Sun™ SNMP Management Agent 利用ガイド (Sun Fire™ B1600 用)

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No. 817-2501-10
2003 年 4 月 Revision A

コメントの宛先: docfeedback@sun.com

Copyright 2003 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

米国 Sun Microsystems, Inc. (以下、米国 Sun Microsystems 社とします) は、本書に記述されている製品に採用されている技術に関する知的所有権を有しています。これら知的所有権には、http://www.sun.com/patents に掲載されているひとつまたは複数の米国特許、および米国ならびにその他の国におけるひとつまたは複数の特許または出願中の特許が含まれています。

本書およびそれに付属する製品は著作権法により保護されており、その使用、複製、頒布および逆コンパイルを制限するライセンスのもとにおいて頒布されます。サン・マイクロシステムズ株式会社の書面による事前の許可なく、本製品および本書のいかなる部分も、いかなる方法によっても複製することが禁じられます。

本製品のフォント技術を含む第三者のソフトウェアは、著作権法により保護されており、提供者からライセンスを受けているものです。

本製品の一部は、カリフォルニア大学からライセンスされている Berkeley BSD システムに基づいていることがあります。UNIX は、X/Open Company Limited が独占的にライセンスしている米国ならびに他の国における登録商標です。

本製品は、株式会社モリサワからライセンス供与されたリュウミン L-KL (Ryumin-Light) および中ゴシック BBB (GothicBBB-Medium) のフォント・データを含んでいます。

本製品に含まれる HG 明朝 L と HG ゴシック B は、株式会社リコーがリョービイマジクス株式会社からライセンス供与されたタイプフェースマスタをもとに作成されたものです。平成明朝体 W3 は、株式会社リコーが財団法人 日本規格協会 文字フォント開発・普及センターからライセンス供与されたタイプフェースマスタをもとに作成されたものです。また、HG 明朝 L と HG ゴシック B の補助漢字部分は、平成明朝体W3 の補助漢字を使用しています。なお、フォントとして無断複製することは禁止されています。

Sun、Sun Microsystems、AnswerBook2、docs.sun.com、Sun Fire、Java は、米国およびその他の国における米国 Sun Microsystems 社の商標もしくは登録商標です。サンのロゴマークおよび Solaris は、米国 Sun Microsystems 社の登録商標です。

すべての SPARC 商標は、米国 SPARC International, Inc. のライセンスを受けて使用している同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。SPARC 商標が付いた製品は、米国 Sun Microsystems 社が開発したアーキテクチャーに基づくものです。

OPENLOOK、OpenBoot、JLE は、サン・マイクロシステムズ株式会社の登録商標です。

ATOK は、株式会社ジャストシステムの登録商標です。ATOK8 は、株式会社ジャストシステムの著作物であり、ATOK8 にかかる著作権その他の権利は、すべて株式会社ジャストシステムに帰属します。ATOK Server/ATOK12 は、株式会社ジャストシステムの著作物であり、ATOK Server/ATOK12 にかかる著作権その他の権利は、株式会社ジャストシステムおよび各権利者に帰属します。

本書で参照されている製品やサービスに関しては、該当する会社または組織に直接お問い合わせください。

OPEN LOOK および Sun Graphical User Interface は、米国 Sun Microsystems 社が自社のユーザーおよびライセンス実施権者向けに開発しました。米国 Sun Microsystems 社は、コンピュータ産業用のビジュアルまたは グラフィカル・ユーザーインタフェースの概念の研究開発における米国 Xerox 社の先駆者としての成果を認めるものです。米国 Sun Microsystems 社は米国 Xerox 社から Xerox Graphical User Interface の非独占的ライセンスを取得しており、このライセンスは米国 Sun Microsystems 社のライセンス実施権者にも適用されます。

Use, duplication, or disclosure by the U.S. Government is subject to restrictions set forth in the Sun Microsystems, Inc. license agreements and as provided in DFARS 227.7202-1(a) and 227.7202-3(a) (1995), DFARS 252.227-7013(c)(1)(ii) (Oct. 1998), FAR 12.212(a) (1995), FAR 52.227-19, or FAR 52.227-14 (ALT III), as applicable.

本書は、「現状のまま」をベースとして提供され、商品性、特定目的への適合性または第三者の権利の非侵害の黙示の保証を含みそれに限定されない、明示的であるか黙示的であるかを問わない、なんらの保証も行われないものとします。

本書には、技術的な誤りまたは誤植のある可能性があります。また、本書に記載された情報には、定期的に変更が行われ、かかる変更は本書の最新版に反映されます。さらに、米国サンまたは日本サンは、本書に記載された製品またはプログラムを、予告なく改良または変更することがあります。

本製品が、外国為替および外国貿易管理法 (外為法) に定められる戦略物資等 (貨物または役務) に該当する場合、本製品を輸出または日本国外へ持ち出す際には、サン・マイクロシステムズ株式会社の事前の書面による承諾を得ることのほか、外為法および関連法規に基づく輸出手続き、また場合によっては、米国商務省または米国所轄官庁の許可を得ることが必要です。

原典: *Sun™ SNMP Management Agent Guide for the Sun Fire™ B1600*
Part No: 817-1010-10
Revision A

Adobe PostScript

目次

図目次

表目次

コード例

はじめに

このマニュアルでは、SNMP (簡易ネットワーク管理プロトコル) を使用したプラットフォームハードウェアの管理を支援する、Sun Fire B1600 プラットフォーム用 Sun SNMP Management Agent について説明します。

Management Agent では、登録された資源の監視、構成、環境および障害の報告を実行できます。サービスインジケータの制御と監視、プロセッサブレードの電源、スタンバイ、リセットの制御と監視も行えます。

このマニュアルでは、対象読者として、経験を積んだエンタープライズ管理者および専門の開発者を想定しています。

このマニュアルは 2 部構成になっています。

- 第 1 部 (第 1 章から第 8 章) では、SNMP Management Agent の概要を示し、このソフトウェアの機能を説明します。
- 第 2 部 (第 9 章から第 13 章) では、このソフトウェアをインストールして構成する方法を説明します。

マニュアルの構成

このマニュアルは以下の章で構成されています。

第 **1** 部

第 1 章では、Sun SNMP Management Agent ソフトウェアのコンポーネントについて説明します。

第 2 章では、SNMP の基本機能を簡単に説明します。

第 3 章では、SNMPv3 マスターエージェントの機能と特徴について説明します。

第 4 章では、Sun Fire B1600 の SNMP モデルについて概要を説明します。

第 5 章では、Sun Fire B1600 の管理対象オブジェクトとそれらのオブジェクト間の関係が、SNMP インタフェースでどのように表されるかについて説明します。

第 6 章では、sunPlat の物理クラス階層について説明し、sunPlat モデルで定義された管理対象の物理オブジェクトクラスが SUN-PLATFORM-MIB にどのように記述されているかを示します。

第 7 章では、sunPlat の論理クラス階層について説明し、sunPlat モデルで定義された管理対象オブジェクトクラスが SUN-PLATFORM-MIB にどのように記述されているかを示します。

第 8 章では、SUN-PLATFORM-MIB に定義されている SunPlat 通知クラスおよび属性について説明します。

第 **2** 部

第 9 章では、Sun Fire B1600 用の管理ソフトウェアを構成するコンポーネントについて説明し、SNMP ソフトウェアをインストールする前に実行する必要があるシステムチェックのリストを示します。

第 10 章では、Sun Fire B1600 への管理ソフトウェアのインストール方法を説明します。

第 11 章では、ユーザーが構成を変更できるファイルに関する情報を示します。

第 12 章では、インストール後のデフォルト構成、および構成ファイルを変更する方法について説明します。

第 13 章では、ソフトウェアをアンインストールする方法について説明します。

第 14 章では、ソフトウェアの障害追跡に役立つ情報を示します。

付録 **A** では、J2RE を J2SE と共存するようにインストールする方法、インストールした J2RE を検出するように起動スクリプトを変更する方法について説明します。

書体と記号について

書体または記号	意味	例
AaBbCc123	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コード例。	.login ファイルを編集します。 ls -a を実行します。 % You have mail.
AaBbCc123	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して表します。	**マシン名**% **su** Password:
AaBbCc123 または**ゴシック**	コマンド行の可変部分。実際の名前や値と置き換えてください。	rm *filename* と入力します。 rm **ファイル名** と入力します。
『 』	参照する書名を示します。	『Solaris ユーザーマニュアル』
「 」	参照する章、節、または、強調する語を示します。	第 6 章「データの管理」を参照。 この操作ができるのは「スーパーユーザー」だけです。
\	枠で囲まれたコード例で、テキストがページ行幅をこえる場合に、継続を示します。	% **grep '^#define ** **XV_VERSION_STRING'**

シェルプロンプトについて

シェル	プロンプト
UNIX の C シェル	**マシン名**%
UNIX の Bourne シェルと Korn シェル	$
スーパーユーザー (シェルの種類を問わない)	#

関連マニュアル

用途	タイトル	Part No.
SunMC	『SunMC 3.0 Sun Fire B1600 のための追補マニュアル』	817-2538
リリース ノート	『SNMP Release Notes for the Sun Fire B1600』	817-1006
Sun Fire B1600 プラットフォーム	『Sun Fire B1600 ブレードシステムシャーシハードウェア設置マニュアル』	817-1904
	『Sun Fire B1600 ブレードシステムシャーシソフトウェア設定マニュアル』	817-1888
	『Sun Fire B1600 ブレードシステムシャーシ管理マニュアル』	817-1898
	『Sun Fire B1600 ブレードシステムシャーシスイッチ管理マニュアル』	817-1893

Sun のオンラインマニュアル

各言語対応版を含むサンの各種マニュアルは、次の URL から表示、印刷または購入できます。

`http://www.sun.com/documentation`

サンの技術サポート

この製品に関する技術的な質問については、このマニュアルでは扱っていません。以下の Web サイトにアクセスしてください。

`http://www.sun.com/service/contacting`

コメントをお寄せください

弊社では、マニュアルの改善に努力しており、お客様からのコメントおよびご忠告をお受けしております。コメントは下記宛に電子メールでお送りください。

docfeedback@sun.com

電子メールの表題にはマニュアルの Part No. (817-2501-10) を記載してください。

なお、現在日本語によるコメントには対応できませんので、英語で記述してください。

PART I

技術的な説明と機能

第1章

SNMP Management Agent 追加パッケージ

このリリースの Sun™ SNMP Management Agent は、Sun Fire™ B1600 シェルフと Sun Fire B100s ブレードの監視と制御の機能を提供します。

プラットフォームのタイプに応じて、以下のエージェントを採用できます。

- Sun Fire B100s ブレード上で動作するドメインエージェント (ドメインハードウェア監視)

  監視対象のサーバー上にソフトウェアをローカルにインストールします。監視できるのはそのサーバーだけです。Sun Fire B1600 の場合、各ブレードが個別に監視されます。

  Sun Fire B100s ブレードのドメインハードウェア監視の範囲は、ブレードのハードウェアのみに限定されます。シェルフの他のコンポーネント (サービスインジケータ、PSU、SSC など) とシェルフ自体の識別情報は対象外です。

- システムコントローラをプロキシとして使用するプラットフォームエージェント (プラットフォームハードウェア監視)

  遠隔 (プラットフォームエージェント) サーバーにソフトウェアをインストールします。遠隔サーバーはシステムコントローラを通してプラットフォームの計測機構にアクセスします。そのため、システムコントローラで管理するすべてのハードウェアを監視できます。

  Sun Fire B1600 でのプラットフォームハードウェア監視の範囲には、シェルフ、シェルフの識別情報、サービスインジケータ、およびすべての FRU (現場交換可能ユニット) が含まれます。さらに、ドメインハードウェア監視では対象外である Sun Fire B100s ブレードに関するハードウェア情報 (具体的には電圧監視) も対象になります。

  図 1-1 は、両方のタイプのハードウェア監視の例です。Sun Fire B1600 シェルフ A および B は、プラットフォームエージェントサーバーを経由してネットワーク管理ステーションに接続されています (プラットフォームハードウェア監視)。
  Sun Fire B1600 シェルフ C の場合は、Sun Fire B100s ブレードがネットワーク管理ステーションに直接接続されています (ドメインハードウェア監視)。

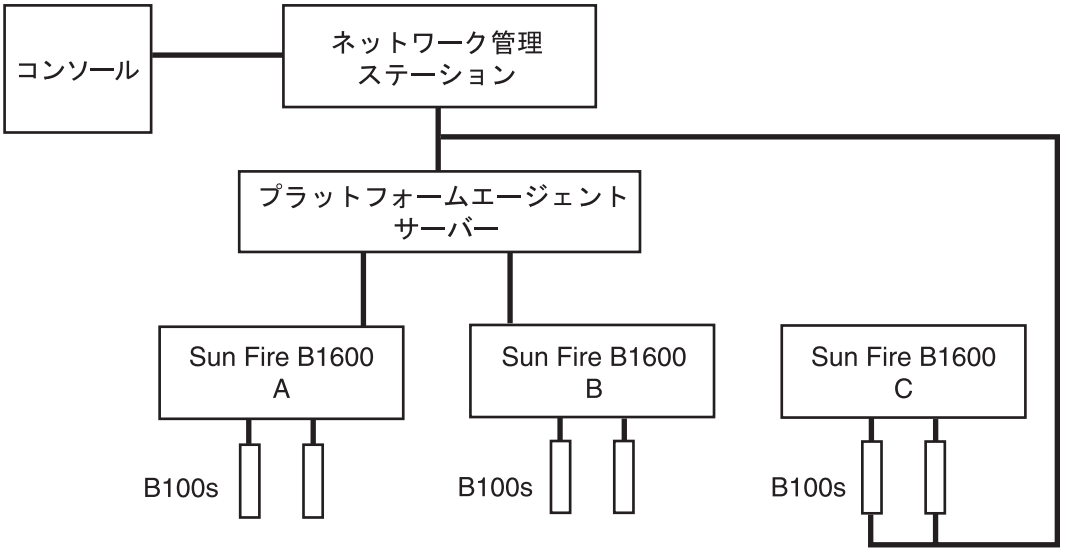

図 1-1　ドメインハードウェア監視とプラットフォームハードウェア監視の例

このソフトウェアは、以下の機能を備えた複数のパッケージで構成されます。

- SNMP サブエージェント

  デフォルトでは、SNMP サブエージェントは Solaris マスターエージェントである snmpdx のサブエージェントとして登録されます。このサブエージェントは「SNMP メディエータ」とも呼ばれます。

- SNMPv3 マスターエージェント

  SNMPv3 マスターエージェントは、プラットフォーム上で動作している SNMP メディエータにアクセスするための、セキュリティー保護された単一アクセスポイントを提供します。マスターエージェントは、`snmpdx` に要求を転送するプロキシとして機能します。

- Sun Fire B1600 および Sun Fire B100s の計測機構

  これらのパッケージは、ドメインベースとプラットフォームベースのいずれのハードウェア監視を採用するかによって、必要に応じてインストールします。

第2章

SNMP の概要

この章では、SNMP (簡易ネットワーク管理プロトコル) の基本機能を簡単に説明します。SNMP は、Sun Fire™ B1600 システム特有の問題に対応した完全なプロトコルです。

この章は以下の節で構成されています。

SNMP のバージョン

SNMP は、ネットワークに接続された装置 (システム) を管理するためのオープンなインターネット規格です。SNMP は他のインターネット規格と同様、IETF (インターネット特別技術調査委員会) が発行する複数の RFC (Requests for Comments) によって定義されています。

承認された規格を定義する 3 つのバージョンの SNMP があります。

- SNMPv1
- SNMPv2 (SNMPv2c とも呼ばれ、このマニュアルでは SNMPv2c と記載します)
- SNMPv3

1988 年に SNMPv1 が最初に定義されました。1993 年に定義された SNMPv2 では、SNMPv1 で不十分だった点を補うためにプロトコル操作とデータ型が追加され、セキュリティー機能が提供されました。セキュリティーモデルにおける制約により、新しいセキュリティーベースの機能を削除した規格が定義されました。これが現在 SNMPv2c として認められている規格です。SNMPv2usec および SNMPv2* と呼ばれる実験的なバージョンも同時期に開発されましたが、これらのバージョンは現在までのところ広範囲に受け入れられておらず、実験的な状態のままになっています。

1999 年に開発された SNMPv3 は、プラグ可能なコンポーネントをサポートする SNMP 管理フレームワークのセキュリティーを含む定義です。

これらの規格の詳細については、IETF の Web サイト (`http://www.ietf.org/rfc.html`) で次の RFC を参照してください。

- SNMPv1: RFC1155、RFC1157、RFC1212、RFC1215
- SNMPv2: RFC2578、RFC2579、RFC2580、RFC3416
- SNMPv3: RFC3410、RFC3411、RFC3412、RFC3413、RFC3414、RFC3415
- 規格間の共存: RFC2576

SNMP マネージャーと SNMP エージェント

SNMP は、ネットワーク管理ステーション (NMS) による装置の遠隔管理を可能にするネットワークプロトコルです。一般に、NMS は「マネージャー」と呼ばれます。

装置を管理するためには、装置に関連付けられた「SNMP エージェント」(「SNMP メディエータ」とも呼ばれる) が必要です。メディエータの目的は次のとおりです。

- 装置の状態を表すデータへの要求をマネージャーから受け取り、適切な応答を戻す。
- 装置状態の制御を有効にするデータをマネージャーから受け取る。
- 「SNMP トラップ」を生成する (SNMP トラップは、装置に関連する重要なイベントを通知するために、選択された 1 つ以上のマネージャーに送信される非請求メッセージ)。

SNMP 管理情報ベース

装置を管理および監視するには、エージェントとマネージャーの両方が理解できる形式で装置の特性を表現する必要があります。装置の特性は、ファン速度などの物理プロパティー、またはルーティングテーブルなどのサービスを表します。このような特性を定義するデータ構造は、「管理情報ベース (MIB)」と呼ばれます。このデータモデルは、通常はテーブル構造ですが、単純な値を格納することもできます。前者の例としてルーティングテーブルがあり、後者の例としては、エージェントの起動日時を示すタイムスタンプがあります。

MIB は、SNMP を使用してアクセス可能な仮想データストアの定義です。MIB の内容には、次のように `get` と `set` の操作を使用してマネージャーからアクセスできます。

- メディエータは、`get` 操作への応答としてデータを取得します。データはローカルに格納されているか、または管理対象の装置から直接取得されます。
- また、`set` 操作への応答として、エージェントは通常、エージェント自体または管理対象装置の状態に影響を及ぼす何らかの処理を実行します。

NMS がエージェントを介して装置を管理するためには、エージェントから示されるデータに対応した MIB をマネージャーに読み込む必要があります。これを行うメカニズムは、ネットワーク管理ソフトウェアの実装によって異なります。そのメカニズムによって、エージェントから示されたデータモデルを正しく識別して解釈するために必要な情報がマネージャーに提供されます。

注 – MIB から他の MIB にある定義を参照することが可能です。そのため、ある特定の MIB を使用するために他の MIB を読み込むことが必要な場合があります。

この仮想データストアの内容を識別するために、MIB は「オブジェクト識別子 (OID)」を使用して定義されます。OID は、一意の名前空間を定義する階層化された一連の整数で構成されます。割り振られた個々の整数には、テキスト名が関連付けられています。たとえば、`1.3.6.1` という OID には、対応する `iso.org.dod.internet` という OID 名があります。また、`1.3.6.1.4` は、OID 名 `iso.org.dod.internet.private` に対応します。

数値形式は SNMP プロトコルのトランザクションで使用され、一方、テキスト形式は読みやすさを助けるためにユーザーインタフェースで使用されます。このような OID で表されたオブジェクトは、通常、簡略形式としてオブジェクト名の最後の構成要素を使用して参照されます。この慣例から生じる混乱を避けるために、MIB に定義するすべてのオブジェクト名に対して、*sunPlat* のような MIB 固有の接頭辞を付けるのが一般的です。それによって、すべての OID がグローバルに一意になります。

注 – MIB は ASN.1 と呼ばれる言語を使用して定義します。この言語については、このマニュアルでは説明していません。参照情報としては、SNMPv2c に対応した MIB の構造を定義する SMI (Structure of Management Information) が RFC2578 で規定されています。SMI は、MIB で使用できる構文と基本データ型を定義します。RFC2579 で規定されたテキスト規則 (型定義) に、追加のデータ型と列挙が定義されています。

MIB のテーブル

MIB に定義されるデータ内容の大部分は表形式であり、一意の OID が割り振られた一連のオブジェクトからなるエントリとして編成されています。たとえば、ファンの特性を示すテーブルは複数行で構成され、1 行が 1 つのファンに対応します。各行に現在の速度、要求される速度、および許容可能な最低速度を示す列があります。

テーブル内の行は、次のいずれかの方法で識別できます。

- 単純な単一次元のインデックス (テーブル内の行番号。例: 6)
- より複雑な複数次元のインスタンス指示子 (IP アドレスやポート番号など。例: `127.0.0.1, 1234`)

MIB 内の各テーブル定義には INDEX 句があり、この句によって、ある特定のエントリを選択するために使用するインスタンス指示子が定義されます。いずれの場合も、要求された行のインデックスを定義するオブジェクト自体が、MIB 内に定義されている必要があります。そのため、単純な単一次元のインデックスには通常インデックス列があり、テーブルの INDEX 句でそのインデックス列を参照します。テーブル内の特定のデータ項目を識別するには、列接頭辞を付けた OID を指定します。

たとえば、`myFanTable.myFanEntry.myCurrentFanSpeed` に接尾辞としてインスタンス指示子 (たとえば前の例に示した `127.0.0.1.1234`) を付けると、`myFanTable.myFanEntry.myCurrentFanSpeed.127.0.0.1.1234` となります。

MIB 構文を定義している SMI には、テーブルを拡張して追加エントリを付加する (実際にはテーブルに追加の列を付加する) ための重要な機能が提供されています。テーブルを拡張するには、拡張するテーブルの INDEX 句と同じ INDEX 句を持つテーブルを定義します。

MIB テーブルを定義する場合、定義するテーブルに含まれているオブジェクトではなく、他のテーブルからインポートされるオブジェクトをインデックスに持つ MIB テーブルを定義することも可能です。インポートするテーブルは他の MIB に定義されていてもかまいません。この構造を使用すると、既存のテーブルに列を追加することが実質的に可能です。

注 – SUN-PLATFORM-MIB では、このメカニズムを活用して、ENTITY-MIBに定義されたテーブルを拡張しています (第 5 章を参照)。

アクセス制御

MIB に定義するすべての識別可能なオブジェクトには、*read-only* や *read-write* などの最大限のアクセス権が関連付けられます。オブジェクトに関連付けられたアクセス権により、エージェントがサポート可能な最大限のアクセス権が決まります。また、マネージャーはこれらのアクセス権を使用して、オペレータに実行を許可する操作を制限できます。エージェントは必要に応じて最大アクセス権より下位のアクセス権を適用できます。つまり、読み取り書き込み可能なオブジェクトへの書き込みを拒否することができます。この拒否は以下を基準とすることができます。

- 識別されるオブジェクトに対する操作の適合度 (たとえば、MIB に定義されたオブジェクトがステートマシンを表す場合、ステートマシンに対しては特定のトランザクションのみが有効)
- 特定の操作を限定されたマネージャーのみに制限するセキュリティー制約

SNMPv1 でセキュリティーアクセス権の伝達に使用されるメカニズムは、「コミュニティー文字列」のメカニズムです。コミュニティー文字列は、各 SNMP データ要求とともに渡される「private」や「public」のような単なるテキスト文字列です。SNMPv1 および SNMPv2 の要求は暗号化されないため、安全と見なすことはできません。エージェントが応答すべきコミュニティー文字列と送信元マネージャーを定義するために使用されるメカニズムは、エージェントの実装に依存しますが、通常はアクセス制御リスト (ACL) に基づきます。ACL は適用可能なアクセス権を記述したファイルです。

