



WDR 開発ガイド

WBEM ベースのシステム管理アプリケーションの作成

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

Part No. 816-7273-11
2002 年 12 月, Revision A

コメントの宛先: docfeedback@sun.com

Copyright 2002 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, CA 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随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, Sun4U, SunSwift, Java, JDK 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。

OPENLOOK 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ぬものとします。

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: WDR Developer's Guide
Part No: 816-1984-11
Revision A



目次

はじめに ix

1. WDR の概要 1

WDR のハードウェア要件 1

Sun Fire 6800/4810/4800/3800 システムでの MSP のハードウェア要件 1

WDR のソフトウェア要件 2

Sun Fire 15K/12K システムでのソフトウェア要件 2

Sun Fire 6800/4810/4800/3800 システムでのソフトウェア要件 2

Web-Based Enterprise Management (WBEM) とは 2

Common Information Model (CIM) 3

プラットフォーム固有および共通の MOF ファイル 4

WDR により実行される操作 4

管理者セキュリティーモデル 5

WDR セキュリティー 5

Solaris WBEM Services 7

CIM Object Manager (CIMOM) 8

WBEM プロバイダ 8

Solaris WBEM ソフトウェア開発キット (SDK) 9

2. WDR での Solaris WBEM Services の使用方法 11

Solaris WBEM Services の概要	11
Solaris WBEM Services の層	12
Solaris WBEM Services のアプリケーション層	12
Sun WBEM User Manager と SMC ユーザーツール	12
Solaris Management Console (SMC) WBEM ログビューア	13
Managed Object Format (MOF) コンパイラ	13
Solaris WBEM Services の管理層	16
CIM Object Managerとは	16
手動による CIM Object Manager の起動と停止	17
▼ CIM Object Manager を起動する	17
▼ CIM Object Manager を停止する	17
Solaris WBEM Services のプロバイダ層	18
Solaris プロバイダ	18
WBEM セキュリティーサービス	19
認証	19
承認	19
再実行保護	20
デジタル署名	20
セキュリティーの実装	20
Sun WBEM User Manager の使用方法	21
▼ Sun WBEM User Manager を起動する	22
▼ ユーザーにデフォルトのアクセス権を与える	22
▼ ユーザーのアクセス権を変更する	23
▼ ユーザーのアクセス権を削除する	23
▼ ネームスペースへのアクセス権を設定する	23
▼ ネームスペースへのアクセス権を削除する	24
API を使用したアクセス制御の設定	24
Solaris_UserAcl クラス	25

- ▼ ユーザーごとにアクセス制御を設定する 26
 - Solaris_NamespaceAcl クラス 27
- ▼ ネームスペースでのアクセス制御を設定する 27
- Solaris Management Console (SMC) ユーザーツールの起動 28
 - ▼ SMC ユーザーツールを起動する 28
- Solaris WBEM ログイングサービス 29
- Solaris WBEM Services のログファイル 30
 - Solaris WBEM Services のログファイル規則 30
 - Solaris WBEM Services のログファイル形式 31
- Solaris WBEM ログクラス 31
 - Solaris_LogRecord クラス 32
 - Solaris_LogService クラス 32
- API を使用した Solaris WBEM ログイングの有効化 33
 - Solaris WBEM ログファイルへのデータの書き込み 33
 - ▼ Solaris_LogRecord のインスタンスを作成してデータを書き込む 34
 - Solaris WBEM ログファイルからのデータの読み取り 36
 - ▼ Solaris_LogRecord クラスのインスタンスを取得してデータを読み取る 36
 - Solaris WBEM ログイングプロパティの設定 39
 - ▼ Solaris WBEM ログイングプロパティを設定する 39
- Solaris WBEM ログビューア 41
 - ▼ SMC と Solaris ログビューアを起動する 41
- 3. プロセスインジケーションの使用 43
 - CIM イベントモデル 43
 - インジケーションの生成方法 45
 - サブスクリプションの作成方法 45
 - CIM リスナーの追加 46
 - ▼ CIM リスナーを追加する 46

イベントフィルタの作成	47
▼ イベントフィルタを作成する	48
イベントハンドラの作成	49
▼ CIM イベントハンドラを作成する	51
イベントフィルタとイベントハンドラのバインド	51
▼ イベントフィルタとイベントハンドラをバインドする	51
4. WDR のクラス、ドメイン、関連、およびインジケーション	53
WDR CIM クラス階層図	54
CIM 接続点クラス	55
CIM Solaris_WDRAttachmentPoint クラス	55
CIM Solaris_CHSystemBoard クラス	60
CIM Solaris_CHCPU クラス	64
CIM Solaris_CHMemory クラス	65
CIM Solaris_CHController クラス	67
CIM スロットクラス	68
CIM Solaris_WDRSlot クラス	68
CIM Solaris_XCSlot クラス	72
CIM Solaris_SGSlot クラス	74
CIM Solaris_WDRDomain クラス	76
CIM Solaris_WDRDomain クラス	76
CIM Solaris_XCDomain クラス	77
CIM Solaris_SGDomain クラス	80
WDR スキーマ の関連と集約	82
CIM Solaris_DomainHasAttachmentPoints 集約	82
CIM Solaris_DomainHasSlots 集約	83
Solaris_SlotHasSystemBoard 関連	84
Solaris_SystemBoardHasProcessors 集約	85
Solaris_SystemBoardHasMemory 集約	85

Solaris_SystemBoardHasControllers 集約	86
CIM プロセスインジケーションクラス	87
WDR インジケーションクラス階層図	88
Solaris_WDRIndication クラス	88
Solaris_SGBoardPresenceChange インジケーション	89
Solaris_SGDomainACLChange インジケーション	89
Solaris_SGDomainStateChange インジケーション	90
Solaris_SGSlotAssignmentChange インジケーション	91
Solaris_SGBoardStateChange インジケーション	92
Solaris_SGSlotAvailabilityChange インジケーション	93
Solaris_XCSystemBoardConfigChange インジケーション	94
Solaris_XCEnvironmentallIndication インジケーション	95
Solaris_XCComponentRemove インジケーション	95
Solaris_XCComponentInsert インジケーション	96
Solaris_XCBoardPowerOn インジケーション	96
Solaris_XCBoardPowerOff インジケーション	96
Solaris_XCDomainIndication インジケーション	96
Solaris_XCDomainConfigChange インジケーション	97
Solaris_XCDomainUp インジケーション	97
Solaris_XCDomainDown インジケーション	97
Solaris_XCDomainStop インジケーション	98
Solaris_XCDomainStateChange インジケーション	98
5. WDR のプログラミング手法	99
システムの状態情報のキャッシュ	99
EventProvider の操作	100
▼ WDR インジケーションを選択して読み取る	100
▼ イベントリスナーを実装する	102
▼ イベントフィルタとイベントハンドラをバインドする	102

InstanceProvider の操作	107
AssociatorProvider の操作	108
MethodProvider の操作	109
A. MOF ファイル	111
索引	139

はじめに

本書『WDR 開発ガイド』は、Web ベースの企業管理では業界標準である WBEM を使用して、DR 操作を遠隔で行うアプリケーションを開発するシステム管理者向けのマニュアルです。

開発者は、Sun WBEM SDK などのソフトウェア開発キット (SDK) を使用して、Java™ 言語などで WDR クライアントアプリケーションを記述することができます。

お読みになる前に

このマニュアルは、UNIX® システム、特に Solaris™ オペレーティング環境ベースのシステムでの作業経験を持つ Sun Fire™ 15K、12K、6800、4810、4800、および 3800 システムのプラットフォーム管理者を対象としています。このような経験がない場合は、まずこのシステムに付属する Solaris ユーザーおよびシステム管理者向けマニュアルを読み、UNIX システム管理のトレーニングの受講を検討してください。

マニュアルの構成

第1章では、WDRの概要と、WDRを使用して実行できる作業について説明します。

第2章では、Solaris オペレーティング環境に組み込まれている Solaris WBEM Services のさまざまな層について説明します。

第3章では、プロセスインジケーションについて説明します。プロセスインジケーションとは、各 WDR クライアントが申請できるシステムイベントの通知のことです。

第4章では、開発者向けに WDR で提供されているすべてのクラス、(システムイベントの) インジケーション、および関連を紹介します。開発者が使用するメソッドとプロパティのすべてについても、この章で説明します。

第5章では、Sun Fire 15K/12K および 6800/4810/4800/3800 システム上でシステム管理を簡略化および自動化する WDR アプリケーションを作成するときに、開発者に役立つプログラミング手法を紹介します。

UNIX コマンド

このマニュアルには、UNIX[®]の基本的なコマンド、およびシステムの停止、システムの起動、デバイスの構成などの基本的な手順の説明は記載されていません。

基本的なコマンドや手順についての説明は、次のマニュアルを参照してください。

- 『Sun 周辺機器 使用の手引き』
- Solaris[™] オペレーティング環境についてのオンラインマニュアル
- 本システムに付属している他のソフトウェアマニュアル

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。	<code>.login</code> ファイルを編集します。 <code>ls -a</code> を実行します。 <code>% You have mail.</code>
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	コマンド行の変換部分を、実際の名前や値と置き換えてください。	<code>rm filename</code> と入力します。 <code>rm ファイル名</code> 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.
WDR のインストール	WDR インストールマニュアル	816-7269
Sun Fire 6800、4810、4800、および 3800 システムでの DR	Sun Fire 6800、4810、4800、3800 システム Dynamic Reconfiguration ユーザーマニュアル	816-3596
Sun Fire 15K および 12K システムでの DR	Sun Fire 15K/12K Dynamic Reconfiguration (DR) ユーザーマニュアル	816-7250
Sun Fire 15K および 12K システムでのシステムレベルのセキュリティ	System Management Services (SMS) 1.2 管理者マニュアル (Sun Fire 15K/12K システム用)	816-7243
Sun Fire 6800、4810、4800、および 3800 システムでのシステムレベルのセキュリティ	Sun Fire 6800/4810/4800/3800 システムプラットフォーム管理ガイド	806-7904
Solaris WBEM Services	Solaris WBEM Services の管理	806-7119

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. (816-7273-11) 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。

第1章

WDR の概要

WDR (WBEM Dynamic Reconfiguration) は、ソフトウェアアプリケーションで使用できるアプリケーションプログラムインタフェース (API) を提供しており、以下のシステム上で動的再構成 (DR) 操作を遠隔実行できます。

- Sun Fire 15K
- Sun Fire 12K
- Sun Fire 6800
- Sun Fire 4810
- Sun Fire 4800
- Sun Fire 3800

ソフトウェア開発者とシステム管理者は、WDR API を使用すると、負荷均衡などの重要なシステム管理機能を遠隔実行する独自のアプリケーションを作成することができます。WDR では DR 操作を実行する従来の方式に代わる方法を提供しており、DR 操作は Sun Fire システムコントローラ (SC) 上と Solaris ドメイン上 (cfgadm システムライブラリ使用による) のいずれかで実行されます。

WDR のハードウェア要件

Sun Fire 6800/4810/4800/3800 システムでは、WDR は *Midframe Service Processor* (MSP) と呼ばれる外部ホスト上で動作します。Sun Fire 15K と 12K システムでは、WDR は システムコントローラ (SC) 上で動作します。

Sun Fire 6800/4810/4800/3800 システムでの MSP のハードウェア要件

MSP に必要な最小構成のハードウェア要件は、次のとおりです。

- Sun4U™ アーキテクチャー

- 8 GB のハードディスク容量
- 128 MB のメモリー
- CD-ROM ドライブ
- SunSwift™ カード、または QuadFast Ethernet カード (最適)

WDR のソフトウェア要件

WDR は、Solaris 8 2/02 または Solaris 9 ソフトウェアが実行されている、Sun Fire 6800/4810/4800/3800 または Sun Fire 15K/12K システムドメイン上で使用できます。WDR は、Solaris オペレーティング環境などのソフトウェアにはバンドルされていません。

Sun Fire 15K/12K システムでのソフトウェア要件

WDR を使用するには、WDR ソフトウェアと Solaris WBEM Services ソフトウェアの両方を SC 上にインストールしておく必要があります。さらに、System Management Services (SMS) バージョン 1.2 ソフトウェアも SC 上にインストールしておく必要があります。

Sun Fire 6800/4810/4800/3800 システムでのソフトウェア要件

WDR を使用するには、WDR ソフトウェアと Solaris WBEM Services ソフトウェアの両方を MSP 上にインストールしておく必要があります。

Web-Based Enterprise Management (WBEM) とは

WDR インタフェースは、業界標準である Web-Based Enterprise Management (WBEM) に基づいて設計され、さまざまなプラットフォーム上でシステム、ネットワーク、およびデバイスを Web ベースで管理することができます。WBEM は、多数の業界リーダー企業で構成される Distributed Management Task Force (DMTF) のメンバーにより開発された標準規格です。

WBEM は、以下の 3 つの主要コンポーネントで構成されています。

- 管理対象オブジェクトのモデル化手法。WBEM では、**Common Information Model (CIM)** を使用して管理対象オブジェクトを表すクラスを作成します。これらのクラスには、管理対象オブジェクトの属性と状態を表すプロパティと、管理対象オブジェクトにおいて実行可能な操作を表すメソッドが含まれます。
- CIM 情報をネットワークで送信できるようにするコード化方法。WBEM では、SGML を強化して拡張性を持たせた **Extensible Markup Language (XML)** を使用して、CIM クラスをコード化します。
- XML 操作をネットワークで伝送するためのカプセル化方法。WBEM では、XML/HTTP または **RMI** を使用して、管理対象オブジェクトからの情報の取得、管理対象オブジェクトのプロパティの設定、および管理対象オブジェクトでの操作を実行するよう送信します。

つまり WBEM では、管理対象オブジェクトを CIM クラス、プロパティ、およびメソッドで表し、CIM 操作を XML/HTTP または RMI メッセージのいずれかで表して、これらのメッセージをネットワークを介して送信します。

このマニュアルでは、WBEM 標準規格について全体的な説明はしません。しかし、WBEM の詳細については、DMTF の Web サイト www.dmtf.org をはじめとして、さまざまな情報源から入手可能です。

Common Information Model (CIM)

WDR は Sun Fire システム専用 CIM スキーマを拡張したものであり、以下のものを表すときに使用されます。

- DR によって管理可能な Sun Fire システム上のリソース
- DR に関連するイベントや、WDR モデルの状態に影響を与えるイベント
- AttachmentPoint クラスとそのサブクラスによって表される DR プラットフォームリソース (接続点など)
- DR プラットフォームリソースのコンテナ (ドメイン、スロットなど)
- WDR スキーマでのオブジェクトの存在や状態、あるいはその両方に影響を与えるイベント
- WDR スキーマでのオブジェクト同士の関連付け
- DR 操作そのもの

Sun Fire 6800/4810/4800/3800 システムのアーキテクチャーは、Sun Fire 15K および 12K システムのアーキテクチャーとかなり異なっています。WDR には、WDR を使用する Sun Fire システムが異なっても、Sun Fire システムのすべてのアーキテクチャーに反映される CIM スキーマが組み込まれています。

CIM スキーマには、すべての Sun Fire システムに共通のオブジェクト、Sun Fire 6800/4810/4800/3800 システム専用のオブジェクト、そして Sun Fire 15K および 12K システム専用のオブジェクトがあります。

システムアーキテクチャー間の共通点はプラットフォームに依存しないスーパークラスに取り込まれ、相違点はプラットフォームに依存しないスーパークラスのプラットフォーム固有のサブクラスに取り込まれます。

プラットフォーム固有および共通の MOF ファイル

WDR で使用される CIM スキーマは、3 つの Managed Object Format (MOF) ファイルに記述されています。これは、Sun Fire システム上の管理対象リソースを表すオブジェクトがすべて定義されている ASCII テキストファイルです。

- WDR_core1.0.mof では、Sun Fire 15K/12K および 6800/4810/4800/3800 システムに共通の要素が定義されています。
- WDR_XC1.0.mof には、Sun Fire 15K/12K システムに固有の要素が定義されています。
- WDR_SG1.0.mof には、Sun Fire 6800/4810/4800/3800 システムに固有の要素が定義されています。

MOF ファイルは、スキーマを提供するだけでなく、ソフトウェア開発者やシステム管理者に WDR CIM スキーマを構成するオブジェクトの形式定義も提供します。

注 – CIM の形式定義については、『Common Information Model, Implementing the Object Model for Enterprise Management』(Winston Bumpus 他共著、Wiley Computer Publishing、copyright 2000、New York、ISBN 0-471-35342-6) を参照してください。

WDR により実行される操作

WDR では、以下の動的再構成操作を遠隔実行することができます。

- Solaris ソフトウェアが動作しているドメインに、システムボード (CPU/メモリーボード) を追加する。まず DR は、ボードをシステムに電氣的に接続し、*connected* (接続された) 状態にします。次に DR は、システムボードをドメインで実行されるすべてのアプリケーションで使用可能となるように構成し、*configured* (構成された) 状態にします。
- 構成解除を行ってから構成操作を行うことにより、1 つのドメインから別のドメインにシステムボードを移動する。
- システムボードをドメインから削除し、他のドメインで使用できるようにする。
- システム上のドメインで現在使用可能な接続点をすべて一覧で表示する。

- 指定されたシステムボードの現在の状態に関する情報 (システムボードの電源の状態、利用可能かどうか、ドメイン割り当てなど) を表示する。
- 構成されているシステムボードのメモリー構成を取り出す。
- メモリーに与える影響に関する情報 (メモリードレイン情報など) を取り出す。この情報によっては、構成されているシステムボードを切り離すことになります。

WDR の機能は DR 自体の基本的な機能と同じものであり、WDR には DR に追加された操作はありません。しかし、ドメインとスロットに関する情報、クラス間の関連、およびイベント通知を提供することによって、WDR では DR の機能を向上させています。

