvector X'05' (hierarchy/resource list)
#   construction from member IHSATALL.
#
CLASS IHSATUSR_UPS2
```

```
  SELECT
    1: ATTR(=,$CDS_GROUP), VALUE(=,"GROUP002");
    2: ATTR(=,specific_trap), VALUE(=,"32");
  MAP
#              |-- Tells trap to alert to continue to third pass
    CONTINUE = NEXT;
END
#
# Defer subvector X'05' definition to GROUP003 generic CLASS
# definition in IHSATALL
#
#
#   Fourth pass for UPS trap, construct subvector X'98' (detailed
#   data) and subvectors X'31'.  ALL selection criteria must be met
#   in order for inclusion of the information defined here in the
#   alert NMVT.
#
#   1) fourth pass, (CDS_GROUP keyword has the value GROUP004)
#   2) generic trap
#
#      Because no VALUE was supplied, we just look for
#      presence of the item, which, for an SNMPv1
#      trap, should always be there.
#
#      A primary reason to look for the presence of
#      of something that should always be there is that
#      this provides the method by which we can retrieve
#      the value, perhaps manipulate it, then put it in
#      the alert NMVT.
#
#   3) specific trap code with value 32 decimal,
#
#   4) MIB variable with name "1.3.6.1.4.1.12270.200.2.1.1.1"
#
#       Because gave no VALUE, we merely expect it to
#       have been present in the trap, again so we
#       retrieve its value and use it.
#
#   5) presence of origin address keyword,
#   6) presence of origin port keyword,
#   7) presence of community information,
#   8) presence of enterprise object ID,
#   9) presence of agent address,
#  10) presence of a timestamp,
#  11) presence of MIB variable "1.3.6.1.4.1.12270.200.2.1.1.2",
#  12) presence of MIB variable "1.3.6.1.4.1.12270.200.2.1.1.3",
#  13) presence of MIB variable "1.3.6.1.4.1.12270.200.2.1.1.5"
#
CLASS IHSATUSR_UPS3

  SELECT
    1: ATTR(=,$CDS_GROUP), VALUE(=,"GROUP004");
    2: ATTR(=,generic_trap);
    3: ATTR(=,specific_trap), VALUE(=,"32");
    4: ATTR(=,"1.3.6.1.4.1.12270.200.2.1.1.1");
    5: ATTR(=,$ORIGIN_ADDR);
    6: ATTR(=,$ORIGIN_PORT);
    7: ATTR(=,community);
    8: ATTR(=,enterpriseOID);
    9: ATTR(=,agent_address);
   10: ATTR(=,timestamp);
   11: ATTR(=,"1.3.6.1.4.1.12270.200.2.1.1.2");
   12: ATTR(=,"1.3.6.1.4.1.12270.200.2.1.1.3");
   13: ATTR(=,"1.3.6.1.4.1.12270.200.2.1.1.5");
  MAP
#                                 |-- Special detail data = hard-code
#                                 |           enterprise information
```

```
#                                    |                        |-- Special detail data
#                                    |                        |     = generic trap code
#                                    |                        |
#                                    |                        |                  |-- Special
#                                    |                        |                  |  detail data
#                                    |                        |                  |  = specific
#                                    |                        |                  |   trap code
#                                    |                        |                  |
#                                    |                        |                  |
#                                    |                        |                  |
#                                    |                        |                  |
#                                    |                        |                  |
#                                    |                        |                  |
#    SV98 = PRINTF("#{98#{8229F811#<Ent_Name#>#}#{8229FA11#<%s#>#}#{8229FB11#<%s#>#}#}",$V2,$V3);
#
# Now we also add some X'31' subvectors to convey additional
# information that came in the SNMP trap.
#
# Notice something in the definitions for SV31_2 and SV31_6.
# Both the community and the information from the variable binding
# expected in the SNMP trap are, for this example, presumed to have
# ASCII text data even though the data type encoded in the trap
# indicates merely OCTET STRING.  When E/AS encounters the OCTET
# STRING data type, E/AS converts it to a string representing the
# value in hexadecimal.  Because we have decided it's really text
# data and we want to be able to read in the display of the alert
# in hardware monitor, we use the escape character sequences (pound-sign followed
# by left bracket and pound-sign followed by right bracket) to delimit
# the string representing hexadecimal data and tell E/AS to convert it
# to EBCDIC text for inclusion in the subvector we're building.
#
# Subvector x'31' showing origin IP address:port number
#
# Note that this should be coded on one continuous line
    SV31_1 = PRINTF("#{31#{0202800000#}#{12#<ENU>#}#{2100#}#{30#<Origin
Address=%s:%s#>#}#}",$V5,$V6);
#
# Subvector x'31' showing community name in EBCDIC
#
# Note that the left square bracket must be x'AD' and the
#  right square bracket must be x'BD'
#
#                                                                        |--Left square
#                                                                        |  bracket must
#                                                                        |  be x'AD'
#                                                                        |  codepage 1047
#                                                                        |
#                                                                        |    |Right square
#                                                                        |    |bracket must
#                                                                        |    |be x'BD'
#                                                                        |    |codepage 1047
#                                                                        |    |
    SV31_2 = PRINTF("#{31#{0202800000#}#{12#<ENU#>#}#{2100#}#{30#<Community=#>#[%s#]#}#}",$V7);
#
# Subvector x'31' showing enterprise object ID
#
# Note that this should be coded on one continuous line
    SV31_3 = PRINTF("#{31#{0202800000#}#{12#<ENU#>#}#{2100#}#{30#<Enterprise
Object ID=%s#>#}#}",$V8);
#
# Subvector x'31' showing agent IP address
#
    SV31_4 = PRINTF("#{31#{0202800000#}#{12#<ENU#>#}#{2100#}#{30#<Agent Address=%s#>#}#}",$V9);
#
# Subvector x'31' showing the timestamp
#
    SV31_5 = PRINTF("#{31#{0202800000#}#{12#<ENU#>#}#{2100#}#{30#<Timestamp=%s#>#}#}",$V10);
```

```
#
# Subvectors x'31' showing what we presume to be text from the
# variable bindings in the trap.  We hard-code the MIB variable names
# in the text.  For the MIB variable values we want to also include
# in the text, we are assuming that the values are, in fact, ASCII
# text that we want to see in EBCDIC when we display the alert NMVT
# in hardware monitor.
#
#
# Note that the left square bracket must be x'AD' and the right square bracket
# must be x'BD'
# Note also that this should be coded on one continuous line
    SV31_6 = PRINTF("#{31#{0202800000#}#{12#<ENU#>#}#{2100#}#
{30#<1.3.6.1.4.1.12270.200.2.1.1.1 =
 #>#[%s#]#}#}",$V4);
#
# Builds SV31 with 1.3.6.1.4.1.12270.200.2.1.1.2 = its value in EBCDIC
#
# Note that this should be coded on one continuous line
    SV31_7 = PRINTF("#{31#{0202800000#}#{12#<ENU#>#}#{2100#}#
{30#<1.3.6.1.4.1.12270.200.2.1.1.2 = #>#[%s#]#}#}",$V11);
#
# Builds SV31 with 1.3.6.1.4.1.12270.200.2.1.1.3 = its value in EBCDIC
#
# Note that this should be coded on one continuous line
    SV31_8 = PRINTF("#{31#{0202800000#}#{12#<ENU#>#}#{2100#}#
{30#<1.3.6.1.4.1.12270.200.2.1.1.3 = #>#[%s#]#}#}",$V12);
#
# Builds SV31 with Message = and the value of the
# 1.3.6.1.4.1.12270.200.2.1.1.5 MIB variable in EBCDIC.
#
# Note that this should be coded on one continuous line
    SV31_9 = PRINTF("#{31#{0202800000#}#{12#<ENU#>#}#{2100#}#
{30#<Message=#>#[%s#]#}#}",$V13);
END
```

クラス定義ステートメント (CDS) の処理について詳しくは、169 ページの『マルチパス方式』および 179 ページの『Trap-to-Alert の CDS 後処理』を参照してください。

ハードウェア・モニターの記録フィルターによってアラート NMVT が保存されるとすれば、trap-to-alert 変換処理で生成されたアラート NMVT はハードウェア・モニターのイベント詳細では以下のようになります。

```
N E T V I E W          SESSION DOMAIN: CNM01    NETOP2    08/19/10 13:27:17
 NPDA-43S                       * EVENT DETAIL *                 PAGE  1 OF  4

  NTV90       10.71.22
             +--------+
  DOMAIN     |  SP    |
             +--------+

 SEL# TYPE AND NAME OF OTHER RESOURCES ASSOCIATED WITH THIS EVENT:
 ( 1) SP      10.71.225.20

 DATE/TIME: RECORDED - 08/19 13:16    CREATED - 08/19/10 13:16:16

 EVENT TYPE: PERMANENT

 DESCRIPTION: LOSS OF EQUIPMENT COOLING

 PROBABLE CAUSES:
    COOLING FAN

N E T V I E W          SESSION DOMAIN: CNM01    NETOP2    08/19/10 13:27:52
```

```
 NPDA-43S                      * EVENT DETAIL *                  PAGE  2 OF  4

  NTV90       10.71.22
             +---------+
  DOMAIN     |  SP     |
             +---------+

 QUALIFIERS:
    1)  ENTERPRISE Ent_Name
    2)  SNMP GENERIC-TRAP NUMBER 6
    3)  SNMP SPECIFIC-TRAP NUMBER 32

    Origin Address = 10.71.225.21:5644

    Community = public

N E T V I E W          SESSION DOMAIN: CNM01    NETOP2    08/19/10 13:28:22
 NPDA-43S                      * EVENT DETAIL *                  PAGE  3 OF  4

  NTV90       10.71.22
             +---------+
  DOMAIN     |  SP     |
             +---------+

    Enterprise Object ID = 1.3.6.1.4.1.12270

    Agent Address = 10.71.225.20

    Timestamp = 44225693

    1.3.6.1.4.1.12270.200.2.1.1.1 = 1493

    1.3.6.1.4.1.12270.200.2.1.1.2 = /L22/O50

N E T V I E W          SESSION DOMAIN: CNM01    NETOP2    08/19/10 13:28:40
 NPDA-43S                      * EVENT DETAIL *                  PAGE  4 OF  4