ACL の構成方法については、第 11 章を参照してください。

SNMP マスターエージェント

マネージャーは、エージェントが動作しているシステムのよく使用されるポート (161) に UDP パケットを送信することによって、エージェントと通信します。ある特定のシステムで複数のエージェントが動作しており、各エージェントが異なる装置を管理している場合は、ポート資源の使用に関して競合が発生する可能性があります。可能な解決策の 1 つは、エージェントごとに異なる非標準ポート番号を使用することです。別の方法として、「マスターエージェント」の考え方を取り入れる方法があります。マスターエージェントは、ある特定のシステムで動作しているすべてのエージェントに代わって SNMP 要求を受け取り、受け取った要求を適切な宛先に転送します。この方法には、マネージャーがすべての SNMP エージェントに一貫した方法でアクセスできるという利点があります。Sun Fire B1600 ではいずれの方法もサポートしています。

マスターエージェントの詳細については、第 3 章を参照してください。

SNMP メディエータと snmpdx

snmpdx は標準 Solaris™ SNMP エージェントであり、SNMPv1 のマスターエージェントです。

デフォルトでは、SNMP メディエータは snmpdx のサブエージェントとして登録されます。この構成では、SNMPv1 の get 要求と set 要求のみがサポートされますが、SNMPv2c の通知が発行されます。

snmpdx と SNMP メディエータの関係については、第 10 章と第 11 章で詳しく説明します。

snmpdx(1M) のマニュアルページも参照してください。

第3章

マスターエージェント

この章では、SNMPv3 マスターエージェントの機能について説明します。

この章は以下の節で構成されています。

機能

SNMPv3 マスターエージェントは、SNMP 管理情報にアクセスするための、セキュリティー保護された単一アクセスポイントを提供します。

SNMPv3 マスターエージェントは SNMP サービスポート (デフォルトでは 161) にバインドし、すべての要求を標準 Solaris マスターエージェントである `snmpdx` に転送します。次に `snmpdx` がこれらの要求を適切な登録済みサブエージェントに転送します。`snmpdx` は標準の Solaris ソフトウェアに付属していますが、SNMPv1 のみをサポートしているため、SNMPv3 に用意されているセキュリティーを直接提供はしていません。マスターエージェントは、SNMPv1、v2c、v3 のいずれであるかにかかわらず、すべての要求を SNMPv1 に変換し、`snmpdx` で処理できるようにします。

マスターエージェントは実質的に、すべての既存サブエージェントへのセキュリティー保護されたアクセスを提供する SNMP のファイアウォールとして機能します。

構成の概要

ここでは、マスターエージェント機能を組み込むために SNMP を構成する方法の概要を説明します。詳細については、第 11 章で説明します。第 11 章では、この節の以降の部分で参照している構成ファイルの詳細な説明と例を記載しています。

SNMP メディエータは、自動的に割り当てられたポート番号を使用して、snmpdx のサブエージェントとして登録されます。

注 – このマニュアルでは、SNMP エージェントのことを SNMP メディエータと呼びます。

マスターエージェントが有効になっている場合は、`snmpdx` の自動起動が無効になり、マスターエージェントはポート 161 で登録されます。新しいポート番号が `snmpdx` に割り当てられ、`snmpdx` の起動はマスターエージェントの起動ファイルによって制御されます。

構成の手段は次のとおりです。

- SNMPv3 のセキュリティーファイル `spama.uacl` および `spama.security`
- SNMPv1/v2 のアクセス制御リスト (ACL) ファイル `spama.acl`
- 構成ファイル `spama.conf`
- 起動スクリプト `spama`

マスターエージェントの実行中にマスターエージェントを動的に構成することはできません。

SNMP メディエータも構成する必要があります。これについては第 11 章で説明します。

第4章

プラットフォーム管理モデル

この章では、Sun プラットフォーム SNMP モデル (sunPlat) を使用した Sun Fire B1600 システムの SNMP モデルについて概要を説明します。

この章は以下の節で構成されています。

Sun Fire B1600 プラットフォームのモデル化

Sun Fire B1600 は、シャーシ内の入れ子になった「ハードウェア資源」の集合として表されます。マザーボードなどの一部の資源はシャーシ内で直接入れ子にすることができます。他の資源内で入れ子になる資源もあります。たとえば、マザーボードにプロセッサを組み込むことができます。シャーシの内側から拡大していくこれらの関係によって、ハードウェア資源の「階層」が形成されます。階層内の各ハードウェア資源は、その資源を格納する「親」に物理的に収納されます。この階層は、ハードウェア資源を表す「管理対象オブジェクト」間の「関係」を使用してモデル化されます。

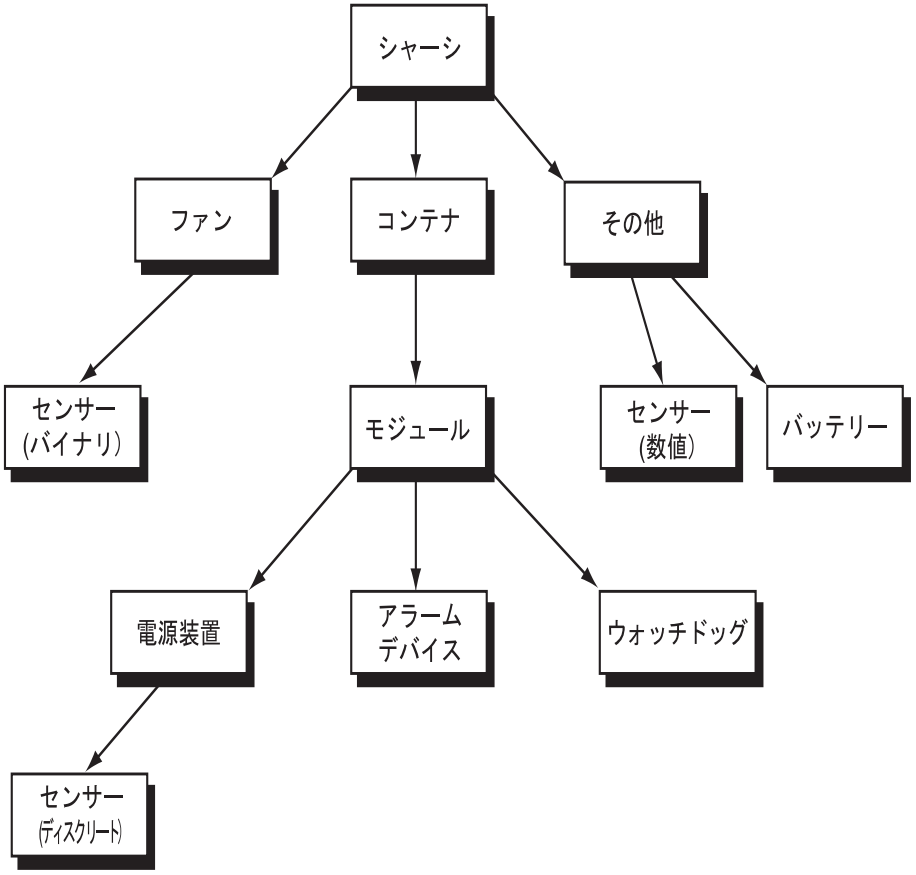

図 4-1　　ハードウェア資源階層の例

管理対象オブジェクト

sunPlat モデルは、基本的なハードウェア資源を表す、便利な共通プラットフォーム構成単位の集合を提供します。プラットフォーム構成単位のインスタンスは、「管理対象オブジェクト」と呼ばれます。管理対象オブジェクトとして表されているハードウェア資源は、監視したり、有用な構成情報を取得したりできます。

管理インタフェースの他の機能を表すために使用される追加の管理対象オブジェクトがあります。たとえば、ハードウェア資源は問題 (アラーム) または構成の変更 (イベント) に対する応答として、非同期状態レポート (通知) を発行できます。

管理対象オブジェクトは、管理対象オブジェクトの「クラス」に基づいて定義されます。資源の特性は管理対象オブジェクトの「プロパティー」によって表されます。「サブクラス」と呼ばれる新しいクラスが、既存のクラスに基づいて定義されます。サブクラスは「スーパークラス」のすべての特性を継承する一方、新しいプロパティーを追加してサブクラス独自の特性を表します。

図 4-2 は、sunPlat モデルに定義されたハードウェア構成単位のクラス継承階層を示しています。

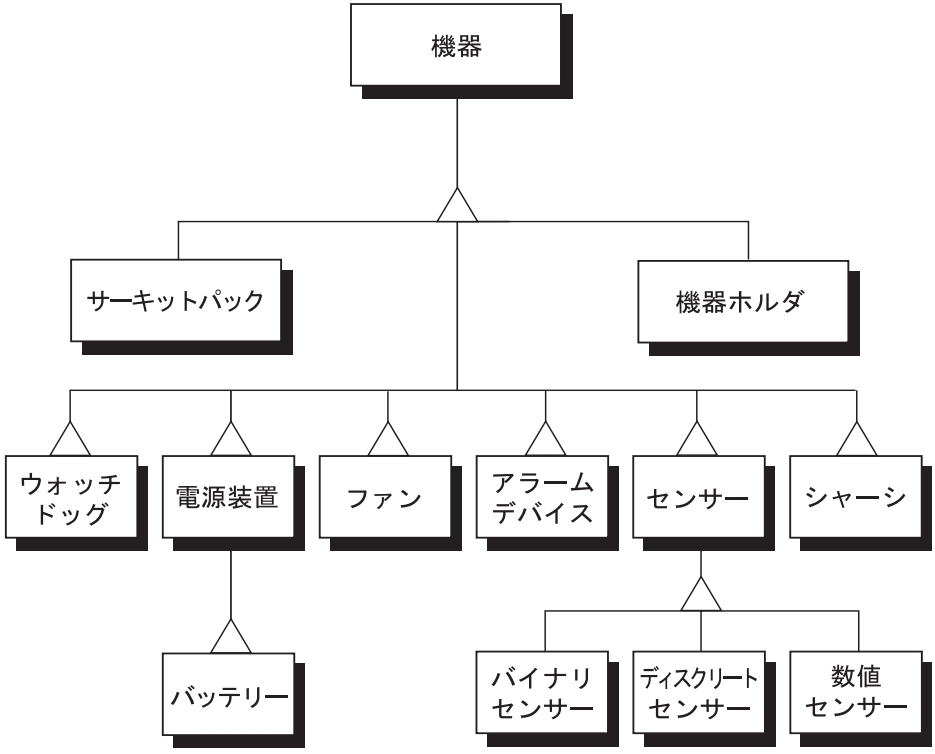

図 4-2 sunPlat 管理対象オブジェクトのクラス継承図

sunPlat のクラスの派生

sunPlat のクラスは、業界標準の管理の概念に基づいています。Sun Fire B1600 システムでは、ITU-T の汎用ネットワーク情報モデルのサブセットを使用してハードウェアのインフラストラクチャーを表しています。それによって、TMN (Telecommunications Management Network) での一貫性のある障害管理と構成管理を支援する、拡張可能な優れた枠組みを提供しています。

また、DMTF (Distributed Management Task Force) 共通情報モデル (CIM) スキーマによって物理環境とイベント定義およびイベント処理をモデル化し、共通モデルにシステム固有の拡張を加えています。

第5章

Sun Fire B1600 の MIB

この章では、Sun Fire B1600 の管理対象オブジェクトとそれらのオブジェクト間の関係が、SNMP インタフェースでどのように表されるかについて説明します。

この章は以下の節で構成されています。

SNMP でのモデルの表現

SNMP メディエータは、ポーリングによる管理およびイベントベースの管理の両方をサポートしています。Sun Fire B1600 システムの物理コンポーネントおよびシステム内の管理ドメインの論理的な表現が、ENTITY-MIB に定義されています。ENTITY-MIB は RFC 2737 で規定されています。SUN-PLATFORM-MIB は ENTITY-MIB の拡張です。

注 – MIB に定義された多数のオブジェクトに `read-write` の `MAX-ACCESS` が設定されていますが、それらのオブジェクトが書き込み可能になるのは、書き込みがモデル化されているコンポーネントに対して適切な操作である場合のみです。

ENTITY-MIB には、管理対象システムの物理要素と論理要素を記述する次のグループが含まれています。

entityPhysical グループ

entityPhysical グループには物理エンティティが記述されています。物理エンティティとは、エージェントによって管理される識別可能な物理資源です (たとえばシャーシ、電源装置、センサーなど)。物理エンティティは *entPhysicalTable* の行として表されます。

entityLogical グループ

entityLogical グループには、エージェントによって管理される論理エンティティが記述されています。entityLogical グループのエンティティは、上位レベルで管理する必要があるサービスを抽象化した上位の論理エンティティを表します。このグループのエンティティは、主にプラットフォームのハードウェア管理に関連していて、OS の再起動、ハードウェアのリセット、電源制御などの機能を含みます。通常、Solaris ドメインなどの管理ドメインまたはサービスコントローラに対応します。

entityMapping グループ

entityMapping グループには、entityPhysical グループと entityLogical グループの関係が記述されています。この機能は SNMP メディエータによって内部的に処理されます。

entityGeneral グループ

entityGeneral グループには、物理エンティティテーブルまたは物理マッピングテーブルのエンティティが変更された最終更新日時のタイムスタンプが記述されています。

entityMIBTraps グループ

entityMIBTraps グループには、ENTITY-MIB 内のオブジェクトに対する変更を伝えるために使用される *entPhysicalChange* 通知が定義されています。

第 2 章では、Sun プラットフォーム SNMP モデルが SNMP の汎用的な要素によってどのように表現されるかについて、概要を説明しています。

物理モデル

sunPlat 物理モデルでは、ENTITY-MIB を使用してハードウェアエンティティの包含階層を定義しています。各エンティティは ENTITY-MIB の entPhysicalTable に個別の行としてモデル化されています。

図 5-1 は、物理的な包含階層の例です。右下の数字は、*entPhysicalTable* 内の対応する行を示すインデックスです (表 5-1 を参照)。

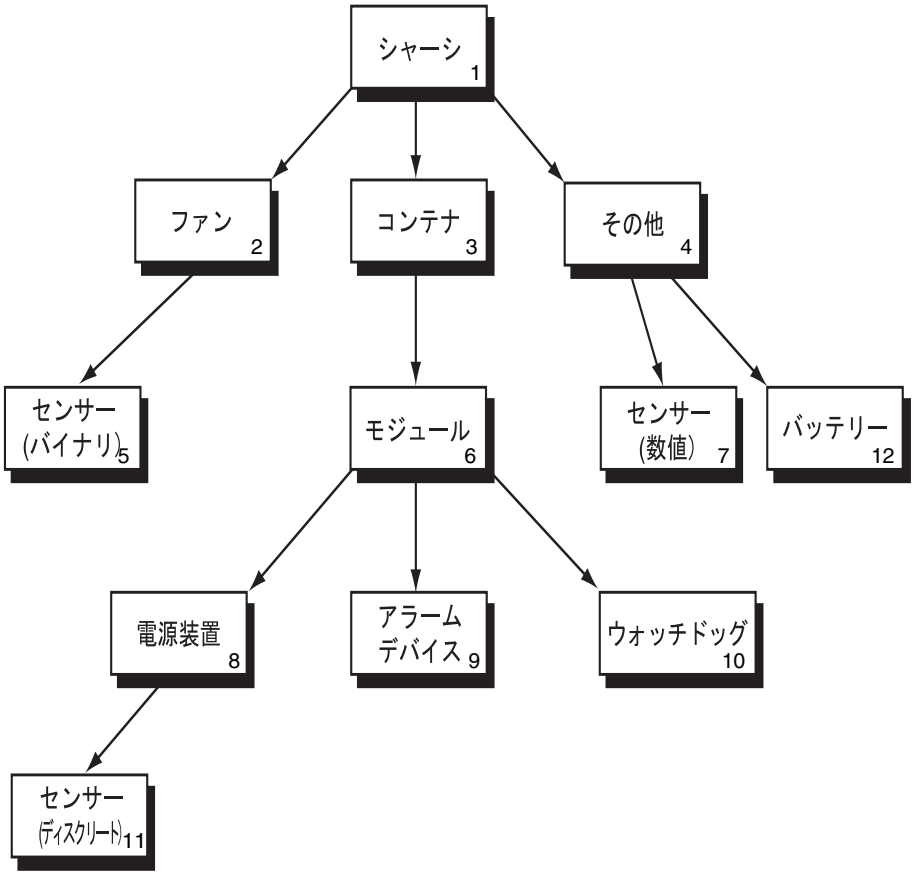

図 5-1　　ハードウェア資源階層の例

この情報は次の SNMP テーブルを使用して表されています。

- 物理エンティティテーブル (*entPhysicalTable*)

  このテーブルでは 1 行で 1 つのハードウェアエンティティを定義しています。ハードウェアエンティティを定義する行は「エントリ」と呼ばれ、特定の行は「インスタンス」と呼ばれます。各エントリの内容は次のとおりです。

  - 物理クラス (*entPhysicalClass*)
  - ハードウェアエンティティの共通特性
  - 一意のインデックス (*entPhysicalIndex*)
  - この資源の「コンテナ」として機能するハードウェアエンティティの行への参照 (*entPhysicalContainedIn*)。別のコンテナに物理的に収容されていないコンポーネント (シャーシなど) の場合は、0 です。

- 物理マッピングテーブル (*entPhysicalContainsTable*)

  このテーブルは、実質的には、物理エンティティテーブルに定義されたハードウェア資源階層のコピーです。このテーブルには 2 次元で、収容する側であるエントリの *entPhysicalIndex* が 1 つ目のインデックスで、収容される側の各エントリの *entPhysicalIndex* が 2 つ目のインデックスです。

表 5-1 は、上記の図の基になる *entPhysicalTable* を示しています。表 5-2 は物理マッピングです。

表 5-1 物理エンティティテーブル

entPhysicalIndex	entPhysicalClass	entPhysicalContainedIn
1	chassis	0
2	fan	1
3	container (例: FRU を搭載したスロット)	1
4	other	1
5	sensor (binary)	2
6	module (例: プラグ可能な FRU)	3
7	sensor (numeric)	4
8	power supply	6
9	alarm device	6
10	watchdog	6
11	sensor (discrete)	8
12	power supply (battery)	4

表 5-2　物理マッピングテーブル

entPhysicalIndex	entPhysicalChildIndex
1	2
1	3
1	4
2	5
3	6
4	7
4	12
6	8
6	9
6	10
8	11

クラス

`entPhysicalClass` は、特定の物理エンティティの汎用的なハードウェアタイプを示す列挙値であり、個々の値は *entPhysicalTable* の行として表されています。

Sun Fire B1600 プラットフォームに適用される列挙値を次に示します (図 5-1 と表 5-1 も参照)。

- `other(1)`

  列挙値 `other` は、物理エンティティが次のいずれかに分類できない場合に適用されます。

- `chassis(3)`

  `chassis` クラスは、機器全体のコンテナを表します。シャーシはあらゆる物理エンティティのクラスを含むことができます。

- `container(5)`

  `container` クラスは、同じタイプまたは異なるタイプの取り外し可能な物理エンティティを 1 つ以上収容できる物理エンティティに適用されます。たとえば、シャーシ内の個々の空スロットまたは装着済みスロットがコンテナとしてモデル化されます。電源装置やファンなどの FRU (現場交換可能ユニット) は、コンテナエンティティ内のモジュールとしてモデル化されます。

- `powerSupply(6)`

  `power supply` クラスは、電源を供給可能なコンポーネントに適用されます。

- `fan(7)`

  `fan` クラスは、物理エンティティがファンまたはその他の冷却装置である場合に適用されます。

- `sensor(8)`

  `sensor` クラスは、物理プロパティーを測定する能力がある物理エンティティに適用されます。

- `module(9)`

  `module` クラスは独立したサブシステムに適用され、`chassis` や他の `module` などの他の物理エンティティ内にモデル化されます。このエンティティは常に `container` 内にモデル化されます。

論理モデル

sunPlat 論理モデルでは、ENTITY-MIB で上位論理エンティティのリストを提供しています。各エンティティは ENTITY-MIB の *entLogicalTable* に個別の行としてモデル化されています。物理モデルとは異なり、論理モデルの構造は階層ではなくフラットです。

ENTITY-MIB では、物理オブジェクトの場合とは異なり、論理オブジェクトのクラスを区別していません。SUN-PLATFORM-MIB に論理オブジェクトのクラス階層が提供されています。これについては第 7 章で説明します。

entLogicalTable 内の情報を使用して、異なる名前付けコンテキストによる複数の有効範囲をサポートできます。ただし、この製品にこの機能は採用されていません。特定の値の情報は *entLogicalDescription* と *entLogicalTAddress* です。後者は論理エンティティにアクセスするための IP アドレスを示します。

論理階層と物理階層のマッピング

ENTITY-MIB には、ENTITY-MIB を構成する論理オブジェクトと物理オブジェクトのマッピングが定義されています。マッピングの定義は *entLPMappingTable* で行われています。これは 2 次元のテーブル (*entPhysicalContainsTable* と同じ) であり、ある特定の論理エンティティを実現する物理エンティティを識別します。それらの物理エンティティは *entLPPhysicalIndex* によって識別されます。これは *entPhysicalIndex* に相当します。

ある特定の論理エンティティに関連付けられたすべての物理エンティティをこのテーブルに記述することは可能ですが、通例として、収容する側の物理エンティティのみを参照します。たとえば、ある物理モジュールによって実現される論理エンティティについて、マッピングではそのモジュールのみを参照し、そのモジュールに含まれるすべての物理エンティティを参照することはしません。

イベントとアラームのモデル

ENTITY-MIB には、*entConfigChange* という 1 つの SNMP 通知が定義されています。この通知は、MIB 内のテーブルに対する変更を伝えるために使用されます。この通知は、最大 5 秒に 1 回トラップを生成するように設定されています。

SUN-PLATFORM-MIB では、他にも特定の通知が定義されています。それらの通知については、第 8 章で説明します。

SUN-PLATFORM-MIB

SUN-PLATFORM-MIB では次の拡張が行われています。

- コンポーネントの新しいクラスを定義するための物理エンティティテーブルの拡張
- 上位プラットフォームオブジェクトおよびサーバーオブジェクトを定義するための論理エンティティテーブルの拡張

注 – SUN-PLATFORM-MIB に定義されたすべてのオブジェクトには *sunPlat* という接頭辞が付いていて、そのためオブジェクトはグローバルに一意になっています。

物理モデルの拡張テーブル

SUN-PLATFORM-MIB では、物理エンティティテーブルに記述されていないクラスの追加属性を定義しています。SUN-PLATFORM-MIB では、一部のみ記述された以下の拡張テーブルを追加することによって、物理エンティティテーブルを拡張しています。

- 拡張機器テーブル

  物理エンティティテーブルを拡張して、Equipment クラスの管理対象オブジェクトに関する追加情報を定義しています。このクラスはすべての Sun Fire B1600 ハードウェア資源に適用可能です。Equipment クラスのサブクラスが追加の拡張テーブルに定義されています。

  - 拡張機器ホルダテーブル

    拡張機器テーブルの拡張です。entPhysicalClass が `container(5)` の管理対象オブジェクトに関連する追加情報を定義しています。

  - 拡張サーキットパックテーブル

    拡張機器テーブルの拡張です。entPhysicalClass が `module(9)` の管理対象オブジェクトに関連する追加情報を定義しています。

- 拡張物理テーブル

  物理エンティティテーブルの拡張です。物理エンティティテーブルの entPhysicalClass 列を補足するために使用されます。entPhysicalClass が `other(1)` の資源が、sunPlat でモデル化されているクラスつまり Watchdog または AlarmDevice クラスに属する場合、その資源の sunPlatPhysicalClass がこのテーブルで識別されています。

- 拡張センサーテーブル

  拡張機器テーブルの拡張です。entPhysicalClass が `sensor(8)` の管理対象オブジェクトに関連する追加情報を定義しています。Sensor クラスのサブクラスが追加の拡張テーブルに定義されています。このテーブルでは sunPlatSensorClass を使用してサブクラスを識別しています。