WDR は、パフォーマンスを著しく低下させることなく、DR 操作を効率的に実行することを目的に設計されています。

管理者セキュリティーモデル

WDR では、Sun Fire 15K/12K および 6800/4810/4800/3800 システム上で管理者セキュリティーモデルが適用されます。

Sun Fire 6800/4810/4800/3800 システムレベルでのセキュリティーの実装についての詳細は、『Sun Fire 6800/4810/4800/3800 システムプラットフォーム管理ガイド』(Part No. 806-7904) を参照してください。

Sun Fire 15K/12K システムレベルでのセキュリティーの実装についての詳細は、『System Management Services (SMS) 1.2 管理者マニュアル (Sun Fire 15K/12K システム用)』(Part No. 816-7243) を参照してください。

また、Solaris WBEM Services により使用可能となるセキュリティーについては、第 2 章で説明します。

WDR セキュリティー

/etc/group ファイルに、現在ログインしているユーザーが割り当てられているグループが表示されます。

Sun Fire 6800/4810/4800/3800 システムグループ

Sun Fire 6800/4810/4800/3800 システム上のグループメンバーシップを示す /etc/group ファイルは、手動で編集することができます。

割り当てられているグループメンバーシップに応じて、ユーザーが実行できるすべての操作を以下の表に示します。

表 1-1 グループに応じて実行できるタスク - Sun Fire 6800/4810/4800/3800

グループ	ユーザーが実行できるタスク
なし (すべてのユーザー)	ドメインとスロットの列挙。
spltdadm	ボードの割り当ておよび割り当て解除。
spltop	特別な特権はありません。
sdxadm	<p>x はドメインを意味します。</p> <ul style="list-style-type: none">ドメイン x の接続点の列挙。ユーザーがすべてのドメインで <code>sdxadm</code> グループに割り当てられている場合は、すべての接続点の列挙。接続点状態の変更、ドメイン x の使用可能構成要素リストにあるボードの割り当て、割り当て解除、電源投入、および電源切断。
sdxop	<p>x はドメインを意味します。</p> <ul style="list-style-type: none">ドメイン x の接続点の列挙。ユーザーがすべてのドメインで <code>sdxop</code> グループに割り当てられている場合は、すべての接続点の列挙。

Sun Fire 15K および 12K システムグループ

Sun Fire 15K または 12K システム上のグループメンバーシップを示す `/etc/group` ファイルを変更するには、引数を指定して `/opt/SUNWSMS/bin/smsconfig` スクリプトを実行します。詳細は、『System Management Services (SMS) 1.2 管理者マニュアル (Sun Fire 15K/12K システム用)』を参照してください。

割り当てられているグループメンバーシップに応じて、ユーザーが実行できるすべての操作を以下の表に示します。

表 1-2 グループに応じて実行できるタスク - Sun Fire 15K および 12K

グループ	ユーザーが実行できるタスク
platadm	ボードの割り当て、割り当て解除、電源投入、および電源切断。
platoper	特別な特権はありません。
dmnxadm	<p>x はドメインを意味します。</p> <ul style="list-style-type: none">ドメイン x の接続点の列挙。ユーザーがすべてのドメインで <code>dmnxadm</code> グループに割り当てられている場合は、すべての接続点の列挙。接続点状態の変更、ドメイン x の使用可能構成要素リストにあるボードの割り当て、割り当て解除、電源投入、および電源切断。
dmnxrcfg	<p>x はドメインを意味します。</p> <ul style="list-style-type: none">ドメイン x の接続点の列挙。ユーザーがすべてのドメインで <code>dmnxrcfg</code> グループに割り当てられている場合は、すべての接続点の列挙。接続点状態の変更、ドメイン x の使用可能構成要素リストにあるボードの割り当て、割り当て解除、電源投入、および電源切断。

Solaris WBEM Services

WDR は、Solaris 8 2/02 および Solaris 9 オペレーティング環境に搭載されている Solaris WBEM Services ソフトウェアを拡張したものです。Solaris WBEM Services ソフトウェアを使用すると、管理データに安全にアクセスして操作することができるため、ソフトウェア開発者は Solaris 環境でシステムリソースを管理するクライアントアプリケーションを作成することができます。

Solaris WBEM Services ソフトウェアは、以下の 3 つのレベルで機能するコンポーネントから構成されています。

- アプリケーション層。この層では、WBEM クライアントによって、管理対象リソースのデータが処理および表示されます。アプリケーション層のサービスには、WBEM Workshop、WBEM User Manager、および MOF コンパイラがあります。管理者は、WBEM User Manager を使用すると、承認 WBEM ユーザーの追加と削除、および WBEM ユーザーのアクセス権の設定を行うことができます。
- 管理層。この層では、管理者は CIM API (アプリケーション層と管理層の境界を形成する) を使用して、CIMOM から管理対象リソースのクラスとインスタンスの表示や作成などの操作を実行できます。CIMOM、CIM Repository、およびプロバイダインタフェースはすべて管理層に存在しています。

- プロバイダ層。この層には Solaris プロバイダが常駐しています。このプロバイダでは、Solaris オペレーティング環境の管理対象リソースの CIMOM インスタンスを提供し、管理対象リソースに関する情報を取得します。Solaris プロバイダが、CIMOM と管理対象システムリソース間のインタフェースを形成します。

Solaris WBEM Services コンポーネントは、Solaris ソフトウェアとも、システムハードウェアとも対話します。Solaris WBEM Services ソフトウェアについての詳細は、Solaris WBEM Web サイト www.sun.com/software/solaris/wbem を参照してください。

負荷均衡などのシステム管理アプリケーションの開発者は、Solaris WBEM Services ソフトウェアを使用すると、Sun Fire システムドメイン上での現在使用されているリソースの使用率レベル情報を入手することができます。システムパフォーマンスデータは、WDR 自体からは提供されません。

CIM Object Manager (CIMOM)

WBEM システム上では、CIMOM が CIM オブジェクトの管理を行います。CIMOM は、WBEM クライアント間の情報、CIMOM Repository の情報、および管理対象リソースの情報をプロバイダ経由で転送します。CIMOM は RMI プロトコルを使用して管理アプリケーションからの接続を受け付け、接続されているクライアントに以下のサービスを提供します。

- 管理サービス。CIMOM が CIM データの意味と構文をチェックし、アプリケーション間、CIM Repository、および管理対象リソースのデータを配布します。
- セキュリティーサービス。管理者は、CIM 情報にアクセスするユーザーを制御できます。
- ロギングサービス。このサービスは、ログに動的に CIMOM イベントデータを記録し、ログからそのデータを取り出すアプリケーションを作成するのに使用できるクラスからなります。
- XML サービス。このサービスは XML データを CIM クラスに変換し、XML ベースの WBEM クライアントが CIMOM と通信できるようにします。

WBEM プロバイダ

WDR にはいくつかのプロバイダクラスがあり、これは MOF ファイルに記述されています。WBEM プロバイダは、システム上の CIMOM と管理対象オブジェクトとの間の仲介を行うクラスです。WBEM プロバイダは、管理対象デバイス上で情報の取得と設定を行い、さらに操作を実行することもあります。WBEM プロバイダでは、Solaris WBEM Services ソフトウェアに含まれている CIMOM に取り出した情報を転送して、要求側クライアントに配布します。

CIMOM が CIMOM Repository で取得できない情報に関する要求を受け取った場合は、その要求をプロバイダに転送します。プロバイダではその情報に対する要求を受け取ると、API を使ってその情報を返します。

Solaris WBEM ソフトウェア開発キット (SDK)

WDR アプリケーションの開発者は、Solaris WBEM SDK を使用することができます。ただし、WDR では標準のプロトコルセットが使用されているので、Solaris WBEM SDK を使用するための要件はありません。Solaris WBEM SDK についての詳細は、以下の Sun Developer Connection の Web サイトをご覧ください。

www.sun.com/solaris/wbem

第2章

WDR での Solaris WBEM Services の使用方法

Solaris WBEM Services の概要

Solaris WBEM Services では、WDR アプリケーション開発者に対して、Solaris 8 2/02 または Solaris 9 オペレーティング環境のいずれかが実行されているドメイン上でさまざまな WBEM サービスを提供しています。Solaris WBEM Services は Solaris ソフトウェアに含まれており、WBEM を使用して Solaris ソフトウェアが実行されているシステムを管理するアプリケーションを簡単に作成できるようになります。

この開発者向けマニュアルには、WDR アプリケーション開発者が理解しておかなければならない Solaris WBEM Services に関する情報のみを記載しています。Solaris WBEM Services についての詳細は、以下の Web サイトを参照してください。

<http://www.sun.com/solaris/wbem>

Solaris WBEM Services を使用すれば、管理対象リソース情報に安全にアクセスできるので、WDR を使用するアプリケーションでシステムリソース情報の取得とシステムリソースの管理を行えるようになります。管理対象リソース情報とは、ハードウェアおよびソフトウェアの状態情報、パフォーマンスメトリクス、または負荷均衡の実行やデバイスのフェイルオーバーに対する応答を行うときに管理アプリケーションで必要となるその他のデータなどのことです。この管理対象リソース情報にアクセスできるようにするプログラムが、Solaris WBEM Services に組み込まれている Solaris プロバイダです。

Solaris WBEM Services では、Common Information Model (CIM) を使用して、Solaris ソフトウェアが実行されているシステムにある管理対象オブジェクトを表すスキーマを作成します。CIM オブジェクトは Managed Object Format (MOF) ファイルに指定されています。このファイルは WDR で提供され、WDR のインストール時にコンパイルされます。

Solaris WBEM Services の層

Solaris WBEM Services は、以下の 3 つの層に存在するソフトウェアパッケージです。各層には、WDR アプリケーション開発者にとって重要なソフトウェアコンポーネントが存在しています。

- アプリケーション層
- 管理層
- プロバイダ層

Solaris WBEM Services のアプリケーション層

WDR アプリケーション開発者にとって特に役立つ、以下の Solaris WBEM Services アプリケーション層のソフトウェアプログラムについて、この章で詳しく説明します。

- 13 ページの「Solaris Management Console (SMC) WBEM ログビューア」
- 13 ページの「Managed Object Format (MOF) コンパイラ」
- 21 ページの「Sun WBEM User Manager の使用方法」
- 28 ページの「Solaris Management Console (SMC) ユーザーツールの起動」

Sun WBEM User Manager と SMC ユーザーツール

Sun WBEM User Manager および SMC ユーザーツールアプリケーションを使用すると、システム管理者は、承認ユーザーの追加と削除、管理対象リソースに対する承認ユーザーのアクセス権の設定を行うことができます。

Solaris ソフトウェアが実行されているドメインでセキュリティーを管理するメカニズムには、WBEM アクセス制御リスト (ACL) と Solaris 役割によるアクセス制御 (RBAC) という異なる 2 つのメカニズムがあります。

ユーザーを既存の ACL に追加し、そのユーザーに読み取りアクセス権または読み取り・書き込みアクセス権のいずれかを与えるときには、WBEM User Manager を使用します。

RBAC を使ってユーザーを追加して、そのユーザーの役割と特権を与えるときには、Solaris Management Console (SMC) のユーザーツールを使用します。

ACL と RBAC によるシステムセキュリティーの詳細を含めた、WBEM セキュリティーの管理方法についての詳細は、19 ページの「WBEM セキュリティーサービス」を参照してください。

Solaris Management Console (SMC) WBEM ログビューア

SMC WBEM ログビューアには、コマンドを発行したユーザー名、コマンドが発行されたクライアントコンピュータなどの情報が記録されているログファイルが表示されます。

Solaris WBEM Services には、システムイベントのロギングを使用可能にする API が含まれています。ログファイル、ログファイルに関連する規則、ログファイル形式、開発者がシステムイベントの記録に使用するクラス、およびログサービスを使用可能にする API の使用方法についての詳細は、29 ページの「Solaris WBEM ロギングサービス」(およびそれ以降の節) を参照してください。

Managed Object Format (MOF) コンパイラ

MOF ファイルのコンパイルには MOF コンパイラを使用します。この MOF ファイルは、Solaris ソフトウェアが実行されているシステムの管理対象オブジェクトを表すオブジェクトが CIM スキーマで指定されている ASCII テキストファイルです。

WDR には MOF ファイルが 3 つ含まれており、これらファイルで、管理対象リソースを表すオブジェクトから構成されるスキーマを定義します。MOF ファイルの 1 つは、すべての Sun Fire システムに対して使用されます。もう 1 つのファイルは Sun Fire 15K および 12K システム用で、3 つめのファイルは Sun Fire 6800、4810、4800 または 3800 システム用です。

MOF コンパイラは、クラスとインスタンスが定義されている MOF ファイルのステートメントを読み取ってから、そのステートメントを CIM Object Manager Repository に追加します。これは、管理データに関する情報が集められている記憶領域です。

mofcomp コマンド

MOF コンパイラを起動し、MOF ファイルをコンパイルするには、mofcomp コマンドを使用します。

```
/usr/sadm/bin/mofcomp [-help] [-v] [-sc] [-si] [-sq] [-version]
[-c cimom_ホスト名] [-u ユーザー名] [-p パスワード] ファイル名
```

コマンドの各引数は以下を意味します。

表 2-1 mofcomp コマンドの引数

引数	説明
-help	mofcomp コマンドの引数を一覧表示します。
-v	すべてのコンパイラメッセージが表示される冗長モードで、コンパイラを実行します。
-sc	“set class” オプションを付けてコンパイラを実行します。このオプションは、クラスがすでに存在していて、そのクラスにインスタンスが含まれていない場合はクラスを更新し、クラスが存在していない場合はエラーを返します。-sc オプションを指定しないときは、コンパイラでは接続されているネームスペースに CIM クラスを追加し、クラスがすでに存在している場合にエラーを返します。
-si	“set instance” オプションを付けてコンパイラを実行します。このオプションは、インスタンスがすでに存在している場合はインスタンスを更新し、インスタンスが存在していない場合はエラーメッセージを返します。-si オプションを指定しないときは、コンパイラでは接続されているネームスペースに CIM インスタンスを追加し、インスタンスがすでに存在している場合にエラーを返します。
-sq	“set qualifier types” オプションを付けてコンパイラを実行します。このオプションは、修飾子がすでに存在している場合は修飾子を更新し、修飾子が存在していない場合はエラーメッセージを返します。-sq オプションを指定しないときは、コンパイラでは接続されているネームスペースに CIM 修飾子を追加し、修飾子がすでに存在している場合にエラーを返します。
-version	MOF コンパイラのバージョン番号を表示します。
-c cimom_ホスト名	CIM Object Manager を実行しているシステムを指定します。

表 2-1 mofcomp コマンドの引数 (続き)

引数	説明
-u ユーザー名	<p>CIM Object Manager に接続する際のユーザー名を指定します。CIM Object Manager へのアクセス権を要求するコンパイルでは、-u ユーザー名 オプションを使用してください。</p> <p>-p と -u の両方を指定する場合は、セキュリティーに危険をもたらす可能性があるため、コマンド行にパスワードを入力する必要があります。より安全にパスワードを指定する方法は、コンパイラからパスワードを入力するプロンプトが表示されるように、-p ではなく -u を指定する方法です。16 ページの「mofcomp のパスワード保護のアドバイス」を参照してください。</p>
-p パスワード	<p>CIM Object Manager に接続する際のパスワードを指定します。CIM Object Manager へのアクセス権を要求するコンパイルでは、このオプションを使用してください。</p> <p>-p と -u の両方を指定する場合は、セキュリティーに危険をもたらす可能性があるため、コマンド行にパスワードを入力する必要があります。より安全にパスワードを指定する方法は、コンパイラからパスワードを入力するプロンプトが表示されるように、-p ではなく -u を指定する方法です。16 ページの「mofcomp のパスワード保護のアドバイス」を参照してください。</p>
ファイル名	コンパイルする MOF ファイルの名前。

MOF ファイルのコンパイル

MOF ファイル名に .mof という拡張子が含まれていなくても、MOF ファイルをコンパイルできます。CIM スキーマと Solaris スキーマが記述されている MOF ファイルは、/usr/sadm/mof にあります。

▼ MOF ファイルのコンパイル方法

1. オプションを付けずに MOF コンパイルを実行するときは、以下のコマンドを入力します。

```
# mofcomp ファイル名
```

以下に例を示します。

```
# mofcomp /usr/sadm/mof/Solaris_Application1.0.mof
```

Solaris_Application1.0.mof という名前の MOF ファイルが、CIM Object Manager Repository にコンパイルされます。

mofcomp のパスワード保護のアドバイス

mofcomp コマンドを、-p オプション、または -p と -u オプションを付けて実行し、コマンド行にパスワードを入力した場合には、他のユーザーが後で ps コマンドまたは history コマンドを実行して、その前に入力されたパスワードを表示する可能性があります。このときシステムからは、セキュリティー警告は表示されません。

注 – コマンド行でパスワードを入力するよう求めるコマンドを実行したときは、そのコマンドの実行後、ただちにパスワードを変更してください。これにより、自分の現在のパスワードを他のユーザーが表示することはありません。

安全ではない (セキュリティーが保護されない) コマンドの使用例を以下に示します。

```
% mofcomp -p Log8Rif
```

```
% mofcomp -up molly Log8Rif
```

上記のいずれかの方法で mofcomp コマンドを使用する場合は、コマンド実行後、ただちにパスワードを変更してください。

Solaris WBEM Services の管理層

WDR アプリケーション開発者にとって役立つ Solaris WBEM Services の管理層のソフトウェアプログラムは、Common Information Model (CIM) Object Manager です。