  NTV90       10.71.22
             +---------+
  DOMAIN     |  SP     |
             +---------+

    1.3.6.1.4.1.12270.200.2.1.1.3 = 2005-01-10T16:13:00

    Message = DIGIN ON    OCCURRED

 FLAGS:
    SNMP TRAP



 UNIQUE ALERT IDENTIFIER: PRODUCT ID - 5697-ENV0  ALERT ID - 673BB726
```

もちろん、NetView 自動化を使用することにより、アラート NMVT をスキャンし、その中にある情報に基づいてアクションをとる 1 つ以上のコマンドを実行できる場合があります。

Alert-to-Trap の CDS 後処理

alert-to-trap サービスの CDS 後処理では、CDS プロセスから作成される EIF イベントを SNMP トラップに変換します。

トラップ内の非変数結合情報はすべて、alert-to-trap サービスによって直接、構成されたトラップに書き込まれます。CDS ファイルを使用してカスタマイズすることはありません。この例外は、特定のトラップ値だけです。

alert-to-trap アダプターは、次のように非変数結合フィールドを設定します。

version
0

community
alert-to-trap 構成ファイル (IHSAATCF) からのコミュニティー・ステートメントの値

enterpriseOID
alert-to-trap 構成ファイルからの enterpriseOID ステートメントの値

IP address
ローカル・ホスト IP アドレス

generic type
6

timestamp
0

個々のタイプは、CDS 処理によって作成される個々のイベント属性の値から取られます。

他のスロット/値の組はすべて、トラップ上で変数結合にエンコードされます。alert-to-trap アダプターの CDS ファイル内のスロット名が有効なオブジェクト ID である場合、そのスロット名は変数結合のオブジェクト ID として使用され、そのスロット値はその変数結合の値となります。CDS ファイル内のスロット名が有効なオブジェクト ID でない場合は、オブジェクト ID 1.3.6.1.4.1.2.5.1.4.1.4.x が変数結合に使用され、その変数結合値は *slot=value* です。ここで、slot は CDS ファイルのスロット名であり、value は CDS ファイルの値です。値 x は 1 から始まる指標で、トラップ内の各変数結合ごとに 1 ずつ増加します。

例えば、スロット名 `1.3.6.1.4.1.2.5.1.4.1.4.1` を値 *examplevalue* にマップする CDS ファイルの MAP ステートメントは、最終トラップ内でオブジェクト ID が `1.3.6.1.4.1.2.5.1.4.1.4.1` で、値が *examplevalue* の変数結合をもちます。

スロット名 *source* を値 *examplevalue* にマップする CDS ファイルの MAP ステートメントは、最終トラップ内でオブジェクト ID が `1.3.6.1.4.1.2.5.1.4.1.4.1` で、値が *source=examplevalue* の変数結合をもちます。この例でオブジェクト ID が想定しているのは、オブジェクト ID を alert-to-trap アダプターに作成させることを要求する変数結合がほかに無かったということであるため、このオブジェクト ID の開始指標は 1 です。

第 9 章 NetView インスツルメンテーション

NetView インスツルメンテーションはサブシステムからなります。NetView 管理コンソール プログラムまたは Tivoli Business Service Manager プログラムがインストール済みであれば、トポロジー・ディスプレイ・サブシステムを使用することができます。他のサブシステム (イベント・フロー・サブシステムを含む) については、Tivoli Business Service Manager プログラムがインストールされていなければなりません。

考慮事項

NetView インスツルメンテーション用の REXX プログラムは、ALTERNATE オプション付きでコンパイルされています。NetView から REXX ランタイム・ライブラリーにアクセスする場合、インスツルメンテーション用の REXX プログラムは、コンパイル・モードで実行します。それ以外の場合は、REXX 代替ライブラリーが使用され、インスツルメンテーション用の REXX プログラムは、解釈モードで実行します。ページング可能リンク・パック域 (PLPA) から REXX ランタイム・ライブラリーまたは REXX 代替ライブラリーにアクセスできない場合、NetView 始動プロシージャーを変更して、これらのライブラリーのいずれかにアクセスする必要があります。

管理情報をトポロジー・サーバーに搬送するイベントは、キーワード/値の組を含むメッセージとして開始されます。API により発行されるこれらのメッセージは BNH351I、BNH352I、BNH353I、および BNH354I です。これらのメッセージは変換され、トポロジー・サーバーに転送されます。

カスタマイズ

下記のサンプルはアプリケーション管理インスツルメンテーション向けに更新されています。これらをユーザーの環境に合わせてカスタマイズする必要があります。

- **CNMSTYLE**

  DSIAMIAT 自動化テーブルおよび AUTOAMI 自動タスクを追加するには CNMSTYLE %INCLUDE のメンバー CNMSTUSR または C*xx*STGEN を使用します。また、TOWER ステートメントを CNMSTYLE から CNMSTUSR または C*xx*STGEN にコピーし、 AMI タワーからアスタリスク (*) を除去します。

- **DSIAMIAT**- サンプル DSIPARM 内

  アプリケーション管理インスツルメンテーション用の別の自動化テーブル。下記の include ステートメントのうちの 1 つをコメント行ではなくする必要があります。

  – %INCLUDE DSIAMIR - BNH351 から BNH354 のメッセージを別の NetView プログラム宛てに送付します。NetView バージョン 2 リリース 4 およびバージョン 3 リリース 1 にはこれを使用します。

– %INCLUDE DSIAMIT - BHN351 から BNH354 のメッセージをメッセージ・アダプター宛てに送付します (イベント自動化サービスが開始済みでなければなりません)。イベント自動化サービス・メッセージ・アダプターの PPI 受信側 ID を変更する必要があります (デフォルト値は IHSATEC)。メッセージ・アダプターは、メッセージを変換して Tivoli Enterprise Console または Tivoli Netcool/OMNIbus に送信します。このプログラム・ルールは、変換後のメッセージをフォーマット設定して、Tivoli Business Service Manager プログラムに送信します。

  IHSAAPMF をメッセージ・アダプター・フォーマット・ファイルに組み込むことで、メッセージ・アダプターを構成します。詳しくは、「*IBM Tivoli NetView for z/OS インストール:追加コンポーネントの構成*」を参照してください。

  ファイル interapp.baroc および interapp_o.rls が事前に ihsttec.sh スクリプトでルール・ベースに追加されていない場合は、それらをルール・ベースにインポートし、Tivoli Enterprise Console イベント・コンソールまたは Tivoli Netcool/OMNIbus を構成します。詳しくは、Tivoli Business Service Manager ライブラリーを参照してください。
– %INCLUDE DSIAMIN - BHN351 から BNH354 のメッセージを NetView 管理コンソール・トポロジー・サーバー宛てに NETCONV 経由で直接送付します (これがデフォルトです)。

• **DSIAMII**- サンプル DSIPARM 内

  アプリケーション管理インスツルメンテーション・メンバー
– フォーカル・ポイント NetView (トポロジー・サーバーまたはメッセージ・アダプター宛てにメッセージを送付する NetView システム) において、すべてのリモート NetView プログラム (ある場合) の NetView ドメインを RMTLU=luname キーワード付きでコーディングします。
– モニターのデフォルトしきい値仕様およびポーリング間隔をユーザーの環境に合わせてカスタマイズします。ここで定義されるデフォルト値は、すべてのコンポーネント・インスタンスまたは接続タイプに適用されます。特定インスタンスのしきい値仕様およびポーリング間隔は、しきい値設定タスクまたはポーリング間隔設定タスクを起動することにより変更することができます。

  複数のしきい値仕様を定義することができます。しきい値仕様それぞれは 3 つの値で構成されます。1 番目の値はしきい値、2 番目の値は演算子、3 番目の値はしきい値イベントの重大度です。例えば、次のようにします。

```
BEGIN_THRESHOLD
 SS=Tivoli;TME10NVCNMTAMEL;1.2
 MONITOR=('STATE'UP,6,0,DOWN,6,5 MVR=CNMETDMV 10)
 MONITOR=('IPC QUEUE' 25,8,2)
 MONITOR=('VIEWMGR QUEUE' 25,8,2)
 MONITOR=('VSTATMGR QUEUE' 25,8,2)
END_THRESHOLD
```

  この例の IPC QUEUE モニターの場合、現行値が 25 を超えると (演算子 **8**)、WARNING (2) しきい値イベントが送信されます。

  それぞれの値の意味は以下のとおりです。

1. 現行モニター値としきい値の値が比較されます。
2. 現行モニター値をしきい値の値と比較するときに使われる比較演算子。

```
0 = greater than
1 = greater than or equal
2 = less than
3 = less than or equal
4 = equal
5 = not equal
6 = changes to
7 = changes from
8 = goes over
9 = goes less than
10 = matches
11 = does not match
```

3. 一致したときに送信されるしきい値イベントの重大度。

```
0 = "NORMAL"
1 = "INFORMATIONAL"
2 = "WARNING"
3 = "SEVERE"
4 = "CRITICAL"
5 = "FATAL"
```

– 下記のリストは、1 つまたは全部のコンポーネントを活動化するために DSIAMII でカスタマイズすることができるものの明細です。

- ハードウェア・モニター・コンポーネント

```
INIT=CNME3016(60)
TERM=CNME3017()
```

CNME3016 のパラメーターは heartbeat_interval です。

- イベント自動化サービス・コンポーネント (メッセージ・アダプター、アラート・アダプター、イベント受信側)

```
INIT=CNME9503(60 IHSAEVNT.IHSATEC)
TERM=CNME9531()
```

INIT=CNME9503 ステートメントを変更し、PROC 名 (procname) およびアダプターの PPI 受信側 ID を組み込みます。

- MSM エージェント・インスツルメンテーション

```
INIT=FLCAPMIN(60)
TERM=FLCAPMTR()
```

FLCAPMIN のパラメーターは heartbeat_interval です。

- トポロジー・ディスプレイ・サブシステム・コンポーネント。これらの DSIAMII メンバーには、インスツルメンテーションの初期化用に複数のステートメントがあります。そのステートメントは、次のとおりです。