  - 拡張バイナリセンサーテーブル

    拡張センサーテーブルの拡張です。entPhysicalClass が `sensor(8)` で sunPlatSensorClass が `binary(1)` の管理対象オブジェクトに関連する追加情報を定義しています。

  - 拡張数値センサーテーブル

    拡張センサーテーブルの拡張です。entPhysicalClass が `sensor(8)` で sunPlatSensorClass が `numeric(2)` の管理対象オブジェクトに関連する追加情報を定義しています。

  - 拡張ディスクリートセンサーテーブル

    拡張センサーテーブルの拡張です。entPhysicalClass が `sensor(8)` で sunPlatSensorClass が `discrete(3)` の管理対象オブジェクトに関連する追加情報を定義しています。

- 拡張ファンテーブル

  拡張機器テーブルの拡張です。entPhysicalClass が `fan(7)` の管理対象オブジェクトに関連する追加情報を定義しています。

- 拡張アラームテーブル

  拡張機器テーブルの拡張です。entPhysicalClass が `other(1)` で sunPlatPhysicalClass が `alarm(8)` の管理対象オブジェクトに関連する追加情報を定義しています。

- 拡張ウォッチドッグテーブル

  拡張機器テーブルの拡張です。entPhysicalClass が `other(1)` で sunPlatPhysicalClass が `watchdog(3)` の管理対象オブジェクト (通常はサービスインジケータ LED を表す) に関連する追加情報を定義しています。

- 拡張電源装置テーブル

  拡張機器テーブルの拡張です。entPhysicalClass が `powerSupply(6)` の管理対象オブジェクトに関連する追加情報を定義しています。

表 5-3 は、物理エンティティテーブルの拡張テーブルの例です。entPhysicalIndex (このテーブルの列 1) は、図 5-1 に示したハードウェア資源階層の例に対応しています。

ENTITY-MIB			SUN-PLATFORM-MIB											
entPhysicalIndex	entPhysicalClass				sunPlatPhysicalClass		sunPlatFanClass		sunPlatSensorClass					sunPlatPowerSupplyClass
1	chassis													
3	container													
8	power supply													G
12	power supply													H
2	fan						A							
	fan						B							
	fan						C							
5	sensor								D					
7	sensor								E					
11	sensor								F					
6	module													
9	other				alarm									
10	other				watch dog									
4	other				other									
entPhysicalTable		sunPlatEquipmentTable	sunPlatEquipmentHolderTable	sunPlatCircuitPackTable	sunPlatPhysicalTable	sunPlatWatchdogTable	sunPlatFanTable	sunPlatAlarmTable	sunPlatSensorTable	sunPlatBinarySensorTable	sunPlatNumericSensorTable	sunPlatDiscreteSensorTable	sunPlatDiscreteSensorStatusTable	sunPlatPowerSupplyTable

表 5-3　物理エンティティテーブルの拡張

表 5-4　物理エンティティテーブルの拡張 (表 5-3) の記号の意味

記号	説明
A	ファン
B	冷却装置
C	ヒートシンク
D	バイナリ
E	数値
F	ディスクリート
G	電源装置
H	バッテリー

論理モデルテーブルの拡張

SUN-PLATFORM-MIB では、論理エンティティテーブルでサポートされていないクラスの追加属性を定義しています。SUN-PLATFORM-MIB では、一部のみ記述された以下の拡張テーブルを追加することによって、論理エンティティテーブルを拡張しています。

- 拡張論理クラステーブル

  このテーブルでは、entLogicalTable を拡張して、論理エンティティのクラス (*SunPlatLogicalClass*) と状態 (*sunPlatLogicalStatus*) を定義しています。sunPlatLogicalTable は entLogicalTable 内のすべてのエントリに対して有効です。さらに次の拡張テーブルに論理クラスの Computer System サブクラスが定義されています。

  - 拡張コンピュータシステムテーブル

    このテーブルでは、entLogicalTable を拡張して、コンピュータシステムのインスタンスに共通する属性を定義しています。sunPlatUnitaryComputerSystemTable は、sunPlatLogicalClass が `computerSystem(2)` である entLogicalTable の行に対して有効です。

    各コンピュータシステムの論理エンティティに、読み込み情報テーブル (*sunPlatInitialLoadInfoTable*) にあるエントリの組が関連付けられています。このエントリの組は、コンピュータシステムの起動に関する設定を制御するパラメタになります。

イベントとアラームのログテーブル

SNMP トラップが必ず配信される保証はありません。この理由のため、また管理アプリケーションでプラットフォームの現在のアラーム状態を正確に追跡できるように、MIB には管理対象オブジェクトごとに現在の問題リストが維持されます。これはオブジェクトごとの未解決アラームのテーブルです。アラーム条件が解消されるとアラームは自動的に解除されます。

SUN-PLATFORM-MIB には、イベントまたはアラームをそのタイプ別に、または影響を受けるエンティティ別にグループ化して記録するためのログが定義されています。Sun Fire B1600 は、すべての監視対象エンティティに関する未解決アラームのリストを、これらのログを使用して維持するように実装されています。理由は前の段落に示したとおりです。

MIB に定義された、アラームを生成する可能性があるエンティティ (つまり、すべての物理エンティティと論理エンティティ) については、ログテーブル (sunPlatLogTable) にエントリがあります。このテーブルで管理の状態を確認し、現在の問題リストを制御できます。ログは永久に有効化された状態であり、サイズに制限はありません。

ログテーブルの各エントリに対応したログレコードテーブルには 0 個以上のエントリがあります。これらのエントリの詳細については、第 8 章で説明します。

イベントレコード

イベントレコードは、sunPlat のトラップ通知を形成します。モデルに変更が発生すると、管理アプリケーションに変更が通知されます。通知にはイベントとアラームという 2 つのカテゴリの SNMP トラップが使用されます。

イベント

- Object Creation Record

  このレコードは、資源がオブジェクトモデルに追加されたことを示します。

- Object Deletion Record

  このレコードは、資源がオブジェクトモデルから削除されたことを示します。

- State Change Record

  このレコードは、資源の状態に変更が発生したことを示します。

- Integer Attribute Value Change Record

  このレコードは、`INTEGER` 型の属性によってモデル化されている資源の特性に変更が発生したことを示します。変更されたオブジェクトに応じて、整数は符号付きまたは符号なしのいずれかです。

- String Attribute Value Change Record

  このレコードは、OCTET STRING 型の属性によってモデル化されている資源の特性に変更が発生したことを示します。
- OID Attribute Value Change Record

  このレコードは、OBJECT IDENTIFIER 型のオブジェクト識別子属性に変更が発生したことを示します。

アラーム

- Communications Alarm Record

  このレコードは、資源がサポートしている通信サービスに障害が発生したことを示します。
- Environmental Alarm Record

  このレコードは、資源に関係する環境条件を示します。
- Equipment Alarm Record

  このレコードは、資源に障害が発生したことを示します。
- Processing Error Alarm Record

  このレコードは、資源に関連したソフトウェア障害または処理障害が発生したことを示します。
- Quality of Service Alarm Record

  このレコードは、サービス品質アラームが発生したことを示します。
- Indeterminate Alarm Record

  このレコードは、タイプが不明なアラームが発生したことを示します。

第6章

物理モデル

この章では、sunPlat の物理クラス階層について説明し、sunPlat モデルで定義された管理対象の物理オブジェクトクラスが SUN-PLATFORM-MIB にどのように記述されているかを示します。

この章は以下の節で構成されています。

sunPlat 物理クラス階層

図 6-1 は sunPlat クラスの継承階層を示しています。Sun Fire B1600 内のハードウェア資源はこれらのクラスを使用してモデル化されています。

Physical Entity スーパークラスには、管理対象オブジェクト間の関係を定義する属性が提供されています。また、Equipment クラスの属性に対応した標準 SNMP 属性も提供されています。

sunPlat Equipment クラスは Physical Entity スーパークラスから派生したクラスです。障害監視に適用可能な対応するクラスに定義された、追加属性が提供されています。

sunPlat Equipment Holder クラスと sunPlat Circuit Pack クラスは sunPlat Equipment スーパークラスから派生したクラスです。それぞれ、受容装置と、受容装置に接続されるコンポーネントを表します。

sunPlat Equipment クラスをさらに特殊化して DMTF の派生クラスが提供されています。

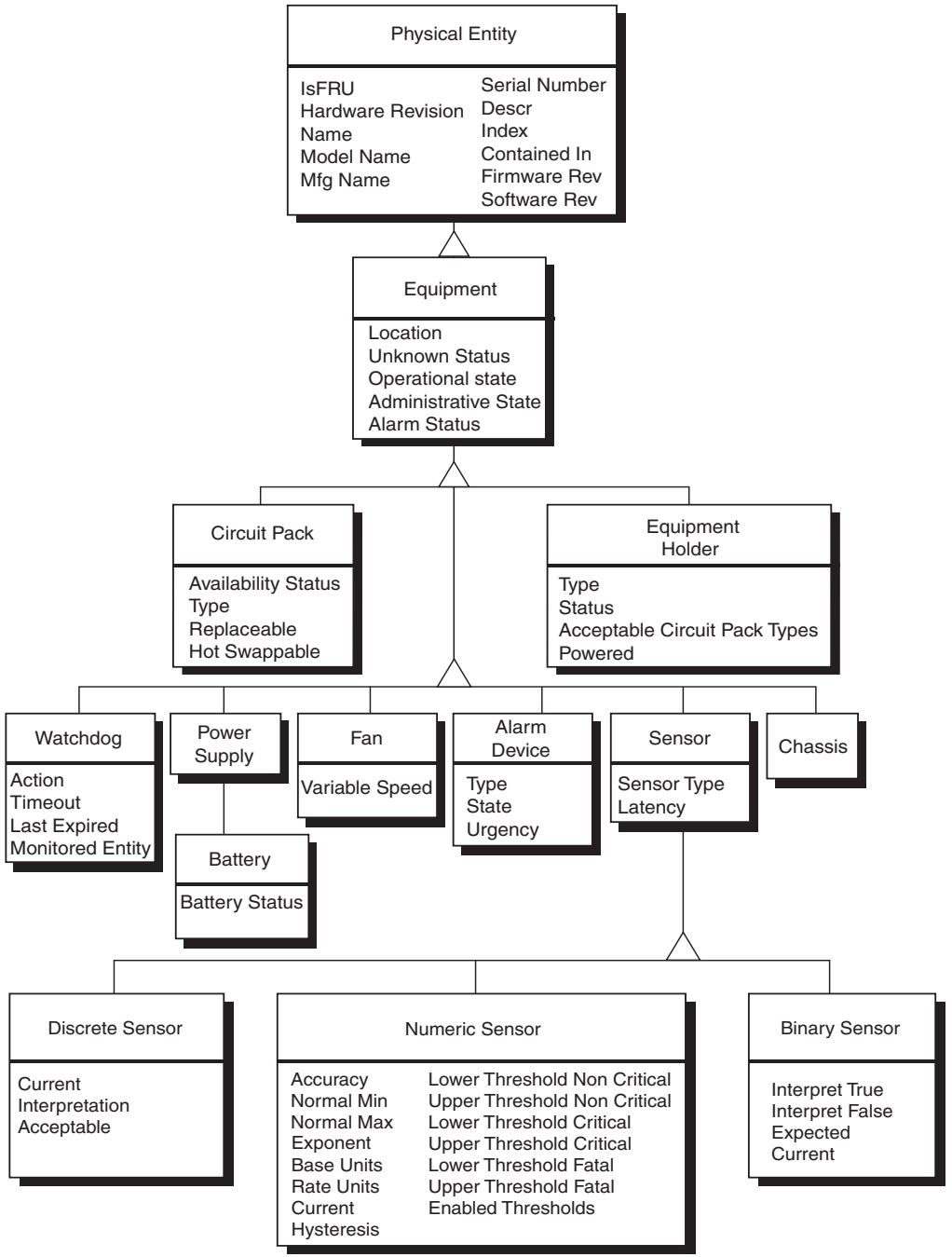

図 6-1 sunPlat 物理資源の継承クラス図

sunPlat クラス定義

sunPlat クラスの属性はハードウェア資源の特性を表すために使用されます。マネージャーに対する資源の使用可能性と操作可能性が管理対象オブジェクトの「State (状態)」によって表されます。さまざまな sunPlat クラスに管理対象オブジェクトの状態の特徴を表す多様な属性があります。

Physical Entity

Physical Entity スーパークラスは、すべての資源について汎用的な特性を表すために使用されます。

注 – 以下に示す属性名では、簡潔にするために接頭辞 *entPhysical* を省略してあります。

- ***Descr***
  資源の既知の名前を含むテキスト文字列です。この名前は通常、製品マニュアルや製品ラベルで資源を記述するために使用される名前です。ファームウェアに記憶されている名前の場合もあります。
- ***Is FRU***
  資源が FRU (現場交換可能ユニット) かどうかを表すブール値です。*sunPlatCircuitPack* クラスのハードウェア資源のみが FRU とみなされます。
- ***Hardware Rev***
  資源の製造元のハードウェアバージョン情報を含むテキスト文字列です。ハードウェアバージョン情報が関連付けられていないハードウェア資源もあります。
- ***Name***
  資源の論理名を含むテキスト文字列です。オペレーティングシステムおよび関連ユーティリティーではこの名前によって資源を識別します。この名前はデバイスノードか、システムユーティリティーで使用される定義済みの名前 (該当する場合) です。デバイス名がない資源もあります。
- ***Model Name***
  ユーザーが視認できる製造元のパーツ番号またはパーツ定義を含むテキスト文字列です。パーツの番号や定義が関連付けられていないハードウェア資源もあります。
- ***Serial Num***
  資源の製造元のシリアル番号を含むテキスト文字列です。シリアル番号が関連付けられていないハードウェア資源もあります。

- ***Mfg Name***
  資源の製造元の名前を含むテキスト文字列です。製造元の名前が関連付けられていないハードウェア資源もあります。

Physical Entity スーパークラスには、ハードウェア資源の階層を記述するために使用される属性もあります。

- ***Class***
  この列挙型には、特定の物理資源の一般的なハードウェアタイプを示すインジケータが含まれます。このクラスのサポートされている値は、ENTITY-MIB に定義されています。この属性は、管理対象オブジェクトに関連した拡張テーブルを指示するものとして使用できます。ENTITY-MIB のクラスと sunPlat のクラス間のマッピングは、表 6-1 に示すとおりです。

表 6-1 Physical Entity スーパークラスの「Class」属性のマッピング

entPhysicalClass	sunPlat クラス
chassis(3)	*sunPlat Chassis*
backplane(4)	未実装
container(5)	*sunPlat Equipment Holder*
powerSupply(6)	*sunPlat Power Supply*
fan(7)	*sunPlat Fan*
sensor(8)	*sunPlat Sensor*、サブクラスあり
module(9)	*sunPlat Circuit Pack*
port(10)	未実装
stack(11)	未実装
other(1)	*sunPlat Equipment*、サブクラスあり
unknown(2)	未実装

- ***Index***
  この整数は、物理エンティティテーブル内で管理対象オブジェクトを特定するエントリを一意に識別します。値は事前に割り当てられないため、エージェントを起動するたびに異なることがあります。
- ***Contained In***
  この整数は、管理対象オブジェクトを収容している管理対象オブジェクトの *Index* 属性を表します。したがって、この属性は管理対象オブジェクト間の関係をモデル化します。

注 – 物理的な包含階層のルートにあるオブジェクト (通常はシャーシ) は、テーブルに記述された他のエンティティに物理的に収容されていません。このことを示すために、そのようなオブジェクトの *entPhysicalContainedIn* 値は `0` に設定されています。

- ***Firmware Rev***
  資源の製造元のファームウェアバージョン情報を含むテキスト文字列です。ファームウェアバージョン情報が関連付けられていないハードウェア資源もあります。
- ***Software Rev***
  資源の製造元のソフトウェアバージョン情報を含むテキスト文字列です。ソフトウェアバージョン情報が関連付けられていないハードウェア資源もあります。

sunPlat Equipment クラス

sunPlat Equipment クラスは、すべてのハードウェア資源について汎用的な特性を表すために使用されます。このクラスには、構成および汎用的な健全性の状態情報を表す属性があります。さらに、特定のタイプの資源に関する詳細な構成情報と監視データを提供するサブクラスがあります。

entPhysicalClass は、記述するサブクラスに依存します。

sunPlat Equipment クラスの属性は次のとおりです。

注 – 以下に示す属性名では、簡潔にするために接頭辞 *sunPlatEquipment* を省略してあります。

- ***Administrative State***
  この読み取り書き込み属性は、資源の現在の管理状態を示す次のいずれかの列挙値を取ります。
  - `locked(1)`
  - `unlocked(2)`
  - `shuttingDown(3)`

- ***Operational State***
  この読み取り専用属性は、資源が物理的に装着されていてサービスを提供可能かどうかを示す列挙型です。この属性は管理対象オブジェクトの状態に影響を与え、表 6-2 に示す値を取ります。

表 6-2 Operational State の属性値

属性値	説明
`disabled(1)`	資源は完全に動作不能な状態であり、ユーザーにサービスを提供できません。
`enabled(2)`	資源は部分的または完全に動作可能であり、使用可能な状態です。

- ***Alarm Status***
  この読み取り専用属性は、資源の現在のアラーム状態を示す列挙値を取ります。管理対象オブジェクトの未解決アラームのうち最も高い重要度を示します。この属性は次の値を取ります。
  - `critical(1)`
  - `major(2)`
  - `minor(3)`
  - `indeterminate(4)`
  - `warning(5)`
  - `pending(6)`
  - `cleared(7)`
- ***Unknown Status***
  この読み取り専用属性は、他の状態属性が資源の実際の状態を反映していない可能性があるかどうかを示します。この属性は、管理対象オブジェクトが資源に対する障害を正確に報告できるかどうかを示すブール値を取ります。資源が資源自身の状態を正しく反映できない場合、この属性は `true` に設定されます。
- ***Location Name***
  この読み取り専用属性には、資源の場所を特定する情報が含まれます。シャーシに直接収納されている資源の場合、この属性はスロットおよび製品マニュアルに記載されている名称と関連付けられるか、またはシャーシ内の資源の位置を示す地理的な場所を指定します。その他のハードウェア資源には、通常、その資源を収納している資源の管理対象オブジェクトの *Name* に対応した *location* が設定されます。

sunPlat Circuit Pack クラス

sunPlat Circuit Pack クラスは、交換可能な資源または FRU について汎用的な特性を表すために使用されます。交換可能な資源は、内部のハードウェアコンポーネントを認知されたフォームファクタとしてパッケージ化することが目的のハードウェアモジュールとして定義されます。通常、FRU のフォームファクタと物理的な外観は決

まっています。コネクタに挿入するプラグ可能な取り外し可能ユニットや、ベイ内に固定されたユニットや、ドロワー、ラック、またはシェルフに搭載するユニットがあります。

このクラスの entPhysicalClass は `module(9)` です。

sunPlat Circuit Pack クラスの属性は次のとおりです。

注 – 以下に示す属性名では、簡潔にするために接頭辞 *sunPlatCircuitPack* を省略してあります。

- ***Type***
  この読み取り専用属性は、コンテナに対する資源の互換性を評価するために使用されるテキスト文字列です。この属性で資源の機能とフォームファクタの特性を特定できます。
- ***Availability Status***
  この読み取り専用属性は、管理対象オブジェクトの *Operational State* をさらに分類します。これは BITS 構文を使用するオブジェクトであり、表 6-3 に示すように、0 個以上の値を取ります。これらの値がすべて管理対象オブジェクトのすべてのクラスに適用されるとは限りません。この属性は管理対象オブジェクトの状態に影響を与えます。

表 6-3 Availability Status の属性値

属性値	ビット番号	16 進数	説明
`inTest(0)`	0	80	資源はテスト手順を実行中です。
`failed(1)`	1	40	資源に内部障害が発生し、資源の動作を妨げています。*Operational State* は `disabled(1)` です。
`powerOff(2)`	2	20	資源に電源が入っていないため電源を投入する必要があります。
`offLine(3)`	3	10	資源をオンラインにして使用可能にするために所定の操作を実行する必要があります。*Operational State* は `disabled(1)` です。
`offDuty(4)`	4	08	内部制御プロセスにより資源は非アクティブ化されました。

表 6-3　Availability Status の属性値 (続き)

属性値	ビット番号	16 進数	説明
dependency(5)	5	04	依存している他の資源が使用できないため、資源は動作できません。*Operational State* は disabled(1) です。
degraded(6)	6	02	速度や操作能力など、資源が提供するサービスの一部に機能低下が発生しています。しかし、資源はサービスを提供できる状態です。*Operational State* は enabled(2) です。
notInstalled(7)	7	01	管理対象オブジェクトに記述された資源が存在しないか、不完全です。*Operational State* は disabled(1) です。

- ***Replaceable***
  この読み取り専用属性は、資源が交換可能ユニットかどうかを示すブール値を取ります。
- ***Hot Swappable***
  この読み取り専用属性は、交換可能な資源がホットスワップ対応かどうかを示すブール値を取ります。

sunPlat Equipment Holder

sunPlat Equipment Holder クラスは、取り外し可能なハードウェア資源を収納できるハードウェア資源の特性を表すために使用されます。

このクラスの entPhysicalClass は container(5) です。

sunPlat Equipment Holder クラスの属性は次のとおりです。

注 – 以下に示す属性名では、簡潔にするために接頭辞 *sunPlatEquipmentHolder* を省略してあります。

- ***Type***
  この読み取り専用属性は、表 6-4 に示すように、資源のホルダタイプを表す列挙型です。

表 6-4 Equipment Holder の Type の属性値

属性値	説明
bay(1)	通常、ベイはラック内の垂直方向のスペースを持つユニットで、シェルフやドロワーを組み込んで通信用機器を収納します。シャーシ内でベイを使用している場合、sunPlat では、信号接続のためにケーブルを必要とする物理的な受容装置と解釈されます。
shelf(2)	通信用機器をラック内に収納するための水平方向の棚またはサブラック。
drawer(3)	通信用機器をラック内に収納するための水平方向の格納装置。
slot(4)	取り外し可能な機器に使用する信号接続用コネクタを内蔵した物理的な受容装置。
rack(5)	ラックは、独立した格納装置に通信用機器、ホルダ、およびケーブル管理システムを収納するための支持構造です。

- ***Acceptable Types***
  この読み取り専用属性は、ホルダによって支持される取り外し可能な資源 (サーキットパック) を表すテキスト文字列のリストです。これらのタイプは、取り外し可能な資源の *Type* 属性に対して互換性テスト済みです。
- ***Status***
  この読み取り専用属性は、ホルダの状態を、ホルダに装着する交換可能なハードウェア資源 (サーキットパック) との関連で示す列挙型です。表 6-5 を参照してください。

表 6-5 Equipment Holder の Status の属性値

属性値	説明
holderEmpty(1)	ホルダ内に取り外し可能な資源はありません。
inTheAcceptableList(2)	*Acceptable Circuit Pack Types* リストのいずれかのタイプである取り外し可能な資源がホルダに装着されています。
notInTheAcceptableList(3)	ネットワーク要素によって認識できる取り外し可能な資源がホルダに装着されています。しかし、その資源は *Acceptable Circuit Pack Types* リストのいずれのタイプでもありません。
unknownType(4)	認識できない取り外し可能な資源がホルダに装着されています。

- ***Powered***
  この読み取り書き込み属性は、資源の電源状態を示す列挙型です。示される値は次のとおりです。
  - `other(1)`
  - `unknown(2)`
  - `powerOff(3)`
  - `powerOn(4)`

sunPlat Power Supply

sunPlat Power Supply クラスは、電源装置を表すために使用されます。このクラスは sunPlat Equipment クラスの特性を拡張しません。一般に、電源装置は監視対象プロパティー (たとえば電圧、電流、温度など) を表すセンサーを備えています。ファンなどの他のハードウェア資源を内蔵している場合もあります。これは管理対象オブジェクト間の関係を使用してモデル化されます。