CIM Object Manager とは

Solaris WBEM Services には、WBEM 対応システムでオブジェクトを管理する CIM Object Manager が組み込まれています。個々の CIM オブジェクトは、CPU、入出力ボード、接続点などの管理対象システムオブジェクトを表します。

まず CIM Object Manager では、RMI または XML/HTTP プロトコルのいずれかを使用して、管理アプリケーションへの接続を受け付けて、CIM Object Manager Repository への接続を設定してから、クライアントアプリケーションからのサービス要求を待ちます。サービスには以下のものがあります。

- 管理サービス。CIM データ操作の意味と構文をチェックして最新の CIM 仕様に準拠していることを確認し、アプリケーション (WDR アプリケーションなど) 間の管理データ、CIM Repository、および管理対象リソースを配布します。

- セキュリティーサービス。ユーザーのログイン要求を認証し、システムリソースへのアクセスを制御します。
- ロギングサービス。システムイベントを記録します。

WBEM クライアントは WBEM 対応システムに接続されると、WBEM 操作を要求できます。要求する WBEM 操作には、CIM クラスとインスタンスの作成、表示、および削除、プロパティ値の取り出し、指定したクラス階層内のクラスまたはそのインスタンスの列挙などがあります。

手動による CIM Object Manager の起動と停止

通常、CIM Object Manager は、インストール時と、`/etc/init.d/init.wbem` というユーティリティによりドメインを起動するたびに、自動的に起動されます。このコマンドは、CIM Object Manager だけでなく Solaris Management Console (SMC) も起動します。両者はそれぞれ単一のプロセスとして実行されます。

CIM Object Manager を手動で起動したり停止したりする必要はありませんが、その必要が生じたときには手動で行うこともできます。`init.wbem` ユーティリティの構文は次の通りです。

```
/etc/init.d/init.wbem start|stop|status
```

`start` オプションを指定すると、このコマンドによって呼び出されたドメイン上で CIM Object Manager が起動します。`stop` オプションを指定すると、そのドメイン上で CIM Object Manager が停止します。`status` オプションを指定すると、そのドメイン上での CIM Object Manager の状態が表示されます。

▼ CIM Object Manager を起動する

1. システムプロンプトで以下のコマンドを入力し、スーパーユーザーとなります。
% su
2. root システムプロンプト (#) に、ドメインのスーパーユーザーのパスワードを入力します。
3. 以下のコマンドを入力して、CIM Object Manager を起動します。
/etc/init.d/init.wbem start

▼ CIM Object Manager を停止する

1. システムプロンプトで以下のコマンドを入力し、スーパーユーザーとなります。
% su

2. プロンプトが表示されたら、root システムプロンプト (#) に、ドメインのスーパーユーザーのパスワードを入力します。
3. 以下のコマンドを入力して、CIM Object Manager を停止します。

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

Solaris WBEM Services のプロバイダ層

Solaris WBEM Services のプロバイダ層には、WDR アプリケーション開発者に特に役立つ Solaris プロバイダというソフトウェアプログラムが含まれています。

Solaris プロバイダ

Solaris プロバイダは、管理対象オブジェクトと通信するクラスです。プロバイダは、CIM Object Manager に Solaris オペレーティング環境が実行されているシステム上の管理対象リソースのインスタンスを提供し、管理対象デバイスの情報を取り出して設定します。

WDR アプリケーションが管理対象リソースに関する CIM データにアクセスするときには、まず WBEM でドメイン上のユーザーログイン情報を確認します。デフォルトでユーザーに与えられているのは、Read Only (読み取り専用) アクセス権です。WBEM システムのセキュリティーについての詳細は、19 ページの「WBEM セキュリティーサービス」を参照してください。

CIM Object Manager では、オブジェクトプロバイダ API を使用してプロバイダと通信します。アプリケーションから CIM Object Manager の動的データの要求があるとき、CIM Object Manager はプロバイダ API 経由で応答し、要求された情報をプロバイダに渡します。

プロバイダは、マシン固有の独自のプロバイダであっても、Java Native Interface (JNI) を使って記述した移植性がある (マシンに依存しない) ものでも構いません。なお、Java Native Interface (JNI) は、Java™ Development Kit (JDK™) に含まれていません。

WBEM セキュリティーサービス

WBEM 対応システム上で不正アクセスから CIM オブジェクトを保護する主なセキュリティ機能には、以下の 3 つがあります。

- 認証
- 承認
- 再実行保護

認証

認証とは、Sun Fire システムにおいて、ユーザー、デバイスなどのエンティティーの識別情報を確認するプロセスのことです。正当なユーザーにはシステムリソースへのアクセスを許可し、認証できないユーザーにはアクセスを拒否する場合、認証がよく使用されます。

ユーザーがログインして、ユーザー名とパスワードを入力すると、クライアントではそのパスワードを使って、サーバーで確認される暗号化ダイジェストを生成します。ユーザーが認証されると、CIM Object Manager は MAC トークンを与えて、クライアントセッションを確立します。それ以降の操作はすべてセキュリティ保護されたクライアントセッション内で行われ、すべての操作には、認証プロセス時にネゴシエートされたセッションキーを使用する MAC トークンが含まれます (MAC トークンとは、メッセージを認証するとき使用されるセキュリティ情報が格納され、遠隔呼び出しに追加されるトークンパラメタのことです)。

承認

承認とは、ユーザー、プログラム、プロセスに対し、システムリソースにアクセスする権利を与えるプロセスのことです。承認は、認証に続いて行われます。

CIM Object Manager でユーザーの識別情報が認証された後、その識別情報を使用して、ユーザーがアプリケーションや関連タスクの実行を許可されているかどうか確認できます。CIM Object Manager では、機能ベースの承認をサポートしています。この承認では、特権ユーザーが他のユーザーに読み取り・書き込みアクセス権を割り当てることができます。このようにして承認されたユーザーは、既存の Solaris ユーザーアカウントに追加されます。

再実行保護

再実行保護は、セッションキーを確認することにより、未承認クライアントが、他のクライアントのメッセージを受信してサーバーに送信するのを防止するサービスです。

クライアントは、他のクライアントから CIM Object Manager に送信された最新のメッセージをコピーすることはできません。CIM Object Manager は、認証時にネゴシエートされたセッションキーに基づいて、すべてのメッセージで MAC を使用することにより、そのセッションを開始してクライアントサーバー認証に加わっていたクライアントとの間で、クライアントサーバーセッションの全通信が確かに行われていることを保証します。

MAC を使用して、メッセージがそのセッションに対してもともと認証を受けていたクライアントから送信されたものであること、およびそのメッセージが他のクライアントによって再実行されたものでないことを確認します。このタイプのメカニズムは、RMI メッセージを検証するときに WBEM で使用されます。ユーザー認証の交換中にネゴシエートされたセッションキーは、メッセージの MAC トークンのセキュリティー情報の暗号化に使用されます。

デジタル署名

WBEM セキュリティーサービスでは、メッセージのデジタル署名は実行されません。

セキュリティーの実装

Solaris オペレーティング環境内でセキュリティーを管理するときは、WBEM アクセス制御リストを使用します。

WBEM アクセス制御リスト

アクセス制御リストによるセキュリティーは、Solaris_Acl1.0.mof ファイルに定義されているクラスを使って実装します。Solaris WBEM Services 固有のアクセス制御リストによるセキュリティーは、Solaris WBEM Services のデフォルトの承認方式であり、すべての CIM 操作に適用されます。定義されているクラスのインスタンスによって、WBEM ユーザーやネームスペース、あるいはその両方に割り当てられるデフォルトの承認が決定されます。

ユーザーを既存のアクセス制御リストに追加して、そのユーザーに読み取りアクセス権または読み取り・書き込みアクセス権のいずれかを割り当てるときは、Sun WBEM User Manager を使用します。これについては、「Sun WBEM User Manager」の節で説明します。Sun WBEM User Manager は、`/usr/sadm/bin/wbemadmin` ディレクトリにあります。

詳細は、21 ページの「Sun WBEM User Manager の使用方法」を参照してください。

Sun WBEM User Manager の使用方法

Sun WBEM User Manager を使用すると、特権ユーザーは、承認ユーザーの追加と削除、および承認ユーザーの CIM オブジェクトへのアクセス権の設定を、WBEM 対応システム上で行うことができます。すべてのユーザーは Solaris ユーザーアカウントを持っている必要があります。

Sun WBEM User Manager では、個々のネームスペースで、またはユーザーとネームスペースの組み合わせで、アクセス権を設定できます。ユーザーを追加してネームスペースを選択すると、ユーザーは指定したネームスペース内の CIM オブジェクトへの読み取りアクセス権をデフォルトで持つことができます。

1 つのネームスペースに対するすべてのユーザーのアクセスを制限してから、ユーザーごとに、そのネームスペースに対する読み取り、読み取り・書き込み、または書き込みアクセスを許可することができます。

個々の管理対象オブジェクトにはアクセス権を設定できません。ただし、ネームスペース内と各ユーザーについては、すべての管理対象オブジェクトのアクセス権を設定できます。

スーパーユーザーとしてログインしている場合は、WBEM User Manager を使用して、以下のような CIM オブジェクトに対するアクセス権を設定できます。

- **Read Only (読み取り専用)** — CIM スキーマ内のオブジェクトの読み取りのみを許可します。読み取り専用アクセス権を持っているユーザーは、インスタンスとクラスを取り出すことはできますが、CIM オブジェクトの作成、削除、変更はできません。デフォルトのユーザーアクセス権です。
- **Read/Write (読み取り・書き込み)** — すべての CIM クラスおよびインスタンスに対する読み取り、書き込み、および削除をすべて許可します。
- **Write (書き込み)** — すべての CIM クラスおよびインスタンスに対する書き込みと削除を許可しますが、読み取りは許可しません。
- **None (アクセス権なし)** — CIM クラスおよびインスタンスに対するアクセスを許可しません。

▼ Sun WBEM User Manager を起動する

1. スーパーユーザーとして、以下のコマンドをコマンド行に入力します。

```
# /usr/sadm/bin/wbemadmin
```

Sun WBEM User Manager が読み込まれ、「ログイン」ダイアログボックスが表示されます。コンテキストヘルプを使用するには、ダイアログのフィールドをクリックして、「コンテキストヘルプ」パネルを表示します。

2. 「ログイン」ダイアログボックスの「ユーザー名」フィールドにユーザー名を入力します。

ログインするには、`root\security` ネームスペースへの読み取りアクセス権を持っている必要があります。デフォルトでは、Solaris ユーザーは `guest` アクセス権を持っています。このアクセス権は、ユーザーにデフォルトのネームスペースへの読み取りアクセスを許可します。読み取りアクセス権を持っているユーザーは、ユーザー特権を表示することはできますが、変更はできません。

ユーザーにアクセス権を与えるには、スーパーユーザーか、または `root\security` ネームスペースへの書き込みアクセス権を持っているユーザーとしてログインする必要があります。

3. 「ログイン」ダイアログボックスの「パスワード」フィールドに、ユーザーアカウントのパスワードを入力します。
4. 「了解」をクリックします。

「User Manager」ダイアログボックスが表示されます。ここには、ユーザーの一覧と、現在のドメイン上のネームスペース内の WBEM オブジェクトに対する各ユーザーのアクセス権が表示されています。

▼ ユーザーにデフォルトのアクセス権を与える

1. Sun WBEM User Manager を起動します。
2. 「User Manager」ダイアログボックスの「ユーザーアクセス」部分で「追加」をクリックします。

ドメイン上で使用可能なネームスペースがすべて表示されているダイアログボックスが表示されます。

3. 「ユーザー名」フィールドに Solaris ユーザーのアカウント名を入力します。
4. 使用可能なネームスペースのリストからネームスペースを 1 つ選択します。
5. 「了解」をクリックします。

「User Manager」ダイアログボックスに表示されているユーザーのリストに、そのユーザー名が追加されます。

6. 「了解」をクリックして変更を保存し、「User Manager」ダイアログボックスを閉じます。または、「適用」をクリックして変更を保存し、「User Manager」ダイアログボックスを閉じます。

これで、指定したユーザーに、選択したネームスペースの CIM オブジェクトに対する読み取り専用アクセス権が与えられました。

▼ ユーザーのアクセス権を変更する

1. Sun WBEM User Manager を起動します。
2. ユーザーのアクセス権が表示されているリストから、変更するユーザーを選択します。
3. そのユーザーに読み取り専用アクセス権を与えるには、「読み取り」チェックボックスをクリックします。そのユーザーに書き込みアクセス権を与えるには、「書き込み」チェックボックスをクリックします。
4. 「了解」をクリックして変更を保存し、「User Manager」ダイアログボックスを閉じます。または、「適用」をクリックして変更を保存し、「User Manager」ダイアログボックスを閉じます。

▼ ユーザーのアクセス権を削除する

1. Sun WBEM User Manager を起動します。
2. 「User Manager」ダイアログボックスの「ユーザーアクセス」部分で、ユーザーのアクセス権が表示されているリストから、アクセス権を削除するユーザーを選択します。
3. 「削除」をクリックし、そのユーザーのネームスペースに対するアクセス権を取り消します。
確認を求めるダイアログボックスが表示され、そのユーザーのアクセス権を取り消してよいか確認が求められます。「了解」をクリックして、次に進みます。
4. 「了解」をクリックして変更を保存し、「User Manager」ダイアログボックスを閉じます。または、「適用」をクリックして変更を保存し、「User Manager」ダイアログボックスを閉じます。

▼ ネームスペースへのアクセス権を設定する

1. Sun WBEM User Manager を起動します。

2. 「User Manager」ダイアログボックスの「ネームスペースへのアクセス」部分で「追加」をクリックします。

ドメインで使用可能なネームスペースがすべて表示されているダイアログボックスが表示されます。

3. アクセス権を設定するネームスペースを選択します。

ユーザーにはデフォルトでネームスペースへの読み取り専用アクセスが許可されており、「読み取り」チェックボックスがチェックされています。書き込みアクセス権を与えるには、「書き込み」チェックボックスをクリックします。読み取り・書き込みアクセス権を与えるには、「読み取り」と「書き込み」チェックボックスをクリックします。ネームスペースへのアクセスを許可しないときは、「読み取り」と「書き込み」の両方のチェックボックスがチェックされていないことを確認します。

4. 「了解」をクリックして変更を保存し、「User Manager」ダイアログボックスを閉じます。または、「適用」をクリックして変更を保存し、「User Manager」ダイアログボックスを閉じます。

▼ ネームスペースへのアクセス権を削除する

1. Sun WBEM User Manager を起動します。
2. 「User Manager」ダイアログボックスの「ネームスペースへのアクセス」部分で、アクセス権を削除するネームスペースを選択し、「削除」をクリックします。

これにより、選択したネームスペースからアクセス権が削除され、「User Manager」ダイアログボックスに表示されているネームスペースのリストからそのネームスペースが削除されます。

3. 「了解」をクリックして変更を保存し、「User Manager」ダイアログボックスを閉じます。または、「適用」をクリックして変更を保存し、「User Manager」ダイアログボックスを閉じます。

API を使用したアクセス制御の設定

Sun WBEM SDK API を使用して、ネームスペースでのアクセス制御や、ユーザーごとのアクセス制御を設定することができます。以下のセキュリティークラスが `root\security` ネームスペースに格納されています。

- `Solaris_Acl` - Solaris アクセス制御リスト (ACL) の基底クラス。このクラスでは、文字列プロパティ `capability` を定義し、そのデフォルト値を `"r"` (読み取り専用) に設定します。
- `Solaris_UserAcl` - 指定されたネームスペース内で、CIM オブジェクトに対してユーザーが持っているアクセス制御です。

- Solaris_NamespaceAcl - ネームスペース上のアクセス制御です。

Solaris_UserACL クラスのインスタンスを作成してから、API を使用してそのインスタンスのアクセス権を変更すると、ネームスペース内の CIM オブジェクトに対するアクセス制御を、ユーザーごとに設定できます。同様に、Solaris_NameSpaceACL クラスのインスタンスを作成してから、API (setInstance メソッドなど) を使用して、そのインスタンスのアクセス権を設定すると、ネームスペースでのアクセス制御を設定することができます。

この 2 つのクラスを効果的に組み合わせて使用するには、まず Solaris_NameSpaceACL クラスを使って、ネームスペースのオブジェクトに対するすべてのユーザーのアクセスを制限し、次に Solaris_UserACL クラスを使って、選択したユーザーにそのネームスペースへのアクセスを許可する方法があります。

注 - アクセス制御リスト (ACL) は、DMTF が作成中の標準規格によって管理されます。現在 Solaris ACL スキーマは CIM に準拠していますが、DMTF により ACL 標準規格が最終的に策定されたときには、このスキーマを変更する必要があります。Solaris ACL スキーマクラスを使用して記述しているプログラムも、この変更の対象となる可能性があります。

Solaris_UserAcl クラス

Solaris_UserAcl クラスは Solaris_Acl 基底クラスを拡張したもので、この基底クラスから “r” (Read Only) のデフォルト値を持つ文字列プロパティ `capability` を継承します。

Solaris_UserAcl クラスの `capability` プロパティを以下のいずれかの値に設定すると、アクセス権を設定できます。

表 2-2 `capability` プロパティの設定

アクセス権	説明
r	Read Only (読み取り専用)
rw	Read/Write (読み取り・書き込み)
w	Write (書き込み)
none	Only (単独使用)

Solaris_UserAcl クラスでは、capability プロパティーのほかにも以下の2つの主要なプロパティーが定義されます。1つのネームスペースに存在できるのは、この2つの namespace-username ACL でどちらか1つのインスタンスだけです。

表 2-3 Solaris_UserAcl クラスの主要プロパティー

プロパティー	データ型	目的
nspace	文字列	この ACL が適用されるネームスペースを特定する。
username	文字列	この ACL が適用されるユーザーを特定する。