```
INIT=CNMETDIN(HBEAT,60)
INIT=CNMETDIN(QDEPTH,10)
INIT=CNMETDIN(GMFHS,CNMSJH10.C)
INIT=CNMETDIN(GPARM,DOMAIN=CNM01)
INIT=CNMETDIN(RODM,EKGXRODM.X)
INIT=CNMETDIN(COLDPARM,TYPE=COLD,INIT=EKGLISLM)
INIT=CNMETDIN(WARMPARM,TYPE=WARM)
INIT=CNMETDIN(COMPLETE)
```

パラメーターは次のとおりです。

- HBEAT はハートビートを指定します。これは必須です。

- QDEPTH はキュー項目数を指定します。これは必須です。
- GMFHS は GMFHS 始動プロシージャーとその別名を指定します。これは必須です。
- GPARM は、GMFHS 始動プロシージャーで使用されるパラメーターを指定します。これは必須ではありませんが、ドメイン値がここで指定されない場合、GMFHS は、初期化メンバー DUIGINIT で、または指定された GMFHS 始動プロシージャーでドメインを見つけようとします。
- RODM は RODM 始動プロシージャーとその別名を指定します。これは必須です。
- COLDPARM は、ユーザーが RODM のコールド・スタートを選択するときに、RODM 始動プロシージャーのパラメーターを指定します。これは必須ではありません。
- WARMPARM は、ユーザーが RODM のウォーム・スタートを選択するときに、RODM 始動プロシージャーのパラメーターを指定します。これは必須ではありません。

インスツルメンテーションを作成する場合、DSIAMII を変更してデフォルトのしきい値仕様およびインスツルメンテーション初期化ルーチンと終了ルーチンの呼び出しを追加する必要があります。API の説明については、 Tivoli Business Service Manager ライブラリーを参照してください。

インスツルメンテーションの開始と停止

インスツルメンテーションを開始するには、フォーカル・ポイント NetView (メッセージ・アダプター宛てにメッセージを送付する NetView プログラム) において、INITAMI コマンドを発行します。INITAMI は、リモートとして DSIAMII に定義されている複数の NetView プログラムで自動的に発行されます。INITAMI コマンドは NetView プログラムのフォーカル・ポイントにおいて AUTOAMI を開始します (まだ開始されていない場合)。AUTOAMI のコンソール ID は AMI*xxxxx* に設定されます。ただし *xxxxx* は、NetView ドメインの右端の 5 文字です。したがって、コンソールはシスプレックス内で固有となり、また自動タスクから発行されるコマンドは互いに関連付けられます。

ただし、AUTOAMI においてインスツルメンテーションが強制的に実行されることはありません。したがって、システム内に複数の NetView プログラムが存在している環境、つまりシスプレックス内では、自動タスク AUTOAMI において INITAMI コマンドを発行しなければなりません。

INITAMI コマンドは、DSIAMII に RMTLU ステートメントでドメイン名がコーディングされている任意の NetView システムとの RMTCMD セッションも確立します。これはその NetView プログラム上の AUTOAMI 自動タスクにログオンします。

インスツルメンテーションを停止するには、TERMAMI コマンドを発行します。TERMAMI は、リモートとして DSIAMII に定義されている複数の NetView プログラムで自動的に発行されます。このほかに、フォーカル・ポイント NetView 上の AUTOAMI 自動タスクを停止します。これにより、INITAMI で確立された RMTCMD セッションは終了します。

トポロジー・サーバーは、TERMAMI コマンドの発行後に、インスツルメンテーション関連コマンドを発行することができます。ただし、そのコマンドが機能するためには AUTOAMI 自動タスクが開始されていなければなりません。

IBM Tivoli Enterprise Console のカスタマイズ

イベント自動化サービス・メッセージ・アダプターを介して IBM Tivoli Enterprise Console にインスツルメンテーション・メッセージを送付するには、そのコンソールをカスタマイズする必要があります。

ACB モニターのカスタマイズ

アプリケーション制御ブロック (ACB) モニター・フォーカル・ポイントは、フォーカル・ポイントの仮想記憶通信アクセス方式 (VTAM) およびエントリー・ポイント VTAM から ACB の更新状況を受け取ります。 Tivoli Business Service Manager プログラムと ACB モニターとを併用すれば、ACB モニターで下記のものが検出されます。

- 総称リソース
- ユーザー指定のアプリケーション
- ユーザー指定モデルに一致するアプリケーション

ACB モニターは以下のものもモニターします。

- ACB 状況
- セッション・カウント
- ACB アプリケーション用持続リカバリー・イベント

Tivoli Business Service Manager プログラムあるいは NetView 管理コンソール TN3270 管理と ACB モニターとを併用すれば、ACB モニターで TN3270 サーバーとクライアントとが検出されます。オプションで ACB データを DB2® データベースに保管することができます。

それぞれのシステム複合体 (またはシスプレックス、つまり z/OS システムの集合) ごとに、 ACB モニター・フォーカル・ポイントを 1 つ定義します。シスプレックス環境においてアプリケーション力学のインスツルメンテーションを十分に生かすには、シスプレックス内の他のすべてのイメージをそのフォーカル・ポイントのエントリー・ポイントとして定義してください。

ACB データを DB2 に保管することにより、telnet クライアントを、IP アドレス、ホスト名、またはアプリケーション名で照会することができます (TN3270 クライアントを見付ける TBSM タスクを使用)。また、ACB モニターを再始動せずに、重要な TN3270 クライアント・リソースのリストを変更することも可能になります。

注:

1. ACB データを DB2 に保管するには、ACB モニター・フォーカル・ポイントで DB2 が操作可能であること、および NetView SQL パイプ・ステージが使用可能になっていることが必要です。

2. ACB モニター・インスツルメンテーションを使用可能にするためには、ACB モニター・フォーカル・ポイントにおいて AMI が使用可能になっていなければなりません。

パーツ

ACB モニターの一部として出荷されるパーツのリストを表 17 に示します。

表 17. Tivoli Business Service Manager パーツ・リスト

パーツ名	言語	機能
TN3270.BSDF	MIF	TN3270 ビジネス・システム記述ファイル
TN3270.BCDF	MIF	TN3270 ビジネス・コンポーネント記述ファイル
TN3270.BMDF	MIF	TN3270 ビジネス・マッピング記述ファイル
TN3270.CDF	MIF	TN3270 コンポーネント定義ファイル
Ltn3270loc.ddf	DDF	TN3270 クライアントを見付けるローカル・ダイアログ定義
Ltn3270glob.ddf	DDF	TN3270 クライアントを見付けるグローバル・ダイアログ定義
TN3270.html	HTML	ヘルプ・ファイル
GENRSC.BSDF	MIF	総称リソース・ビジネス・システム記述ファイル
GENRSC.BCDF	MIF	総称リソース・ビジネス・コンポーネント記述ファイル
GENRSC.BMDF	MIF	総称リソース・ビジネス・マッピング記述ファイル
GENRSC.CDF	MIF	総称リソース・コンポーネント定義ファイル
GENRSC.html	HTML	ヘルプ・ファイル
VTAMAPPL.BSDF	MIF	VTAM アプリケーション・ビジネス・システム記述ファイル
VTAMAPPL.BCDF	MIF	VTAM アプリケーション・ビジネス・コンポーネント記述ファイル
VTAMAPPL.BMDF	MIF	VTAM アプリケーション・ビジネス・マッピング記述ファイル
VTAMAPPL.CDF	MIF	VTAM アプリケーション・コンポーネント定義ファイル
VTAMAPPL.html	HTML	ヘルプ・ファイル

フォーカル・ポイントの定義

下記のステップに従って ACB モニター・フォーカル・ポイントを定義します。

1. サンプル `DSIAMIAT` 内の自動化テーブルをカスタマイズします。`%INCLUDE CNMSVTFT` をコメント行ではなくします。
2. 下記のステップに従って、サンプル `DSIAMII` 内の AMI 構成メンバーをカスタマイズします。
   a. 各 ACB モニター・エントリー・ポイントの NetView ドメイン名を `AMONLU=`*keyword* でコーディングします。

b. ACB データを DB2 に保管するかどうかで次のように分かれます。
   - 保管する場合は、2c と 2d のステップを実行してください。
   - 保管しない場合は、ステップ 2e に進んでください。

c. AMONDB2=*y* とコーディングします。

d. DB2 ボリュームを DB2VOL=*keyword* でコーディングします。

e. DB2 ボリューム・カタログを DB2VCAT=*keyword* でコーディングします。

f. モニターされる定義済み VTAM アプリケーションごとに、DB2 バッファー・プールを DB2BUFFERPOOL=*keyword* でコーディングします。

3. VTAM アプリケーションとモデルがサンプルの DSIAMII 内で見つかるように、これらのリストをカスタマイズします。
   a. モニターされる定義済み VTAM アプリケーションごとに、APPLCOMPONENT=*applname* でアプリケーション名をコーディングします。
   b. モニターされる VTAM モデルごとに、 MODELCOMPONENT=*modelname* でモデル名をコーディングします。
4. ACB データを DB2 に保管するかどうかで次のように分かれます。
   - 保管しない場合は、ステップ 5 に進んでください。
   - 保管する場合は、以下のステップを行ってサンプル DSIAMII 内の DB2 パラメーターをカスタマイズしてください。
     a. AMONDB2=*Y* とコーディングします。
     b. DB2 ボリュームを DB2VOL=*keyword* でコーディングします。
     c. DB2 ボリューム・カタログを DB2VCAT=*keyword* でコーディングします。
     d. DB2 バッファー・プールを DB2BUFFERPOOL=*keyword* でコーディングします。
5. サンプル DSIAMII 内のデフォルトのしきい値をカスタマイズします。以下のどれでもカスタマイズすることができます。
   - ACB 状況モニターに対してしきい値イベントが発行されたとき
   - ACB 状況モニターに対して発行されたイベントの重大度
   - セッション・カウント・モニター
   - 持続リカバリー・モニター

   DSIAMII をカスタマイズすることで、デフォルトのしきい値を定義します。しきい値設定タスクを用いれば、それぞれのインスタンス (アイコン) ごとにしきい値をカスタマイズすることもできます。

   例えば、APPLCOMPONENT と MODELCOMPONENT のアプリケーションについて CONCT 状態と RESET 状態のしきい値重大度を、情報 (1) から重大 (3) に変更する場合は、以下の行を変更します。

   ```
   ACT,6,0,CONCT,6,1,RESET,6,1,INACT,6,2,UNKNOWN,6,2,PINACT,6,4,PACT,6,4
   ```

   次のように変更してください。

   ```
   ACT,6,0,CONCT,6,3,RESET,6,3,INACT,6,2,UNKNOWN,6,2,PINACT,6,4,PACT,6,4
   ```

あるいは、セッション・カウントが 999 を超えたならば警告しきい値イベントを発行し、セッション・カウントが 1000 より下がったならば正常しきい値イベントを発行するには、以下の行を

```
MONITOR=('SESSION COUNT' 0,1,0 EVENT)
```

次のように変更してください。

```
MONITOR=('SESSION COUNT' 999,8,2,1000,9,0 EVENT)
```

6. ACB モニター VTAM 出口をインストールします。CSECT CNMIETMN を、VTAM 用 VTAMLIB DD のロード・モジュール ISTIETMN にリンク・エディットします。

エントリー・ポイントの定義

下記のステップに従って ACB モニター・エントリー・ポイントを定義します。

1. サンプル DSIAMIAT 内の自動化テーブルをカスタマイズします。%INCLUDE CNMSVTET をコメント行ではなくします。
2. ACB モニター VTAM 出口をインストールします。CSECT CNMIETMN を、VTAM の VTAMLIB DD にあるロード・モジュール ISTIETMN にリンクします。

VTAM ACB モニターの開始

以下のものに対するインスツルメンテーションを使用可能にするために、フォーカル・ポイント NetView システムにおいて **INITAMI** コマンドを発行し、AMI を開始します。

- 総称リソース
- TN3270 サーバー
- APPLCOMPONENT VTAM アプリケーション
- MODELCOMPONENT VTAM アプリケーション

VTAM ACB モニターを開始するには、フォーカル・ポイント NetView において **INITAMON** コマンドを発行します。このフォーカル・ポイント、および AMONLU=*keyword* で識別されるすべてのエントリー・ポイントが活動化されます。

VTAM ACB モニターを活動化してから、**INITAMON** *entry_point* コマンドを発行してその他のエントリー・ポイントを活動化します。ここで、*entry_point* には、エントリー・ポイントの NetView ドメイン名を指定します。

VTAM ACB モニター・エントリー・ポイントのリカバリー:

エントリー・ポイントとフォーカル・ポイントとの間の RMTCMD LU 6.2 セッションで障害が起こると、そのエントリー・ポイントは自動的に停止されます。この 2 つの NetView プログラム間の通信障害の原因となったエラーを訂正してから、エントリー・ポイントをリカバリーするために、フォーカル・ポイントから **INITAMON** *entry_point* コマンドを発行します。

VTAM ACB モニターの停止

VTAM ACB モニターを停止するには、フォーカル・ポイント NetView において **TERMAMON** コマンドを発行します。フォーカル・ポイントおよびすべてのアクティブ・エントリー・ポイントは非活動化されます。

特定のエントリー・ポイントを停止するには、**TERMAMON** *entry_point* コマンドを発行します。ここで、 *entry_point* には、停止させたいエントリー・ポイントの NetView ドメイン名を指定します。その NetView システムに関連する VTAM 上のすべてのアプリケーションの状況がデータベースから除去されます。

第 10 章 NetView Web サーバー用の HTML ファイルの設計

NetView プログラムには、Web ブラウザー・インターフェースからコマンドを受け取る Web アプリケーション・サーバーが用意されています。これにより、ユーザー独自の Web ページ用の HTML ファイルを設計することができます。

ファイルおよびコマンドの参照

NetView プログラムに Web ブラウザーからアクセスするための HTML は、Web アプリケーション・サーバーに動的に生成されます。

HTML (ユーザー作成の HTML を含む) は、 Web アプリケーション・サーバー (ワークステーション) と、 NetView for z/OS ホストに分割されることがあります。これには、ホスト HTML からのワークステーション・ファイルの参照が含まれます。

基本 URL について

以下は、NetView プログラムをブラウズする場合の一般的な URL です。

`https://`*`web_application_server:port`*`/netview/`*`domain_ID`*`/`

ここで、*web_application_server:port* は TCP ホスト名および NetView Web アプリケーションがインストール済みの HTTPS サーバーのポート番号、`netview` は NetView Web アプリケーションのコンテキスト・ルート、および *domain_ID* は接続したい NetView for z/OS プログラムのドメイン ID です。

この例の URL は、基本 *URL* と見なされます。URL の中に疑問符 (?) が入っている場合には、残りのデータは照会データと見なされて、基本 URL の一部とは見なされません。

Web アプリケーション・サーバー上のワークステーション・ファイルの参照

他のソースの参照は、基本 URL を基準にした相対パスを用います。 Web アプリケーション・サーバー上のファイルの場合は、 `../` を使用して、`/netview/` ディレクトリーにバックアップします。

NetView コマンドの参照

次の例では、実行する NetView コマンドを指定します。コマンドが正しく解析されるように、コマンド内のブランクはすべて正符号 (+) として指定する必要があります。NetView Web サーバー (DSIWBTSK) は、正符号 (+) をブランクに変更してから、コマンドを実行します。