電源装置が取り外し可能な資源である場合、電源装置は sunPlat Circuit Pack クラスの管理対象オブジェクト内にモデル化されます。

このクラスの entPhysicalClass は `powerSupply(6)` です。

sunPlat Power Supply クラスの属性は次のとおりです。

注 – 以下に示す属性名では、簡潔にするために接頭辞 ***sunPlatPowerSupply*** を省略してあります。

- ***Class***

  この読み取り専用属性は、電源装置のクラスを示す列挙型であり、次の値を取ります。
  - `other(1)`
  - `powerSupply(2)`
  - `battery(3)`

sunPlat Battery

sunPlat Battery クラスは、バッテリーから電源を供給する電源装置を表すために使用されます。

このクラスの entPhysicalClass は `powerSupply(6)`、SunPlatPowerSupplyClass は `battery(3)` です。

sunPlat Battery クラスの属性は次のとおりです。

注 – 以下に示す属性名では、簡潔にするために接頭辞 *sunPlatBattery* を省略してあります。

- ***Status***
  この読み取り専用属性は、バッテリーの状態を示す列挙型であり、次の値を取ります。
  - `other(1)`
  - `unknown(2)`
  - `fullyCharged(3)`
  - `low(4)`
  - `critical(5)`
  - `charging(6)`
  - `chargingAndHigh(7)`
  - `chargingAndLow(8)`
  - `chargingAndCritical(9)`
  - `undefined(10)`
  - `partiallyCharged(11)`

sunPlat Watchdog

sunPlat Watchdog クラスは、オペレーティングシステムまたはアプリケーションの状態のハードウェアによる監視を可能にする、タイマーハードウェア資源の特性を表すために使用されます。

このクラスの entPhysicalClass は `other(1)`、sunPlatPhysicalClass は `watchdog(3)` です。

sunPlat Watchdog クラスの属性は次のとおりです。

注 – 以下に示す属性名では、簡潔にするために接頭辞 *sunPlatWatchdog* を省略してあります。

- ***Timeout***
  この読み取り専用属性は、リセットしないとウォッチドッグがタイムアウトする間隔をミリ秒単位で示す整数です。

- ***Action***
  この読み取り専用属性は、*Timeout* で指定した秒数内にウォッチドッグをリセットしなかった場合に実行されるウォッチドッグのアクションを示す列挙型です。示される値は表 6-6 のとおりです。

表 6-6　Watchdog の Action の属性値

アクション	説明
`statusOnly(1)`	ウォッチドッグはソフトウェアによる読み取りが可能ですが、アクションは実行しません。
`systemInterrupt(2)`	ウォッチドッグは監視対象システムに対してハードウェア割り込みを生成します。
`systemReset(3)`	ウォッチドッグは監視対象システムをリセットします。
`systemPowerOff(4)`	ウォッチドッグは監視対象システムの電源を切断します。
`systemPowerCycle(5)`	ウォッチドッグは監視対象システムの電源を切断してから再投入します。

- ***Last Expired***
  この読み取り専用属性は、ウォッチドッグが最後にタイムアウトした日時を示します。
- ***Monitored Entity***
  この読み取り専用属性は、ウォッチドッグで監視できるエンティティを表す列挙型です。示される値は次のとおりです。
  - `unknown(1)`
  - `other(2)`
  - `operatingSystem(3)`
  - `operatingSystemBootProcess(4)`
  - `operatingSystemShutdownProcess(5)`
  - `firmwareBootProcess(6)`
  - `biosBootProcess(7)`
  - `application(8)`
  - `serviceProcessors(9)`

sunPlat Alarm

sunPlat Alarm クラスは、問題のある状況に連動したインジケータ (ブザー、LED、リレー、バイブレータ、ソフトウェアアラームなど) を備えたハードウェア資源の特性を表すために使用されます。

このクラスの entPhysicalClass は `other(1)`、sunPlatPhysicalClass は `alarm(2)` です。

sunPlat Alarm クラスの属性は次のとおりです。

注 – 以下に示す属性名では、簡潔にするために接頭辞 *sunPlatAlarm* を省略してあります。

- ***Type***
  この読み取り専用属性は、アラーム条件の通知手段を表す列挙型です。示される値は表 6-7 のとおりです。

表 6-7 Alarm の Type の属性値

属性値	説明
other(1)	アラームデバイスタイプが次のいずれでもない
audible(2)	アラームデバイスは装置の音が変化する
visible(3)	アラームにより装置に視覚的変化が発生する
motion(4)	アラームにより装置に動作が発生する
switch(5)	アラームにより電気的な信号の変化が発生する

- ***State***
  この読み取り書き込み属性は、アラームの状態を表す列挙型です。示される値は表 6-8 に示すとおりです。

表 6-8 Alarm の State の属性値

属性値	説明
unknown(1)	アラームの状態は未定義または監視不能です。
off(2)	アラームは非アクティブです。
steady(3)	アラームはアクティブです。
alternating(4)	アラームは非アクティブ状態とアクティブ状態を繰り返しています。

- ***Urgency***
  この読み取り書き込み属性は、アラームが点滅したり、振動したり、音を出したりする相対的な回数を示す列挙型です。示される値は次のとおりです。
  - other(1)
  - unknown(2)
  - notSupported(3)
  - informational(4)
  - nonCritical(5)
  - critical(6)
  - unrecoverable(7)

sunPlat Fan

sunPlat Fan クラスは、アクティブな冷却装置の特性を表すために使用されます。通常、ファンは回転速度を示すセンサーを備えています。これは、sunPlat Fan 管理対象オブジェクトと sunPlat Sendor クラスのタコメータ管理対象オブジェクト間の物理的な包含関係を使用してモデル化されます。

このクラスの entPhysicalClass は `fan(7)` です。

sunPlat Fan クラスの属性は次のとおりです。

注 – 以下に示す属性名では、簡潔にするために接頭辞 *sunPlatFan* を省略してあります。

- ***Class***
  この読み取り専用属性は、冷却装置のクラスを示す列挙型であり、次の値を取ります。
  - `other(1)`
  - `fan(2)`
  - `refrigeration(3)`
  - `heatPipe(4)`

sunPlat Sensor

sunPlat Sensor スーパークラスは、他のハードウェア資源のプロパティーを測定するハードウェア資源の汎用的な特性を表すために使用されます。

このクラスの entPhysicalClass は `sensor(8)` です。

sunPlat Sensor クラスの属性は次のとおりです。

注 – 以下に示す属性名では、簡潔にするために接頭辞 *sunPlatSensor* を省略してあります。

- ***Class***
  この読み取り専用属性は、センサーのクラスを示す列挙型であり、次の値を取ります。
  - `binary(1)`
  - `numeric(2)`
  - `discrete(3)`

- ***Type***
  この読み取り専用属性は、センサーによって測定されるプロパティーを識別する列挙型です。*Type* に示される値の一部を表 6-9 に示します。

表 6-9　Sensor の Type の属性値

タイプ	説明
`temperature(3)`	環境温度を測定するセンサー
`voltage(4)`	電圧を測定するセンサー
`current(5)`	電流を測定するセンサー
`tachometer(6)`	装置の回転速度を測定するセンサー
`counter(7)`	定義されたイベントをカウントする汎用センサー

- ***Latency***
  この読み取り専用属性は次の情報を示します。
  - センサーがポーリングされる場合、この整数は更新間隔をミリ秒単位で表します。
  - センサーがイベントによって駆動される場合、この値は、そのイベントを処理するために予期される最長応答時間を表します。

sunPlat Binary Sensor

sunPlat Binary Sensor クラスは、バイナリ出力を返すセンサーの特性を表すために使用されます。このクラスは、sunPlatSensor テーブルを拡張してバイナリセンサー固有の属性を提供します。

このクラスの entPhysicalClass は `sensor(8)`、sunPlatSensorClass は `binary(1)` です。

sunPlat Binary Sensor クラスの属性は次のとおりです。

注 – 以下に示す属性名では、簡潔にするために接頭辞 *sunPlatBinarySensor* を省略してあります。

- ***Current***
  この読み取り専用属性は、センサーの最新値を示すブール値を取ります。
- ***Expected***
  この読み取り専用属性は、センサーの予期されている値を示すブール値を取ります。

- ***Interpret True***
  この読み取り専用属性は、センサーの `true` 値の解釈を示すテキスト文字列です。
- ***Interpret False***
  この読み取り専用属性は、センサーの `false` 値の解釈を示すテキスト文字列です。

sunPlat Numeric Sensor

sunPlat Numeric Sensor クラスは、指示値を数値で返すことができるセンサーの特性を表すために使用されます。数値センサーの値は、次に定義する測定単位によって修飾されます。

測定単位 = *Base Unit* * $10^{Exponent}$

この修飾によってミリアンペアやマイクロボルトなどの測定単位が使用できます。*Rate Unit* が定義されている場合、測定単位は次のように詳細定義されます。

測定単位 = (*Base Unit* * $10^{Exponent}$) / (*Rate Unit*)

この修飾によって rpm や km/hr などの測定単位が使用できます。

このクラスの entPhysicalClass は `sensor(8)`、sunPlatSensorClass は `numeric(2)` です。

sunPlat Numeric Sensor クラスの属性は次のとおりです。

注 – 以下に示す属性名では、簡潔にするために接頭辞 *sunPlatNumericSensor* を省略してあります。

- ***Base Units***
  この読み取り専用属性は、上記の修飾を適用する前の測定単位を示す列挙型です。この型の値の例を次に示します。
  - `degC(3)`
  - `volts(6)`
  - `amps(7)`
- ***Exponent***
  この読み取り専用属性は、10 の累乗によって *Base Unit* の精度を決定するために使用される整数です。たとえば、*sunPlatNumericSensorBaseUnits* が `volts` に設定され、*sunPlatNumericSensorExponent* が `-6` に設定されている場合、返される値の単位は `microVolts` です。

- ***Rate Units***
  この読み取り専用属性は、センサーで測定する値が絶対値 (属性値が none の場合) か割合かを示す列挙型です。後者の場合、*sunPlatNumericSensorBaseUnits* で指定された単位は「単位時間あたり」で示されます。たとえば、*sunPlatNumericSensorBaseUnits* が degC に設定され、*sunPlatNumericSensorRateUnits* が perSecond に設定されている場合、示される値の単位は degC/second です。

  この型の値の例を次に示します。

  - perMicrosecond(2)
  - perMillisecond(3)
  - perSecond(4)
  - perMinute(5)
  - perHour(6)
  - none(1)
- ***Current***
  この読み取り専用属性は、センサーの最新値を示す整数です。
- ***Normal Min***
  この読み取り専用属性は、センサー指示値の予期される最小値として定義されたしきい値を示す整数です。この値は前述の測定単位で表されます。この属性が適用されないセンサーもあります。
- ***Normal Max***
  この読み取り専用属性は、センサー指示値の予期される最大値として定義されたしきい値を示す整数です。この値は前述の測定単位で表されます。この属性が適用されないセンサーもあります。
- ***Accuracy***
  この読み取り専用属性は、センサーで測定されたプロパティーの誤差を小数点以下 2 桁までの割合で示す整数です。この値は、センサー指示値がダイナミックレンジに対して直線的であるかどうかによって変化します。
- ***Lower Non Critical Threshold***
  この読み取り専用属性は、nonCritical 条件を発生させるしきい値の下限を示す整数です。
- ***Upper Non Critical Threshold***
  この読み取り専用属性は、nonCritical 条件を発生させるしきい値の上限を示す整数です。
- ***Lower Critical Threshold***
  この読み取り専用属性は、critical 条件を発生させるしきい値の下限を示す整数です。
- ***Upper Critical Threshold***
  この読み取り専用属性は、critical 条件を発生させるしきい値の上限を示す整数です。
- ***Lower Fatal Threshold***
  この読み取り専用属性は、fatal 条件を発生させるしきい値の下限を示す整数です。

- ***Upper Fatal Threshold***
  この読み取り専用属性は、`fatal` 条件を発生させるしきい値の上限を示す整数です。
- ***Hysteresis***
  この読み取り専用属性はしきい値のヒステリシスを表します。
- ***Enabled Thresholds***
  書き込みを受けるとセンサーをデフォルト値にリセットする読み取り専用属性です。

sunPlat Discrete Sensor

sunPlat Discrete Sensor クラスは、sunPlat Numeric Sensor クラスまたは sunPlat Binary Sensor クラスで表現できないセンサーに使用されます。

このクラスの entPhysicalClass は `sensor(8)`、sunPlatSensorClass は `discrete(3)` です。

このクラスは 2 つのテーブルで構成されます。sunPlatDiscreteSensor テーブルには *sunPlatDiscreteSensorCurrent* という 1 つの属性があります。この属性は、sunPlatDiscreteSensorStates テーブルのインデックスで表した、センサーの現在の状態を示します。

sunPlat Discret Sensor クラスの属性は次のとおりです。

注 – 以下に示す属性名では、簡潔にするために接頭辞 *sunPlatDiscreteSensorState* を省略してあります。

- ***Index***
  この読み取り専用属性は、sunPlatDiscreteSensorStates テーブル内の、このセンサー状態を示す行のインデックスを表す数値を取ります。
- ***Interpretation***
  この読み取り専用属性は、sunPlatDiscreteSensorStatesTable の対応する行で表された状態を説明する文字列です。
- ***Acceptable***
  この読み取り専用属性は、テーブルのこの行で表された状態が許容可能かどうかを示すブール値を取ります。

sunPlat Chassis

sunPlat Chassis クラスは、主要な格納装置を表すために使用されます。このクラスは sunPlat Equipment クラスの特性を拡張しません。シャーシはモデル化されたすべてのハードウェア資源を格納し、他のどの資源にも格納されません。

このクラスの entPhysicalClass は `chassis(3)` です。

第7章

論理モデル

この章では、sunPlat の論理クラス階層について説明し、sunPlat モデルで定義された管理対象オブジェクトクラスが SUN-PLATFORM-MIB にどのように記述されているかを示します。

この章は以下の節で構成されています。

sunPlat 論理クラス階層

図 7-1 は、sunPlat 論理クラスの継承階層を示しています。

Logical Entity クラスでは、すべての論理オブジェクトに共通する情報が提供されます。

Unitary Computer System クラスは、モデル化されたコンピュータシステム (たとえば、Sun Fire B1600 シャーシに装着された 1 つの Sun Fire B100s ブレード) の電源状態の報告に関連したプロパティーを追加します。これらのプロパティーは強制リセットを実行するために使用することもできます。

Administrative Domain クラスは、モデル化されたシステムとの管理上の接点を示すオブジェクトを表すために使用されます。追加プロパティーはありません。Sun Fire B1600 プラットフォームの場合、このクラスはシステムコントローラを表すために使用されます。

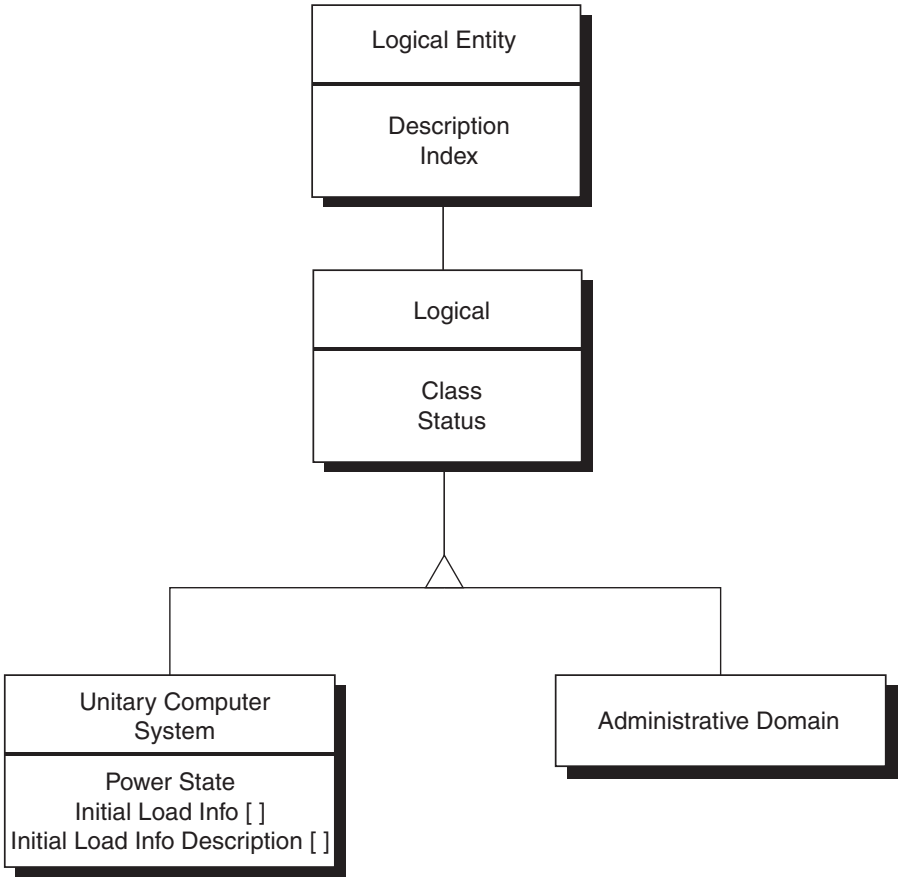

図 7-1 sunPlat 論理資源継承クラス図

sunPlat 論理クラス定義

sunPlat 論理クラスの属性は論理資源の特性を表すために使用されます。論理資源は、マルチドメインシステムにおけるドメインなどの上位オブジェクトを表します。マネージャーに対する資源の使用可能性と操作可能性が管理対象オブジェクトの状態によって表されます。さまざまな sunPlat クラスに管理対象オブジェクトの状態の特徴を表す多様な属性があります。

Logical Entity

このクラスは識別情報を提供する論理エンティティを表します。有効なオブジェクトは次のとおりです。

注 – 以下に示すオブジェクト名では、簡潔にするために接頭辞 *entLogical* を省略してあります。

- ***Description***

  このオブジェクトは管理されるオブジェクトのタイプを識別します。

- ***TAddress***

  エンティティを直接管理できる IP アドレスと UDP ポート番号を指定します。Sun Fire B1600 システムに装着する Sun Fire B100s ブレードの場合は、ブレードの IP アドレス、およびブレードの標準 Solaris SNMP エージェントと通信できるポート 161 を指定します。

Logical

このクラスは、論理エンティティが表す資源の状態のタイプを表します。このクラスには以下のオブジェクトが含まれます。

注 – 以下に示す属性名では、簡潔にするために接頭辞 *sunPlatLogical* を省略してあります。

- ***Class***

  この属性は論理クラスのタイプを示す列挙型であり、次の値を取ります。

  - `other(1)`
  - `computerSystem(2)`
  - `adminDomain(3)`

- ***Status***

  この属性は、論理クラスの状態を示す列挙型です。次の値を取ります。

  - `ok(1)`
  - `error(2)`
  - `degraded(3)`
  - `unknown(4)`
  - `predFail(5)`
  - `starting(6)`
  - `stopping(7)`
  - `service(8)`

- stressed(9)
- nonRecover(10)
- noContact(11)
- lostComm(12)
- stopped(13)

sunPlat Unitary Computer System

このクラスの特定のプロパティーは sunPlatUnitaryComputerSystemTable を使用して記述されます。このクラスの sunPlat Logical クラスは computerSystem(2) です。

このクラスには以下のオブジェクトが含まれます。

注 – 以下に示す属性名では、簡潔にするために接頭辞 *sunPlatUnitaryComputerSystem* を省略してあります。

- ***Power State***

  この属性は、読み取り時の現在の電源状態を示します。電源状態の遠隔制御を有効にすることもできます。たとえば、ブレードの電源を投入または切断したり、強制リセットを実行するために電源を再投入できます。

  この属性は次の値を取ります。

  - unknown(1)
  - fullPower(2)
  - psLowPower(3)
  - psStandby(4)
  - psOther(5)
  - powerCycle(6)
  - powerOff(7)
  - psWarning(8)
  - hibernate(9)
  - softOff(10)
  - reset(11)

- ***Apply Settings***

  このプロパティーに書き込むと、適用するデフォルトまたはカスタムの起動パラメタが有効になります。

  sunPlatUnitaryComputerSystemTable の各エントリには *sunPlatInitialLoadInfoTable* のエントリが関連付けられています。関連付けられたエントリに、現在の起動パラメタ設定と、*sunPlatUnitaryComputerSystemApplySettings* オブジェクトへの書き込みによって適用できる代替設定の両方が定義されています。

sunPlat Administrative Domain

このクラスは Logical Entity クラスにプロパティーを追加しません。したがって関連付けられた MIB オブジェクトはありません。このクラスの sunPlat 論理クラスは `adminDomain(3)` です。

第8章

sunPlat の通知

この章では、SUN-PLATFORM-MIB に定義されている SunPlat 通知クラスおよび属性について説明します。

sunPlat 通知クラスは、登録されているネットワークマネージャーにエージェントから送信される非同期メッセージです。通知を使用すると、管理対象オブジェクトへのポーリングよりも効率的な方法でイベント情報を伝達することが可能です。