▼ ユーザーごとにアクセス制御を設定する

1. 以下のコードを使用して、Solaris_UserAcl クラスのインスタンスを作成します。

```
...
/* Create a namespace object initialized with root\security
   (name of namespace) on the local host. */
CIMNameSpace cns = new CIMNameSpace("", "root\security");
// Connect to the root\security namespace as root.
cc = new CIMClient(cns, "root", "root_password");
// Get the Solaris_UserAcl class
cimclass = cc.getClass(new CIMObjectPath("Solaris_UserAcl");
// Create a new instance of the Solaris_UserAcl
class ci = cimclass.newInstance(); ...
```

2. 以下のコードを使用して、capability プロパティーを目的のアクセス権に設定します。

```
...
/* Change the access rights (capability) to read/write for
   user Guest
on objects in the root\molly namespace.*/
ci.setProperty("capability", new CIMValue(new String("rw")));
ci.setProperty("nspace", new CIMValue(new String("root\
molly")));
ci.setProperty("username", new CIMValue(new String("guest")));
...
```

- 以下のコードを使用して、新規作成したインスタンスを更新します。

```
...
// Pass the updated instance to the CIM Object Manager
cc.setInstance(new CIMObjectPath(), ci);
...
```

Solaris_NamespaceAcl クラス

Solaris_NamespaceAcl クラスは Solaris_Acl 基底クラスを拡張したもので、この基底クラスから "r" (GUEST と全ユーザーに対する読み取り専用アクセス権) のデフォルト値を持つ文字列プロパティ `capability` を継承します。Solaris_NamespaceAcl クラスでは、以下の主要なプロパティが定義されます。

プロパティ	データ型	目的
<code>namespace</code>	文字列	このアクセス制御リスト (ACL) が適用されるネームスペースを特定する。1 つのネームスペースに存在できるのは、 <code>namespace ACL</code> で 1 つのインスタンスだけです。

▼ ネームスペースでのアクセス制御を設定する

- 以下のコードを使用して、Solaris_namespaceACL クラスのインスタンスを作成します。

```
...
/* Create a namespace object initialized with root\security
   (name of namespace) on the local host. */
CIMNameSpace cns = new CIMNameSpace("", "root\security");
// Connect to the root\security namespace as root.
cc = new CIMClient(cns, "root", "root_password");
// Get the Solaris_namespaceAcl class
cimclass = cc.getClass(new
    CIMObjectPath("Solaris_namespaceAcl");
// Create a new instance of the Solaris_namespaceAcl
class ci = cimclass.newInstance();
...
```

2. 以下のコードを使用して、目的のアクセス権を許可するよう capability プロパティを設定します。

```
...
/* Create a namespace object initialized with root\security
   (name of namespace) on the local host. */
CIMNameSpace cns = new CIMNameSpace("", "root\security");
// Connect to the root\security namespace as root.
cc = new CIMClient(cns, "root", "root_password");
// Get the Solaris_namespaceAcl class
cimclass = cc.getClass(new
    CIMObjectPath("Solaris_namespaceAcl");
// Create a new instance of the Solaris_namespaceAcl
class ci = cimclass.newInstance();
...
```

3. 以下のコードを使用して、新しく作成したインスタンスを更新します。

```
// Pass the updated instance to the CIM Object Manager
cc.setInstance(new CIMObjectPath(), ci);
```

Solaris Management Console (SMC) ユーザーツールの起動

SMC ユーザーツールを使用すると、ユーザーを既存の役割に追加し、既存のユーザーに RBAC 権を許可することができます。RBAC 権は、SMC ユーザーツールの「権利」部分で管理されます。

▼ SMC ユーザーツールを起動する

1. 以下のコマンドを入力して、SMC 起動コマンドの場所に移動します。

```
# cd /usr/sbin
```

2. 以下のコマンドを入力して、SMC を起動します。

```
# smc
```

3. アプリケーションが読み込まれ、ユーザーインターフェースが表示されたら、左側の「ナビゲーション」パネルの「このコンピュータ」をダブルクリック (または、「このコンピュータ」の横にある展開・縮小アイコンをクリック) して、「このコンピュータ」の下位のツリーを展開します。
4. 左側の「ナビゲーション」パネルの「システム構成」をダブルクリック (または、「システム構成」の横にある展開・縮小アイコンをクリック) して、「システム構成」の下位のツリーを展開します。「ユーザー」アイコンが表示されます。
5. 「ユーザー」アイコンをクリックして、ユーザーツールを起動します。

注 – Solaris Management Console の使用方法の詳細は、smc (1m) マニュアルページを参照してください。

Solaris WBEM ロギングサービス

WBEM ロギングサービスを使用すると、システム管理者はシステムイベントを監視し、その発生状況を判定することができます。

ロギングサービスでは、サービスプロバイダから返されるようにプログラムされているアクションと、Solaris WBEM Services コンポーネントが実行するアクションをすべて記録します。また、情報とエラーメッセージもログに記録できます。

たとえば、ユーザーがシリアルポートを使用不可にしたときに、シリアルポートプロバイダによって自動的にこの情報をログに記録することができます。あるいは、システムエラーなどの障害が発生したときには、管理者はログを調べて、その障害の原因を突き止めることができます。

イベントに応じて、すべてのコンポーネント、アプリケーション、およびプロバイダが自動的にロギングを開始します。たとえば、CIM Object Manager はインストールされて起動されると、自動的にイベントを記録します。

WBEM 環境用に開発するアプリケーションとプロバイダでログを記録するよう設定することができます。詳細は、33 ページの「API を使用した Solaris WBEM ロギングの有効化」を参照してください。

Solaris Management Console (SMC) ログビューアにログデータを表示して、設定したログ機能をデバッグすることができます。ログファイルの表示方法の詳細は、41 ページの「Solaris WBEM ログビューア」と smc(1m) マニュアルページを参照してください。

Solaris WBEM Services のログファイル

イベントを記録するようにアプリケーションやプロバイダを設定すると、そのイベントはログファイルに記録されます。ログの記録はすべて `/var/sadm/wbem/log` のパスに格納されます。ログファイルでは、次の命名規則が使用されます。

`wbem_log.#`

ここで、# はログファイルのバージョンを示すために追加される数字です。

`wbem_log.1` のように “.1” が付いているログファイルが最後に保存されたバージョンです。“.2” が付いているログファイルはその前のバージョンで、このように順に番号が付いています。すべてのバージョンのログファイルは、`/var/sadm/wbem/log` にアーカイブとしてまとめて入れられています。

以下の 2 つの条件のいずれかが満たされたときに、ログファイルは .1 というファイル拡張子付きで名前変更され、保存されます。

- 現在のファイルが、`Solaris_LogServiceProperties` クラスで指定されているファイルサイズの限界に達した場合。デフォルト値は `wbemService.properties` ファイルで設定されています。
ログファイルの使用方法を `Solaris_LogServiceProperties` クラスのプロパティがどのように制御するかについては、30 ページの「Solaris WBEM Services のログファイル規則」を参照してください。
- `Solaris_LogService` クラスの `clearLog()` メソッドが、現在のログファイルで呼び出された場合。
`Solaris_LogService` クラスとそのメソッドについては、32 ページの「`Solaris_LogService` クラス」を参照してください。

Solaris WBEM Services のログファイル規則

`Solaris_LogServiceProperties` クラスは `Solaris_Core1.0.mof` で定義されています。`Solaris_LogServiceProperties` クラスには、以下のログファイルの属性を制御するプロパティがあります。

- ログファイルを書き込むディレクトリ
- ログファイルの名前
- ファイル拡張子付きで名前変更されて保存されるまでに、ログファイルにログを格納できる最大サイズ
- アーカイブに保持できるログファイル数
- Solaris オペレーティング環境のデフォルトのログシステムである SysLog にログデータを書き込む機能

これらの属性のいずれかを、ログファイルにデータを書き込むアプリケーションで指定するには、`Solaris_LogServiceProperties` クラスの新しいインスタンスを作成し、その関連プロパティの値を設定します。新しいインスタンスのプロパティ値の設定方法についての詳細は、39 ページの「Solaris WBEM ログイングプロパティの設定」を参照してください。

Solaris WBEM Services のログファイル形式

ログイングサービスには、アプリケーション、システム、およびセキュリティーという3つのログ記録のカテゴリがあります。ログ記録は、情報であることも、エラーや警告から生成された記録データであることもあります。ログに記入可能なデータについて、標準的なフィールド設定が定義されていますが、必ずしもログではすべてのフィールドを使用する必要はありません。たとえば、情報ログの場合は、イベントを表す簡潔なメッセージを記録することができます。エラーログの場合は、より詳細なメッセージを記録することができます。

一部のログデータフィールドでは、`CIM Repository` のデータを特定する必要があります。これらのフィールドは、`Solaris_LogRecord` クラスの読み取り専用キー修飾子でフラグが立てられているプロパティです。これらのフィールドには値を設定することはできません。ただし、ログファイルの以下のフィールドであれば、いずれも値を設定できます。

- `Category` — ログ記録のタイプ
- `Severity` — データをログファイルに書き込む条件の重大度
- `AppName` — データが取得されたときに使用していたアプリケーションの名前
- `UserName` — ログデータが生成されたときに、アプリケーションを使用していたユーザー名
- `ClientMachineName` — ログデータを生成したイベントが発生したコンピュータ名
- `ServerMachineName` — ログデータを生成したイベントが発生したサーバー名
- `SummaryMessage` — イベントの発生を説明する簡単なメッセージ
- `DetailedMessage` — イベントの発生を説明する詳細なメッセージ
- `Data` — アプリケーションとプロバイダがログメッセージを解釈するために提示できるコンテキスト情報

Solaris WBEM ログクラス

Solaris WBEM ログイングサービスでは、`Solaris_LogRecord` と `Solaris_LogService` の2つの Solaris スキーマクラスを使用します。

Solaris_LogRecord クラス

Solaris_LogRecord クラスは、Solaris_Core1.0.mof ファイルで定義され、ログファイルのエントリをモデル化します。イベントに応じてアプリケーションやプロバイダにより Solaris_LogRecord クラスが呼び出されると、Solaris_LogRecord クラスでは、イベントによって生成されたすべてのデータをログファイルに書き込みます。Solaris プロバイダの一部として Solaris_LogRecord クラスの定義を見るには、テキストエディタに Solaris_Core1.0.mof ファイルを表示します。Solaris_Core1.0.mof ファイルは /usr/sadm/mof にあります。

Solaris_LogRecord クラスでは、プロパティのベクトルとキー修飾子を使用して、イベント、システム、ユーザー、およびデータを生成するアプリケーションまたはプロバイダの属性を指定します。読み取り専用修飾子の値は、アプリケーションと CIM Repository との間で使用するために透過的に生成されます。たとえば、RecordID という値はログエントリを一意に特定しますが、生成されたデータを見るときには、ログ形式の一部としては表示されません。

書き込み可能な修飾子の値を設定できます。たとえば、イベントが発生したシステムを特定する、ClientMachineName や ServerMachineName といったプロパティの修飾子の値を設定することができます。

SysLogFlag プロパティを真に設定すると、ログ記録の詳細メッセージが自動的に UNIX システムの syslog デーモンに送信されます。

Solaris_LogService クラス

Solaris_LogService クラスは、ロギングサービスの操作を制御し、ログデータの処理方法を定義します。このクラスには、発行元アプリケーションから CIM Object Manager に特定のイベントに関するデータを配布するときにアプリケーションで使用できる 1 組のメソッドがあります。そのデータが CIM Object Manager からの応答を生成するトリガーとなり、CIM Repository からのデータの取得などが行われます。

Solaris_LogService クラスでは、以下のメソッドを使用します。

- clearLog — 現在のログファイルの名前変更と保存、保存されているログファイルの削除を行う。
- getNumRecords — 特定のログファイルに含まれているレコード数を返す。
- listLogFiles — /usr/sadm/wbem/log に格納されているすべてのログファイルのリストを返す。
- getCurrentLogFileName — 最新のログファイル名を返す。
- getNumLogFiles — /usr/sadm/wbem/log に格納されているログファイル数を返す。
- getLogFileSize — 特定のログファイルのサイズをメガバイト単位で返す。

- `getSyslogSwitch` — Solaris オペレーティング環境のロギングサービスである `SysLog` に、ログデータを送信できるようにする。
- `getLogStorageName` — ログファイルが格納されているホストコンピュータまたはデバイスの名前を返す。
- `getLogFileDir` — ログファイルが格納されているディレクトリのパスと名前を返す。

`Solaris_LogServiceProperties` クラスを使用すると、ロギングのプロパティを設定できます。39 ページの「Solaris WBEM ロギングプロパティの設定」を参照してください。

`Solaris_LogService` クラスの定義は、`/usr/sadm/mof` にある `Solaris_Core1.0.mof` ファイルで見ることができます。

API を使用した Solaris WBEM ロギングの有効化

現在、ログファイルの内容はログビューアで見ることができます。しかし、カスタマイズされた方法でのログファイルの表示を望む場合は、独自のログビューアを開発することができます。ロギングアプリケーションプログラミングインタフェース (API) を使用すると、ログビューアを開発できます。この API では、以下のことを実行できます。

- アプリケーションからログファイルへのデータの書き込み
- ログファイルからログビューアへのデータの読み取り
- ログデータの処理方法を指定するロギングプロパティの設定

Solaris WBEM ログファイルへのデータの書き込み

アプリケーションでログファイルにデータを書き込めるようにするには、以下の主なタスクが必要です。

- `Solaris_LogRecord` クラスの新しいインスタンスを作成する
- ログファイルに書き込まれるプロパティを指定し、プロパティ修飾子の値を設定する
- 出力する新しいインスタンスとプロパティを設定する

▼ Solaris_LogRecord のインスタンスを作成してデータを書き込む

1. 必要な Java クラスをすべてインポートします。以下に挙げるクラスは最小限必要なクラスです。

```
import java.rmi.*;
import com.sun.wbem.client.CIMClient;
import com.sun.wbem.cim.CIMInstance;
import com.sun.wbem.cim.CIMValue;
import com.sun.wbem.cim.CIMProperty;
import com.sun.wbem.cim.CIMNameSpace;
import com.sun.wbem.cim.CIMObjectPath;
import com.sun.wbem.cim.CIMClass;
import com.sun.wbem.cim.CIMException;
import com.sun.wbem.solarisprovider.*;
import java.util.*;
```

2. **public** クラス `CreateLog` を宣言し、`CIMClient`、`CIMObjectPath`、および `CIMNameSpace` クラスのインスタンスを作成します。

```
public class CreateLog {
    public static void main(String args[]) throws CIMException {
        if ( args.length != 3 ) {
            System.out.println("Usage: CreateLog host username password");
            System.exit(1);
        }
        CIMClient cc = null;
        CIMObjectPath cop = null;
        try {
            CIMNameSpace cns = new CIMNameSpace(args[0]);
            cc = new CIMClient(cns, args[1], args[2]);
```

3. 返されるプロパティのベクトルを指定します。修飾子のプロパティの値を設定します。

```
Vector keys = new Vector();
CIMProperty logsvKey;
logsvKey = new CIMProperty("category");
logsvKey.setValue(new CIMValue(new Integer(2)));
keys.addElement(logsvKey);
```

```

logsvcKey = new CIMProperty("severity");
logsvcKey.setValue(new CIMValue(new Integer(2)));
keys.addElement(logsvcKey);
logsvcKey = new CIMProperty("AppName");
logsvcKey.setValue(new CIMValue("SomeApp"));
keys.addElement(logsvcKey);
logsvcKey = new CIMProperty("UserName");
logsvcKey.setValue(new CIMValue("molly"));
keys.addElement(logsvcKey);
logsvcKey = new CIMProperty("ClientMachineName");
logsvcKey.setValue(new CIMValue("dragonfly"));
keys.addElement(logsvcKey);
logsvcKey = new CIMProperty("ServerMachineName");
logsvcKey.setValue(new CIMValue("spider"));
keys.addElement(logsvcKey);
logsvcKey = new CIMProperty("SummaryMessage");
logsvcKey.setValue(new CIMValue("brief_description"));
keys.addElement(logsvcKey);
logsvcKey = new CIMProperty("DetailedMessage");
logsvcKey.setValue(new CIMValue("detailed_description"));
keys.addElement(logsvcKey);
logsvcKey = new CIMProperty("data");
logsvcKey.setValue(new CIMValue("0xfe 0x45 0xae 0xda"));
keys.addElement(logsvcKey);
logsvcKey = new CIMProperty("SyslogFlag");
logsvcKey.setValue(new CIMValue(new Boolean(true)));
keys.addElement(logsvcKey);

```

4. ログ記録に対して、CIMObjectPath クラスの新しいインスタンスを宣言します。

```

CIMObjectPath logreccop = new CIMObjectPath("Solaris_LogRecord",
keys);

```

5. Solaris_LogRecord の新しいインスタンスを宣言します。ファイルに書き込むプロパティのベクトルを設定します。

```

CIMInstance ci = new CIMInstance();
    ci.setClassName("Solaris_LogRecord");
    ci.setProperties(keys);
    //System.out.println(ci.toString());

```

```

        cc.setInstance(logreccop, ci);
    }
    catch (Exception e) {
        System.out.println("Exception: "+e);
        e.printStackTrace();
    }
}

```

6. データがログファイルに書き込まれたならば、セッションを閉じます。

```

// close session.
if(cc != null) {
    cc.close();
}

```

Solaris WBEM ログファイルからのデータの読み取り

アプリケーションでログファイルからログビューアにデータを読み取れるようにするには、以下のタスクが必要です。

- Solaris_LogRecord クラスのインスタンスを列挙する
- 目的のインスタンスを取得する
- 出力デバイス (通常はログビューアのユーザーインターフェース) にインスタンスのプロパティを出力する