```
https://web_application_server:port/netview/domain_ID/?DSICMDS+=+command
```

ポートフォリオへのタスクおよびリンクの追加

ご自分の Web ページまたは他の Web ページにタスク (リンク) を追加して、NetView Web アプリケーションをカスタマイズすることができます。ポートフォリオへタスクを追加するには、 webmenu ステートメントを使用します。詳しくは、「*IBM Tivoli NetView for z/OS アドミニストレーション・リファレンス*」または CNMSTWBM メンバーを参照してください。

例えば、新規タスク・グループを追加する場合は、既存の webmenu.groups ステートメントにそのグループ ID を追加します。グループに名前を割り当てるには、webmenu.*group_ID*.name ステートメントを使用します。グループに 1 つ以上のタスク (リンク) を割り当てるには、webmenu.*group_ID*.groups ステートメントを使ってタスク ID を指定します。

新規グループに個々のタスクを定義するには、以下のように行います。

- webmenu.*group_ID*.*task_ID*.name ステートメントを使用して、タスクに名前を付ける。
- webmenu.*group_ID*.*task_ID*.action ステートメントを使用して、タスクにアクションを割り当てる。

例えば、NetView プログラム・ベースの *myhtml* という名前の HTML 生成ルーチンを呼び出すには、以下のような webmenu ステートメントを使用します。

```
webmenu.group_ID.task_ID.name = My HTML
webmenu.group_ID.task_ID.action =
      https://web_application_server:port/netview/domain_ID/?DSICMDS=myhtml
```

Web サイト (例えば、http://www.ibm.com/) を起動するには、以下の webmenu ステートメントに示すように、action ステートメントの中に http: (または https:) を組み込み、さらに個々のスラッシュ (/) をコーディングします。

```
webmenu.group_ID.task_ID.action = http:&slash;&slash;www.ibm.com/
```

注:

1. Web サイトの起動例については、CNMSTWBM メンバーにあるLaunch Sample URL タスクの webmenu ステートメントを参照してください。
2. ユーザー定義の URI (Uniform Resource ID) では、スラッシュを 2 つ連続 (//) して指定しないことにご注意ください。代わりに、URI では以下のいずれかの方法で連続した 2 つのスラッシュを指定します。
   - &SLASH;/
   - /&SLASH:
   - &SLASH;&SLASH;

REXX の使用による HTML の生成

NetView プログラムは、REXX プロシージャー用のコモン・ゲートウェイ・インターフェース (CGI) に類似したインターフェースをサポートします。ユーザーのプロシージャーが NetView Web サーバーによって呼び出されたのかどうかを、REXX 関数 CGI ( ) を使用して判別してください。CGI ( ) からの戻りが **1** であれば、最初の出力行の初めの文字を以下のものにすることにより、プロシージャーは動的 Web ページを作成することができます。

- <HTML
- <!DOCTYPE

注: HTML と DOCTYPE は大文字でなければなりません。

この場合、NetView プログラムは出力の変更も追加も行いません。パイプ・ステージ CONSOLE ONLY を使用して出力を作成して、HTML 出力のログ記録および自動化を防ぐことができます。

注: カスタマイズを行うには、CGI 関数をお勧めします。

パフォーマンスを向上させるため、静的な HTML ファイルまたはバイナリー・ファイルを Web アプリケーション・サーバー上に置くことができます。

NetView Web アプリケーションは、ユーザー作成の HTML で POST メソッドと GET メソッドの両方を引き続き処理します。 POST メソッドを使用している場合、NetView Web アプリケーションは、処理の前にそれを GET に変更します。GET メソッドの場合は、関連するすべてのデータが URL の照会ストリング部分に配置され、ブラウザー・ウィンドウの上部に表示されます。 HTML に TITLE エレメントを追加すると、照会ストリングのデータではなく、TITLE を表示できます。

第 11 章 Common Base Events を使用したカスタマイズ

Common Base Events は、状況変更や問題報告といったシステム・イベントに関する XML ベースの表記法です。NetView プログラムは、メッセージおよび MSU の XML フォーマットのイベントへの変換と、それらのストレージ用イベント・サーバーへの転送、および必要な相手への配布をサポートします。「*IBM Tivoli NetView for z/OS 自動操作ガイド*」には、Common Base Events を含む自動化に関するセクションがあります。このサポートを対象としたカスタマイズには、メッセージや MSU を変換する際に NetView プログラムが使用する XML テンプレートの作成および変更が含まれます。テンプレートは、イベントにメッセージや MSU からどんな情報を含めるかを制御する簡単な方法を提供します。

Common Base Event のフォーマットはスキーマで定義され、「*Autonomic Computing Toolkit Developer's Guide*」(SC30-4083) に記述されています。この資料の「Common Base Event XML Schema」セクション内の「Understanding Common Base Event Specification」セクションを参照してください。

XML フォーマット

NetView プログラムは、事前定義されたイベント・テンプレートのセットを提供します。これにより、ユーザーは自動化テーブルのアクションを使用して、またはアラートの場合にはハードウェア・モニターのフィルター設定値を使用して Common Base Events を作成することができます。テンプレートはサンプル・ファイル CNMSCBET の中にあります。テンプレートは、初期設定時または RESTYLE CBE コマンドの結果として、CNMSTYLE の CBE.TEMPLATES ステートメントが処理されるときに NetView プログラムによってストレージにロードされます。

イベントのカスタマイズには、イベント用の XML テンプレートの変更や追加、または、自動化テーブル処理の一環としてのイベントの変更が含まれます。自動操作中のイベントの変更例については、サンプル CNMSCBEA を参照してください。このセクションの後半では、NetView プログラムが提供するイベント・テンプレート、およびそのテンプレートの変更方法について記述します。テンプレートは XML スケルトンであり、実行時にメッセージや MSU からのデータを使って入力されるいくつかの変数シンボルを使用します。テンプレートを正しく処理するためには、イベントの XML フォーマットについての知識が必要です。詳細については、前述の「*Autonomic Computing Toolkit Developer's Guide*」のセクションを参照してください。

Common Base Event の主要な XML エレメントには、以下のものが含まれます。

CommonBaseEvent エレメント

このエレメントは、Common Base Event のルートです。ここには、以下のものを含め、イベントを記述する多くの属性が含まれています。

- 固有のグローバル ID
- イベントの作成時刻
- イベントのテキスト記述 (`msg` 属性)

CommonBaseEvent エレメント用に NetView プログラムが提供するデフォルトは、 CNMSCBET 内の *root* テンプレートです。

reportingComponent エレメント
イベントを生成するシステム・コンポーネントを示します。ここには、プロダクト名やネットワーク・ロケーションなどの情報が含まれます。また、スレッド ID やプロセス ID などの詳細情報も提供します。NetView プログラムが生成するイベントの場合、このコンポーネントは通常 `NetView` を指定します。

sourceComponent エレメント
イベントを生成させたシステム・コンポーネントを示します。例えば、イベントを生成するのに使用されたメッセージを出した z/OS ジョブを指定することができます。

situation エレメント
コンポーネント始動の結果生成される startSituation イベントのような、ハイレベル・ビューのイベント・タイプを提供します。テンプレート *repsitm* (メッセージの場合) および *repsita* (MSU の場合) は、このエレメント用に NetView プログラムが提供するデフォルトです。

ExtendedDataElements
これらはイベントの拡張機能を提供し、製品固有の情報を保持します。NetView プログラムは、イベントを生成しているメッセージやアラートに関する追加情報を保持するために使用できる多くの拡張機能を提供します。

Common Base Event フォーマット・ルール

有効な Common Base Event XML 文書は、「 *Autonomic Computing Toolkit Developer's Guide* 」に XML スキーマで記述されています。つまり、XML イベントのエレメントが CommonBaseEvent エレメントの中に含まれています。以下のエレメントは、下記の順序で組み込むことができます。

contextDataElements
オプション

extendedDataElements
オプション

associatedEvents
オプション

reporterComponentId
必須 (ソース・コンポーネントと同じである場合を除き、ソース・コンポーネントに組み込まれていない場合)

sourceComponentId
必須

msgDataElement
オプション

situation
必須

テンプレート・ファイル CNMSCBET

NetView のテンプレート・サンプル CNMSCBET には、msgdefault および msudefault のような完全なイベントを定義するいくつかのテンプレートがあります。その他のエレメント (例えば、拡張データ・エレメント) では、完全イベントを形成するために結合できる、イベントの断片の定義のみを行います。これらのテンプレートは、パイプ EDIT ステージにおいて CBETEMP グローバル・オーダーで使用することを意図しています。このオーダーはイベント自動化テーブルのアクションと一緒に使用され、テンプレートへの読み取りおよび完全イベントを構成するのに使用できます。もっとも単純な CBETEMP の使用は、完全イベント・テンプレートの中の 1 つ (例えば、msgdefault) に読み取ることですが、オーダーはイベントの断片 (これは、後で編集仕様で構成されます) を定義するテンプレートの読み取りにも使用することができます。

テンプレート・ファイル自体は XML 文書です。<cbedata name='xxxx'> タグは、イベントの XML データを定義し、またここで、このデータに名前をつけることができます。name 属性値は、<?include> の処理の際に CBETEMP グローバル編集オーダーで使用されます。テンプレート・ファイルは 2 つの XML エレメント (*cbedata* と *templates*)、および 1 つの処理命令 (*include*) から構成されています。ほぼすべてのケースにおいて、*cbedata* が定義する XML は XML CDATA セクションに含まれます。 CDATA セクション内の XML マークアップは、テンプレート・ファイルが NetView プログラムによってロードされるときに、純粋な文字データとして処理されます。この XML は、イベントが構成されるまで XML として処理されず、また解析もされません。

<?include name='xxx'?> は、テンプレート・ファイル内の他の部分で定義された <cbedata> エレメントにプルする処理エレメントを定義します。これにより、複数のテンプレートで使用可能な XML エレメントを事前定義することが可能になります。例で示すように、root は msgdefault および msudefault の両方で使用されます。

組み込み処理は、CNMSTYLE の CBE.TEMPLATES ステートメントが処理されるときに実行されます。XML テンプレートの結果は内部的にメモリー内に格納され、CBETEMP で参照されます。DISPCBET コマンドを使用して、テンプレートが処理されロードされた後、そのテンプレートのメモリー内の内容を表示することができます。

以下はテンプレートの例です。