この章は以下の節で構成されています。

sunPlat 通知クラス階層

図 8-1 は、sunPlat 通知クラスの継承階層を示しています。

通知クラスは、抽象クラスおよび具象クラスの両方からなる階層を使用して表され、これらのクラスに共通する属性が適用されます。

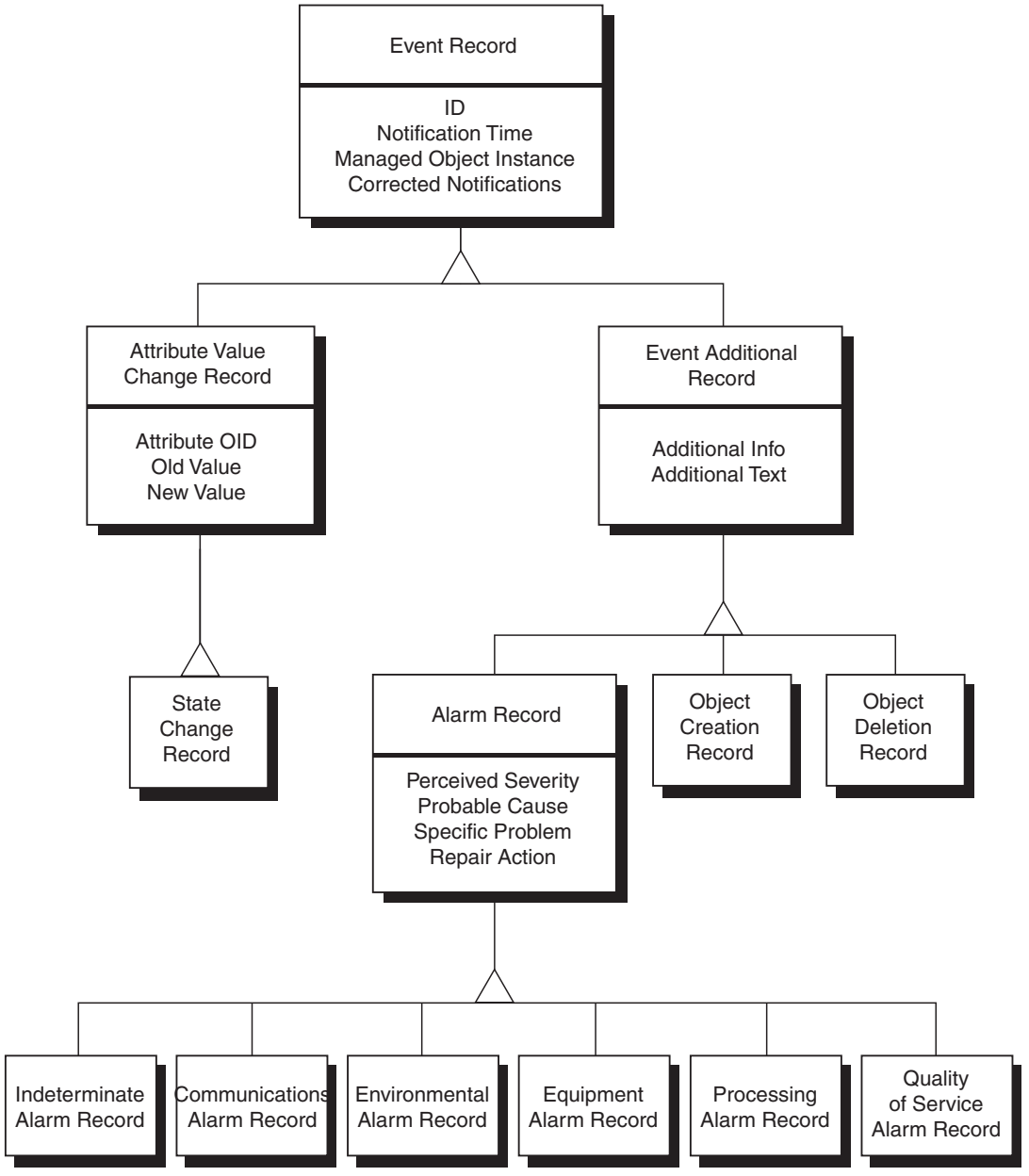

図 8-1　　イベントレコード継承クラス図

sunPlat クラス定義

sunPlat Event Record

sunPlat Event Record スーパークラスは、すべての通知に共通する属性を表します。このクラスには、このクラスに記録される特定のイベントに関係した追加情報を提供するサブクラスがあります。

sunPlat Event Record の属性は次のとおりです。

注 – 以下に示す属性名では、簡潔にするために接頭辞 *sunPlatLogRecord* を省略してあります。

- ***ID***
  この整数は通知を一意に識別するとともに、エージェントによって通知が生成された順序を示します。エージェントによる通知生成の順序は、通知を生成させたイベントの発生順序を必ずしも反映していないことに注意してください。
- ***Notification Time***
  この読み取り専用属性は、現時点では通知が生成された日時を示すタイムスタンプです。
- ***Managed Object Instance***
  この読み取り専用属性は、イベントが関連付けられている資源を表す MIB のエントリを直接参照する OID です。
- ***Correlated Notifications***
  この読み取り専用属性は、このイベントが関連付けられている他のイベントを識別する ID 値をコンマで区切ったリストです。

sunPlat Event Additional Record

sunPlat Event Additional Record スーパークラスは、次に示すイベントの発生時に生成される通知に共通する追加属性を表します。

- オブジェクト作成
- オブジェクト削除
- アラーム

このクラスには、このクラスに記録される特定のイベントに適用される追加情報を提供するサブクラスがあります。

sunPlat Event Additional Record の属性は次のとおりです。

注 – 以下に示す属性名では、簡潔にするために接頭辞 *sunPlatLogRecord* を省略してあります。

- ***Additional Info***
  この読み取り専用属性は、この通知に関係する追加情報を提供することが可能なオブジェクトの OID (オプション) です。
- ***Additional Text***
  この読み取り専用属性は、この通知に関係する追加情報を提供するテキスト文字列 (オプション) であり、ラベルと `entPhysical` 名によって、影響を受けるコンポーネントを識別します。

sunPlat Object Creation Record

sunPlat Object Creation Record クラスは、階層内の関連付けられた資源の下に資源が追加されたことを示します。この追加はホットプラグイベントによって発生する場合があります。*Additional Info* 属性には、追加された資源を表す物理エンティティテーブルエントリの OID が含まれます。

論理オブジェクトはマネージャーオブジェクトのインスタンス `0.0` のもとで作成されます。

sunPlat Object Deletion Record

sunPlat Object Deletion Record は、階層内の関連付けられた資源の下から資源が削除されたことを示します。*Additional Info* 属性には、削除された資源を表す物理エンティティテーブルエントリの OID が含まれます。

注 – この OID はすでに無効なものですが、受信側マネージャーにとっては有用です。

---

sunPlat Alarm Record

sunPlat Alarm Record スーパークラスは、アラームを示すすべての通知に共通する追加属性を表します。

このクラスには、発生したアラームのクラスを識別するサブクラスがあります。

sunPlat Alarm Record の属性は次のとおりです。

注 – 以下に示す属性名では、簡潔にするために接頭辞 *sunPlatAlarmRecord* を省略してあります。

- ***Perceived Severity***
  この読み取り専用属性は、問題が資源のサービスに及ぼした影響の度合いを示す 6 つの重要度レベルを定義する列挙型です。この属性の値は表 8-1 のとおりです。

表 8-1 sunPlat Alarm Record の Perceived Severity の値

検出された重要度	説明
`indeterminate(1)`	アラームの重要度レベルを判定できません。
`critical(2)`	サービスに影響を及ぼす条件が発生したため、ただちに訂正処置が必要です。
`major(3)`	サービスに影響を及ぼす条件が発生したため、緊急訂正処置が必要です。
`minor(4)`	サービスに影響を及ぼさない条件が発生しました。より重大な条件の発生を防ぐために訂正処置が必要です。
`warning(5)`	サービスに影響を及ぼす潜在的または緊急性を持った障害条件が検出されました。より重大な条件の発生を防ぐための措置が必要です。
`cleared(6)`	この資源に関するアラームのうち、アラームクラスが同じ、つまり *Probable Cause* と、示されている場合は *Specific Problem* が同じであるアラームをすべて消去します。

- ***Probable Cause***
  この読み取り専用属性は、アラーム生成の原因となった条件のタイプをさらに分類するオプションの列挙型です。この型の値の例を次に示します。
  - `coolingSystemFailure(134)`
  - `IODeviceError(75)`
  - `powerProblem(58)`
  - `softwareProgramError(283)`
- ***Specific Problem***
  この読み取り専用属性は、アラームの *Probable Cause* をさらに詳細に記述するオプションのテキスト文字列です。
- ***Repair Action***
  この読み取り専用属性は、推奨される修復処置をリストした文字列です。

sunPlat Indeterminate Alarm Record

sunPlat Indeterminate Alarm Record クラスは、sunPlat Alarm Record クラスによって提供される情報を拡張しません。このクラスは、次のいずれのクラスにも属さないアラームを記録するために使用されます。

sunPlat Communications Alarm Record

sunPlat Communications Alarm Record クラスは、sunPlat Alarm Record クラスによって提供される情報を拡張しません。このクラスは、関連付けられた資源で通信エラーが検出されたことを記録するために使用されます。

sunPlat Environmental Alarm Record

sunPlat Environmental Alarm Record クラスは、sunPlat Alarm Record クラスによって提供される情報を拡張しません。このクラスは、関連付けられた資源で環境の問題が検出されたことを記録するために使用されます。

sunPlat Equipment Alarm Record

sunPlat Equipment Alarm Record クラスは、sunPlat Alarm Record クラスによって提供される情報を拡張しません。このクラスは、関連付けられた資源で障害が検出されたことを記録するために使用されます。

sunPlat Processing Alarm Record

sunPlat Processing Alarm Record クラスは、sunPlat Alarm Record クラスによって提供される情報を拡張しません。このクラスは、関連付けられた資源でソフトウェアまたは処理の障害が検出されたことを記録するために使用されます。

sunPlat Quality of Service Alarm Record

sunPlat Quality of Service Alarm Record クラスは、sunPlat Alarm Record クラスによって提供される情報を拡張しません。このクラスは、関連付けられた資源でサービス品質の変化が検出されたことを記録するために使用されます。

sunPlat Attribute Value Change Record

sunPlat Attribute Value Change Record スーパークラスは、関連付けられた資源の属性の変更を示す通知に共通する追加属性を表します。

このクラスには、可能な属性タイプごとにサブクラスがあります。

sunPlat Attribute Value Change Record の属性は次のとおりです。

注 – 以下に示す属性名では、簡潔にするために接頭辞 *sunPlatLogRecordChange* を省略してあります。

- ***OID***
  この読み取り専用属性は、値が変更された管理対象オブジェクトの属性を示す、物理エンティティテーブルまたは論理エンティティテーブルのオブジェクトを直接参照する OID です。

変更された属性の構文に応じて、新しい値と前の値が次のいずれかのオブジェクトの組を使用して表されます。

- ***New Integer***
  この読み取り専用属性は、変更された管理対象オブジェクトの属性の新しい INTEGER 値を識別します。型は、変更された属性の型に応じて、符号付きまたは符号なしです。
- ***Old Integer***
  この読み取り専用属性は、変更された管理対象オブジェクトの属性の前の INTEGER 値を識別します。型は、変化した属性の型に応じて、符号付きまたは符号なしです。
- ***New String***
  この読み取り専用属性は、属性変更通知に含まれる新しい OCTET STRING 値を識別します。
- ***Old String***
  この読み取り専用属性は、属性変更通知に含まれる前の OCTET STRING 値を識別します。
- ***New OID***
  この読み取り専用属性は、属性変更通知に含まれる新しい OCTET IDENTIFIER 値を識別します。
- ***Old OID***
  この読み取り専用属性は、属性変更通知に含まれる前の OCTET IDENTIFIER 値を識別します。

sunPlat State Change Record

sunPlat State Change Record クラスは、sunPlat Attribute Value Change Record クラスによって提供される情報を拡張しません。このクラスは、資源の「状態」の特徴を反映した、管理対象オブジェクトの属性の変更を示すために使用されます。

PART II　インストールと構成

第9章

管理ソフトウェアのコンポーネント

この章では、Sun Fire B1600 用の管理ソフトウェアを構成するコンポーネントについて説明し、SNMP ソフトウェアをインストールするための要件を示します。

この章は以下の節で構成されています。

システム管理オプション

Sun Fire B1600 には次のシステム管理オプションが用意されています。

- SNMP を使用したシステムの監視と制御
- セキュリティー保護された管理をサポートする SNMPv3 機能
- Sun Management Center 3.0 を使用したシステム監視[1]

1. インストールと構成を含む詳細な説明については、『Sun Management Center 3.0 Sun Fire B1600 のための追補マニュアル』(Part No. 817-2538-10) を参照してください。

計測機構

プラットフォームのタイプに応じて、以下のエージェントを採用できます。

- Sun Fire B100s ブレード上で動作するドメインエージェント (ドメインハードウェア監視)

  監視対象のサーバー上にソフトウェアをローカルにインストールします。監視できるのはそのサーバーだけです。Sun Fire B1600 の場合、各ブレードが個別に監視され、エージェントの各インスタンスで 1 つのブレードのみを監視できます。

- システムコントローラをプロキシとして使用するプラットフォームエージェント (プラットフォームハードウェア監視)

  システムコントローラを通してプラットフォームの命令にアクセスする遠隔サーバーにソフトウェアをインストールします。そのため、システムコントローラで管理するすべてのハードウェアを監視できます。Sun Fire B1600 の場合、ブレードを装着したシェルフ全体を監視できます。これには任意のタイプのすべてのブレード、電源装置およびシステムコントローラが含まれます。

図 9-1 では、Sun Fire B1600 シェルフ A および B にプラットフォームハードウェア監視、Sun Fire B1600 シェルフ C にドメインハードウェア監視を採用しています。

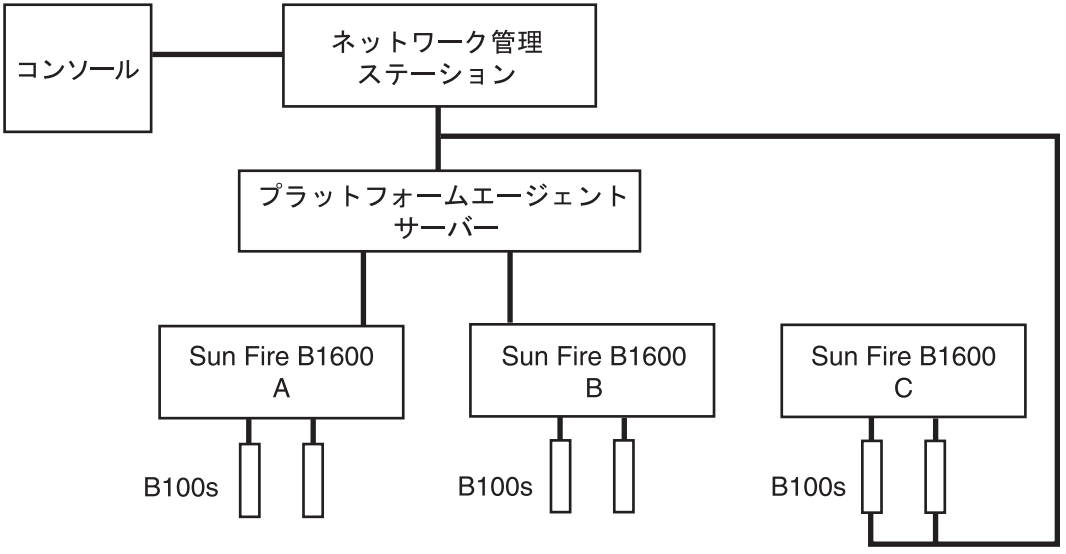

図 9-1 ドメインハードウェア監視とプラットフォームハードウェア監視の例

システム要件

SNMP Management Agent をインストールする前に、この節で説明する必要条件と依存条件をシステムが満たしていることを確認してください。

オペレーティング環境

SNMP Management Agent ソフトウェアを使用するには Solaris 8 Update 3 以降が必要です。

必要な空きディスク容量

プラットフォームエージェントサーバー上に 512Mバイト以上 (1.0Gバイトを推奨)

パッチ

標準の Solaris オペレーティング環境ソフトウェアに加えて、次のパッチをインストールする必要があります。

Solaris 8

Sun Fire B100s ブレード用のパッチは必要ありません。

SNMP Management Agent ソフトウェアをインストールする前に、プラットフォームエージェントサーバーおよびブレード (プラットフォームハードウェア監視およびドメインハードウェア監視の両方の場合) に Java 1.4 をインストールする必要があります (68 ページの「Java 環境」を参照)。

Solaris 9

パッチは必要ありません。

Java 環境

Sun Fire B100s ブレードを完全に監視するには、監視対象の各 Sun Fire B100s ブレードおよびプラットフォームエージェントサーバーに Java J2SE 1.4 のコンポーネントをあらかじめインストールしておく必要があります。

注 – このインストールを実行すると、既存の J2SE ソフトウェアがアップグレードされます。ソフトウェアをアップグレードしたくない場合、たとえばデフォルトの J2SE バージョン 1.3.1 で動作保証されているアプリケーションがある場合などは、デフォルトシステムである J2SE と共存するように J2RE をインストールできます。そのためには、Sun SNMP Management Agent ソフトウェアに追加の構成を行う必要があります。これについては付録 A で説明します。

ターゲットの計測機構がない Sun Fire B1600 シェルフのみを監視する場合は、Java J2SE 1.4 のコンポーネントをプラットフォームエージェントサーバーに事前インストールするだけでかまいません。この場合、ハードディスクドライブ、CPU 情報および Ethernet MAC アドレスの計測機構は使用できません。

Java 1.4 ファイルを適切な場所 (`/usr/j2se`) にインストールするには、`j2sdk-1_4_0_03-solaris-sparc.tar.Z` パッケージを使用します。

ファイルは次の場所から入手できます。

`http://java.sun.com/j2se/1.4/download.html`

`Solaris SPARC 32-bit tar.Z` の SDK をダウンロードするように選択します。

上記のページに示されているダウンロードの手順に従います。

注 – このファイル名は、マニュアルの作成時点のものです。ファイルのバージョンが最新であることを確認する必要があります。ファイル名の形式は `j2sdk-1_4_0_<ver>-solaris-sparc.tar.Z` で、`<ver>` はソフトウェアのバージョンです。

このパッケージをインストールするとシステムの J2SE が置き換えられるため、既存の Java アプリケーションを継続して正常に動作させるには、64 ビットの J2SE 1.4 パッケージもインストールする必要があります。このパッケージは `j2sdk-1_4_0_<ver>-solaris-sparcv9.tar.Z` ファイルに含まれています。

注意 – Solaris 8 では、J2SE 1.3.1 が J2SE 1.4 に置き換えられるため、J2SE 1.3.1 をアンインストールしてから J2SE 1.4 をインストールする必要があります。その後、四半期に 1 度の Solaris 8 のアップデートをインストールすると、J2SE 1.4 パッケージの一部が J2SE 1.3.1 パッケージで上書きされます。J2SE 1.4 を適切な場所にインストールするには、`pkgadd` を使用します。

インストールの確認

インストールが正しく行われたことを確認するには、次のコマンドを実行します。

```
# /usr/j2se/bin/java -version
java version "1.4.1_03"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.1_03-
b04)
Java HotSpot(TM) Client VM (build 1.4.1_03-b04, mixed mode)
```

システムにインストールされているバージョンが報告されます。

Java SNMP API

インストールパッケージには、Java SNMP API である `SUNWjsnmp` の新しいバージョンが収められています。`pkgrm` を使用してこのパッケージの既存バージョンを削除してから、Management Agent ソフトウェアをインストールしてください。

インストールパッケージ

管理ソフトウェアを構成するパッケージは、次のグループに分類できます。

- ドメインハードウェア監視に必要なパッケージ
- プラットフォームエージェントサーバーでのプラットフォームハードウェア監視に必要なパッケージ
- ターゲットマシンでのプラットフォームハードウェア監視に必要なパッケージ

これらのパッケージを表 9-1 に示します。

表 9-1 SNMP Management Agent ソフトウェアパッケージの説明

パッケージ	パッケージ名	機能
SUNWbgpc	SPA Personality Module Framework	パーソナリティモジュールをサポートするフレームワーク
SUNWbgptk	SPA Personality Module Toolkit	再利用可能なコンポーネントモデルのライブラリとデータアクセスライブラリ
SUNWbgpr	SPA Personality Module (root)	RDP 起動スクリプト
SUNWbgcm	SPA HW Platform Object Manager	プラットフォームオブジェクトマネージャー
SUNWbgcmr	SPA HW Platform Object Manager (root)	プラットフォームオブジェクトマネージャー起動スクリプト
SUNWbgpm	SPA SNMP Protocol Mediator/Master Agent	SNMP のプロトコルサポートとマスターエージェント
SUNWbgpmr	SPA SNMP Protocol Mediator/Master Agent (root)	SNMP コンポーネント起動スクリプト
SUNWbgidr	SPA Domain Discovery (root)	Domain Agent Discovery 起動スクリプト
SUNWbgod	SPA Platform Discovery	Platform Agent Discovery デーモン
SUNWbgodr	SPA Platform Discovery (root)	Platform Agent Discovery 起動スクリプト
SUNWbgpji	SPA Sun Fire B100s Domain Personality Module	B100s ドメイン計測機構
SUNWbgpjo	SPA Sun Fire B1600 Platform Personality Module	B1600 プラットフォーム計測機構

注 – パッケージ間には内部的な依存関係があるため、パッケージは特定の順序でインストールする必要があります (77 ページの「SNMP ソフトウェアのインストール」を参照)。

ソフトウェアのアップグレード

ソフトウェアをアップグレードするには、新しいバージョンをインストールする前に既存のソフトウェアを削除する必要があります (第 13 章を参照)。

パッケージの配布

パッケージは tar アーカイブバンドルで提供されます。表 9-2 は、`SUNWspa.1.0.tar.Z` アーカイブバンドルの内容を示しています。この tar アーカイブバンドルには、Sun Fire B1600 のプラットフォーム管理および Sun Fire B100s のドメイン管理のための SNMP サポートを提供するパッケージが収められています。

展開したパッケージは表に示したディレクトリに格納されます。ソフトウェアの詳細なインストール手順については、第 10 章を参照してください。

注 – ファイルのバージョンが最新であることを確認する必要があります。

表 9-2　SNMP Management Agent パッケージのバンドル

バンドル	説明	内容
SUNspa.1.0.22.tar.Z	プラットフォームエージェントサーバーにインストールする SNMP プラットフォームエージェント用のパッケージ	platform/proxy/SUNWbgpc platform/proxy/SUNWbgpk platform/proxy/SUNWbgcm platform/proxy/SUNWbgcmr platform/proxy/SUNWbgod platform/proxy/SUNWbgodr platform/proxy/SUNWbgpjo platform/proxy/SUNWbgpm platform/proxy/SUNWbgpmr platform/proxy/SUNWjdrt platform/proxy/SUNWjsnmp
	Sun Fire B100s ブレードにインストールする SNMP 計測機構用のパッケージ	platform/target/SUNWbgpc platform/target/SUNWbgptk platform/target/SUNWbgpr platform/target/SUNWbgcm
	Sun Fire B100s ブレードにインストールする SNMP ドメインエージェント用のパッケージ	domain/SUNWbgpc domain/SUNWbgptk domain/SUNWbgcm domain/SUNWbgcmr domain/SUNWbgidr domain/SUNWbgpji domain/SUNWbgpm domain/SUNWbgpmr domain/SUNWjdrt domain/SUNWjsnmp

tar ファイルを展開するには、次のように入力します。

```
$ zcat SUNWspa.1.0.tar.Z | tar xf -
```

ドメイン用またはターゲット用パッケージの Sun Fire B100s へのインストール

ドメイン用 (ドメインハードウェア監視の場合) またはターゲット用 (プラットフォームハードウェア監視の場合) のパッケージを Sun Fire B100s ブレードにインストールする必要があります。これを行うには、次に示す複数の方法があります。

- 個々の Sun Fire B100s ブレード上にドメイン用またはターゲット用のパッケージを展開する
- ソフトウェアのインストール先となる各 Sun Fire B100s ブレードの root ユーザーがアクセスできる共有ディレクトリに、ドメイン用またはターゲット用のパッケージを展開する
- ドメイン用またはターゲット用のパッケージをネットワーク経由でインストールできるように設定する

システムファイルへの影響

表 9-3 に示すように、複数の新しい起動ファイルが `/etc/init.d` に作成され、`/etc/rc<n>.d` へのリンクが設定されます。

表 9-3 起動スクリプト

コンポーネント	起動スクリプト	パッケージ名	パッケージの説明
Platform Object Manager	`spapom`	`SUNWbgcmr`	Platform Object Manager (root)
Domain Hardware Discovery Module	`spaibdm`*	`SUNWbgidr`	Domain Hardware Discovery Module (root)
Remote Data Plug-ins	`spardp`†	`SUNWbgpr`	Personality Module (root)
SNMP Protocol Mediator/Master Agent	`spama`	`SUNWbgpmr`	SSNMP Protocol Mediator SNMP Master Agent (root)

* ドメイン用パッケージをインストールした Sun Fire B100s ブレードの場合のみ。

† プラットフォーム用またはターゲット用のパッケージをインストールした Sun Fire B100s ブレードの場合のみ。

Discovery Module は `inetd` で自動的に起動され、それによって `/etc/inetd.conf` ファイルに新しいエントリが作成されます。

Platform Object Manager (POM) は、IP ポートのアクティビティを監視してクライアントからの要求を検出します。監視対象ポートは `/etc/services` ファイルに登録されたポートです。

第 10 章

インストール

この章では、Sun Fire B1600 への管理ソフトウェアのインストール方法を説明します。

この章は以下の節で構成されています。

インストールの種類の選択

インストールするソフトウェアの構成を決定する場合、主な考慮事項が 2 つあります。

1. 計測機構の構成
2. 管理インタフェースの構成

計測機構の構成

プラットフォームのタイプに応じて、以下のエージェントを採用できます。

- 監視対象システム上で動作するドメインエージェント

  監視対象の Sun Fire B100s ブレード (ドメイン) にソフトウェアをローカルにインストールします。各ブレードは個別に監視され、一度に 1 つのブレードだけを監視できます。これは「ドメインハードウェア監視」と呼ばれます。

■ システムコントローラをプロキシとして使用するプラットフォームエージェント

Sun Fire B1600 システムコントローラを通してプラットフォームの命令にアクセスする遠隔 (プラットフォームエージェント) サーバー、および監視対象の Sun Fire B100s (ターゲット) ブレードにソフトウェアをインストールします。そのため、システムコントローラで管理するすべてのハードウェアを監視できます。これには電源装置、システムコントローラおよびすべてのブレードが含まれます。これは「プラットフォームハードウェア監視」と呼ばれます。