▼ Solaris_LogRecord クラスのインスタンスを取得してデータを読み取る

1. 必要な Java クラスをすべてインポートします。以下に挙げるクラスは最小限必要なクラスです。

```

import java.rmi.*;
import com.sun.wbem.client.CIMClient;
import com.sun.wbem.cim.CIMInstance;
import com.sun.wbem.cim.CIMValue;
import com.sun.wbem.cim.CIMProperty;
import com.sun.wbem.cim.CIMNameSpace;
import com.sun.wbem.cim.CIMObjectPath;
import com.sun.wbem.cim.CIMClass;

```

```

import com.sun.wbem.cim.CIMException;
import com.sun.wbem.solarisprovider.*;
import java.util.*;
import java.util.Enumeration;

```

2. クラス ReadLog を宣言します。

```

public class ReadLog
{
    public static void main(String args[]) throws
        CIMException
    {
        if ( args.length != 3)
        {
            System.out.println("Usage: ReadLog host username password");
            System.exit(1);
        }
    }
}

```

3. ReadLog クラスの CIMClient、CIMObjectPath、および CIMNameSpace の値を設定します。

```

CIMClient cc = null;
CIMObjectPath cop = null;
try { CIMNameSpace cns = new CIMNameSpace(args[0]);
    cc = new CIMClient(cns, args[1], args[2]);
    cop = new CIMObjectPath("Solaris_LogRecord");
}

```

4. Solaris_LogRecord のインスタンスを列挙します。

```

Enumeration e = cc.enumInstances(cop, true);
for (; e.hasMoreElements(); ) {

```

5. 出力デバイスにプロパティ値を送信します。

```

System.out.println("-----");
CIMObjectPath op = (CIMObjectPath)e.nextElement();
CIMInstance ci = cc.getInstance(op);
System.out.println("Record ID : " +

    (((Long)ci.getProperty("RecordID").getValue().getValue()).longVal
ue()));
System.out.println("Log filename : " +
    ((String)ci.getProperty("FileName").getValue().getValue()));

```

```

int categ = 0
    (((Integer)ci.getProperty("category").getValue().intValue()
    .intValue());
if (categ == 0)
    System.out.println("Category : Application Log");
else if (categ == 1)
    System.out.println("Category : Security Log");
else if (categ == 2)
    System.out.println("Category : System Log");
int severity =
    (((Integer)ci.getProperty("severity").getValue().intValue()
    .intValue());
if (severity == 0)
    System.out.println("Severity : Informational");
else if (severity == 1)
    System.out.println("Severity : Warning Log!");
else if (severity == 2)
    System.out.println("Severity : Error!!");
System.out.println("Log Record written by : " +
    ((String)ci.getProperty("AppName").getValue().getValue());
System.out.println("User : " +
    ((String)ci.getProperty("UserName").getValue().getValue());
System.out.println("Client Machine : " +
    ((String)ci.getProperty("ClientMachineName").getValue().getValue(
    )));
System.out.println("Server Machine : " +
    ((String)ci.getProperty("ServerMachineName").getValue().getValue()
    ));
System.out.println("Summary Message : " +
    ((String)ci.getProperty("SummaryMessage").getValue().getValue()
    ));
System.out.println("Detailed Message : " +
    ((String)ci.getProperty("DetailedMessage").getValue().getValue()
    ));
System.out.println("Additional data : " +
    ((String)ci.getProperty("data").getValue().getValue()));
boolean syslogflag =
    ((Boolean)ci.getProperty("syslogflag").getValue().getValue()).
    booleanValue();
if (syslogflag == true) {

```

```

        System.out.println("Record was written to syslog as well");
    } else {
        System.out.println("Record was not written to syslog");
    }
    System.out.println("-----");

```

6. エラー条件が発生した場合は、ユーザーにエラーメッセージを返します。

```

...
catch (Exception e) {
    System.out.println("Exception: "+e);
    e.printStackTrace(); }
...

```

7. データがファイルから読み取られたならば、セッションを閉じます。

```

// close session.
    if(cc != null) {
cc.close();
    }
}
}

```

Solaris WBEM ロギングプロパティの設定

`Solaris_LogServiceProperties` クラスのインスタンスを作成して、そのインスタンスのプロパティ値を設定すると、アプリケーションやプロバイダがログを処理する方法を制御できます。

▼ Solaris WBEM ロギングプロパティを設定する

以下のコード例では、ロギングプロパティの設定方法を示します。プロパティは、`/var/sadm/lib/wbem/WbemServices.properties` ファイルに格納されます。

```

public class SetProps {
    public static void main(String args[]) throws CIMException {
if ( args.length != 3) {
    System.out.println("Usage: SetProps host username password");
    System.exit(1);

```

```

}
CIMClient cc = null;
try {
    CIMNameSpace cns = new CIMNameSpace(args[0]);
    cc = new CIMClient(cns, args[1], args[2]);
    CIMObjectPath logpropcop = new
    CIMObjectPath("Solaris_LogServiceProperties");
    Enumeration e = cc.enumInstances(logpropcop, true);
    for (; e.hasMoreElements(); ) {
        CIMObjectPath op = (CIMObjectPath)e.nextElement();
        CIMInstance ci = cc.getInstance(op);
        ci.setProperty("Directory", new CIMValue("/tmp/bar1/"));
        ci.setProperty("FileSize", new CIMValue("10"));
        ci.setProperty("NumFiles", new CIMValue("2"));
        ci.setProperty("SyslogSwitch", new CIMValue("off"));
        cc.setInstance(logpropcop,ci);
    }
}
catch (Exception e) {
    System.out.println("Exception: "+e);
    e.printStackTrace();
}
// close session.
if(cc != null) {
    cc.close();
}
}

```

Solaris WBEM ログビューア

Solaris Management Console (SMC) ログビューアは、記録されたデータ表示用のグラフィカルユーザーインターフェースを提供するアプリケーションであり、このログビューアにログ記録の詳細をすべて表示することができます。SMC についての詳細は、マニュアルページ `smc(1M)` を参照してください。

ログ記録を作成した後、SMC を起動してから SMC ログビューアを起動します。

▼ SMC と Solaris ログビューアを起動する

1. 以下のコマンドを入力して、SMC 起動コマンドの場所に移動します。

```
# cd /usr/sbin
```

2. 以下のコマンドを入力して SMC を起動します。

```
# smc
```

3. 「ナビゲーション」パネルの「このコンピュータ」をダブルクリック (または、「このコンピュータ」の横にある展開・縮小アイコンをクリック) して、「このコンピュータ」の下位のツリーを展開します。「システムステータス」をダブルクリックすると、「ログビューア」アイコンが表示されます。
4. 「ログビューア」アイコンをクリックして、ログビューアを起動します。

第3章

プロセスインジケーションの使用

この章では、CIM プロセスインジケーションの概要、CIM プロセスインジケーションをイベント発生時の通信に使用する方法、および CIM プロセスインジケーションの受信をクライアントで申請できるようにするクラスについて説明します。この章は、以下のトピックで構成されています。

- 43 ページの「CIM イベントモデル」
- 45 ページの「インジケーションの生成方法」
- 45 ページの「サブスクリプションの作成方法」
- 46 ページの「CIM リスナーの追加」
- 47 ページの「イベントフィルタの作成」
- 49 ページの「イベントハンドラの作成」
- 51 ページの「イベントフィルタとイベントハンドラのバインド」

プロセスインジケーションクラスについての詳細は、第4章「WDR のクラス、ドメイン、関連、および指示」を参照してください。

注 – CIM イベントモデルについての詳細は、
<http://www.dmtf.org/education/whitepapers.php> から入手可能な
Distributed Management Task Force 白書を参照してください。

CIM イベントモデル

参考 – CIM Event API は、
`/usr/sadm/lib/wbem/doc/javax/wbem/client/CIMEvent.html` にあります。

イベントとは、実世界でのできごとのことです。プロセスインジケーションとは、イベントが発生した結果として作成されるオブジェクトです。プロセスインジケーションはイベントの通知であり、イベントとプロセスインジケーションを区別することが重要です。CIM ではイベントが発行されるのではなく、プロセスインジケーションが発行されます。

プロセスインジケーションは、ゼロ個以上のトリガー (イベントにより生成されたデータ変更の記述) と関連を持っているクラスのサブタイプであり、Indication クラスのインスタンスを作成できるのがこのトリガーです。WBEM の実装には、トリガーを表す明示的に定義されたオブジェクトはありません。トリガーは、システムの基本オブジェクトでの操作 (クラス、インスタンス、およびネームスペースでの create、delete、および modify)、または管理対象環境でのイベントのいずれかによって、暗黙的に定義されます。イベントが発生すると、WBEM プロバイダでは、システムで何かが起こったことを示すプロセスインジケーションを生成します。

たとえば Service クラスでは、サービスが停止してトリガーが開始されると、サービス停止の通知としての役割を果たすプロセスインジケーションが発行されます。

Solaris WBEM Services スキーマの関連 CIM クラスは、
</usr/sadm/lib/wbem/doc/mofhtml/index.html> をご覧ください。クラスは以下のように構成されています。

- ルートクラス : CIM_Indication
 - スーパークラス : CIM_ClassIndication
 - サブクラス : CIM_ClassCreation
 - CIM_ClassDeletion
 - CIM_ClassModification
 - スーパークラス : CIM_InstIndication
 - サブクラス : CIM_InstCreation
 - CIM_InstDeletion
 - CIM_InstMethodRecall
 - CIM_InstRead
 - スーパークラス : CIM_ProcessIndication

CIM_ProcessIndication スーパークラスは、88 ページの「WDR インジケーションクラス階層図」の最上位に位置しています。

インジケーションの生成方法

CIM イベントは、ライフサイクルイベントとプロセスイベントのいずれかに分類できます。ライフサイクルイベントは組み込み (固有の) CIM イベントで、クラスやクラスのインスタンスの作成、変更、削除といった、データへの変更に応答して発生します。プロセスイベントは、ライフサイクルイベントでは表されない、ユーザーが定義した (固有でない) イベントです。

管理者は `cimom.properties` ファイルのプロパティを編集することにより、CIM Object Manager のイベントポーリング間隔とデフォルトのポーリング動作を変更できます。`cimom.properties` ファイルの編集方法については、『Solaris WBEM Services の管理』(Part No. 806-7119-10) を参照してください。

CIM Object Manager からの要求に応じてインジケーションを生成するのは、イベントプロバイダです。CIM Object Manager では、サブスクリプション要求を分析し、EventProvider インタフェースを介して該当するプロバイダに連絡し、適切なインジケーションを生成するよう要求します。プロバイダでインジケーションが生成されると、CIM Object Manager は、CIM_IndicationHandler インスタンスで指定されている送信先にそのインジケーションを送信します。このインスタンスは、サブスクリバ (申請元) によって作成されます。

サブスクリプションの作成方法

クライアントアプリケーションでは、CIM イベントが通知されるように申請できます。サブスクリプションは、1 つまたは複数のインジケーションストリームに対する宣言です。

CIM イベントが通知されるように申請するアプリケーションでは、以下について記述します。

- 対象とするイベント
- イベントの発生時に CIM Object Manager がとるべきアクション

イベントの発生は、CIM_Indication クラスのいずれかのサブクラスのインスタンスとして表されます。インジケーションが生成されるのは、クライアントがイベントについて申請をしている場合のみです。

サブスクリプションを作成するには、CIMListener インタフェースのインスタンスを指定して、以下の CIM_Indication クラスのサブクラスのインスタンスを作成します。

CIM_IndicationFilter — インジケーションを生成する基準と、そのインジケーションで返されるようにするデータを定義します。

CIM_IndicationHandler — インジケーションの処理と操作方法を示します。インジケーションを配信する宛先とプロトコルが含まれる場合もあります。

CIM_IndicationSubscription — イベントフィルタとイベントハンドラをバインドして関連付けます。

アプリケーションでは、1つまたは複数のイベントハンドラを使って、1つまたは複数のイベントフィルタを作成することができます。アプリケーションがイベントサブスクリプションを作成するまで、イベントインジケーションは配信されません。

CIM リスナーの追加

CIM イベントのインジケーションに関する登録を行うときは、CIMListener インタフェースのインスタンスを追加します。CIM Object Manager では、クライアントサブスクリプションが作成されたときに、イベントフィルタで指定された CIM イベントのインジケーションを生成します。

CIMListener インタフェースでは、引数 CIMEvent をとる indicationOccured メソッドを実装している必要があります。このメソッドは、インジケーションが配信できるようになったときに呼び出されます。

▼ CIM リスナーを追加する

以下のようなコードを使用して、CIM リスナーを追加します。

```
// Connect to the CIM Object Manager
cc = new CIMClient();
// Register the CIM Listener
cc.addCIMListener(new CIMListener() {
    public void indicationOccured(CIMEvent e) {
    }
});
```

イベントフィルタの作成

イベントフィルタには、配信されるイベントのタイプと、配信される条件が記述されます。アプリケーションでは、CIM_IndicationFilter クラスのインスタンスを作成し、そのプロパティの値を定義することにより、イベントフィルタを作成します。イベントフィルタは、ネームスペースに含まれます。各イベントフィルタは、そのフィルタと同じネームスペースに含まれているイベントに対してのみ動作します。

CIM_IndicationFilter クラスには文字列プロパティがあります。アプリケーションは、このプロパティで、一意にフィルタを特定するよう設定し、照会文字列を指定し、照会文字列の構文解析に使用する照会言語を設定することができます。現在、照会言語でサポートされているのは、WBEM Query Language のみです。

表 3-1 CIM_IndicationFilter クラスのプロパティ

プロパティ	説明	必須・オプション
SystemCreationClassName	フィルタを作成するクラスが存在する、またはそのクラスが適用されるシステムの名前。	オプション。この基本プロパティのデフォルトは CIMSystem.CreationClassName です。
SystemName	フィルタが存在する、またはそのフィルタが適用されるシステムの名前。	オプション。この基本プロパティのデフォルトは、CIM Object Manager が実行されているシステムの名前です。
CreationClassName	フィルタの作成に使用されたクラスまたはサブクラスの名前。	オプション。この基本プロパティのデフォルトとして、CIM Object Manager により CIM_IndicationFilter が割り当てられます。
Name	フィルタの一意の名前。	オプション。CIM Object Manager により一意の名前が割り当てられます。

表 3-1 CIM_IndicationFilter クラスのプロパティ (続き)

プロパティ	説明	必須・オプション
SourceNamespace	CIM インジケーションが生成されるローカルの名スペースへのパス。	オプション。デフォルトは NULL です。
Query	インジケーションが生成される条件を定義する照会式。現在、Level 1 WBEM Query Language 式のみサポートされています。WQL 照会式の作成方法については、『Sun WBEM SDK 開発ガイド』(Part No. 816-0093-10)の「照会」を参照してください。	必須。
QueryLanguage	照会式を記述する言語。	必須。デフォルトは WQL (WBEM Query Language) です。

▼ イベントフィルタを作成する

1. 以下のコードを使用して、CIM_IndicationFilter クラスのインスタンスを作成します。

```
CIMClass cimfilter = cc.getClass
    (new CIMObjectPath(``CIM_IndicationFilter``), true, true,
    true, null);CIMInstance ci = cimfilter.newInstance();
```

2. 以下のコードを使用して、イベントフィルタの名前を指定します。

```
Name = ``filter_all_new_solarisdiskdrives``;
```

3. 以下のコードを使用して、返されるイベントインジケーションを特定する WQL 文字列を作成します。

```
String filterString = ``SELECT *
    FROM CIM_InstCreation WHERE sourceInstance is
    ISA Solaris_DiskDrive``
```

4. 以下のコードを使用して、cimfilter インスタンスのプロパティ値を設定して、フィルタ名、CIMイベントを選択するフィルタ文字列、および照会文字列の構文解析に使用する照会言語を指定します。

注 - 現在、照会文字列の構文解析に使用できるのは WBEM Query Language のみです。

```
ci.setProperty(`Name`, new  
    CIMValue("filter_all_new_solarisdiskdrives&rdquo;));  
ci.setProperty("Query", new CIMValue(filterString));  
ci.setProperty("QueryLanguage", new CIMValue("WQL");)
```

5. 以下のコードを使用して、cimfilter インスタンスから 1 つのインスタンスを作成し、そのインスタンスを CIM Object Manager Repository に格納します。

```
CIMObjectPath filter = cc.createInstance(new CIMObjectPath(),  
ci);
```

イベントハンドラの作成

Solaris Event MOF では、RMI プロトコルを使ってクライアントアプリケーションへの CIM イベントのインジケーションの配信を処理する

Solaris_JAVAXRMIDelivery クラスを作成することにより、CIM_IndicationHandler クラスを拡張しています。RMI クライアントは、Solaris_JAVAXRMIDelivery クラスをインスタンス化して、RMI による配信場所を設定する必要があります。クライアントがイベントの受信に使用できるのは RMI のみで、HTTP はサポートされていません。

アプリケーションでは、CIM_IndicationHandler クラスのプロパティを設定し、ハンドラに一意の名前を付け、その所有者の UID を特定します。

表 3-2 CIM_IndicationHandler クラスのプロパティ

プロパティ	説明	必須・オプション
SystemCreationClassName	ハンドラを作成するクラスが存在する、またはそのクラスが適用されるシステムの名前。	オプション。CIM Object Manager により設定されます。
SystemName	ハンドラが存在する、またはそのハンドラが適用されるシステムの名前。	オプション。この基本プロパティのデフォルトは、CIM Object Manager が実行されているシステムの名前です。
CreationClassName	ハンドラの作成に使用されるクラスまたはサブクラスの名前。	オプション。この基本プロパティのデフォルトとして、CIM Object Manager により適切なクラスが割り当てられています。
Name	ハンドラの一意的名前。	必須。クライアントアプリケーションで一意的名前を割り当てる必要があります。
Owner	このハンドラを作成した、または保持しているエンティティの名前。プロバイダがこの値をチェックし、ハンドラでのインジケーションの受け取りを承認するかどうかを判定します。	オプション。デフォルトの値は、このインスタンスを作成しているユーザーの Solaris ユーザー名です。