```
<cbedata name='root'>
<![CDATA[
<CommonBaseEvent
elapsedTime='0'
version='1.0.1'
msg='&TEXT'
priority='&PRIORITY'
severity='&SEVERITY'
repeatCount='0'
sequenceNumber='0'
>
]]>
</cbedata>
```

このエレメントはイベントの先頭を定義します。他のテンプレートは *include* 処理オーダーを使用して、このテンプレートにプルします。例えば、次のようにします。

```
<cbedata name='msgdefault'>
<!-- Beginning of CBE goes here -->
<?include name='root'?>
```

<?include と <cbedata は両方とも name='xxxx' 属性を使用します。この値を単一引用符で囲みます。name 属性値は、大/小文字の区別をしません。ただし、ファイル内の XML エレメント名 (cbedata、include、および templates) は、CDATA セクションに含まれる XML マークアップのように大/小文字の区別をします。

NetView プログラムは完全イベントを構成するいくつかのテンプレートを提供します。以下は msgdefault テンプレートです。

```
<cbedata name='msgdefault'>
<-- Beginning of CBE goes here -->
<?include name='root'?>
<!-- Context Data Elements, if any, go here -->
<!-- Extended Data Elements start here -->
<?include name='nvobjtypems'?>
<?include name='correlate'?>
<?include name='actionmg'?>
<?include name='mlwtolns'?>
<!-- Associated Events, if any, go here -->
<!-- Reporting component goes here -->
<?include name='nvrepcompmsg'?>
<! -- Source component goes here -->
<?include name='nvsrccompmsg'?>
<!-- Message Data Element goes here -->
<?include name='nvmsgdataelm'?>
<!-- Situation goes here -->
<?include name='repsitm'?>
<?include name='tail'?>
</cbedata>
```

コード・ページの考慮事項

CNMSCBET ファイルは、コード・ページ 037 コード (US EBCDIC) を使用してエンコードされます。解析コードは、コード・ページ 037、コード・ページ 1047、またはコード・ページ 939 を処理することができます。コード・ページ間の主な違いは大括弧文字です。大括弧文字は、コード・ページ 037、1047、および 939 では異なる 16 進数コード・ポイントにマップします。エンコード方式は、ファイルの XML 処理命令でエンコード属性により指定されます。指定するエンコード方式は、そのファイルで使用する文字と一致する必要があります。エンコードの値は、xml 命令で属性として指定します。次に例を示します。

```
<?xml version="1.0" encoding="ebcdic-cp-us"?>
```

encoding に有効な値およびそのマップ対象となるコード・ページは、以下のとおりです。

ebcdic-cp-us
037

IBM-037
037

IBM037
037

IBM-1047
1047

IBM1047
1047

X-EBCDICJapaneseAndJapaneseLatin
939

IBM-939
939

IBM939
939

大括弧用に使用する文字 (特に CDATA セクションで) は、指定されたコード・ページに応じた 16 進数のエンコードに厳密に一致する必要があります。コード・ページ 037 の場合、左大括弧文字 [は X'BA' であり、右大括弧文字] は X'BB' です。しかし、コード・ページ 1047 および 939 の場合は、左大括弧文字 [は X'AD' であり、右大括弧文字] は X'BD' です。

CDATA セクションで大括弧を使用する場合、xml 命令で指定されたエンコード方式に関係なく、英語のシステムではその大括弧を 037 エンコードにマップし、日本語のシステムでは 1047 または 939 エンコードにマップしてください。XML イベントを処理する際に、システムが英語か日本語かに応じて、大括弧文字を含んだ文字ストリームはコード・ページ 037 または 939 に基づいて変換されます。互換性がなくならないように、コード・ページに依存しないシーケンスである [を左大括弧に、また、右大括弧には] が使えます。次に例を示します。