プラットフォームハードウェア監視を実施する場合は、ハードディスクドライブ、CPU、および Ethernet MAC アドレスに関する情報を取得するために、監視対象の各 Sun Fire B100s ブレードにターゲット用のパッケージをインストールする必要があります。

注 – Sun Fire B100s ブレードは、ドメインハードウェア監視では「ドメイン」と呼ばれ、プラットフォームハードウェア監視では「ターゲット」と呼ばれます。

---

66 ページの「計測機構」も参照してください。

管理インタフェースの構成

管理ソフトウェアはさまざまな方法で配置できるように設計されており、現時点では次の配置方法がサポートされています。

■ マスターエージェントなしで `snmpdx` を使用する SNMP (これがデフォルトのインストールです)

■ マスターエージェントと `snmpdx` を使用する SNMP

この場合は、第 11 章および第 12 章に従ってインストールを手動で構成する必要があります。

SNMP ソフトウェアのインストール

この節では、監視ソフトウェアのインストール手順の概要を示します。インストールの種類ごとの詳細な手順については、この章の以降の節で説明します。

このソフトウェアをインストールする前に、次の作業が完了していることを確認してください。

- ドメインまたはターゲット (Sun Fire B100s) とプラットフォームエージェントサーバーの両方に、必要なレベルの Solaris をインストールする (67 ページの「オペレーティング環境」を参照)。
- 必要なすべてのパッチ (67 ページの「パッチ」を参照) と、SNMP ソフトウェアに含まれていないその他の必須パッケージ (68 ページの「Java 環境」) をインストールする。
- 68 ページの「Java 環境」に従って既存の J2SE をアップグレードするか J2RE を別個にインストールして、Java 1.4 をインストールする。

システムがこれらの要件をすべて満たしていることを確認できたら、SNMP ソフトウェアのインストールに進むことができます。

ここで、ドメインハードウェア監視を実施するか、プラットフォームハードウェア監視を実施するかを決定する必要があります。

- ドメインハードウェア監視を選択する場合は、77 ページの「ドメインハードウェア監視のためのソフトウェアのインストール」にある手順に従います。
- プラットフォームハードウェア監視を選択する場合は、79 ページの「プラットフォームハードウェア監視のためのソフトウェアのインストール」にある手順に従います。

ドメインハードウェア監視のためのソフトウェアのインストール

監視対象の各ブレードに必要なパッケージ (73 ページの「ドメイン用またはターゲット用パッケージの Sun Fire B100s へのインストール」を参照) をインストールします。

▼ ソフトウェアをインストールする

1. **Java 1.4 をインストール済みであることを確認します。**
   68 ページの「Java 環境」を参照してください。
2. **既存バージョンの `SUNWjsnmp` を削除したことを確認します。**
   69 ページの「Java SNMP API」を参照してください。

3. Sun Fire B100s ブレードにドメインエージェントパッケージをインストールします。依存関係の問題を避けるために、示されている順序でインストールしてください。

```
# pkgadd -d . SUNWbgptk SUNWbgpc SUNWbgcm SUNWbgcmr SUNWbgidr \
SUNWbgpji SUNWjsnmp SUNWjdrt SUNWbgpm SUNWbgpmr
```

4. Java 環境を構成します。

   a. 68 ページの「Java 環境」に従って J2SE 1.4 をインストールした場合は、この手順を省略してください。

   b. 127 ページの「J2RE 1.4 のインストール」に従って J2RE 1.4 をインストールした場合は、129 ページの「ドメインハードウェア監視」に示すようにドメインハードウェア監視の起動スクリプトを編集します。

5. ソフトウェアを構成します。

   第 11 章を参照してください。

6. Sun Fire B100s ブレードを再起動します。

   以上でドメインハードウェア監視のためのソフトウェアのインストールが完了しました。85 ページの「インタフェースのオプション」から続行してください。

7. 次のように入力して、プロセスが正常に起動されたことを確認します。

```
# ps -ef | grep spa.snmp
    root 15789     1  1 13:44:01 pts/2    0:00 /usr/j2se/bin/java
-Dcom.sun.spa.snmp.LOG_LEVEL=INFO -Djdmk.security.file=//etc
#
# ps -ef | grep spa.wbem

    root   278     1  0   Feb 24 ?       44:19 /usr/j2se/bin/java
    -Dcom.sun.spa.wbem.pomi.port=3333 -Xms64m -Xmx768m -Dcom.sun
#
```

   上記のように出力された場合、プロセスは実行中です。

プラットフォームハードウェア監視のためのソフトウェアのインストール

- Java 1.4 を Sun Fire B100s ブレードにインストールするかどうかを決定します (68 ページの「Java 環境」の説明を参照)。ターゲット計測機構を使用するには、Java 1.4 をインストールする必要があります。
  - ターゲット計測機構を使用するソフトウェアをインストールするには、79 ページの「ターゲット計測機構を使用するソフトウェアをインストールする」からインストール手順を開始します。
  - ターゲット計測機構を使用しないソフトウェアをインストールするには、82 ページの「ターゲット計測機構を使用しないソフトウェアをインストールする」からインストール手順を開始します。
- プラットフォームエージェントサーバーにプラットフォームエージェント用のパッケージをインストールします。
- 必要な場合は、監視対象の各ブレードにターゲットプラットフォームエージェント用のパッケージをインストールします。
- システムコントローラの SMS IP アドレスを設定します。

▼ ターゲット計測機構を使用するソフトウェアをインストールする

1. **プラットフォームのエージェントとして動作するサーバーに Java 1.4 をインストール済みであることを確認します。**

   68 ページの「Java 環境」およびこの後の手順 4 を参照してください。

2. **プラットフォームエージェントサーバーから既存バージョンの `SUNWjsnmp` を削除したことを確認します。**

   69 ページの「Java SNMP API」を参照してください。

3. **プラットフォームエージェントサーバーにプラットフォームエージェント用のパッケージをインストールします。依存関係の問題を避けるために、示されている順序でインストールしてください。**

```
# pkgadd -d . SUNWbgptk SUNWbgpc SUNWbgcm SUNWbgcmr SUNWbgod \
SUNWbgodr SUNWbgpjo SUNWjsnmp SUNWjdrt SUNWbgpm SUNWbgpmr
```

4. Java 環境を構成します。

   a. 68 ページの「Java 環境」に従って J2SE 1.4 をインストールした場合は、この手順を省略してください。

   b. 127 ページの「J2RE 1.4 のインストール」に従って J2RE 1.4 をインストールした場合は、130 ページの「プラットフォームハードウェア監視」に従ってプラットフォームハードウェア監視の起動スクリプトを編集します。

5. **ソフトウェアを構成します。**

   第 11 章を参照してください。

6. **次のように入力して、プラットフォームエージェントを手動で起動します。**

```
# /etc/init.d/spapom start
# /etc/init.d/init.snmpdx stop
# /etc/init.d/spama stop
# /etc/init.d/snmpdx start
# pkill -1 inetd
```

   または、プラットフォームエージェントサーバーを再起動します。

7. **次のように入力して、プロセスが正常に起動されたことを確認します。**

```
# ps -ef | grep spa.snmp
    root 15789     1  1 13:44:01 pts/2    0:00 /usr/j2se/bin/java
-Dcom.sun.spa.snmp.LOG_LEVEL=INFO -Djdmk.security.file=//etc
#
# ps -ef | grep spa.wbem

    root   278     1  0   Feb 24 ?       44:19 /usr/j2se/bin/java
    -Dcom.sun.spa.wbem.pomi.port=3333 -Xms64m -Xmx768m -Dcom.sun
#
# netstat -a | grep mismi
     *.mismi            *.*              0      0 24576      0 LISTEN
      *.mismi                             *.*  0
0 24576      0 LISTEN
#
```

   上記のように出力された場合、プロセスは実行中です。

8. **監視対象のターゲット Sun Fire B100s ブレードに Java 1.4 をインストール済みであることを確認します。**

   68 ページの「Java 環境」およびこの後の手順 11 を参照してください。

9. **ターゲットブレードから既存バージョンの** `SUNWjsnmp` **を削除したことを確認します。**

   69 ページの「Java SNMP API」を参照してください。

10. **ターゲットブレードにプラットフォームエージェント用のパッケージをインストールします。**

    これらのパッケージをインストールすると、監視対象マシンの Solaris インタフェースを使用して計測機構データにアクセスできるようになります。

    依存関係の問題を避けるため、これらのパッケージは示されている順序でインストールしてください。

```
# pkgadd -d . SUNWbgptk SUNWbgpc SUNWbgcm SUNWbgpr
```

11. **Java 環境を構成します。**

    a. 68 ページの「Java 環境」に従って J2SE 1.4 をインストールした場合は、この手順を省略してください。

    b. 127 ページの「J2RE 1.4 のインストール」に従って J2RE 1.4 をインストールした場合は、130 ページの「プラットフォームハードウェア監視」に従ってターゲットハードウェア監視の起動スクリプトを編集します。

12. **次のように入力して、ターゲット計測機構を手動で起動します。**

```
# /etc/init.d/spardp start
```

    または、システムを再起動します。

13. **次のように入力して、プロセスが正常に起動されたことを確認します。**

```
# netstat -an | grep 1099
      *.1099        *.*          0      0 24576      0 LISTEN
```

    上記のように出力された場合、プロセスは実行中です。

14. `setupsc` **を使用して SMS IP アドレスを設定します。**

    83 ページの「システムコントローラの構成」から続行してください。

▼ ターゲット計測機構を使用しないソフトウェアをインストールする

1. プラットフォームのエージェントとして動作するサーバーに Java 1.4 をインストール済みであることを確認します。

   68 ページの「Java 環境」を参照してください。

2. プラットフォームエージェントサーバーから既存バージョンの `SUNWjsnmp` を削除したことを確認します。

   69 ページの「Java SNMP API」を参照してください。

3. プラットフォームエージェントサーバーにプラットフォームエージェント用のパッケージをインストールします。依存関係の問題を避けるために、示されている順序でインストールしてください。

```
# pkgadd -d . SUNWbgptk SUNWbgpc SUNWbgcm SUNWbgcmr SUNWbgod \
SUNWbgodr SUNWbgpjo SUNWjsnmp SUNWjdrt SUNWbgpm SUNWbgpmr
```

4. Java 環境を構成します。

   a. 68 ページの「Java 環境」に従って J2SE 1.4 をインストールした場合は、この手順を省略してください。

   b. 127 ページの「J2RE 1.4 のインストール」に従って J2RE 1.4 をインストールした場合は、130 ページの「プラットフォームハードウェア監視」に従ってプラットフォームハードウェア監視の起動スクリプトを編集します。

5. ソフトウェアを構成します。

   第 11 章を参照してください。

6. 次のように入力して、プラットフォームエージェントを手動で起動します。

```
# /etc/init.d/spapom start
# /etc/init.d/init.snmpdx stop
# /etc/init.d/spama stop
# /etc/init.d/init.snmpdx start
# pkill -1 inetd
```

   または、プラットフォームエージェントサーバーを再起動します。

7. 次のように入力して、プロセスが正常に起動されたことを確認します。

```
# ps -ef | grep spa.snmp
    root 15789     1  1 13:44:01 pts/2    0:00 /usr/j2se/bin/java
-Dcom.sun.spa.snmp.LOG_LEVEL=INFO -Djdmk.security.file=//etc
#
# ps -ef | grep spa.wbem

    root   278     1  0   Feb 24 ?       44:19 /usr/j2se/bin/java
    -Dcom.sun.spa.wbem.pomi.port=3333 -Xms64m -Xmx768m -Dcom.sun
#
# netstat -a | grep mismi
     *.mismi            *.*              0      0 24576      0 LISTEN
      *.mismi                              *.*  0
0 24576      0 LISTEN
#
```

上記のように出力された場合、プロセスは実行中です。

8. setupsc を使用して SMS IP アドレスを設定します。

続けて以下の手順を実行します。

システムコントローラの構成

SNMP ソフトウェアをインストールした後、システムコントローラの SMS IP アドレスとして、プラットフォームエージェントサーバーの SMS IP アドレスを設定する必要があります。この設定を行うには、システムコントローラのコンソールにログオンして setupsc を実行し、プラットフォームエージェントサーバーの IP アドレスを追加します。

次に示す例では、IP アドレスを 10.5.1.1 に設定しています。

次の行が表示されるまで、確認メッセージすべてに ENTER キーを押して現在の値をそのまま使用します。

```
Enter the SMS IP address
```

IP アドレスを入力して ENTER を押し、以降の確認メッセージへの応答として ENTER を押します。

注 – setupsc コマンドについては、『Sun Fire B1600 ブレードシステムシャーシソフトウェア設定マニュアル』に説明があります。

コード例 10-1　SMS IP アドレスの設定

```
hornet-sc>setupsc
Entering Interactive setup mode.
Use Ctrl-z to exit & save. Use Ctrl-c to abort.
Do you want to configure the enabled interfaces [y]?
Should the SC network interface be enabled [y]?
Should the SC telnet interface be enabled for new connections [y]?
Do you want to configure the network interface [y]?
Should the SC use DHCP to obtain its network configuration [n]?
Enter the SC IP address [129.156.174.140]:
Enter the SC IP netmask [255.255.255.0]:
Enter the SC IP gateway [129.156.174.1]:
Do you want to configure the SC private addresses [y]?
Enter the SSC0/SC IP private address [129.156.174.118]:
Enter the SSC1/SC IP private address [129.156.174.128]:
Do you want to enable a VLAN for the SC [n]?
Enter the SMS IP address [0.0.0.0]: 10.5.1.1
<truncated>

hornet-sc>
```

インタフェースのオプション

デフォルトのインストールでは、snmpdx のサブエージェントとして動作する SNMP を使用した管理が可能です。構成後に配置をカスタマイズできますが、インストール中はユーザー入力は要求されません。

構成ファイルを編集して、SNMP と snmpdx にマスターエージェントの機能を追加できます。

注 – すべての種類のインストールで、SNMP アクセス制御リスト (ACL) はアクセスを禁止するデフォルトの構成になります。これらの ACL を、アクセスを有効にするように構成する必要があります (第 11 章を参照)。

---

snmpdx を使用する SNMP (デフォルト)

このオプションでは、自動的に割り当てられる メディエータ宛ての要求の送り先 UDP ポート番号を使用して、SNMP メディエータを snmpdx のサブエージェントとして登録します。メディエータ宛ての要求は snmpdx を経由して送信されるか、または メディエータの ACL ファイルで有効になっている場合は (第 12 章を参照)、図 10-1 に示す点線のように直接送信されます。

表 10-1 図 10-1 のポートの要約

番号	機能	spama.conf 内のパラメタ	デフォルト
1	snmpdx が SNMP メディエータ宛ての要求を転送する転送先ポート	SPAPM_REQ_PORT	
2	メディエータが SNMP マネージャー宛てのトラップを送信する送信先ポート	SPAPM_TRAP_PORT	162

注 – デフォルトの構成は自動的に行われますが、snmpdx または メディエータ(あるいはその両方) の ACL ファイルを、マネージャーの構成をサポートするように構成する必要があります。

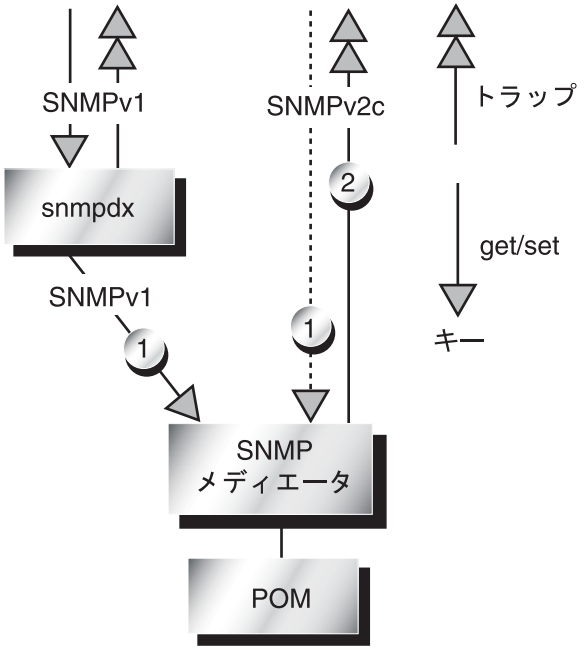

図 10-1　SNMP が `snmpdx` のサブエージェントである場合のデータフロー

マスターエージェントと snmpdx を使用する SNMP

このオプションでは、マスターエージェントの SNMPv3 セキュリティー機能が SNMP と snmpdx に追加されます。snmpdx の自動起動は無効になり、マスターエージェントはポート 161 に登録されます。snmpdx に新しいポート番号が自動的に割り当てられます。

メディエータから送信されるトラップは、オプションでマスターエージェントによって SNMPv3 に変換されるか、または直接送信されます。

snmpdx から送信されるトラップは、SNMP マネージャーに直接転送されるだけであり、マスターエージェントによる変換はできません。

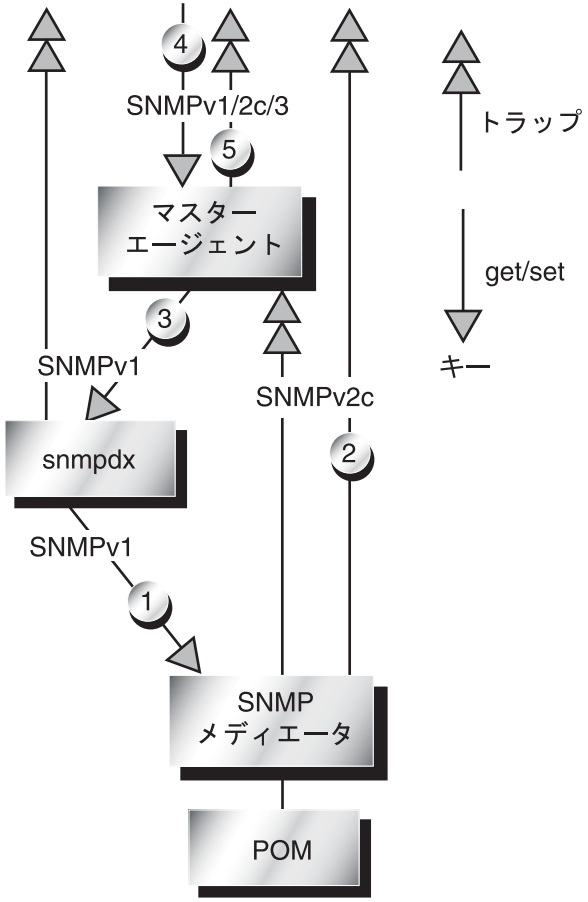

図 10-2　マスターエージェントを採用する場合のデータフロー

表 10-2　図 10-2 のポートの要約

番号	機能	spama.conf 内のパラメタ	デフォルト
1	snmpdx がサブエージェントである SNMP メディエータへ要求を転送する送信先ポート	SPAPM_REQ_PORT	
2	メディエータが SNMP マネージャーへトラップを直接送信する送信先ポート	SPAPM_TRAP_PORT	162
3	マスターエージェントが snmpdx へ要求を転送する送信先ポート	SNMPDX_REQ_FORWARD_PORT	
4	マスターエージェントが要求を監視するポート	MASTER_AGENT_REQ_PORT	161
5	マスターエージェントが SNMP マネージャーへトラップを送信する送信先ポート	SPAPM_TRAP_PORT	162

サードパーティのマスターエージェントと SNMP

このオプションでは、手動で割り当てるか サードパーティのマスターエージェントによって割り当てられるポート番号を使用して、SNMP メディエータをサブエージェントとして登録します。直接アクセスを有効にするには、第 12 章に従って メディエータの ACL ファイルを手動で構成する必要があります。

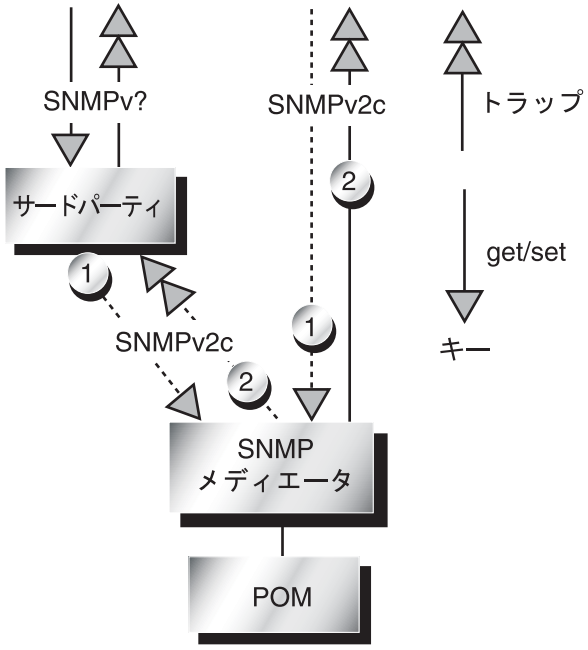

図 10-3　サードパーティのマスターエージェントを採用する場合のデータフロー

表 10-3　図 10-3 のポートの要約

番号	機能	spama.conf 内のパラメタ	デフォルト
1	直接アクセスを有効にする SNMP メディエータが使用するポート、またはサードパーティのマスターエージェントが SNMP メディエータへ要求を転送する送信先ポート	SPAPM_REQ_PORT	
2	SNMP メディエータが SNMPv2c トラップをサードパーティのマスターエージェントに送信するため、または SNMP マネージャーに直接送信するために使用するポート	SPAPM_TRAP_PORT	

第 11 章

構成ファイル

この章では、このソフトウェアを構成するために編集できるファイルの概要を説明します。構成可能なパラメタの一覧を示し、アクセス制御の概念を簡単に説明します。

第 12 章を参照する前に、この章を読んでください。第 12 章では、この章で説明するファイルを使用して SNMP オプションを構成する方法について説明しています。

この章は以下の節で構成されています。

注 – SNMP パッケージとそのインストール方法については、第 9 章および第 10 章を参照してください。

構成ファイル

次のファイルは /etc/opt/SUNWspa/ に格納され、SNMP の構成を決定します。

- 全般的な構成ファイル
  - spama.conf—マスターエージェントとメディエータがどのように構成されているかを定義します。
- メディエータの構成ファイル
  - spapm.acl—メディエータのアクセス制御を定義します。
  - spapm_snmpdx.acl—snmpdx のサブエージェントとしてのメディエータのアクセス制御を定義します。
- マスターエージェントの構成ファイル

  マスターエージェントを使用しない場合、これらのファイルを構成する必要はありません。
  - spama.acl—マスターエージェントのアクセス制御を定義します。
  - spama.uacl—マスターエージェントに対する SNMPv3 ユーザーおよびコンテキストのアクセス制御を定義します。
  - spama.security—spama.uacl で参照されている SNMPv3 ユーザーを定義します。

これらのファイルの詳細について、以降の節で説明します。

全般的な構成ファイル

spama.conf

spama.conf ファイルには、この後の節および表 11-1 に示す多数の構成可能なパラメタを指定します。spama.conf ファイルの例をコード例 11-1 に示します。

全般オプション

START_MEDIATOR

メディエータを実行する場合は、このパラメタを yes に設定します。実行しない場合は no に設定します (図 10-1 も参照)。

デフォルト値は次のとおりです。

```
START_MEDIATOR=yes
```

START_MASTER_AGENT

SNMPv3 のセキュリティー機能を使用する必要があり、その機能を提供するサードパーティのマスターエージェントを使用しない場合は、このパラメタを yes に設定してマスターエージェントを起動します (図 10-2 およびその後の表を参照)。

SNMPv3 のセキュリティー機能を使用する必要がなく、別のマスターエージェント (snmpdx を含む) を使用する場合、またはマスターエージェントを使用しない場合は、このパラメタを no に設定します (図 10-1、および図 10-3 とその後の表も参照)。