▼ CIM イベントハンドラを作成する

CIM イベントハンドラを作成するときは、以下のコードを使用します。

```
// Create an instance of the Solaris_RMIDelivery class.
CIMClass rmidelivery = cc.getClass(new CIMObjectPath
    ('`Solaris_RMIDelivery`'), false, true, true, null);

CIMInstance ci = rmidelivery.newInstance();

//Create a new instance (delivery) from
//the rmidelivery instance.
CIMObjectPath delivery = cc.createInstance(new
CIMObjectPath(), ci);
```

イベントフィルタとイベントハンドラの バインド

アプリケーションは、CIM_IndicationSubscription クラスのインスタンスを作成することにより、イベントフィルタとイベントハンドラをバインドします。CIM_IndicationSubscription が作成されると、イベントフィルタによって指定されたイベントのインジケーションが配信されます。

▼ イベントフィルタとイベントハンドラをバインドする

以下のコード例では、サブスクリプション (filterdelivery) を作成し、47 ページの「イベントフィルタの作成」で作成した filter オブジェクトに filter プロパティを定義し、51 ページの「CIM イベントハンドラを作成する」で作成した delivery オブジェクトに handler プロパティを定義しています。

```
CIMClass filterdelivery = cc.getClass(new
    CIMObjectPath('`CIM_IndicationSubscription`'),
    true, true, true, null);
ci = filterdelivery.newInstance();

//Create a property called "filter" that refers to the filter
//instance.
ci.setProperty("filter", new CIMValue(filter));
```

```
//Create a property called handler that refers to the delivery
//instance.
ci.setProperty("handler", new CIMValue(delivery));

CIMObjectPath indsub = cc.createInstance(new CIMObjectPath(),
ci);
.
```

第4章

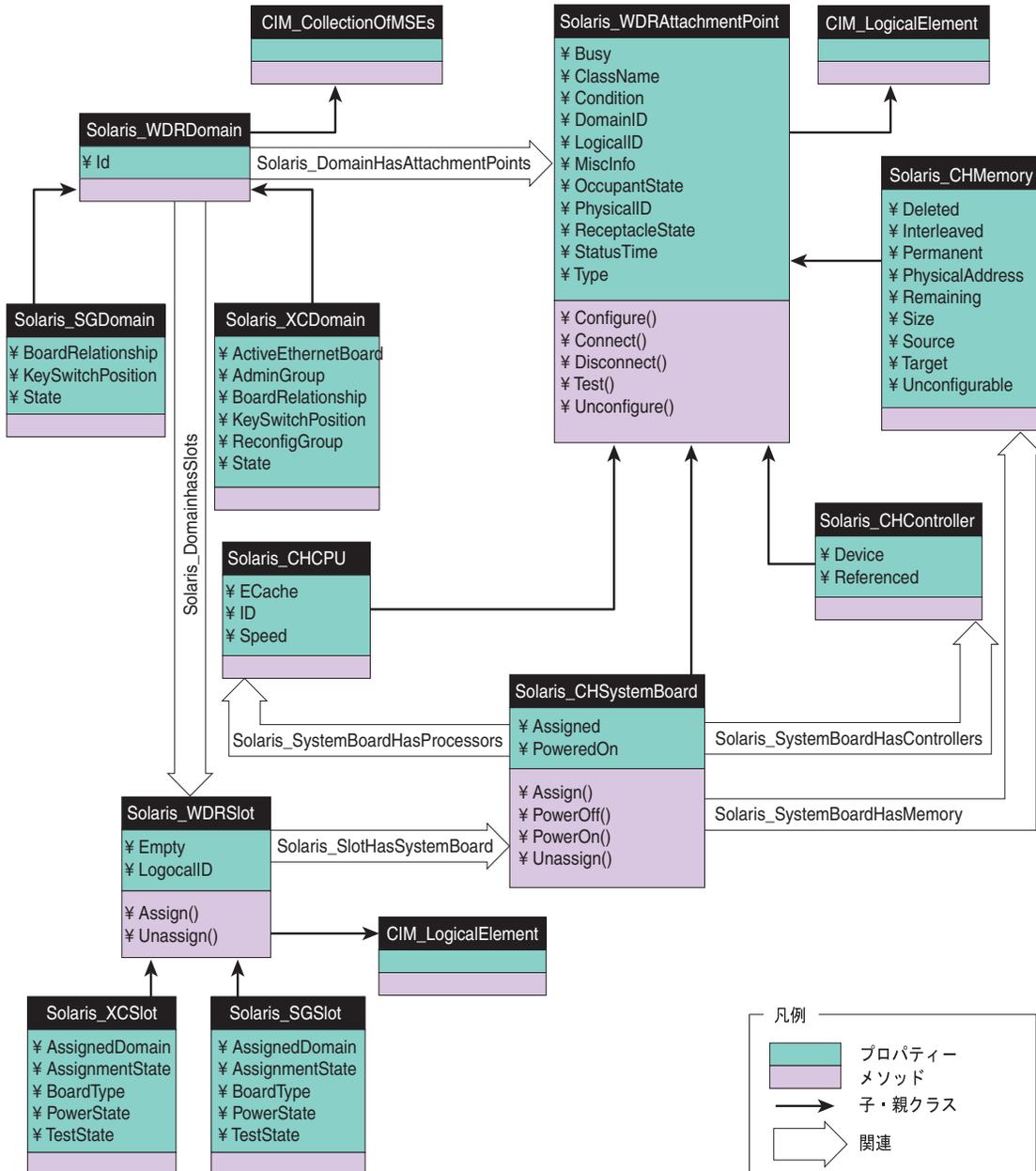
WDR のクラス、ドメイン、関連、 およびインジケーション

この章では、以下の図に示す WDR CIM クラス階層を構成するクラス、ドメイン、関連、およびインジケーションについて説明します。

第4章は、以下の5つの項目について説明します。

- 55 ページの「CIM 接続点クラス」
- 68 ページの「CIM スロットクラス」
- 76 ページの「CIM Solaris_WDRDomain クラス」
- 82 ページの「WDR スキーマ の関連と集約」
- 87 ページの「CIM プロセスインジケーションクラス」

WDR CIM クラス階層図



CIM 接続点クラス

接続点クラスでは、Sun Fire 15K、12K、6800、4810、4800、または 3800 システムの接続点を表す論理要素を提供します。接続点とは、Sun Fire 15K、12K、6800、4810、4800、または 3800 システムの物理的な位置へのインタフェースです。この接続点では、WDR を使用して、Solaris オペレーティング環境が実行されているドメインのシステムボード、CPU、およびメモリーモジュールを構成することができます。接続点は、受容体と占有装置からなります。占有装置を受容体に挿入または受容体から取り外すと、接続点の状態が変更されます。

注 – 接続点についての詳細は、`cfgadm(1M)` マニュアルページ (Sun Fire モデル全般) と `cfgadm_sbd(1M)` マニュアルページ (Sun Fire 15K および 12K 専用) を参照してください。

接続点クラスは、WDR を使用できる Sun Fire 15K、12K、6800、4810、4800、または 3800 システムの物理的な位置を表すという点においては、スロットクラスと同じです (68 ページの「CIM スロットクラス」を参照)。しかしスロットクラスでは、論理的な要素についてはシステムボードと入出力ボードのみを表し、CPU、メモリー、および入出力コントローラについては表しません。スロットは、範囲がボードに限定されている接続点の 1 つのタイプといえます。

CIM Solaris_WDRAttachmentPoint クラス

クラス階層での位置

```
CIM_LogicalElement
|
+--Solaris_WDRAttachmentPoint
```

説明

コアの構成管理情報を表します。(詳細は、`cfgadm(1M)` マニュアルページを参照してください。) この情報は、ドメイン上で `libcfgadm` ライブラリを使用して収集されます。

直系の既知のサブクラス

CIM Solaris_CHCPU クラス、CIM Solaris_CHSystemBoard クラス、CIM Solaris_CHController クラス、および CIM Solaris_CHMemory クラス

CIM Solaris_WDRAttachmentPoint クラスのプロパティ

注 - 接続点についての詳細は、`cfgadm(1M)` マニュアルページ (Sun Fire システム全般) と `cfgadm_sbd(1M)` マニュアルページ (Sun Fire 15K および 12K システム専用) を参照してください。

表 4-1 CIM Solaris_WDRAttachmentPoint のプロパティ

プロパティ	データ型	説明
ClassName	string	接続点のクラス。たとえば "sbd" はシステムボードを表します。
Busy	uint32	接続点が、現在状態の切り替え中であるかどうかを示します。
Condition	uint32	接続点の条件。値: Unknown、OK、Failing、Failed、および Unusable
LogicalID	string	接続点の論理的な識別子。
PhysicalID	string	接続点の物理的な識別子。例: /devices/pseudo/dr@0::SB6
DomainID	uint32	この接続点が割り当てられている、または使用可能であるドメイン。Sun Fire 15K システムのドメインには 0 ~ 17 の番号が割り当てられます。Sun Fire 12K システムのドメインには 0 ~ 8 の番号が割り当てられます。Sun Fire 3800、4800、および 4810 システムのドメインには 0 と 1 の番号が割り当てられます (最大 2 つのドメイン)。Sun Fire 6800 システムのドメインには 0 ~ 3 の番号が割り当てられます (最大 4 つのドメイン)。
OccupantState	uint32	接続点の占有装置の状態。値: None、Configured、および Unconfigured
ReceptacleState	uint32	接続点の受容体の状態。値: None、Empty、Disconnected、および Connected
Type	string	接続点のタイプ。cpun、pcin、または memn のいずれか (n はコンポーネントの番号)。

表 4-1 CIM Solaris_WDRAttachmentPoint のプロパティ (続き)

MiscInfo	string	<p>ドライバによって設定されるドライバ固有情報。名前と値の組み合わせのリスト。Type プロパティの値によって決まります。</p> <p>たとえば Type プロパティが <code>cpu1</code> の場合、MiscInfo プロパティの内容は、プロセッサ ID、プロセッサ速度、および Ecache メモリーサイズ (MB) の情報から生成されます。</p>
StatusTime	datetime	<p>接続点の状態が最後に変更された日付と時刻。次の書式で表されません。</p> <p>yyyyymmddhhmmss.mmmmmmsutc</p> <p>各文字は以下を表します。 yyyy は年、 mm は月、 dd は日付、 hh は時間、 mm は分、 ss は秒</p>

CIM Solaris_WDRAttachmentPoint クラスのメソッド

Solaris_WDRAttachmentPoint メソッドには、動作中のドメインへの接続点リソースの追加、このドメインからの接続点リソースの削除、接続点の状態のテストを行うときに使用する 5 つのメソッドがあります。

注 - Solaris_WDRAttachmentPoint クラスのメソッドの詳細については、`cfgadm(1M)`、`cfgadm_sbd(1M)`、および `rcfgadm(1M)` の各マニュアルページを参照してください。

メソッドのリターンコード

すべての Solaris_WDRAttachmentPoint メソッドは、メソッドが正常に実行されたかどうかを示す `sint32` 値を 1 つ返します。戻り値が 0 (ゼロ) のときは正常に実行されたことを示し、ゼロ以外の値のときは以下のようなエラーが発生したことを示します。

0 = 構成操作は成功しました

1 = 構成操作は取り消されました

- 2 = 構成管理はサポートされていません
- 3 = 構成操作はサポートされていません
- 4 = 特権が不十分です
- 5 = コンポーネントシステムが使用中のため、再試行してください
- 6 = システムが使用中のため、再試行してください
- 7 = データエラーです
- 8 = ライブラリエラーです
- 9 = ライブラリが見つかりませんでした
- 10 = 条件が不十分です
- 11 = 無効な構成操作です
- 12 = ハードウェア固有の障害です
- 13 = 接続点が見つかりませんでした
- 14 = 指定された属性を持つ接続点が見つかりませんでした

注 - クライアントからのメソッドの呼び出し方法についての詳細は、WBEM SDK の *Sun WBEM API Specification* を参照してください。これは、`/usr/sadm/lib/wbem/doc/index.html` にあります。`invokeMethod()` を使用する前に、表示されている正確な順序で `inParams` ベクトルをすべての [IN] (入力) パラメータを使って生成し、さらに `outParams` ベクトルを空の文字列で生成します。`invokeMethod()` が戻ると、`outParams` ベクトルには、対応する DR 操作によって生成されたエラー文字列があればその文字列が含まれ、生成されたエラー文字列がない場合は空の文字列が含まれます。

CIM Solaris_WDRAttachmentPoint メソッドの説明

表 4-2 CIM Solaris_WDRAttachmentPoint メソッド

名前	説明
Configure	<p>Solaris ドメインへの接続点を構成します。</p> <p>パラメタ :</p> <ul style="list-style-type: none"> • boolean — force [IN] 強制的に Configure 操作を行います。行わない場合は、接続点の条件、またはその他のハードウェアに依存した事項によって、失敗する場合があります。ハードウェア特有の安全性および完全性のチェックを行うと、force オプションを実行しても何も処理されません。 • string — hardwareOpts [IN] 指定したオプションは cfgadm というハードウェア特有のプラグインに渡されます。現在 WDR は cfgadm_sbd プラグインと間接的に接続して機能します。-o nopoweroff を指定すると、disconnect 機能によりボードの電源が投入されたままになります。-o unassign を指定すると、disconnect 機能によりドメインからボードの割り当てが解除されます。 ドメインからボードの割り当てを解除すると、そのボードを別のドメインに割り当てることができます。ただし、ボードを別のドメインに割り当てた場合、割り当て解除されたドメインではそのボードを使用できません。 • uint32 — retries [IN] ドメインで、動的再構成 (DR) 要求が再試行される回数を指定します。デフォルトはゼロです。 • uint32 — retryDelay [IN] 再試行を行う際の時間間隔 (秒) を指定します。このオプションは単独では使用できず、必ず -r retry_count オプションとともに指定します。デフォルト値は 0 で、これは DR 要求がすぐに再試行されることを意味します。 • string — error [OUT] 対応する DR コマンドによって返されたエラー文字列があれば、この文字列が含まれます。DR コマンドによって何もエラー文字列が返されない場合は、空の文字列が含まれます。
Unconfigure	<p>現在接続点が構成されている Solaris ドメインから、接続点のリソースを削除します。</p> <p>このメソッドで使用されるパラメタは、上記の Configure メソッドのパラメタと同じです。</p>
Connect	<p>受容体の状態を接続状態に変更します。</p> <p>このメソッドで使用されるパラメタは、上記の Configure メソッドのパラメタと同じです。</p>

表 4-2 CIM Solaris_WDRAttachmentPoint メソッド (続き)

Disconnect	<p>受容体に装着されている占有装置との間の正常な通信を使用不可にします。</p> <p>このメソッドで使用されるパラメータは、上記の Configure メソッドのパラメータと同じです。</p>
Test	<p>ボードのテストが完了していても、POST を使用したボードのテストを明示的に要求します。</p> <p>注: Connect メソッドを呼び出すと POST を使用してボードのテストが行われるため、Test メソッドを呼び出す必要はありません。</p> <p>パラメータ：</p> <ul style="list-style-type: none"> • boolean — verbose [IN] DR コマンド出力はクライアントでは使用できないため、このメソッドの機能にはほとんど影響しません。 • string — hardwareOpts [IN] 上記の Configure メソッドで説明した hardwareOpts パラメータと同様に使用します。 • string — error [OUT] 上記の Configure メソッドで説明した error パラメータと同様に使用します。

CIM Solaris_CHSystemBoard クラス

クラス階層での位置

```

CIM_LogicalElement
|
+--Solaris_WDRAttachmentPoint
|
+--Solaris_CHSystemBoard

```

説明

Dynamic Reconfiguration モデル 2.0 の機能をサポートしている UltraSPARC-III 世代のシステムボードをモデル化する論理要素を表します。

54 ページの「WDR CIM クラス階層図」に示すように、CIM Solaris_CHSystemBoard クラスは、Solaris_CHMemory、Solaris_CHController、Solaris_WDRSlot、および Solaris_CHCPU の各 CIM クラスと関連関係があります。

直系の既知のサブクラス

なし

CIM Solaris_CHSystemBoard クラスのプロパティ

表 4-3 CIM Solaris_CHSystemBoard のプロパティ

名前	データ型	説明
Assigned	boolean	ボードが Solaris ドメインに割り当てられていることを示します。
PoweredOn	boolean	ボードの電源が入っていることを示します。

CIM Solaris_CHSystemBoard クラスのメソッド

Solaris_CHSystemBoard メソッドには、システムボードの電源投入と電源切断、現在動作しているドメインへのボードの割り当てとそのドメインからの割り当ての解除に使用する 4 つのメソッドがあります。

注 - Solaris_CHSystemBoard クラスのメソッドについての詳細は、`cfgadm(1M)`、`cfgadm_sbd(1M)`、および `rcfgadm(1M)` の各マニュアルページを参照してください。

メソッドのリターンコード

すべての Solaris_CHSystemBoard メソッドは、メソッドが正常に実行されたかどうかを示す `sint32` 値を 1 つ返します。戻り値が 0 (ゼロ) のときは正常に実行されたことを示し、ゼロ以外の値のときは以下のようなエラーが発生したことを示します。