```
<cbedata name='example'>
<![CDATA[
<extendedDataElements name='example' type='string'>
<values>This code will have a bracket character coded as &#x5B' in it.</values>
</extendedDataElements>
]]>
</cbedata>
```

注: [および] を、CDATA セクションで大括弧文字に使用することはできません。

事前定義された変数

テンプレートの多くは、&DOMAIN のような変数を参照します。変数は実行時にテキスト・ストリングで置換されます。例えば、&DOMAIN は NTVE4 で置換されます。変数に利用可能なデータがなければ、何も戻されません。すべての変数ではないが、そのほとんどが自動化テーブルの条件項目にならって作られています。すべてではないが、ほとんどの自動化テーブルの条件項目が提供されます。いくつかの変数が追加で提供されます。ビット項目はバイナリー・ストリングで表示されます。例えば、ビット値 one は文字 1 です。条件項目および考えられる値については詳しくは、「*IBM Tivoli NetView for z/OS 自動操作ガイド*」を参照してください。

以下の変数が使用可能です。

&ACTIONMG
アクション・メッセージを示します

&ALRHOSTNAME
アラートが発信されたノードの SNA 名 (アラートの場合)

&AUTOTOKE
MVS メッセージ自動化トークン

&AUTOMATED
NetView プログラムがメッセージに応じた自動化を実行したかどうかを示す

&CART
MVS コマンド/応答トークン

&CURSYS
z/OS システムのローカル名

&DESC
メッセージ記述子コード

&DOMAIN
現行 NetView ドメイン ID

&DOMAINID
メッセージが発信された NetView ドメイン (メッセージの場合)

&HDRMTYPE
NetView メッセージ・タイプ・インディケーター

&HIER
アラート名/タイプのリスト

&HMASPRID
ハードウェアまたはソフトウェアのプロダクト・セット ID (アラートの場合)

&HMBLKACT
アラートのブロック ID およびアクション・コード

&HMCPLINK
複合リンク・インディケーター

&HMEPNAU
アラートが発信されたエントリー・ポイント・ノードの NAU 名

&HMEPNET
アラートが発信されたネットワークのエントリー・ノード

&HMEPNETV
エントリー・ポイント・ノードが NetView プログラムであったかどうかを明確にする

&HMEVTYPE
MSU イベント・タイプ

&HMFWDED
アラートが LUC を使用して転送されたかどうかを示す

&HMFWDSNA
アラートが LU 6.2 を使用して転送されたかどうかを示す

&HMGENCAU
MSU の汎用原因コード

&HMONMSU
MSU が自動化されたかどうかを示す

&HMORIGIN
名前/タイプ階層リストにあるラストネーム (アラートの場合)

&HMSECREC
MSU 用の 2 次記録が行われたかどうかを示す

&HMSPECAU
MSU の特定のコンポーネント・コード

&HMUSRDAT
MSU のサブベクトル 33 からのユーザー・データ

&JOBNAME
メッセージの発信元の z/OS ジョブ名

&JOBNUM
メッセージの発信元の z/OS ジョブ番号

&KEY メッセージに関連付けられたキー

&MCSFLAG
MVS 複数コンソール・サポート・フラグ

&MSGAUTH
許可プログラム・インディケーター

&MSGCATTR
MVS メッセージ属性フラグ

&MSGCMISC
MVS のその他の経路指定フラグ

&MSGCMLVL
MVS メッセージ・レベル・フラグ

&MSGCMSGT
MVS メッセージ・タイプ・フラグ

&MSGCOJBN
起点ジョブ名

&MSGCPROD
z/OS レベル

&MSGCSPLX
メッセージ送信側のシスプレックス

&MSGDOMFL
MVS DOM フラグ

&MSGGDATE
メッセージに関連付けられた日付

&MSGGMFLG
MVS 汎用メッセージ・フラグ

&MSGGMID
MVS メッセージ ID

&MSGGTIME
メッセージが出された時刻

&MSGID
メッセージ ID (メッセージの場合)

&MVSLEVEL
現行 MVS プロダクト・レベル

&MVSRTAIN
MVS AMRF フラグ

&MSGSRCNM
ソース・オブジェクトからのソース名

&NETID
VTAM ネットワーク ID

&NETVIEW
NetView のバージョンおよびリリース

&NVASID
ローカル NetView プログラムの z/OS アドレス・スペース ID

&NVCLOSE
NetView CLOSE プロセス・フラグ

&NVHOSTNAME
NetView プログラムが使用している TCP スタックの完全修飾名

&NVJOBNAME
ローカル NetView プログラムの z/OS ジョブ名

&NVTASKID
メッセージが自動化されているタスクの NetView 名

&OPID
NetView オペレーター ID またはタスク ID

&OPSYSTEM
オペレーティング・システム

&PRIORITY
イベントの重要度を表す 0 から 100 の間の整数

&ROUTECD
MVS 宛先コード

&SEVERITY
イベントの重大度を表す 0 から 70 の間の整数

&SYSID
メッセージの発信元の z/OS システム ID

&SYSPLEX
ローカルの z/OS シスプレックス名

&TEXT
テキスト・メッセージ、または長いエラーの説明の先頭行

&VTAM
VTAM レベル

&VTCOMPID
VTAM コンポーネント ID

付録 A. ハードウェア・モニター・パネルのカラー・マップ

表 18 は、ハードウェア・モニター・パネルのパネル名、パネル番号、およびカラー・マップのリストです。カラー・マップについての詳細は、 87 ページの『第 6 章 ハードウェア・モニターの表示データのカスタマイズ』を参照してください。

注: ハードウェア・モニター・ヘルプ・パネルおよびコマンド記述パネル用のカラー・マップは、NetView プログラムの以前のリリースでのみ利用可能です。また、BNJMP1 で始まるカラー・マップもサポートされなくなりました。

表 18. ハードウェア・モニター・パネルのカラー・マップ

パネル名	パネル番号	カラー・マップ
Alerts-Dynamic (アラート動的) Alerts-History (アラート・ヒストリー) Alerts-Static (アラート静的) Common Format Glossary (共通形式用語集)	NPDA-30A NPDA-31A NPDA-30B NPDA-02C	BNJMP30A BNJMP31A BNJMP30A BNJMP2C1
Controller Information Display (コントローラー情報表示) Controller (CTRL) Selection Menu (コントローラー (CTRL) 選択メニュー) Downstream Member of Token-Ring LAN Fault Domain (トークンリング LAN の障害領域のダウンストリーム・メンバー)	NPDA-02E NPDA-CTRL NPDA-44B	BNJMP02E BNJMPCTL BNJMP4BH
DSU/CSU and Line Status (DSU/CSU と回線状況) DSU/CSU and Line Parameters (DSU/CSU と回線パラメーター) Link Segment Level *n* (リンク・セグメント・レベル n)	NPDA-22C、ページ 1	BNJMPDL1
DSU/CSU and Line Status (DSU/CSU と回線状況) Remote DSU/CSU Interface-Remote Device (リモート DSU/CSU インターフェース-リモート装置) Status-Link Segment Level *n* (状況-リンク・セグメント・レベル n)	NPDA-22C、ページ 2	BNJMPDL2
DSU/CSU and Line Status (DSU/CSU と回線状況) Configuration Summary (構成要約)、 Link Segment Level *n* (リンク・セグメント・レベル n)	NPDA-22C、ページ 3	BNJMPDL3
Event Detail (イベント詳細) Event Detail (イベント詳細) Event Detail (イベント詳細) Event Detail (イベント詳細) Event Detail (イベント詳細)	NPDA-43B NPDA-43M NPDA-43N, 43Q NPDA-43C NPDA-43T	BNJMP43B BNJMP43M BNJMP43N BNJMP43C BNJMP43T
Event Detail (イベント詳細) Event Detail (イベント詳細) Event Detail (イベント詳細) Event Detail (イベント詳細)、代替 Event Detail (イベント詳細)、代替	NPDA-43A NPDA-43P NPDA-43S NPDA-43T NPDA-43S	BNJMP43A BNJMP43P BNJMP43S BNJMP434 BNJMP433
Event Detail for BSC Line (イベント詳細 (BSC 回線)) Event Detail for BSC Station (イベント詳細 (BSC 端末)) Event Detail for BSC/SS Line (イベント詳細 (BSC/SS 回線)) Event Detail for BSC/SS Station (イベント詳細 (BSC/SS 端末)) Event Detail for Channel-Attached Station (イベント詳細 (チャネル接続端末))	NPDA-43T NPDA-43T NPDA-43B NPDA-43B NPDA-43B	BNJMP43T BNJMP43T BNJMP43B BNJMP43B BNJMP43B

表 18. ハードウェア・モニター・パネルのカラー・マップ (続き)

パネル名	パネル番号	カラー・マップ
Event Detail for Channel Link (イベント詳細 (チャネル・リンク)) Event Detail for Instruction Exception (イベント詳細 (命令例外)) Event Detail for Miscellaneous Interrupts (イベント詳細 (各種割り込み)) Event Detail for Scanner-Type 1/4 (イベント詳細 (スキャナー・タイプ 1/4)) Event Detail for Scanner-Type 2/3 (イベント詳細 (スキャナー・タイプ 2/3))	NPDA-43B NPDA-43J NPDA-43K NPDA-43G NPDA-43H	BNJMP43B BNJMP43J BNJMP43D BNJMP43D BNJMP43D
Event Detail for Scanner-Type 1 (イベント詳細 (スキャナー-タイプ 1)) Event Detail for Scanner-Type 2 (イベント詳細 (スキャナー-タイプ 2)) Event Detail for Scanner-Type 3 (イベント詳細 (スキャナー-タイプ 3)) Event Detail for Scanner-Type 4 (イベント詳細 (スキャナー-タイプ 4)) Event Detail for SDLC Line (イベント詳細 (SDLC 回線))	NPDA-43D NPDA-43E NPDA-43F NPDA-43I NPDA-43P	BNJMP43D BNJMP43D BNJMP43D BNJMP43D BNJMP43B
Event Detail for SDLC Line (イベント詳細 (SDLC 回線)) Event Detail for SDLC Station (イベント詳細 (SDLC 端末)) Event Detail for SDLC Station (イベント詳細 (SDLC 端末)) Event Detail for 3270 Non-SNA Controller (イベント詳細 (3270 非 SNA コントローラー)) Event Detail Menu (イベント詳細メニュー) Event Detail Menu (イベント詳細メニュー)	NPDA-43T NPDA-43B NPDA-43T NPDA-43L NPDA-43R NPDA-43R	BNJMP43T BNJMP43B BNJMP43T BNJMP43L BNJMP43R BNJMP43R
Event Detail Menu (イベント詳細メニュー)、代替 Event Detail Menu for BSC Line (イベント詳細メニュー (BSC 回線)) Event Detail Menu for BSC Line (イベント詳細メニュー (BSC 回線))、代替 Event Detail Menu for BSC Station (イベント詳細メニュー (BSC 端末)) Event Detail Menu for BSC Station (イベント詳細メニュー (BSC 端末))、代替	NPDA-43R NPDA-43R NPDA-43T NPDA-43R NPDA-43T	BNJMP432 BNJMP43R BNJMP434 BNJMP43R BNJMP434
Event Detail Menu for SDLC Line (イベント詳細メニュー (SDLC 回線)) Event Detail Menu for SDLC Line, alternate (イベント詳細メニュー (SDLC 回線))、代替 Event Detail Menu for SDLC Station (イベント詳細メニュー (SDLC 端末)) Event Detail Menu for SDLC Station (イベント詳細メニュー (SDLC 端末))、代替 Event Summary (イベント要約)	NPDA-43R NPDA-43T NPDA-43R NPDA-43T NPDA-42A	BNJMP43R BNJMP434 BNJMP43R BNJMP434 BNJMP42A
Event Summary (イベント要約) Event Summary (イベント要約) Glossary displays (用語集表示) HELP Menu (ヘルプ・メニュー) Hexadecimal Display of Error Record (エラー・レコードの 16 進数表示)	NPDA-42B NPDA-42C (表示画面多数) NPDA-44C NPDA-02B	BNJMP42B BNJMP42C BNJMPGLO BNJMP44C BNJMP02B
Line Analysis-Link Segment Level *n* (回線分析-リンク・セグメント・レベル n) Link Configuration (リンク構成) Link Configuration (リンク構成) Link Configuration (リンク構成)、代替	NPDA-24B NPDA-44A1 NPDA-44A2 NPDA-44A1	BNJMPLNA BNJMP441 BNJMP442 BNJMP443

表 18. ハードウェア・モニター・パネルのカラー・マップ (続き)

パネル名	パネル番号	カラー・マップ
Link Configuration Summary-Level Selection (リンク構成要約-レベル選択) Link Data for SNA Controller (リンク・データ (SNA コントローラー)) Link Problem Determination Aid (LPDA-1) Data (リンク問題判別援助機能 (LPDA-1) データ) Link Problem Determination Aid (LPDA-1) LDM Data(リンク問題判別援助機能 (LPDA-1) LDM データ)	NPDA-LSLS NPDA-23A NPDA-52A NPDA-52AL	BNJMPLSL BNJMP23A BNJMP52A BNJMP52L
(LPDA-2) Data Link Segment Level 1 (データ・リンク・セグメント・レベル 1) (LPDA-2) Data Link Segment Level 1 (データ・リンク・セグメント・レベル 1、代替) (LPDA) Data Link Segment Level 2 (データ・リンク・セグメント・レベル 2)	NPDA-52B NPDA-52B NPDA-52C	BNJMP52B BNJMP522 BNJMP52B
Link Status and Test Results (リンク状況とテスト結果) Link Status and Test Results for LDM (リンク状況とテスト結果 (LDM)) LPDA-1 Command Menu (LPDA-1 コマンド・メニュー) LPDA-2 Command Menu (LPDA-2 コマンド・メニュー) Menu (メニュー)	NPDA-24A NPDA-24AL NPDA-LPDA1 NPDA-LPDA2 NPDA-01A	BNJMP24A BNJMP24L BNJMPLP1 BNJMPLP2 BNJMP01A