デフォルト値は次のとおりです。

```
START_MASTER_AGENT=no
```

AGENT_INTERFACE_NAME

マスターエージェントを有効にした場合 (上記を参照)、ここで設定する値は、マスターエージェントがバインドするネットワークインタフェース、つまり メディエータを localhost にバインドするプロトコルを指定します。

マスターエージェントを有効にしない場合、ここで設定する値は、メディエータがバインドするネットワークインタフェースのホスト名を指定します。値を指定しない場合、デフォルトでは、デフォルトのインタフェースによるアクセスが設定されます。

メディエータをたとえば snmpdx のサブエージェントとして使用する場合は、値を localhost に設定します。

デフォルト値は次のとおりです。

```
AGENT_INTERFACE_NAME=localhost
```

マスターエージェントのオプション

マスターエージェントを snmpdx とともに使用するために、マスターエージェントの起動スクリプトは snmpdx を停止し、その SNMP ポートを使用して、snmpdx を別のポート上のサブエージェントとして再起動します。

SNMPDX_REQ_FORWARD_PORT パラメタに null 値が設定されている場合、マスターエージェント起動スクリプトは一時的な匿名用の 32768 から 65535 の範囲にある空きポートを検索し、snmpdx をそのポート上のサブエージェントとして再起動します。起動スクリプトは /etc/services も検索し、そこに記述されているポートは使用しません。

ただし、SNMPDX_REQ_FORWARD_PORT の値を指定した場合には、マスターエージェントはこのポートを使用して snmpdx に要求を転送します。この場合、マスターエージェントはポートがすでに使用中かどうかを確認しません。

MASTER_AGENT_REQ_PORT

これは、マスターエージェントがマネージャーからの要求を受信するポートです。通常の構成では、値を変更する必要はありません (図 10-2 およびその後の表も参照)。

未指定の場合、デフォルト値は 161 です。

ENABLE_SNMPV2C_SETS

このパラメタは、マスターエージェントで SNMPv1 または SNMPv2c を使用した集合操作を実行できるかどうかを制御します。SNMPv1 および SNMPv2c プロトコルには本質的にセキュリティー保護がないため、値を yes に設定すると、セキュリティーが大幅に低下します。

デフォルト値は次のとおりです。

```
ENABLE_SNMPV2C_SETS=no
```

SNMPDX_REQ_FORWARD_PORT

このパラメタは、マスターエージェントが snmpdx に要求を転送するポートを制御します。値を指定しない場合は、マスターエージェントによって自動構成が実行されます (この節の最初の部分と、図 10-2 およびその後の表を参照)。

値を指定した場合は、snmpdx をこのポートで待機するように手動で構成することも必要です。

デフォルト値は次のとおりです。

```
SNMPDX_REQ_FORWARD_PORT=
```

SNMPV3_USER

このパラメタでは、SNMPv3 のトラップを発行する SNMPv3 ユーザーを決定します。

デフォルト値は次のとおりです。

```
SNMPV3_USER=defaultUser
```

注 – SNMPv3 のトラップを送信するには、SPAPM_TRAPS_ARE_V3=yes を設定する必要があります。

プロトコルメディエータのオプション

SUB_AGENT

このパラメタでは、起動時に自動的に起動するのではなく、snmpdx などのマスターエージェントによって メディエータまたはマスターエージェントを起動するかどうかを決定します。

デフォルト値は次のとおりです。

```
SUB_AGENT=yes
```

yes を指定した場合は、<port> パラメタを次のように指定して メディエータを起動する必要があります。

```
# /etc/init.d/spama start <port>
```

<port> には、メディエータが SNMP 要求を待機する UDP ポートを指定します。たとえば、メディエータを snmpdx とともに使用した場合、メディエータの起動は /etc/snmp/conf/spapm.rsrc ファイルにある次の行によって制御されます。

```
command = "etc/init.d/spama start $PORT"
```

注 – SUB-AGENT=YES の場合、START_MASTER_AGENT の値は無視されます。マスターエージェントの機能を有効にした場合は、SUB_AGENT を no に設定する必要があります。

---

no を指定すると、メディエータは起動時に (起動スクリプト /etc/rc3.d/S80spama によって) 起動され、メディエータが SNMP 要求を待機する UDP ポートは、次に説明する SPAPM_REQ_PORT の設定によって定義されます。

SPAPM_REQ_PORT

このパラメタでは、SUB_AGENT が no に設定されている場合にメディエータが要求を受信するポートを決定します。図 10-1、図 10-2、図 10-3 およびその後の表を参照してください。

デフォルト値は次のとおりです。

```
SPAPM_REQ_PORT=
```

デフォルト設定を使用すると、START_MASTER_AGENT=yes の場合のポート番号は自動的に割り当てられます。START_MASTER_AGENT=no の場合、デフォルトのポート番号である 33000 が使用されます。

START_MASTER_AGENT=no の場合は、ローカルマスターエージェントから固定ポートを使用して、または遠隔 SNMP マネージャーから直接メディエータにアクセスできるように、SPAPM_REQ_PORT に必要な値を設定します。

SPAPM_TRAPS_ARE_V3

このパラメタでは、メディエータのトラップが SNMPv3 か SNMPv2c かを決定します。

デフォルト値は次のとおりで、トラップを SNMPv2c に設定します。

```
SPAPM_TRAPS_ARE_V3=no
```

注 – SNMPv3 のトラップを有効にした場合 (SPAPM_TRAPS_ARE_V3=yes)、START_MASTER_AGENT=yes と START_MEDIATOR=yes も設定する必要があります。

SPAPM_TRAP_PORT

このパラメタでは、メディエータのトラップの送信先となるポート番号を決定します。図 10-1、図 10-2、図 10-3 およびその後の表を参照してください。

デフォルト値は次のとおりです。

```
SPAPM_TRAP_PORT=162
```

SPAPM_TRAP_INTERFACE

このパラメタでは、メディエータの SNMP トラップの送信元となるインタフェースを決定します。未定義の場合、トラップはホストのデフォルトインタフェースから発行されます。

注 – SNMPv2c のトラップは、`snmpdx` を経由せずに直接送信されます。

デフォルト値は未定義です。

```
SPAPM_TRAP_INTERFACE=
```

SPAPM_OPTIONS

このパラメタでは、次に示す 1 つ以上のオプションを指定してメディエータの動作を変更できます。

- `-a` 属性変更通知を送信する
- `-s` 状態変更通知を送信する
- `-c` オブジェクト作成通知を送信する
- `-C` 初期化実行中のオブジェクト作成通知を有効にする
- `-d` オブジェクト削除通知を送信する
- `-l` デフォルトで現在の問題リストのログを有効にする

このパラメタの形式とデフォルト値は次のとおりです。

```
SPAPM_OPTIONS="-ascCdl"
```

表 11-1 `spama.conf` にあるデフォルト値

パラメタ	デフォルトのインストール時の値
START_MEDIATOR	START_MEDIATOR=yes
START_MASTER_AGENT	START_MASTER_AGENT=no
AGENT_INTERFACE_NAME	AGENT_INTERFACE_NAME=localhost
MASTER_AGENT_REQ_PORT	MASTER_AGENT_REQ_PORT=161 (未指定の場合のデフォルトもこの値になる)
ENABLE_SNMPV2C_SETS	ENABLE_SNMPV2C_SETS=no
SNMPDX_REQ_FORWARD_PORT	SNMPDX_REQ_FORWARD_PORT=
SNMPV3_USER	SNMPV3_USER=defaultUser
SUB_AGENT	SUB_AGENT=yes
SPAPM_REQ_PORT	SPAPM_REQ_PORT=
SPAPM_TRAPS_ARE_V3	SPAPM_TRAPS_ARE_V3=no

表 11-1　spama.conf にあるデフォルト値 (続き)

パラメタ	デフォルトのインストール時の値
SPAPM_TRAP_PORT	SPAPM_TRAP_PORT=162
SPAPM_TRAP_INTERFACE	SPAPM_TRAP_INTERFACE=
SPAPM_OPTIONS	SPAPM_OPTIONS=”-ascCdl”

コード例 11-1　spama.conf ファイルの例

```
#!/sbin/sh
#
#ident  "@(#)spama.conf1.17 01/29/03 SMI"
#
# Copyright 2003 Sun Microsystems, Inc. All rights reserved.
# Use is subject to license terms.

#
# This file is used to control the configuration of the Master Agent and
# Protocol Mediator
#

#
# Master Agent / Mediator configuration
#

##############################################################################
# General options
##############################################################################

#
# Set to "yes" if the mediator component should be started
#
START_MEDIATOR=yes

#
# Set to "yes" to enable the master agent
#
START_MASTER_AGENT=no

#
# Hostname of the network interface for the agent to bind to. If this
# is not specified the agent will be accessible via the default
# interface.
#
# If the mediator is being used as a sub-agent this should be
# set to localhost.
#
```

コード例 11-1　spama.conf ファイルの例 (続き)

```
# If the master agent is enabled, this setting applies to its interface,
# the protocol mediator being bound to localhost.
AGENT_INTERFACE_NAME=localhost

##############################################################################
# Master Agent options
##############################################################################

#
# SNMP port for master agent to receive SNMP get/set requests.
#
# This port number will be used to listen for SNMP get/set
# requests.
#
# If this value is blank, default will be 161 if START_MASTER_AGENT=yes
#
MASTER_AGENT_REQ_PORT=

#
# set to "yes" to enable SNMPv1/SNMPv2c SET operations via the master agent.
#
ENABLE_SNMPV2C_SETS=no

#
# SNMP sub-agent port to which non-SNMP Protocol Mediator (snmpdx) requests
# will be sent.
#
# If this port setting is blank (default), automatic configuration will be
# performed. The port number for snmpdx will be dynamically determined
# if snmpdx is already using the UDP port where the Master Agent listens
# for SNMP get/set requests (by default UDP port 161) at Master Agent startup.
#
# If this port setting is blank but snmpdx is not using the same port
# as the Master Agent will listen for SNMP get/set requests, the Master
# Agent will use the port number which is being used by snmpdx to forward
# the SNMP set/get requests to snmpdx.
#
# If this port is set, it is expected that the user should perform
# the "listening" port configuration for the sub-agents and the Master
# Agent will use this port number to forward non-SNMP Protocol Mediator
# requests.
#
SNMPDX_REQ_FORWARD_PORT=

#
# SNMPv3 user
#
```

コード例 11-1　spama.conf ファイルの例 (続き)

```
# The Mediator will use this user to send the V3 traps (if enabled with
# SPAPM_TRAPS_ARE_V3).
#
# If this value is blank, default will be 'defaultUser'.
SNMPV3_USER=

##############################################################################
# Protocol Mediator options
##############################################################################

#
#
# Sub-agent configuration
#
# If the master agent is not being used (i.e. START_MASTER_AGENT=no), then
# setting SUB_AGENT=yes indicates that the mediator should be started with a
# port number argument by snmpdx or a third party master agent. Otherwise, set
# to no if the mediator is to be started with a manually configured port
# number.
#
# If START_MASTER_AGENT=yes then this setting is ignored.
#
SUB_AGENT=yes

#
# Mediator request port. If the START_MASTER_AGENT="no" and SUB_AGENT="no", the
# default is 33000, otherwise it is dynamically allocated.
#
SPAPM_REQ_PORT=

#
# set to yes to enable v3 mediator traps (requires START_MASTER_AGENT=yes and
# START_MEDIATOR=yes)
#
SPAPM_TRAPS_ARE_V3=no

#
# Default port for traps
#
SPAPM_TRAP_PORT=162

# This is also used to define the interface to which SNMP traps will be
# sent by the protocol mediator independently of the setting of
# AGENT_INTERFACE_NAME. If not defined, traps will be issued from the
# host's default interface.
#
SPAPM_TRAP_INTERFACE=
```

コード例 11-1　spama.conf ファイルの例 (続き)

```
#
# Agent option flags
#
# -a    Send attribute change notifications
# -s    Send state change notifications
# -c    Send object creation notifications
# -C    Enable object creation notifications during initialization
# -d    Send object deletion notifications
# -l    Enable current problem list logs by default
#
SPAPM_OPTIONS="-ascCdl"
```

アクセス制御

アクセス制御は、マネージャーのホストマシンの IP アドレスとコミュニティー、および各サブエージェントに対して指定されたコミュニティーに基づいて実行されます。コミュニティーおよびホストマシンのアクセス権は、ACL ファイルで定義します。

ACL ファイルでは、エージェントがトラップを送信する送信先ホストも定義します。トラップを送信する場合、エージェントは ACL ファイルの *<trapInterestHostList>* に記述されているすべてのホストにトラップを送信します。

ACL ファイルには次に示す 3 つのファイルがあります。

- `spapm.acl` ではメディエータへのアクセスを制御します。メディエータへのアクセスは、管理アプリケーションからの直接アクセスの場合もありますが、通常は `snmpdx` からのアクセスです。SNMP トラップに必要な受信側もこのファイルで定義します。
- `spapm_snmpdx.acl` では、メディエータが `snmpdx` のサブエージェントとして構成されている場合の、メディエータへのアクセスを制御します。
- `spama.acl` では、SNMPv3 マスターエージェントを経由したアクセスを制御します。

SNMPv3 のアクセス制御、コミュニティーおよびトラップ転送のパラメタは、`spama.uacl` および `spama.security` で定義します。

ACL ファイルの形式

ACL ファイルには、コミュニティーとマネージャーのアクセス権を定義する `acl` グループ、およびトラップを送信するためのコミュニティーとホストを定義する `trap` グループを記述します。ACL ファイルには、行頭にハッシュ記号 (#) を付けたコメント行も記述できます。

注 – `spapm_snmpdx.acl` ファイルの構文にはある特定の違いがあります。詳細については、スクリプト中のコメントを参照してください。

`acl` グループ

`acl` グループには、次の構文を使用してコミュニティー構成のリストを 1 つ以上記述します。

```
acl = {
    <list1>
    <list2>
    ...
    <listN>
}
```

このファイルの `acl` グループでは、特定のコミュニティーおよびマネージャーのアクセス権を指定します。このグループには、次の形式でコミュニティー構成のリストを指定します。

```
{
   communities = <communityList>
   access = <accessRights>
   managers = <hostList>
}
```

<communityList> は、このアクセス制御を適用する対象である SNMP コミュニティー名のリストです。このリストのコミュニティー名はコンマで区切ります。

<accessRights> では、マネージャーの項目に指定したマシンで実行されるすべてのマネージャーに付与する権限を指定します。指定できる値は次の 2 つです。

- `read-write`
- `read-only`

<hostList> 項目では、アクセス権を付与するマネージャーのホストマシンを指定します。*<hostList>* はコンマで区切ったホストのリストです。各ホストは次のいずれかで表すことができます。

- ホスト名 (例: `hubble`)
- IP アドレス (例: `123.456.789.12`)
- サブネットマスク (例: `123!255!255!255`)

注 – ACL ファイル内では、IP アドレスとサブネットマスクを区別するために、サブネットマスクを構成する個々の整数をドットではなく感嘆符 (!) で区切ります。

コード例 11-2 `acl` グループの例

```
acl = {
    {
        communities = public, private
        access = read-only
        managers = rag, tag, bobtail
    }
    {
        communities = tigger
        access = read-write
        managers = brittas
    }
}
```

trap グループ

trap グループでは、エージェントがトラップを送信できるホストを指定します。構成が必要なのは、メディエータが SNMPv2 トラップを送信する必要がある場合のみであり、メディエータが SNMP マスターエージェントを使用して SNMPv3 トラップを送信する場合は構成不要です。

このグループには、次の構文を使用してトラップコミュニティーの定義を 1 つ以上記述します。

```
trap = {
    <community1>
    <community2>
    ...
    <communityN>
}
```

ホストに送信するトラップに含まれる SNMP コミュニティー文字列とホストとの関連付けを各行に定義します。個々の trap-community 定義の形式は次のとおりです。

```
{
   trap-community = <trapCommunityString>
   hosts = <trapInterestHostList>
}
```

<trapCommunityString> 項目では SNMP コミュニティー文字列を指定します。指定したコミュニティー文字列は、hosts 項目に指定したホストに送信されるトラップに格納されます。

<trapInterestHostList> 項目では、コンマで区切ったホストのリストを指定します。次の例に示すように、各ホストは、ホスト名または完全な IP アドレスで指定する必要があります。

コード例 11-3　　trap グループの例

```
trap = {
     {
          trap-community = tigger
          hosts = gandalf, frodo
     }
}
```

メディエータの構成ファイル

この節では、次のファイルの形式について説明します。

105 ページの「spapm.acl ファイル」

106 ページの「spapm_snmpdx.acl ファイル」

spapm.acl ファイル

spapm.acl ファイルでは、プロトコルメディエータのアクセス制御を定義します。102 ページの「ACL ファイルの形式」でこのファイルの一般的な形式を説明しています。この節では、spapm.acl ファイルに固有の情報を示します。コード例 11-4 にこのファイルの例を示します。

このファイルは、デフォルトでは /etc/opt/SUNWspa にあります。

ACL ファイルがある場合は、SNMP アダプタ経由でエージェントにアクセスするすべてのマネージャーまたはプラットフォームエージェントサーバーに、ACL ファイルに定義されたアクセス権が適用されます。エージェント起動時に ACL ファイルが存在しない場合、SNMP アダプタを経由したエージェントへのフルアクセス権がすべてのマネージャーに付与され、トラップは生成されません。

SNMP アダプタのアクセス制御とトラップを有効にするには、エージェント起動時に ACL ファイルが存在していることが必要です。ACL ファイルにはセキュリティー関連情報が含まれています。そのため、ACL ファイルには、root による読み取りのみを許可する限定されたアクセス権を割り当てます。

acl グループと trap グループは、それぞれ、102 ページの「acl グループ」と104 ページの「trap グループ」で説明している形式に従います。

メディエータがマスターエージェント (snmpdx など) のサブエージェントとして登録されている場合は、spapm.acl ファイルに localhost をマネージャーとして指定する必要があります。これがマスターエージェントによって転送される SNMP パケットの発行元になるためです。snmpdx を使用すると、コミュニティー文字列は変更されずに転送されます。したがって、このファイルに記述したコミュニティーを spapm_snmpdx.acl ファイルにも指定する必要があります (コード例 11-4 を参照)。

コード例 11-4 spapm.acl ファイルの例

```
#
#  @(#)spapm.acl     1.6 03/01/29 SMI
#
# Copyright 2003 Sun Microsystems, Inc. All rights reserved.
#
```

コード例 11-4 spapm.acl ファイルの例 (続き)

```
# Template ACL file for Sun SNMP Management Agent for Sun Fire B1600

acl = {
    {
        communities = public, private
        access = read-only
        managers = rag, tag, bobtail
    }
    {
        communities = tigger
        access = read-write
         managers = brittas
     }
}

trap = {
    {
        trap-community = tigger
        hosts = brittas
    }
}
```

trap グループでは、SNMPv2c 通知の送信先を定義します。

spapm_snmpdx.acl ファイル

デフォルトの構成では、メディエータは snmpdx のサブエージェントとして動作します。このファイルを変更して発信元ホスト名に基づくアクセスを有効にすることができます。コミュニティーとアクセス権は、spama.acl ファイルでの定義と一致している必要があります。

このファイルの acl グループでは、特定のコミュニティーおよびマネージャーのアクセス権を指定します。このグループには、次の形式でコミュニティー構成のリストを指定します。

```
#    {
#        communities = <communityList>
#        access = <accessRights>
#        managers = <hostList>
#    }
```

- *communityList* は、このアクセス制御を適用する対象であるコミュニティーの名前をコンマで区切ったリストです。
- *accessRights* では、*hostList* に指定したマネージャーに付与されるアクセス権を指定します。
- *hostList* は、*accessRights* で指定したアクセス権が付与されるホストの名前をコンマで区切ったリストです。

コード例 11-5 の最初の例では、rag、tag および bobtail の各システムには、public および private コミュニティーに対する read-write アクセス権が構成されています。システム brittas には、コミュニティー tigger への read-write アクセス権が構成されています。

2 番目の例は、SNMPv3 マスターエージェントを使用する構成 (spama.conf に START_MASTER_AGENT=yes を設定) に適用されます。この構成では、snmpdx が受信する SNMP パケットはローカルホストから発信されると想定します。public および private のコミュニティーには読み取り専用アクセスが設定されています。コミュニティー tigger には読み取り書き込みアクセス権が設定されています。これらのコミュニティーはマスターエージェントによって SNMPv3 のコンテキストからマッピングされます。したがって、このアクセスが必要な SNMPv3 コンテキストについては、対応するコミュニティーがこのファイルに指定されている必要があります。

コード例 11-5 spapm_snmpdx.acl ファイルの例

```
#  @(#)spapm_snmpdx.acl1.8 03/01/29 SMI
#
# Copyright 2003 Sun Microsystems, Inc. All rights reserved.
# Use is subject to license terms.
#
# Template snmpdx Access Control file for Sun SNMP Management Agent for Sun
# Fire B1600
#
# Example 1:
#
# acl = {
#     {
#         communities = public, private
#         access = read-only
#         managers = rag, tag, bobtail
#     }
#     {
#         communities = tigger
#         access = read-write
#         managers = rag, tag, bobtail
#     }
# }
#
# Example 2:
#
```

コード例 11-5　spapm_snmpdx.acl ファイルの例 (続き)

```
 acl = {
     {
         communities = public, private
         access = read-only
         managers = localhost
     }
     {
         communities = tigger
         access = read-write
         managers = localhost
     }
 }
#
# Trap destinations are defined in spapm.acl and spama.acl.
# This entry does not need to be edited.
trap = {
}
```

マスターエージェントの構成ファイル

これらのファイルを構成する必要があるのは、マスターエージェント機能を使用する (spama.conf に START_MASTER_AGENT=yes を設定する) 場合のみです。

この節では、次のファイルの形式について説明します。

- 108 ページの「spama.acl ファイル」
- 109 ページの「spama.uacl ファイル」
- 110 ページの「spama.security ファイル」

spama.acl ファイル

spama.acl ファイルでは、マスターエージェントのアクセス制御を定義します。102 ページの「ACL ファイルの形式」でこのファイルの一般的な形式を説明しています。この節では、spama.acl ファイルに固有の情報を示します。

このファイルは、デフォルトでは /etc/opt/SUNWspa にあります。

SNMP アダプタのアクセス制御とトラップを有効にするには、エージェント起動時に ACL ファイルが存在していることが必要です。ACL ファイルにはセキュリティー関連情報が含まれています。そのため、ACL ファイルには、root による読み取りのみを許可する限定されたアクセス権を割り当てます。

このファイルでは、SNMPv1 および SNMPv2c のアクセス権と SNMP 通知の受信側を定義します。spama.conf で SPAPM_TRAPS_ARE_V3=yes である場合、トラップは SNMPv3 トラップとして送信されます。それ以外の場合は、SNMPv2c トラップとして送信されます。

acl グループ

SNMPv3 を使用する場合は、SNMPv1 および SNMPv2c での書き込みアクセスを禁止することをお勧めします。したがって、この acl では通常、読み取り専用アクセスのみを許可します。

コード例 11-6　acl グループの例

```
acl = {
     {
          communities = public, private
          access = read-only
          manager = localhost
     }
}
```

trap グループ

このファイルの trap グループは、104 ページの「trap グループ」で説明している形式に従います。

spama.uacl ファイル

このファイルは spama.security ファイルと組み合わせて使用し、マスターエージェントがアクティブな場合に SNMPv3 セキュリティーを有効にします。102 ページの「ACL ファイルの形式」でこのファイルの一般的な形式を説明しています。この節では追加構成パラメタについて説明します。簡潔にするためほとんどのコメントを削除したファイルの例をコード例 11-7 に示します。