0 = 構成操作は成功しました

1 = 構成操作は取り消されました

2 = 構成管理はサポートされていません

3 = 構成操作はサポートされていません

4 = 特権が不十分です

5 = コンポーネントシステムが使用中のため、再試行してください

- 6 = システムが使用中のため、再試行してください
- 7 = データエラーです
- 8 = ライブラリエラーです
- 9 = ライブラリが見つかりませんでした
- 10 = 条件が不十分です
- 11 = 無効な構成操作です
- 12 = ハードウェア固有の障害です
- 13 = 接続点が見つかりませんでした
- 14 = 指定された属性を持つ接続点が見つかりませんでした

注 - クライアントからのメソッドの呼び出し方法についての詳細は、WBEM SDK の *Sun WBEM API Specification* を参照してください。これは、`/usr/sadm/lib/wbem/doc/index.html` にあります。`invokeMethod()` を使用する前に、表示されている正確な順序で `inParams` ベクトルをすべての [IN] (入力) パラメータを使って生成し、さらに `outParams` ベクトルを空の文字列で生成します。`invokeMethod()` が戻ると、`outParams` ベクトルには、対応する DR 操作によって生成されたエラー文字列があればその文字列が含まれ、生成されたエラー文字列がない場合は空の文字列が含まれます。

CIM Solaris_CH_SystemBoard メソッドの説明

表 4-4 CIM Solaris_CH_SystemBoard メソッド

名前	説明
Assign	<p>指定された Solaris ドメインにボードを割り当てます。</p> <p>パラメタ :</p> <ul style="list-style-type: none"> • boolean — force [IN] 強制的に Assign 操作を行います。行わない場合は、接続点の条件、またはその他のハードウェアに依存した事項によって、失敗する場合があります。ハードウェア特有の安全性および完全性のチェックを行うと、force オプションを実行しても何も処理されません。 • string — hardwareOpts [IN] 指定したオプションは cfgadm というハードウェア特有のプラグインに渡されます。現在 WDR は cfgadm_sbd プラグインと間接的に接続して機能します。 ドメインからボードの割り当てを解除すると、そのボードを別のドメインに割り当てることができます。ただし、ボードを別のドメインに割り当てた場合、割り当て解除されたドメインではそのボードを使用できません。 • string — error [OUT] 対応する DR コマンドによって返されたエラー文字列があれば、この文字列が含まれます。DR コマンドによって何もエラー文字列が返されない場合は、空の文字列が含まれます。
PowerOn	<p>ボードの電源を投入します。</p> <p>パラメタ :</p> <p>このメソッドで使用されるパラメタは、上記の Assign メソッドのパラメタと同じです。</p>
PowerOff	<p>ボードの電源を切断します。</p> <p>パラメタ :</p> <p>このメソッドで使用されるパラメタは、上記の Assign メソッドのパラメタと同じです。</p>
Unassign	<p>現在割り当てられているドメインから、ボードの割り当てを解除します。</p> <p>パラメタ :</p> <p>このメソッドで使用されるパラメタは、上記の Assign メソッドのパラメタと同じです。</p>

CIM Solaris_CHCPU クラス

クラス階層での位置

```
CIM_LogicalElement
|
+---Solaris_WDRAttachmentPoint
    |
    +---Solaris_CHCPU
```

説明

システムボード上のプロセッサを表す論理要素。UltraSPARC-III 世代のシステムボード上では、1つのシステムボードに4個のプロセッサを搭載可能です。プロセッサは物理的にシステムボード上のCPUソケットに取り付けられており、しかも構成および構成解除などのDR操作を接続点で実行できることから、CIM Solaris_CHCPUクラスはCIM Solaris_WDRAttachmentPointクラスから派生しています。

54 ページの「WDR CIM クラス階層図」に示すように、CIM Solaris_CHCPUクラスはCIM Solaris_CHSystemBoardクラスと集約関係があります。

直系の既知のサブクラス

なし

CIM Solaris_CHCPU クラスのプロパティ

表 4-5 Solaris_CHCPU のプロパティ

名前	データ型	説明
ID	uint32	プロセッサの一意の識別子。
Speed	uint32	プロセッサのクロック数 (MHz)。
ECache	uint32	ECache メモリーのサイズ (MB)。

CIM Solaris_CHCPU クラスのメソッド

なし

CIM Solaris_CHMemory クラス

クラス階層での位置

```
CIM_LogicalElement
|
+--Solaris_WDRAttachmentPoint
|
+--Solaris_CHMemory
```

説明

システムボードのメモリー情報を表す論理要素。Solaris_CHSystemBoard と Solaris_CHMemory の CIM クラスのインスタンスには一対一の関係があります。さらに、メモリーはシステムボード上の接続点であることから、CIM Solaris_CHMemory クラスは CIM Solaris_WDRAttachmentPoint クラスから派生しています。

直系の既知のサブクラス

なし

CIM Solaris_CHMemory のプロパティ

表 4-6 CIM Solaris_CHMemory のプロパティ

名前	データ型	説明
Deleted	uint32	メモリードレインの実行中、Deleted プロパティはすでに削除されたメモリー全部を格納します。それ以外の場合は、Deleted プロパティは NULL です。
Interleaved	boolean	ボードが他のボードとのインタリーブに加わっている場合は True です。
Permanent	uint32	ページング不可のメモリーページ数を、ボードのメモリーに格納します (KB)。
PhysicalAddress	uint64	ボード上のメモリーの基底物理アドレス。
Remaining	uint32	メモリードレインの実行中に、Remaining プロパティは、ドレインする必要がある残りのメモリー量を格納します (MB)。それ以外の場合は、Remaining プロパティは NULL です。
Size	uint32	ボード上のメモリーの容量 (MB)。
Source	string	コピー & 名前変更のソースである接続点の名前。コピー & 名前変更操作が行われていない場合は、Source プロパティは NULL です。
Target	string	コピー & 名前変更の対象である接続点の名前。コピー & 名前変更操作が行われていない場合は、Target プロパティは NULL です。
Unconfigurable	boolean	このメモリーの構成解除を許可しないようにオペレーティングシステムが構成されている場合は True です。

CIM Solaris_CHMemory クラスのメソッド

なし

CIM Solaris_CHController クラス

クラス階層での位置

```
CIM_LogicalElement
|
+--Solaris_WDRAttachmentPoint
|
+--Solaris_CHController
```

説明

入出力ボード上で入出力コントローラの接続点をモデル化する論理 CIM 要素。

直系の既知のサブクラス

なし

CIM Solaris_CHController クラスのプロパティ

表 4-7 Solaris_CHController のプロパティ

名前	データ型	説明
Device	string	/devices バスの入出力コンポーネントの物理バス。
Referenced	boolean	入出力コンポーネントが参照されている場合は True です。

CIM Solaris_CHController クラスのメソッド

なし

CIM スロットクラス

CIM スロットクラスは、Sun Fire 15K、12K、3800、4800、4810、および 6800 システム上のシステムボードスロットをモデル化します。スロットにボードが装着されていなくても構いません。接続点と同様に、スロットをドメインに割り当てたり、割り当て解除することができます。ただし、接続点と異なるのは、スロットはいずれのドメインからも独立して存在することができ、常に存在する点です。

注 - 名前に “XC” が含まれているクラスは、Sun Fire™ 15K および 12K システム用です。名前に “SG” が含まれているクラスは、Sun Fire 6800、4810、4800、および 3800 システム用です。

CIM Solaris_WDRSlot クラス

クラス階層での位置

```
CIM_LogicalElement
|
+--Solaris_WDRSlot
```

抽象的な CIM Solaris_WDRSlot クラスは、プラットフォームに依存しないスロットをモデル化します。

説明

Sun Fire 15K、12K、6800、4810、4800、または 3800 シャーシのスロットをモデル化する論理 CIM 要素にスーパークラスを提供する論理 CIM 要素。スロットには、システムボードと入出力ボードのどちらも装着できます。

54 ページの「WDR CIM クラス階層図」に示すように、CIM Solaris_WDRSlot クラスは、Solaris_CHSystemBoard および Solaris_WDRDomain の各 CIM クラスと関連関係があります。

直系の既知のサブクラス

CIM Solaris_XCSlot クラスおよび CIM Solaris_SGSlot クラス

CIM Solaris_WDRSlot のプロパティ

表 4-8 CIM Solaris_WDRSlot のプロパティ

名前	データ型	説明
LogicalID	string	<p>スロットの論理名。</p> <p>Sun Fire 15K システムには 18 個の拡張スロットがあり、その各スロットにシステムボードと入出力ボードをそれぞれ 1 つ装着できます。システムボードのスロットは SB0、SB1、... SB17 と表記され、入出力ボードのスロットは IO0、IO1、... IO17 と表記されます。</p> <p>Sun Fire 12K システムには 9 個の拡張スロットがあり、その各スロットにシステムボードと入出力ボードをそれぞれ 1 つ装着できます。システムボードのスロットは SB0、SB1、... SB8 と表記され、入出力ボードのスロットは IO0、IO1、... IO8 と表記されます。</p> <p>Sun Fire 6800、4810、4800、または 3800 システムには、最大 6 個のシステムボードと最大 4 個の入出力ボードを装着できます。システムボードのスロットは SB0、SB1、... SB5 と表記され、入出力ボードのスロットは IB6、IB7、IB8、および IB9 と表記されます。</p>
Empty	boolean	<p>このスロットにボードが装着されているかどうかを示します。値が NULL の場合は、スロットの状態が不明であることを示します。</p> <p>Empty プロパティが True の場合には、サブクラス CIM Solaris_XCSlot クラスおよび CIM Solaris_SGSslot クラスのプロパティである、AssignmentState、BoardType、PowerState、および TestState は NULL です。</p>

CIM Solaris_WDRSlot のメソッド

Solaris_WDRSlot メソッドには、スロットの割り当てと割り当て解除に使用する 2 つのメソッドがあります。

メソッドのリターンコード

Sun Fire 15K と 12K システムでは、すべての Solaris_WDRSlot メソッドは、メソッドが正常に実行されたかどうかを示す sint32 値を 1 つ返します。戻り値が 0 (ゼロ) のときは正常に実行されたことを示し、ゼロ以外の値のときは以下のようなエラーが発生したことを示します。

- 0 = 構成操作は成功しました
- 1 = 構成操作は取り消されました
- 2 = 構成管理はサポートされていません
- 3 = 構成操作はサポートされていません
- 4 = 特権が不十分です
- 5 = コンポーネントシステムが使用中のため、再試行してください
- 6 = システムが使用中のため、再試行してください
- 7 = データエラーです
- 8 = ライブラリエラーです
- 9 = ライブラリが見つかりませんでした
- 10 = 条件が不十分です
- 11 = 無効な構成操作です
- 12 = ハードウェア固有の障害です
- 13 = 接続点が見つかりませんでした
- 14 = 指定された属性を持つ接続点が見つかりませんでした

注 - クライアントからのメソッドの呼び出し方法についての詳細は、WBEM SDK の *Sun WBEM API Specification* を参照してください。これは、
 /usr/sadm/lib/wbem/doc/index.html にあります。invokeMethod() を使用する前に、表示されている正確な順序で inParams ベクトルをすべての [IN] (入力) パラメタを使って生成し、さらに outParams ベクトルを空の文字列で生成します。invokeMethod() が戻ると、outParams ベクトルには、対応する DR 操作によって生成されたエラー文字列があればその文字列が含まれ、生成されたエラー文字列がない場合は空の文字列が含まれます。

CIM_SolarisWDRSlot メソッドの説明

表 4-9 CIM_Solaris_WDRSlot メソッド

名前	説明
Assign	<p>指定されたドメインにスロットを割り当てます。</p> <p>パラメタ:</p> <ul style="list-style-type: none"> • uint32 — domainID [IN] このスロットに割り当てるドメインを指定します。Sun Fire 15K/12K サーバーでは、18 個までのドメインを構成することができます。Sun Fire 3800、4800、4810 システムでは、1 つまたは 2 つのドメインを構成できます。Sun Fire 6800 システムでは、1 ~ 4 個のドメインを構成できます。 • string — error [OUT] 対応する DR 操作によって返されたエラー文字列があれば、この文字列が含まれます。DR 操作によって何もエラー文字列が返されない場合は、空の文字列が含まれます。
Unassign	<p>ドメインからボードの割り当てを解除します。ドメインでスロットに装着されているボードは、アクティブな (接続または構成されている) 状態ではありません。</p> <p>パラメタ:</p> <p>このメソッドで使用されるパラメタは、上記の Assign メソッドのパラメタと同じです。</p>

CIM Solaris_XCSlot クラス

クラス階層での位置

```
CIM_LogicalElement
|
+--Solaris_WDRSlot
|
+--Solaris_XCSlot
```

説明

Sun Fire 15K または 12K システム上のスロットをモデル化する論理 CIM 要素。スロットには、システムボードと入出力ボードのどちらも装着できます。

Sun Fire 15K システムには 18 個の拡張スロットがあり、その各スロットにシステムボードと入出力ボードをそれぞれ 1 つ装着できます。システムボードのスロットは SB0、SB1、... SB17 と表記され、入出力ボードのスロットは IO0 (ゼロ)、IO1、... IO17 と表記されます。

Sun Fire 12K システムには 9 個の拡張スロットがあり、その各スロットにシステムボードと入出力ボードをそれぞれ 1 つ装着できます。システムボードのスロットは SB0、SB1、... SB8 と表記され、入出力ボードのスロットは IO0 (ゼロ)、IO1、... IO8 と表記されます。

直系の既知のサブクラス

なし

CIM Solaris_XCSlot のプロパティ

表 4-10 CIM Solaris_XCSlot のプロパティ

名前	データ型	説明
AssignedDomain	sint32	このスロットの AssignmentState プロパティの値が Assigned の場合に、このスロットが割り当てられているドメイン。数値 -1 ~ 18 が ValueMap の None、A、B、C、D、E、F、G、H、I、J、K、L、M、N、O、P、Q、および R を表します。
AssignmentState	uint32	現在スロットに割り当てられている状態。値 0 ~ 3 が ValueMap の Unknown、Free、Assigned、および Active を表します。 Empty プロパティ (Solaris_WDRSlot クラスから継承される) が True の場合は、常に NULL です。
BoardType	uint32	スロットに装着されているボードのタイプ (既知の場合)。値 0 ~ 8 が ValueMap の CPU、WIB、HPCI、CPCI、MPCU、WPCI、SPCI、HPCIX、および Unknown を表します。注: Unknown は Empty と同じではありません。 Empty プロパティ (Solaris_WDRSlot クラスから継承される) が True の場合は、常に NULL です。
PowerState	uint32	ボードの電源状態。値 0 ~ 3 が ValueMap の Off、On、Unknown、または Minimal を表します。 Empty プロパティ (Solaris_WDRSlot クラスから継承される) が True の場合は、常に NULL です。
TestState	uint32	ボードのテスト状態。値 0 ~ 4 が ValueMap の Unknown、iPOST、Passed、Degraded、または Failed を表します。 Empty プロパティ (Solaris_WDRSlot クラスから継承される) が True の場合は、常に NULL です。

CIM Solaris_XCSlot のメソッド

なし

CIM Solaris_SGSlot クラス

クラス階層での位置

```
CIM_LogicalElement
|
+---Solaris_WDRSlot
|
+---Solaris_SGSlot
```

説明

Sun Fire 6800、4810、4800、または 3800 システム上のスロットをモデル化する論理 CIM 要素。

注 – Sun Fire 6800、4810、4800、または 3800 システムには、最大 6 個のシステムボードと最大 4 個の入出力ボードを装着できます。システムボードのスロットは SB0、SB1、... SB5 と表記され、入出力ボードのスロットは IB6、IB7、IB8、および IB9 と表記されます。

直系の既知のサブクラス

なし

CIM Solaris_SGSlot のプロパティ

表 4-11 CIM Solaris_SGSlot のプロパティ

名前	データ型	説明
AssignedDomain	sint32	このスロットの AssignmentState プロパティの値が Assigned の場合に、このスロットが割り当てられているドメイン。値 1 ~ 5 が ValueMap の以下の項目を表します。 <ul style="list-style-type: none">• None• A• B• C• D
AssignmentState	uint32	現在スロットに割り当てられている状態。値 1 ~ 4 が ValueMap の以下の項目を表します。 <ul style="list-style-type: none">• Unknown• Free• Assigned• Active
BoardType	uint32	スロットに装着されているボードのタイプ (既知の場合)。値 1 ~ 11 が ValueMap の以下の項目を表します。 <ul style="list-style-type: none">• Unknown• Empty• CPU• IO• CPUWIB• IOWIB• SC• L2• Fan• Power Supply• Logic Analyzer
PowerState	uint32	ボードの電源状態。値 1 ~ 4 が ValueMap の以下の項目を表します。 <ul style="list-style-type: none">• Unknown• On• Off• Failed

表 4-11 CIM Solaris_SGSlot のプロパティ (続き)

TestState	uint32	<p>ボードのテスト状態。値 1 ~ 8 が ValueMap の以下の項目を表します。</p> <ul style="list-style-type: none"> • Unknown • Not Tested • Passed • Failed • Under Test • Start Test • Degraded • Unusable
-----------	--------	--

CIM Solaris_SGSlot のメソッド

なし

CIM Solaris_WDRDomain クラス

CIM Solaris ドメインクラスは、Solaris オペレーティング環境が実行されている Sun Fire システム上のドメインを表します。

CIM Solaris_WDRDomain クラス

クラス階層での位置

```

CIM_CollectionOfMSEs
|
+--Solaris_WDRDomain
    
```

説明

CIM Solaris_WDRDomain クラスは、すべての Sun Fire システム (15K、12K、6800、4810、4800、および 3800 システム) 上のドメイン情報を表す抽象スーパークラスです。