Modem and Line Status (モデムと回線状況) Modem and Line Parameters (モデムと回線パラメーター) Link Segment Level *n* (リンク・セグメント・レベル n)	NPDA-22B、ページ 1	BNJMPML1
Modem and Line Status (モデムと回線状況) Remote Modem Interface-Remote Device Status-Link Segment Level *n* (リモート・モデム・インターフェース-リモート装置状況-リンク・セグメント・レベル n)	NPDA-22B、ページ 2	BNJMPML2
Modem and Line Status (モデムと回線状況) Configuration Summary (構成要約)、 Link Segment Level *n* (リンク・セグメント・レベル n)	NPDA-22B、ページ 3	BNJMPML3
Most Recent Events (最新イベント) Most Recent Statistical Data (最新統計データ) Most Recent Statistical Data (最新統計データ) Most Recent Statistical Data (最新統計データ) Most Recent Statistical Data (最新統計データ) Most Recent Statistical Data (最新統計データ)	NPDA-41A NPDA-51E NPDA-51F NPDA-51G NPDA-51H NPDA-51I	BNJMP41A BNJMP51E BNJMP51F BNJMP51G BNJMP51H BNJMP51I
Most Recent Statistical Data (最新統計データ) Most Recent Statistical Data for Printer (最新統計データ (プリンター)) Most Recent Statistical Data for Tape (最新統計データ (テープ)) Most Recent Traffic Statistics (最新トラフィック統計) Most Recent Traffic Statistics for BSC/SS Station (最新トラフィック統計 (BSC/SS 端末))	NPDA-51B NPDA-51D NPDA-51C NPDA-51A NPDA-51A	BNJMP51B BNJMP51B BNJMP51B BNJMP51A BNJMP51A
Most Recent Traffic Statistics for BSC STA. w/LPDA (最新トラフィック統計 (BSC STA. w/LPDA)) Most Recent Traffic Statistics for Channel Attached STA (最新トラフィック統計 (チャネル接続 STA)) Most Recent Traffic Statistics for Local CTRL (最新トラフィック統計 (ローカル CTRL))	NPDA-51A NPDA-51A NPDA-51A	BNJMP51A BNJMP51A BNJMP51A

表 18. ハードウェア・モニター・パネルのカラー・マップ (続き)

パネル名	パネル番号	カラー・マップ
Most Recent Traffic Statistics for SDLC (最新トラフィック統計 (SDLC)) Station (最新トラフィック統計 (SDLC 端末)) Most Recent Traffic Statistics for SDLC STA. w/LPDA (最新トラフィック統計 (SDLC STA. w/LPDA)) Multiple Entries for Selected Resource (選択リソースの複数項目) Overwrite Map (上書きマップ)	NPDA-51A NPDA-51A NPDA-70A (すべての表示画面)	BNJMP51A BNJMP51A BNJMP70A BNJOVERW
Recommended Action for Selected Event (選択イベントの推奨アクション) Recording and Viewing Filter Status (フィルター状況の記録および表示) Release Level for SNA Controller (リリース・レベル (SNA コントローラー)) Remote DTE Interface Status (リモート DTE インターフェース状況) Remote DTE Interface Status for LDM (リモート DTE インターフェース状況 (LDM))	NPDA-BNIxxxyyy NPDA-20A,20B NPDA-21A NPDA-25A NPDA-25AL	BNJMP45A BNJMP20A BNJMP21A BNJMP25A BNJMP25A
Remote Self-Test Results (リモート自己検査結果) Remote Self-Test Results for LDM (リモート自己検査結果 (LDM)) Reported Resource Hardware (報告されたリソース・ハードウェア) Reported Resource Software Product (報告されたリソース・ソフトウェア・プロダクト) Screen Control/Help (画面制御/ヘルプ) Screen Control/Help (画面制御/ヘルプ)	NPDA-22A NPDA-22AL NPDA-44B NPDA-44B NPDA-02A、ページ 1 NPDA-02A、ページ 2	BNJMP22A BNJMP22L BNJMP44B BNJMP4BS BNJMP2A1 BNJMP2A2
Sender Hardware Product ID (送信側ハードウェア・プロダクト ID) Sender Software Product ID (送信側ソフトウェア・プロダクト ID) Statistical Counter Detail Display (統計カウンター詳細表示)、 ページ 1 Statistical Counter Detail Display (統計カウンター詳細表示)、 ページ *n* Statistical Detail (統計詳細) Statistical Detail (統計詳細) Statistical Detail Display for Ethernet (統計詳細表示 (イーサネット)) Statistical Detail Menu (統計詳細メニュー)	NPDA-44B NPDA-44B NPDA-54D NPDA-54D NPDA-53E NPDA-53F NPDA-53KA NPDA-53R	BNJMP4BH BNJMP4BS BNJMP541 BNJMP54N BNJMP53E BNJMP53F BNJMP53K BNJMP43R
Statistical Detail (統計詳細) Menu for BSC (メニュー (BSC)) Statistical Detail Menu for SDLC (統計詳細メニュー (SDLC)) TEST Information Display (TEST 情報表示) Total Events (合計イベント) Total Statistical Data (合計統計データ)	NPDA-53R NPDA-53R NPDA-02D NPDA-40A NPDA-50A	BNJMP43R BNJMP43R BNJMP02D BNJMP40A BNJMP50A
Transmit Receive Test-Link Segment Level *n* (送受信テスト-リンク・セグメント・レベル n) Upstream Member of Token-Ring Fault Domain (トークンリングの障害領域のアップストリーム・メンバー)	NPDA-25B NPDA-44B	BNJMPTRT BNJMP4BH

付録 B. NetView マクロと制御ブロック

この付録で説明されているマクロおよび制御ブロックは、ユーザーのプログラミング・インターフェースとして、NetView プログラムが提供するものです。

重要: この付録で説明されていない NetView マクロは、プログラミング・インターフェースとして使用しないでください。

汎用プログラミング・インターフェース制御ブロックおよび組み込みファイル

以下の制御ブロックおよび組み込みファイルは、汎用プログラミング・インターフェースとして提供されています。

名前	用途
DSIBC	NetView ブリッジ HLL C 組み込みファイル
DSIBCCALL	NetView ブリッジ HLL C サービス・ルーチンの定義
DSIBCCNM	NetView ブリッジ HLL C 戻りコード
DSIBCHLB	NetView ブリッジ HLL C DSIHLB マッピング
DSIBPCNM	NetView ブリッジ HLL PL/I 戻りコード
DSIBPHLB	NetView ブリッジ HLL PL/I DSIHLB マッピング
DSIBPHLS	NetView ブリッジ HLL PL/I サービス・ルーチンの定義
DSIBPLI	NetView ブリッジ HLL PL/I 組み込みファイル
DSIC	主 HLL C 組み込みファイル
DSICCALL	HLL C サービス・ルーチン定義
DSICCNM	HLL C 戻りコード
DSICCONS	HLL C 定数
DSICHLB	HLL C DSIHLB マッピング
DSICORIG	HLL C 起点ブロック・マッピング
DSICPRM	HLL C NetView ブリッジ・パラメーター・ブロック
DSICVARC	HLL C 可変長文字ストリング
DSIPCNM	HLL PL/I 戻りコード
DSIPCONS	HLL PL/I 定数
DSIPHLB	HLL PL/I DSIHLB マッピング
DSIPHLLS	HLL サービス・ルーチンの PL/I 定義
DSIPLI	主 HLL PL/I 組み込みファイル
DSIPORIG	HLL PL/I 起点ブロック・マッピング
DSIPPRM	HLL PL/I NetView ブリッジ・パラメーター・ブロック
EKG1ACCB	PL/I RODM アクセス・ブロック
EKG1ENTB	PL/I RODM エンティティーのアクセス情報ブロック
EKG1FLDB	PL/I RODM フィールドのアクセス情報ブロック
EKG1IADT	PL/I 要約データ・タイプ
EKG1IEEP	PL/I 外部入り口点宣言
EKG1IINC	PL/I 組み込みステートメント
EKG1LOGT	PL/I ログのレコード・タイプ定義
EKG1TRAB	PL/I RODM トランザクション情報ブロック
EKG11100	EKG_ConnectLong の PL/I 機能ブロック
EKG11101	EKG_Connect の PL/I 機能ブロック
EKG11102	EKG_Disconnect の PL/I 機能ブロック
EKG11201	EKG_Checkpoint の PL/I 機能ブロック

名前	用途
EKG11202	EKG_Stop の PL/I 機能ブロック
EKG11302	EKG_CreateClass の PL/I 機能ブロック
EKG11303	EKG_DeleteClass の PL/I 機能ブロック
EKG11304	EKG_CreateField の PL/I 機能ブロック
EKG11305	EKG_DeleteField の PL/I 機能ブロック
EKG11306	EKG_CreateSubfield の PL/I 機能ブロック
EKG11307	EKG_DeleteSubfield の PL/I 機能ブロック
EKG11401	EKG_ChangeField の PL/I 機能ブロック
EKG11402	EKG_SwapField の PL/I 機能ブロック
EKG11403	EKG_ChangeSubfield の PL/I 機能ブロック
EKG11404	EKG_SwapSubfield の PL/I 機能ブロック
EKG11405	EKG_LinkTrigger の PL/I 機能ブロック
EKG11406	EKG_LinkNoTrigger の PL/I 機能ブロック
EKG11407	EKG_UnLinkTrigger の PL/I 機能ブロック
EKG11408	EKG_UnLinkNoTrigger の PL/I 機能ブロック
EKG11409	EKG_CreateObject の PL/I 機能ブロック
EKG11410	EKG_DeleteObject の PL/I 機能ブロック
EKG11411	EKG_RevertToInherited の PL/I 機能ブロック
EKG11412	EKG_AddNotifySubscription の PL/I 機能ブロック
EKG11413	EKG_DeleteNotifySubscription の PL/I 機能ブロック
EKG11415	EKG_TriggerNamedMethod の PL/I 機能ブロック
EKG11416	EKG_TriggerOIMethod の PL/I 機能ブロック
EKG11417	オブジェクト削除通知予約の PL/I 追加
EKG11418	オブジェクト削除通知予約の PL/I 削除
EKG11501	EKG_QueryField の PL/I 機能ブロック
EKG11502	EKG_QuerySubfield の PL/I 機能ブロック
EKG11503	EKG_QueryEntityStructure の PL/I 機能ブロック
EKG11504	EKG_QueryFieldStructure の PL/I 機能ブロック
EKG11505	EKG_QueryFieldID の PL/I 機能ブロック
EKG11506	EKG_QueryFieldName の PL/I 機能ブロック
EKG11507	EKG_QueryNotifyQueue の PL/I 機能ブロック
EKG11508	PL/I 複数サブフィールドの照会
EKG11509	PL/I 位置付け
EKG11510	EKG_QueryResponseBlockOverflow の PL/I 機能ブロック
EKG11600	EKG_ExecuteFunctionList の PL/I 機能ブロック
EKG12001	EKG_QueryFunctionBlockContents の PL/I 機能ブロック
EKG12002	EKG_LockObjectList の PL/I 機能ブロック
EKG12003	EKG_UnlockAll の PL/I 機能ブロック
EKG12004	EKG_ResponseBlock の PL/I 機能ブロック
EKG12005	EKG_SendNotification の PL/I 機能ブロック
EKG12006	EKG_SetReturnCode の PL/I 機能ブロック
EKG12007	EKG_WhereAmI の PL/I 機能ブロック
EKG12008	EKG_OutputToLog の PL/I 機能ブロック
EKG12009	EKG_MessageTriggeredAction の PL/I 機能ブロック
EKG12011	EKG_QueryObjectName の PL/I 機能ブロック
EKG21415	EKG_TriggerNamedMethod の PL/I 応答ブロック
EKG21416	EKG_TriggerOIMethod の PL/I 応答ブロック
EKG21501	EKG_QueryField の PL/I 応答ブロック
EKG21502	EKG_QuerySubfield の PL/I 応答ブロック
EKG21503	EKG_QueryEntityStructure の PL/I 応答ブロック
EKG21504	EKG_QueryFieldStructure の PL/I 応答ブロック
EKG21505	EKG_QueryFieldID の PL/I 応答ブロック
EKG21506	EKG_QueryFieldName の PL/I 応答ブロック
EKG21507	EKG_QueryNotifyQueue の PL/I 応答ブロック

名前	用途
EKG21508	PL/I 複数サブフィールドの照会
EKG21509	PL/I 位置付け
EKG21510	EKG_QueryResponseBlockOverflow の PL/I 応答ブロック
EKG22001	EKG_QueryFunctionBlockContents の PL/I 応答ブロック
EKG22007	EKG_WhereAmI の PL/I 応答ブロック
EKG22011	EKG_QueryObjectName の PL/I 応答ブロック
EKG3ACCB	C/370™ RODM アクセス・ブロック
EKG3CADT	C/370 RODM 要約データ・タイプ
EKG3CEEP	C/370 外部入り口点宣言
EKG3CINC	C/370 組み込みステートメント
EKG3CLOG	C/370 ログ・レコード定義
EKG3ENTB	C/370 RODM エンティティーのアクセス情報ブロック
EKG3FLDB	C/370 RODM フィールドのアクセス情報ブロック