このファイルは、デフォルトでは /etc/opt/SUNWspa にあります。

acl グループ

acl グループには以下のパラメタがあります。

- *context-names*—コンテキスト名をコンマで区切ったリストです。

- *access*—指定できる値は次のとおりです。
  - read-only
  - read-write
- *security-level*—指定できる値は次のとおりです。
  - noAuthNoPrivacy
  - authNoPrivacy
  - authPrivacy
- *users*—ユーザー名をコンマで区切ったリストです。

次に示す例では、defaultUser にアクセス権が付与され、public および null のコンテキストで、最小限のセキュリティー保護である authNoPrivacy を使用した defaultUser からの要求が許可されます。その他すべての SNMP 要求は拒否されます。

このファイルに trap グループはありません。

コード例 11-7 spama.uacl ファイルの例

```
#ident "@(#)spama.uacl 1.4 01/29/03 SMI"
#
# Copyright 2003 Sun Microsystems, Inc. All rights reserved.
# This software is the proprietary information of Sun Microsystems, Inc.
# Use is subject to license terms.
#
# Template ACL file
#
 acl = {
     {
         context-names = public,null
         access = read-write
         security-level=authNoPriv
         users = defaultUser
     }
 }
```

spama.security ファイル

spama.security ファイルでは、マスターエージェントへのアクセスが許可されるユーザー、SNMPv3 の暗号化鍵および認証鍵を指定します。

このファイルには、次の形式で userEntry 行を指定します。

```
userEntry=<engine ID>,<user name>,<security name>,<authentication algorithm>,
<authentication key>, <privacy algorithm>, <privacy key>, <storage type>,<template>
```

注意 – このファイルでは、userEntry 行以外のパラメタを編集しないでください。

これらのフィールドについて表 11-2 で説明します。

表 11-2 ユーザーが構成可能な spama.security のパラメタ

パラメタ	説明
engine ID	使用する SNMP エンジンの ID。次のいずれかを指定できます。 ■ 16 進文字列 ■ <address>:<port>:<IANA number> の形式で指定した engineID を表すテキスト文字列 ■ 文字列 localEngineID (ほとんどの場合に適した指定)
user name	このエントリを適用する対象のユーザー名。
security name	このユーザー名にマッピングするセキュリティー名。通常、ユーザー名とセキュリティー名は同じです。
authentication algorithm	使用する認証アルゴリズム。次のいずれかとすることができます。 • usmHMACMD5AuthProtocol • usmHMACSHHAuthProtocol • usmNoAuthProtocol
authentication key	認証アルゴリズムで使用する鍵。次のいずれかとすることができます。 • テキストのパスワード (8 文字以上) • ローカライズされた 16 進数の鍵。例: 0x0098768905AB67EFAA855A453B665B12
privacy algorithm	使用するプライバシアルゴリズム。次のいずれかとすることができます。 • usmDESPrivProtocol • usmNoPrivProtocol (未指定の場合のデフォルト)

表 11-2　ユーザーが構成可能な spama.security のパラメタ (続き)

パラメタ	説明
privacy key	プライバシアルゴリズムで使用する鍵。次のいずれかとすることができます。 • テキストのパスワード (8 文字以上) • ローカライズされた 16 進数の鍵。例: 0x0098768905AB67EFAA855A453B665B12
storage type	指定できる値は 3 のみであり、これは未指定の場合のデフォルトです。
template	デフォルトは false です (この値を変更する必要はありません)。

簡潔にするためほとんどのコメントを削除したファイルの例をコード例 11-8 に示します。

このファイルは、デフォルトでは /etc/opt/SUNWspa にあります。

デフォルトの spama.security ファイルには、2 つのサンプルユーザーが指定されています。サンプルを変更し、コメントを解除して独自のユーザーを定義できます。1 つ目のサンプルでは、defaultUser というユーザーを次のように指定しています。

- MD5 アルゴリズムのみを使用した認証
- プライバシなし
- 認証パスワードは “mypassword”

2 つ目のサンプルでは、defaultUser というユーザーを次のように指定しています。

- MD5 アルゴリズムと認証パスワード “mypassword” による認証
- DES アルゴリズムとプライバシパスワード “mypassword” を使用したプライバシ
- DES アルゴリズムを使用したプライバシ

コード例 11-8　spama-security ファイルの例

```
#ident  "@(#)spama.security 1.7 01/29/03 SMI"
#
# Copyright 2002 Sun Microsystems, Inc. All rights reserved.
# This software is the proprietary information of Sun Microsystems, Inc.
# Use is subject to license terms.
#
# Template security file

# localEngineBoots=0

# defaultUser configuration. Authentication only.
# userEntry=localEngineID,defaultUser,,usmHMACMD5AuthProtocol,mypasswd
```

コード例 11-8　spama-security ファイルの例 (続き)

```
# defaultUser configuration. Authentication and encryption.
# userEntry=
localEngineID,defaultUser,null,usmHMACMD5AuthProtocol,mypasswd,usmDESPrivProt
ocol,mypasswd,3,
```

第 12 章

ソフトウェアの構成

この章では、インストール後のデフォルト構成、および第 11 章で説明したファイルを変更する方法について説明します。

この章は以下の節で構成されています。

デフォルト構成

このソフトウェアは、次のデフォルト構成を使用してインストールされます。

- マスターエージェントは無効化されます (`START_MASTER_AGENT=no`)。
- メディエータは有効化されます (`START_MEDIATOR_AGENT=yes`)。
- メディエータは `snmpdx` のサブエージェントとして構成されます。

注 – セキュリティー上の理由により、エージェントを監視しているシステム以外のすべてのシステムを除外するように `snmpdx` ACL ファイルを構成して、アクセスを制限する必要があります。

アクセス制御

メディエータのアクセス制御を有効にするには、105 ページの「spapm.acl ファイル」で説明している メディエータの ACL ファイルを構成します。

`snmpdx` (デフォルト構成) を使用する場合は、`spapm_snmpdx.acl` を変更してアクセス権を設定し、`spapm.acl` を変更してトラップの受信側を設定します。

メディエータの起動と停止

メディエータを起動するには、次のように標準の snmpdx 起動スクリプトを使用します。

```
# /etc/init.d/init.snmpdx start
```

メディエータを停止するには、次の メディエータスクリプトを使用します。

```
# /etc/init.d/spama stop
```

直接アクセスのための手動による構成

snmpdx では SNMPv1 のみをサポートしているため、SNMPv2c 固有の get-bulk 操作を実行する必要があり、マスターエージェントを使用しない場合は、メディエータが使用するポートを SNMPv2c での直接アクセス用に設定できます。

メディエータを手動で構成するには、spama.conf に次の変更を行います。

1. SUB_AGENT=no を設定します。
2. SPAPM_REQ_PORT に必要なポート番号を設定します。
   メディエータに送信する SNMPv2c 要求は、このポートに送信する必要があります。

サードパーティのマスターエージェントのサブエージェントとしてのメディエータ

コマンド行パラメタによってポート番号を指定できるサードパーティのマスターエージェントのサブエージェントとしてメディエータを構成するには、次のようにします。

1. ローカルホストからのアクセスを許可するようにメディエータの ACL ファイルを構成します (105 ページの「spapm.acl ファイル」を参照)。
2. 以下の呼び出しによってメディエータを起動するように、適切なポート番号を使用してマスターエージェントを構成します。

```
/etc/init.d/spama start <port>
```

3. 次の OID サブツリーに要求を転送するようにマスターエージェントを構成します。
   - .iso.org.dod.internet.mgmt.mib-2.entityMIB
   - .iso.org.dod.internet.private.enterprises.sun.products.sunFire.sunPlatMIB

   数値で指定する場合は次のとおりです。
   - .1.3.6.1.2.1.47
   - .1.3.6.1.4.1.42.2.70.101

メディエータと SNMPv3 マスターエージェント

メディエータとマスターエージェントを有効にするには、次に示す変更が最低限必要です。

1. spama.conf ファイルでは、次の変更を行います。
   a. `START_MASTER_AGENT=yes` を設定します。
   b. `SUB_AGENT=no` を設定します。
2. ローカルホストからのアクセスが可能になるようにメディエータの ACL ファイルを構成します (105 ページの「spapm.acl ファイル」を参照)。
3. ローカルホストからのアクセスが可能になるように snmpdx を構成します (106 ページの「spapm_snmpdx.acl ファイル」 を参照)。
4. 必要なマネージャーからのアクセスが可能になるようにマスターエージェントの ACL ファイルを構成します (108 ページの「spama.acl ファイル」を参照)。
5. セキュリティーファイルを構成し、SNMPv3 のユーザー、コンテキスト、および認証と暗号化レベルを定義します (108 ページの「spama.acl ファイル」および110 ページの「spama.security ファイル」を参照)。

エージェントの起動と停止

メディエータとマスターエージェントを起動するには、次のメディエータスクリプトを使用します。

```
# /etc/init.d/spama start
```

メディエータとマスターエージェントを停止するには、次のメディエータスクリプトを使用します。

```
# /etc/init.d/spama stop
```

SNMPv3 トラップの転送

メディエータからの SNMPv3 トラップを転送するようにマスターエージェントを構成するには、spama.conf ファイルで以下の手順を実行します (92 ページの「spama.conf」を参照)。

1. `SPAPM_TRAPS_ARE_V3=yes` を設定します。
2. オプションで、`SNMPV3_USER` を設定します。

注 – `spama.uacl` および `spama.security` でトラップのユーザーを SNMPV3 ユーザーとして構成する必要があります。

第13章

ソフトウェアのアンインストール

この章では、ソフトウェアをアンインストールする方法について説明します。

通常、SNMP をアンインストールするために必要な手順は、pkgrm コマンドを使用して、インストールしたパッケージを削除することだけです。この手順を実行すると、関連するすべてのファイルとリンクが削除され、snmpdx が再度有効化されます。

SNMP ソフトウェアによって自動的に行われた構成の変更は、元の状態に復元されます。ただし、snmpdx の ACL ファイルのような外部ファイルの設定を変更した場合は、SNMP ソフトウェア削除後に手動で復元する必要があります。

注 – 次に示す手順では、Java SNMP API パッケージの SUNWjsnmp はアンインストールされません。このパッケージの Solaris バージョンを再インストールするには、最初に Java SNMP API を削除する必要があります。

プラットフォームエージェント用のパッケージとターゲットエージェント用のパッケージ

プラットフォームエージェントサーバーからプラットフォームエージェント用のパッケージを削除するには、次のように入力します。

```
# pkgrm SUNWbgpmr SUNWbgpm SUNWjdrt SUNWjsnmp SUNWbgpjo \
SUNWbgodr SUNWbgod SUNWbgcmr SUNWbgcm SUNWbgpc SUNWbgptk
```

注意 – SUNWjdrt および SUNWjsnmp パッケージを削除するときは、注意が必要です。これらのパッケージは両方ともシステムパッケージであり、他の製品で使用されている場合があります。

Sun Fire B100s ブレードからターゲットプラットフォーム用のパッケージを削除するには、次のように入力します。

```
# pkgrm SUNWbgpr SUNWbgcm SUNWbgpc SUNWbgptk
```

ドメインエージェント用のパッケージ

Sun Fire B100s ブレードからドメインエージェント用のパッケージを削除するには、次のように入力します。

```
# pkgrm SUNWbgpmr SUNWbgpm SUNWjdrt SUNWjsnmp SUNWbgpji \
SUNWbgidr SUNWbgcmr SUNWngcm SUNWbgpc SUNWbgptk
```

注意 – SUNWjdrt および SUNWjsnmp パッケージを削除するときは、注意が必要です。これらのパッケージは両方ともシステムパッケージであり、他の製品で使用されている場合があります。

第 14 章

障害追跡

この章では、システムの障害追跡に役立つ情報を示します。

問題

- **デフォルト構成 (snmpdx) の使用時、SNMP エージェントからの応答がない。**

1. **次のように入力してメディエータが実行中であることを確認します。**

```
# ps -ef | grep spa.snmp
    root 15789     1  1 13:44:01 pts/2    0:00 /usr/j2se/bin/java
-Dcom.sun.spa.snmp.LOG_LEVEL=INFO -Djdmk.security.file=//etc
```

上記のような応答が返された場合、メディエータプロセスは実行中です。

次のように入力してメディエータを停止し、再起動します。

```
# /etc/init.d/spama stop
# /etc/init.d/init.snmpdx start
```

2. **次のように入力して、正しいバージョンの Java がインストールされていることを確認します。**

```
# /usr/j2se/bin/java -version
java version "1.4.1_03"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.1_03-b04)
Java HotSpot(TM) Client VM (build 1.4.1_03-b04, mixed mode)
```

このコマンドで通知されるバージョンが、バージョン 1.4 以降である必要があります。正しいバージョンでない場合は、68 ページの「Java 環境」に従って Java 1.4 JDK をインストールします。

3. 次のように入力して、正しいバージョンの SUNWjsnmp がインストールされていることを確認します。

```
# pkginfo -l SUNWjsnmp | grep VERSION
   VERSION:  5.0
```

バージョン 1.0 と表示された場合は、SUNWjsnmp パッケージを削除して、SUNWspa.*.tar.Z アーカイブに入っているバージョンを再インストールします (69 ページの「Java SNMP API」を参照)。

4. spama.conf ファイルに次のエントリがあることを確認します。

```
START_MASTER_AGENT=no
START_MEDIATOR=yes
SUB_AGENT=yes
```

5. 次のように入力して、メディエータが snmpdx に正しく登録されていることを確認します。

```
# cat /var/snmp/snmpdx.st
spapm spapm 2516 34050
snmpd snmpd 2567 34053
```

上に示した spapm エントリでは、メディエータを snmpdx のサブエージェントとして登録しています。

6. /etc/snmp/conf/spapm.reg および /etc/snmp/conf/spapm.rsrc が壊れていないことを確認します。

次のように入力してメディエータを停止し、再起動します。

```
# /etc/init.d/spama stop
# /etc/init.d/init.snmpdx start
```

7. ACL ファイルでアクセス権が正しく設定されていることを確認します。

- spapm_snmpdx.acl では、使用する SNMP マネージャーのアクセス権を定義します。
- spapm.acl では、localhost のアクセス権を定義します。

詳細については、第 11 章を参照してください。

問題

- **プラットフォームエージェントの使用時、ハードディスクドライブ (HDD) または Ethernet MAC アドレスの計測機構がない。**

  計測情報は、オペレーティング環境がターゲット Sun Fire B100s ブレードで実行されている場合にのみ利用可能です。

1. **Sun Fire B100s ブレードが起動していることを確認します。**
2. **次のように入力して、Sun Fire B100s ブレードでターゲット計測機構が実行中であることを確認します。**

```
# netstat -an | grep 1099
      *.1099         *.*          0      0 24576      0 LISTEN
```

待機しているポートがない場合、ターゲット計測機構は実行されていません。

次のように入力して、Sun Fire B100s ブレード上の計測機構を起動します。

```
# /etc/init.d/spardp start
```

3. **次のように入力して、正しいバージョンの Java がインストールされていることを確認します。**

```
# /usr/j2se/bin/java -version
java version "1.4.1_03"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.1_03-b04)
Java HotSpot(TM) Client VM (build 1.4.1_03-b04, mixed mode)
```

このコマンドで通知されるバージョンが、バージョン 1.4 以降である必要があります。正しいバージョンでない場合、計測機構を起動することはできますが、すぐに障害が発生します。

正しいバージョンでない場合は、68 ページの「Java 環境」に従って Java 1.4 JDK をインストールします。

問題

- **エージェントにアクセスはできるが、監視対象プラットフォームの計測機構がない。**

1. **次のように入力して検出デーモンが実行中であることを確認します。**

```
# netstat -a | grep mismi
      *.mismi              *.*                0      0 24576      0 LISTEN
      *.mismi                              *.*  0
0 24576      0 LISTEN
```

この出力は、検出デーモンが管理対象プラットフォームからの要求を待機していることを示しています。

a. `/etc/services` **に次のエントリがあることを確認します。**

```
mismi           8265/tcp                # MISMI Discovery
```

b. `/etc/inetd.conf` **に次のエントリがあることを確認します。**

```
# MISMIDISCOVERY - mismiDiscovery daemon
mismi   stream  tcp6    nowait  root    /opt/SUNWspa/bin/mismiDiscovery
mismiDiscovery
```

c. **次のように入力して、**`/etc/inetd.conf` **が** `/etc/inet/inetd.conf` **へのシンボリックリンクであることを確認します。**

```
# ls -l /etc/inetd.conf
lrwxrwxrwx   1 root     root          17 Jan  7 17:08 /etc/inetd.conf ->
./inet/inetd.conf
```

リンクがない場合は、`SUNWbgodr` パッケージのインストール時にファイルのアップデートが失敗します。

`inetd` の構成を修正し、次のように入力して再起動します。

```
# pkill -1 inetd
```

2. 次のように入力して、プラットフォームが検出されていることを確認します。

```
# netstat -a | grep mismi
      *.mismi              *.*                  0      0 24576      0 LISTEN
blade-174-119.36780  hornet-sc.mismi       8192      0 24820      0 ESTABLISHED
      *.mismi                           *.*  0
0 24576      0 LISTEN
```

この出力は、検出デーモンが待機していること、プラットフォームシステムコントローラ (<hornet-sc>) への接続が確立されていることを示しています。

接続が確立されていない場合は、83 ページの「システムコントローラの構成」に従ってシステムコントローラの設定を確認します。

問題

- **SNMPv3 の** `get` **要求と** `set` **要求がタイムアウトする。**
  - 考えられる原因

`spama.security` ファイルにある `localEngineId` または `localEngineBoots` の番号が編集されたか、削除されています。

- 確認

ファイルが編集されているかどうかを判断するのは容易ではありません。

- 修正

次に示すようにエージェントを再起動し、その後に管理アプリケーションを再起動して再度同期をとります。

```
# /etc/init.d/spama stop
# /etc/init.d/spama start
```

問題

- **SNMP の** `get` **要求と** `set` **要求がタイムアウトする。**
  - 考えられる原因

負荷の大きいシステムでは、SNMP メディエータへの `snmpdx` マスターエージェントの要求がタイムアウトする場合があります。このタイムアウトは現在 2s (2000000 μs) に設定されています。

■ 確認

管理アプリケーションによって通知されたタイムアウトが、snmpdx と SNMP メディエータの間で発生したのか、管理アプリケーションと snmpdx の間で発生したのかを判断するのは容易ではありません。

■ 修正

/etc/snmp/conf/spapm.reg ファイルでタイムアウトプロパティーを編集することにより、タイムアウトを延ばすことができます。このファイルを編集した場合は、次のように入力して メディエータを再起動してください。

```
# /etc/init.d/spama stop
# /etc/init.d/init.snmpdx start
```

付録 A

J2SE 1.3.1 と共存する J2RE 1.4 のインストール

この付録では、プラットフォームエージェントサーバーおよび Sun Fire B100s ドメインに Java 2 Runtime Environment (J2RE) Standard Edition 1.4 を J2SE 1.3.1 と共存するようにインストールする方法、インストールした J2RE を検出するように起動スクリプトを変更する方法について説明します。

この付録は以下の節で構成されています。

J2RE 1.4 のインストール

68 ページの「Java 環境」で説明しているように J2RE 1.4 を J2SE 1.3.1 と共存するようにインストールするには、次に示す手順に従います。

J2RE 1.4 は、以下の場所から自己抽出型のバイナリファイルとして入手できます。

```
http://java.sun.com/j2se/1.4/download.html
```

次に示す手順に従って J2RE をインストールします。上記の Web サイトにファイルのダウンロードに関する詳細が記載されています。

注 – この製品に必要なのは 32 ビットサポートのみであるため、64 ビットの J2RE 用追加パッケージをインストールする必要はありません。

次に示す手順では、<ver> を適切な J2RE アップデートバージョン番号に置き換えてください。

たとえば、1.4.0_01 のアップデートをダウンロードする場合、次のコマンドを入力するとします。

```
# chmod +x j2re-1_4_<version>-solaris-sparc.sh
```

実際は次のように入力します。

```
chmod +x j2re-1_4_0_01-solaris-sparc.sh
```

1. ダウンロードを実行し、ファイルのサイズを確認します。

   必要なファイルは次のファイルです。

   j2re-1_4_<ver>-solaris-sparc.sh

   ダウンロードする前に、ダウンロード用のページに示されているファイルのサイズをメモします。ダウンロード完了後、ソフトウェアファイルが完全にダウンロードされ、壊れていないことを確認します。

   ファイルはスーパーユーザーでアクセスできる場所にダウンロードしてください (たとえば /tmp)。

2. su を実行してスーパーユーザーのパスワードを入力し、スーパーユーザーになります。

3. 自己抽出型のバイナリに対する実行権が設定されている必要があります。

```
# chmod +x j2re-1_4_<ver>-solaris-sparc.sh
```

4. ファイルのインストール先ディレクトリに移動します。

```
# cd /usr
```

5. 自己抽出型バイナリを実行します。

   /usr/j2re1.4.<ver> というディレクトリが作成され、そこに J2RE がインストールされます。

6. J2RE が正しくインストールされたことを確認します。

```
# /usr/j2re1.4.1_01/bin/java -version
java version "1.4.1_01"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.1_01-
b01)
Java HotSpot(TM) Client VM (build 1.4.1_01-b01, mixed mode)
```

このコマンドで通知されるバージョンが、バージョン 1.4 以降である必要があります。この例では、バージョン 1.4.1_01 が通知されています。

7. 自己抽出型バイナリを削除します。

8. root シェルを終了します。

起動スクリプトの編集

この節では、J2RE 1.4 をインストールした場合に起動スクリプトを変更する方法を説明します。

この節と併せて第 10 章を参照してください。

ドメインハードウェア監視

次に示す手順は77 ページの「ドメインハードウェア監視のためのソフトウェアのインストール」にある手順 4 に関連しています。

1. 監視対象の各 Sun Fire B100s ブレード上で、次の起動スクリプトを編集します。
   - /etc/init.d/spaibdm
   - /etc/init.d/spapom

   変更するのは次の行です。

   ```
   JAVA=/usr/j2se/bin/java
   ```

   これを次のように変更します。

   ```
   JAVA=/usr/j2re1.4.<ver>/bin/java
   ```

2. 監視対象の各 Sun Fire B100s ブレード上で、次の起動スクリプトを編集します。

   ■ /etc/init.d/spama

   変更するのは次の行です。

   ```
   JAVA_JAVA=/usr/j2se/bin/java
   ```

   これを次のように変更します。

   ```
   JAVA_JAVA=/usr/j2re1.4.<ver>/bin.java
   ```

プラットフォームハードウェア監視

手順 1 と手順 2 は、79 ページの「ターゲット計測機構を使用するソフトウェアをインストールする」にある手順 4、および 82 ページの「ターゲット計測機構を使用しないソフトウェアをインストールする」にある手順 4 に関連しています。

手順 3 は、82 ページの「ターゲット計測機構を使用しないソフトウェアをインストールする」にある手順 11 のみに関連しています。

1. プラットフォームエージェントサーバー上で、次の起動スクリプトを編集します。

   ■ /etc/init.d/spapom

   変更するのは次の行です。

   ```
   JAVA=/usr/j2se/bin/java
   ```

   これを次のように変更します。

   ```
   JAVA=/usr/j2re1.4.<ver>/bin/java
   ```

2. プラットフォームエージェントサーバー上で、次の起動スクリプトを編集します。

   ```
   /etc/init.d/spama
   ```

   変更するのは次の行です。

   ```
   JAVA_JAVA=/usr/j2se/bin/java
   ```

   これを次のように変更します。

   ```
   JAVA_JAVA=/usr/j2re1.4.<ver>/bin.java
   ```

3. 監視対象の各 Sun Fire B100s ブレード (ターゲット) 上で、次の起動スクリプトを編集します。

   - /etc/init.d/spardp

   変更するのは次の行です。

   ```
   JAVA=/usr/j2se/bin/java
   ```

   これを次のように変更します。

   ```
   JAVA=/usr/j2re1.4.<ver>/bin/java
   ```

索引

き

く

け

こ

さ

し

ほ

ま

め

よ

る

ろ