EKG3TRAB	C/370 RODM トランザクション情報ブロック
EKG31100	EKG_ConnectLong の C/370 機能ブロック
EKG31101	EKG_Connect の C/370 機能ブロック
EKG31102	EKG_Disconnect の C/370 機能ブロック
EKG31201	EKG_Checkpoint の C/370 機能ブロック
EKG31202	EKG_Stop の C/370 機能ブロック
EKG31302	EKG_CreateClass の C/370 機能ブロック
EKG31303	EKG_DeleteClass の C/370 機能ブロック
EKG31304	EKG_CreateField の C/370 機能ブロック
EKG31305	EKG_DeleteField の C/370 機能ブロック
EKG31306	EKG_CreateSubfield の C/370 機能ブロック
EKG31307	EKG_DeleteSubfield の C/370 機能ブロック
EKG31401	EKG_ChangeField の C/370 機能ブロック
EKG31402	EKG_SwapField の C/370 機能ブロック
EKG31403	EKG_ChangeSubfield の C/370 機能ブロック
EKG31404	EKG_SwapSubfield の C/370 機能ブロック
EKG31405	EKG_LinkTrigger の C/370 機能ブロック
EKG31406	EKG_LinkNoTrigger の C/370 機能ブロック
EKG31407	EKG_UnLinkTrigger の C/370 機能ブロック
EKG31408	EKG_UnLinkNoTrigger の C/370 機能ブロック
EKG31409	EKG_CreateObject の C/370 機能ブロック
EKG31410	EKG_DeleteObject の C/370 機能ブロック
EKG31411	EKG_RevertToInherited の C/370 機能ブロック
EKG31412	EKG_AddNotifySubscription の C/370 機能ブロック
EKG31413	EKG_DeleteNotifySubscription の C/370 機能ブロック
EKG31415	EKG_TriggerNamedMethod の C/370 機能ブロック
EKG31416	EKG_TriggerOIMethod の C/370 機能ブロック
EKG31417	オブジェクト削除通知予約の C/370 追加
EKG31418	オブジェクト削除通知予約の C/370 削除
EKG31501	EKG_QueryField の C/370 機能ブロック
EKG31502	EKG_QuerySubfield の C/370 機能ブロック
EKG31503	EKG_QueryEntityStructure の C/370 機能ブロック
EKG31504	EKG_QueryFieldStructure の C/370 機能ブロック
EKG31505	EKG_QueryFieldID の C/370 機能ブロック
EKG31506	EKG_QueryFieldName の C/370 機能ブロック
EKG31507	EKG_QueryNotifyQueue の C/370 機能ブロック
EKG31508	C/370 複数サブフィールドの照会
EKG31509	C/370 位置付け
EKG31510	EKG_QueryResponseBlockOverflow の C/370 機能ブロック
EKG31600	EKG_ExecuteFunctionList の C/370 機能ブロック
EKG32001	EKG_QueryFunctionBlockContents の C/370 機能ブロック

名前	用途
EKG32002	EKG_LockObjectList の C/370 機能ブロック
EKG32003	EKG_UnlockAll の C/370 機能ブロック
EKG32004	EKG_ResponseBlock の C/370 機能ブロック
EKG32005	EKG_SendNotification の C/370 機能ブロック
EKG32006	EKG_SetReturnCode の C/370 機能ブロック
EKG32007	EKG_WhereAmI の C/370 機能ブロック
EKG32008	EKG_OutputToLog の C/370 機能ブロック
EKG32009	EKG_MessageTriggeredAction の C/370 機能ブロック
EKG32011	EKG_QueryObjectName の C/370 機能ブロック
EKG41415	EKG_TriggerNamedMethod の C/370 応答ブロック
EKG41416	EKG_TriggerOIMethod の C/370 応答ブロック
EKG41501	EKG_QueryField の C/370 応答ブロック
EKG41502	EKG_QuerySubfield の C/370 応答ブロック
EKG41503	EKG_QueryEntityStructure の C/370 応答ブロック
EKG41504	EKG_QueryFieldStructure の C/370 応答ブロック
EKG41505	EKG_QueryFieldID の C/370 応答ブロック
EKG41506	EKG_QueryFieldName の C/370 応答ブロック
EKG41507	EKG_QueryNotifyQueue の C/370 応答ブロック
EKG41508	C/370 複数サブフィールドの照会
EKG41509	C/370 位置付け
EKG41510	EKG_QueryResponseBlockOverflow の C/370 応答ブロック
EKG42001	EKG_QueryFunctionBlockContents の C/370 応答ブロック
EKG42007	EKG_WhereAmI の C/370 応答ブロック
EKG42011	EKG_QueryObjectName の C/370 応答ブロック
FLBTREM	C/370 例外ビュー更新パラメーター構造
FLBTRSM	C/370 状況変更パラメーター構造

以下のマクロは、汎用プログラミング・インターフェースとして提供されています。

名前	用途
CNMALTDATA	キュー上のデータ変更
CNMAUTOTAB	自動化テーブルの呼び出し
CNMCLOSMEM	NetView 区分データ・セットのクローズ
CNMCODE2TXT	コード・ポイントの変換
CNMCOMMAND	NetView コマンドの呼び出し
CNMCOPYSTR	ストレージのコピー
CNMETINIT	サーバー・サポートの初期設定
CNMETNEXT	次のトランザクション要求の入手
CNMETQUIESCE	データベースの静止
CNMETREADY	次のトランザクションの準備
CNMETRPARM	トランザクション要求の入手
CNMETTERM	サーバー・サポートの終了
CNMETWAIT	トランザクション要求の待機
CNMGETATTR	照会メッセージ属性
CNMGETDATA	データ・キューの操作
CNMGETPARM	トランザクション応答パラメーターの入手
CNMHREGIST	高性能移送アプリケーション登録
CNMHSENDMU	高性能移送メッセージ単位の送信
CNMI	DST のもとでの CNMI アクセス
CNMINFOC	NetView の文字情報の照会
CNMINFOI	NetView の整数情報の照会
CNMKEYIO	DST のもとでのキー付きファイル・アクセス

名前	用途
CNMLOCK	ロック制御
CNMNAMESTR	指定されたストレージ
CNMOPENMEM	NetView 区分データ・セットのオープン
CNMOPREP	リソース・オブジェクト・データ・マネージャー
CNMPRSMDB	メッセージ・データ・ブロックの処理
CNMREADMEM	NetView 区分データ・セットの読み込み
CNMREGIST	アプリケーション登録
CNMSCOPECK	セキュリティーのコマンド許可検査
CNMSENDMSG	メッセージまたはコマンドの送信
CNMSENDMU	メッセージ単位の送信
CNMSENDSTR	NetView の要求側へのトランザクション応答の送信
CNMSENDTR	データベース・サーバーへのトランザクション要求の送信
CNMSSCAN	文字ストリングの解析または変換
CNMSTRCELL	記憶セル
CNMSTRPOOL	ストレージ・プール
CNMVARPOOL	変数のセットまたは検索
DUIFEDST	アセンブラー・マクロ

プロダクト・センシティブ・プログラミング・インターフェース

以下の制御ブロックは、プロダクト・センシティブ・プログラミング・インターフェースとして提供されています。

名前	用途
AAUTISAW	内部セッション認識レコード
AAUTLOGR	NetView SMF ログ・レコードの構造マップ
BNJTBRF	バッチ・レコード・フォーマット・テーブル
DSIAIFRO	自動操作内部機能要求オブジェクトの拡張ベクトル
DSIASYPN	非同期パネル・パラメーター・リスト
DSICBH	制御ブロック・ヘッダー
DSICWB	コマンド作業ブロック
DSIDSB	データ・サービス・ブロック
DSIDSRB	データ・サービス要求ブロック
DSIDTR	データ・トランスポート要求ブロック
DSIELB	外部ログ・ブロック
DSIID	NetView レベル ID
DSIIFR	内部機能要求
DSILOGDS	NetView ログ DSECT
DSIMVT	主ベクトル・テーブル
DSIPDB	コマンド解析記述ブロック
DSISCE	システム・コマンド項目
DSISCT	システム・コマンド・テーブル (組み込みのみ)
DSISVL	サービス・ルーチン・ベクトル・リスト (組み込みのみ)
DSISWB	サービス作業ブロック
DSITECBR	ECB プロセッサー・ロード・モジュールのブランチ・テーブル
DSITIB	タスク情報ブロック
DSITVB	タスク・ベクトル・ブロック
DSIUSE	インストール・システム出口パラメーター・リスト

以下のマクロは、プロダクト・センシティブ・プログラミング・インターフェースとして提供されています。

名前	用途
DSIAUTO	自動操作サービス
DSIBAM	自動操作メッセージの作成
DSIBAMKW	自動操作メッセージ・キーワードの作成
DSICBS	制御ブロック・サービス
DSICES	コマンド項目サービス
DSICVTHE	16 進数への変換
DSIC2T	アラート・コード・ポイントのテキストへの変換
DSIDATIM	日付および時刻
DSIDEL	ユーザー定義モジュールの削除
DSIDKS	ディスク・サービス
DSIFIND	長期実行コマンド・ストレージの検出
DSIFRE	ストレージの解放
DSIFREBS	バッファー構造サービスの解放
DSIGET	ストレージの獲得
DSIGETDS	メッセージの検索
DSIHREGS	高性能登録
DSIHSNDS	高性能送信
DSIKVS	キーワード/値サービス
DSILCS	獲得/解放制御ブロック
DSILOD	ユーザー定義モジュールのロード
DSIMBS	メッセージ・バッファー・サービス
DSIMDS	メッセージ定義サービス
DSIMMDBS	メッセージ・データ・ブロック・サービス
DSIMQS	メッセージ・キューイング・サービス
DSINOR	リソース・オブジェクト・データ・マネージャー
DSIPAS	パラメーター/別名サービス
DSIPOP	長期実行コマンドの除去
DSIPOS	ECB 通知サービス
DSIPRS	解析サービス
DSIPSS	表示サービス
DSIPUSH	長期実行コマンドの確立
DSIQOS	オペレーター・サービスの照会
DSIQRS	リソース・サービスの照会
DSIRDS	リソース定義サービス
DSIRXCOM	REXX 変数アクセス (VM のみ)
DSIRXEBS	EVALBLOK の獲得
DSISRCMV	サブベクトル/サブフィールドの探索
DSISYS	オペレーティング・システム・インディケーター
DSITECBS	DST の動的 ECB リストの管理
DSIVARS	グローバル変数アクセス
DSIWAT	ECB 待機サービス
DSIWCS	コンソール書き出しサービス
DSIWLS	ログ書き込みサービス
DSIZCSMS	CNM データ・サービス
DSIZVSMS	VSAM データ・サービス
DSI6REGS	登録サービス
DSI6SNDS	送信サービス

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

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。

〒103-8510
東京都中央区日本橋箱崎町19番21号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

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。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。お客様は、IBM のアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、 IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次のように、著作権表示を入れていただく必要があります。

© (お客様の会社名) (西暦年). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。 © Copyright IBM Corp. _年を入れる_. All rights reserved.

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 ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corporation の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

Adobe は、Adobe Systems Incorporated の米国およびその他の国における登録商標または商標です。

Java およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。

Linux は、Linus Torvalds の米国およびその他の国における登録商標です。

Microsoft および Windows 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。

プライバシー・ポリシーに関する考慮事項

サービス・ソリューションとしてのソフトウェアも含めた IBM ソフトウェア製品 (「ソフトウェア・オファリング」) では、製品の使用に関する情報の収集、エンド・ユーザーの使用感の向上、エンド・ユーザーとの対話またはその他の目的のために、Cookie はじめさまざまなテクノロジーを使用することがあります。多くの場合、ソフトウェア・オファリングにより個人情報が収集されることはありません。IBM の「ソフトウェア・オファリング」の一部には、個人情報を収集できる機能を持つものがあります。ご使用の「ソフトウェア・オファリング」が、これらの Cookie およびそれに類するテクノロジーを通じてお客様による個人情報の収集を可能にする場合、以下の具体的事項を確認ください。

この「ソフトウェア・オファリング」は、Cookie もしくはその他のテクノロジーを使用して個人情報を収集することはありません。

この「ソフトウェア・オファリング」が Cookie およびさまざまなテクノロジーを使用してエンド・ユーザーから個人を特定できる情報を収集する機能を提供する場合、お客様は、このような情報を収集するにあたって適用される法律、ガイドライン等を遵守する必要があります。これには、エンドユーザーへの通知や同意の要求も含まれますがそれらには限られません。

このような目的での Cookie を含む様々なテクノロジーの使用の詳細については、IBM の『IBM オンラインでのプライバシー・ステートメント』(http://www.ibm.com/privacy/details/jp/ja/) の『クッキー、ウェブ・ビーコン、その他のテクノロジー』および『IBM Software Products and Software-as-a-Service Privacy Statement』(http://www.ibm.com/software/info/product-privacy) を参照してください。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[ナ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

[数字]

A

B

C

D

E

G

H

I

L

M

N

O

P

R

S

T

U

V

W

X

Y

［特殊文字］

Printed in Japan

SA88-4388-01

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21