54 ページの「WDR CIM クラス階層図」に示すように、CIM Solaris_WDRDomain クラスは、Solaris_WDRSlot クラスと関連関係があり、Solaris_WDRAttachmentPoint クラスと集約関係があります。

直系の既知の CIM サブクラス

CIM Solaris_SGDomain クラスおよび CIM Solaris_XCDomain クラス

注 - 名前に “XC” が含まれている CIM ドメインクラスは、Sun Fire™ 15K および 12K システム用です。名前に “SG” が含まれている CIM ドメインクラスは、Sun Fire 6800、4810、4800、および 3800 システム用です。

CIM Solaris_WDRDomain クラスのプロパティー

表 4-12 CIM Solaris_WDRDomain のプロパティー

名前	データ型	説明
Id	uint32	ドメインを一意に特定します。

CIM Solaris_XCDomain クラス

クラス階層での位置

```
CIM_CollectionOfMSEs
|
+---Solaris_WDRDomain
    |
    +---Solaris_XCDomain
```

説明

CIM Solaris_XCDomain クラスは CIM Solaris_WDRDomain クラスのサブクラスで、Sun Fire 15K および 12K システムのドメイン情報を表します。このクラスには、Sun Fire 15K および 12K システムに固有の情報が含まれている CIM プロパティーがいくつかあります。

直系の既知の CIM サブクラス

なし

CIM Solaris_XCDomain クラスのプロパティ

表 4-13 CIM Solaris_XCDomain のプロパティ

名前	データ型	説明
ActiveEthernetBoard	string	内部システムコントローラ (SC) ネットワークでアクティブな Ethernet 接続をホストしている入出力ボード。
AdminGroup	string	ドメイン管理者グループに割り当てられている UNIX グループ名。
BoardRelationship[]	sint32	<p>ドメイン内のボードの状態を示す値の配列 (各ボードについて 1 つの値)。配列の BitMap の個々の位置が 1 つのボードの状態を表し、ValueMap の個々の数値が以下の値のいずれかを表します。</p> <ul style="list-style-type: none">• Not Available• Available• Assigned• Active <p>配列の BitMap の 1 ~ 18 の数値が、各システムボード (SB0 ~ SB17) の状態を表します。配列の BitMap の 19 ~ 18 の数値が、各入出力ボード (IO0 ~ IO17) の状態を表します。</p>
KeyswitchPosition	uint32	<p>ドメインの状態を示します。0 ~ 5 の値が、ドメインの状態を示す ValueMap の項目を表します。</p> <ul style="list-style-type: none">• On• Standby• Off• Diag• Secure• Unknown
ReconfigGroup	string	ドメイン再構成の役割に割り当てられている UNIX グループ名。

表 4-13 CIM Solaris_XCDomain のプロパティ (続き)

State	uint32	<p>ドメインの現在の状態。ValueMap の 0 ~ 36 の各値が以下の値のいずれかを表し、ドメインの現在の状態を示します。</p> <ul style="list-style-type: none"> • Unknown • Powered Off • Keyswitch Standby • Running Domain POST • Running Board POST • Layout OBP • Loading OBP • OBP Booting • OBP Running • OBP Callback • OBP Loading Solaris • OBP Booting Solaris • OBP Domain Exited • OBP Failed • OBP in Sync Callback • OBP Exited • OBP Error Reset • OBP Domain Halt • OBP Environmental Domain Halt • OBP Booting Solaris Failed • OBP Loading Solaris Failed • OBP Debug • OS Running Solaris • OS Quiesce in Progress • OS Quiesced • OS Resume in Progress • OS Panic • OS Panic Debug • OS Panic Continue • OS Panic Dump • OS Halt • OS Panic Exit • OS Environmental Exit • OS Debug • OS Exit • Domain Down • Domain In Recovery
-------	--------	---

CIM Solaris_SGDomain クラス

クラス階層での位置

```
CIM_CollectionOfMSEs
|
+---Solaris_WDRDomain
    |
    +---Solaris_SGDomain
```

説明

CIM Solaris_SGDomain クラスは CIM Solaris_WDRDomain クラスのサブクラスで、Sun Fire 6800、4810、4800、および 3800 システムのドメイン情報を表します。このクラスには、Sun Fire 6800、4810、4800、および 3800 システムに固有の情報が含まれている CIM プロパティがいくつかあります。

直系の既知の CIM サブクラス

なし

CIM Solaris_SGDomain クラスのプロパティ

表 4-14 CIM Solaris_SGDomain のプロパティ

名前	データ型	説明
BoardRelationship[]	sint32	<p>ドメイン内のボードの状態を示す値の配列 (各ボードについて 1 つの値)。配列 BitMap の各位置の ValueMap 項目 0 ~ 4 が、以下のボード状態の値を表します。</p> <ul style="list-style-type: none"> • Nonexistent Slot • Not Available • Available • Assigned • Active <p>Sun Fire 6800 では、BitMap 値 1 ~ 10 ですべてのボードを表します。BitMap 値 1 ~ 6 がシステムボード 0 ~ 5 (SB0 ~ SB5) に対応します。BitMap 値 7 ~ 10 が入出力ボード (IB6 ~ IB9) に対応します。</p> <p>Sun Fire 3800、4800、および 4810 システムでは、CPU ボード用に 3 個、入出力ボード用に 2 個、合計 5 個のスロットのみ使用できます。したがって、BitMap 値 4、5、および 6 (SB3、SB4、および SB5 用) と、BitMap 値 9 と 10 (IB8 と IB9 用) は常に 0 (Nonexistent Slot) です。</p>
KeyswitchPosition	uint32	<p>ドメインの状態を示します。値 1 ~ 16 が ValueMap の以下の項目を表します。</p> <ul style="list-style-type: none"> • Unknown • Off • Standby • On • Diag • Secure • Off To Standby • Off To On • Off To Diag • Off To Secure • Standby To Off • Active To Off • Active To Standby • Reboot To On • Reboot To Diag • Reboot To Secure

表 4-14 CIM Solaris_SGDomain のプロパティ (続き)

State	uint32	ドメインの現在の状態。ValueMap 項目 1 ~ 14 が以下の値を表します。 <ul style="list-style-type: none"> • Unknown • Running POST • Standby • Active • Powered Off • Domain Idle • Running OBP • Booting • Running Solaris • Halted • Reset • Panic • Debugger • Hang Detected
-------	--------	--

WDR スキーマ の関連と集約

CIM 関連クラスは、1 つの WDR クラスまたはインスタンスを、別のクラスやインスタンスに関連付ける特別なクラスです。関連は、一対一の関係であることも、集約であることもあります。

WDR の集約は、1 つの WDR クラスまたはインスタンスを多数のクラスやインスタンスに関連付けます。

CIM Solaris_DomainHasAttachmentPoints 集約

説明

接続点がドメインで使用可能である (ドメインの使用可能構成要素リストに表示される) か、ドメインに割り当てられている場合に、ドメインは接続点を持っているといいます。動作中のドメインだけが接続点を持つことができます。

Solaris_DomainHasAttachmentPoints 集約は、Solaris_WDRDomain クラスのサブインスタンスを、ドメインで使用可能であるか、またはドメインに割り当てられている Solaris_WDRAttachmentPoint クラスのサブインスタンスに関連付けます。

Solaris_DomainHasAttachmentPoints 集約は、ドメインが 1 つまたは複数の接続点で構成されているコンポジション関連です。

Solaris_DomainHasAttachmentPoints 集約の親は、Solaris_WDRDomain クラスのサブインスタンスです。Solaris_DomainHasAttachmentPoints 集約の子は、Solaris_WDRAttachmentPoint クラスのサブインスタンスです。

Solaris_DomainHasAttachmentPoints 集約は、1 つのドメインで複数の接続点で使用できる、または割り当てられている一対多の関係です。

CIM Solaris_DomainHasAttachmentPoints 集約のプロパティ

表 4-15 CIM Solaris_DomainHasAttachmentPoints 集約のプロパティ

名前	データ型	説明
Collection	REF Solaris_WDRDomain	集約関係にある親を参照します。
Member	REF Solaris_WDRAttachmentPoint	集約関係にある子を参照します。

CIM Solaris_DomainHasSlots 集約

説明

ゼロ個以上のスロットが装備されているドメインの 1 つの特性。システムボードが装着されているかどうかに関わらず、スロットをドメインに割り当てることができます。その結果、Solaris_DomainHasSlots 集約は、CIM Solaris_WDRDomain と CIM Solaris_WDRSlot クラス間のバインドを関連付けます。

Solaris_DomainHasSlots 集約は、ドメインが 1 つまたは複数のスロットで構成されているコンポジション関連です。

Solaris_DomainHasSlots 集約では、親は Solaris_XCDomain クラスのインスタンスで、子は Solaris_WDRSlot クラスのインスタンスです。

Solaris_DomainHasSlots 集約は、複数のスロットを 1 つのドメインに割り当てることができる一対多の関係です。ただし、1 つのスロットを一度に複数のドメインに存在させることはできません。

CIM Solaris_DomainHasSlots 集約のプロパティ

表 4-16 CIM Solaris_DomainHasSlots 集約のプロパティ

名前	データ型	説明
Collection	REF Solaris_WDRDomain	集約関係にある親を参照します。
Member	REF Solaris_WDRSlot	集約関係にある子を参照します。

Solaris_SlotHasSystemBoard 関連

説明

スロットがドメインに割り当てられているかどうかに関わらず、ボードを装着できるスロット。CIM Solaris_SlotHasSystemBoard 関連は、CIM Solaris_WDRSlot クラスのインスタンスを、スロットに装着するボードに対応する CIM Solaris_SystemBoard クラスのインスタンスに関連付けます。

CIM Solaris_SlotHasSystemBoard はコンポジション関連であり、ゼロ個以上の CIM Solaris_SystemBoard クラスのインスタンスから構成される CIM Solaris_WDRSlot クラスのインスタンスです。

CIM Solaris_SlotHasSystemBoard 関連のプロパティ

表 4-17 CIM Solaris_SlotHasSystemBoard 関連のプロパティ

名前	データ型	説明
Antecedent	REF Solaris_WDRSlot	関連関係にある親を参照します。
Dependent	REF Solaris_CHSystemBoard	関連関係にある子を参照します。

Solaris_SystemBoardHasProcessors 集約

説明

システムボードは、プロセッサ、メモリーモジュール、および入出力モジュールが搭載されている大規模回路基板です。CIM Solaris_SystemBoardHasProcessors 集約は、Solaris_CHSystemBoard クラスのインスタンスと Solaris_CHCPU クラスのインスタンス間の関係を表し、システムボードを、そのボードに搭載されているプロセッサに関連付けます。

集約は、ボードにゼロ個から 4 個までのプロセッサを搭載できる一対多の関係です。

CIM Solaris_SystemBoardHasProcessors 集約のプロパティー

表 4-18 CIM Solaris_SystemBoardHasProcessors 集約のプロパティー

名前	データ型	説明
GroupComponent	REF Solaris_CHSystemBoard	集約関係にある親を参照します。
PartComponent	REF Solaris_CHCPU	集約関係にある子を参照します。

Solaris_SystemBoardHasMemory 集約

説明

システムボードは、プロセッサ、メモリーモジュール、および入出力モジュールが搭載されている大規模回路基板です。CIM Solaris_SystemBoardHasMemory 集約は、Solaris_CHSystemBoard クラスのインスタンスと Solaris_CHMemory クラスのインスタンスを関連付けて、システムボードと、そのボードに搭載されているメモリーを関連付けます。

Solaris_CHMemory クラスは、システムボード上のメモリーに関する情報の集まりです。特定のシステムボードでは、Solaris_CHMemory クラスの 1 つのインスタンスについて上限があります。

CIM Solaris_SystemBoardHasMemory 集約のプロパティ

表 4-19 CIM Solaris_SystemBoardHasMemory 集約のプロパティ

名前	データ型	説明
GroupComponent	REF Solaris_CHSystemBoard	集約関係にある親を参照します。
PartComponent	REF Solaris_CHMemory	集約関係にある子を参照します。

Solaris_SystemBoardHasControllers 集約

説明

システムボードには、プロセッサとメモリーモジュールだけでなく、ディスク、ネットワークコントローラなどの入出力モジュールも搭載できます。CIM Solaris_SystemBoardHasControllers 集約は、システムボードを、そのボードに装備されているコントローラに関連付けます。

Solaris_SystemBoardHasControllers 集約は、1つのシステムボードに複数の入出力デバイスを搭載できる一対多の関係です。

CIM Solaris_SystemBoardHasControllers 集約のプロパティ

表 4-20 CIM Solaris_SystemBoardHasControllers 集約のプロパティ

名前	データ型	説明
GroupComponent	REF Solaris_CHSystemBoard	集約関係にある親を参照します。
PartComponent	REF Solaris_CHController	集約関係にある子を参照します。

CIM プロセスインジケーションクラス

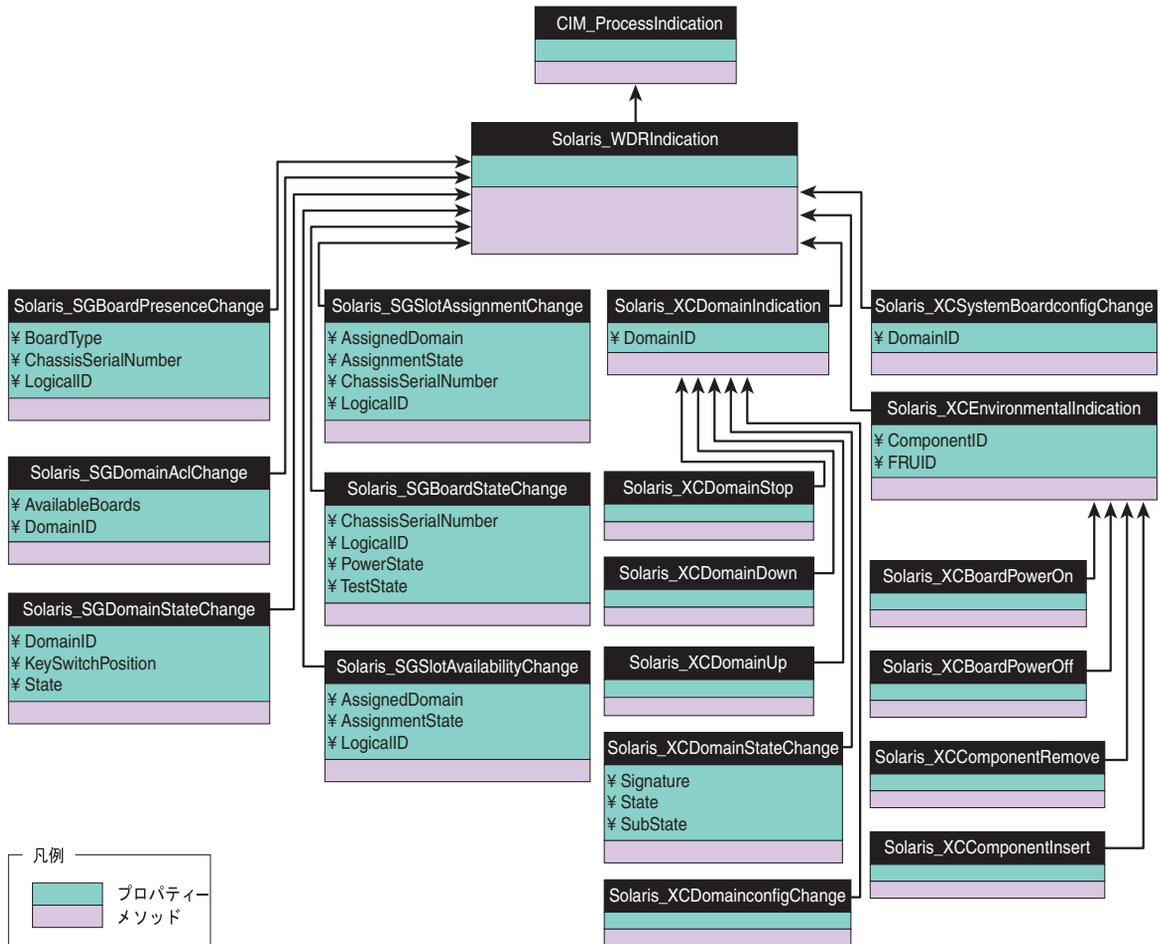
CIM プロセスインジケーションは、CIM_Processindication クラスのサブクラスです。CIM プロセスインジケーションは、Sun Fire 15K、12K、6800、4810、4800、および 3800 システム上でのイベント通知をクライアントアプリケーションに転送するときに WDR が使用します。プロセスインジケーションについては、第 3 章「プロセスインジケーションの使用」で詳細に説明しています。

Sun Fire 6800、4810、4800、および 3800 システム上でのプロセスインジケーションは、システムコントローラ (SC) から受信した SNMP トラップの中から選ばれて生成されます。

Sun Fire 15K および 12K システム上でのプロセスインジケーションは、Sun Fire 15K および Sun Fire 12K SC 上で、システムイベント機能である sysevent によって生成されたイベントの中から選ばれて生成されます。

注 - 名前に "XC" が含まれているプロセスインジケーションクラスは、Sun Fire™ 15K および 12K システム用です。名前に "SG" が含まれているクラスは、Sun Fire 6800、4810、4800、および 3800 システム用です。

WDR インジケーションクラス階層図



Solaris_WDRIndication クラス

Solaris_WDRIndication クラスは、すべての Sun Fire システム上で、すべてのプロセスインジケーションクラスがここから派生する抽象クラスです。Solaris_WDRIndication クラスでは、その基底クラスに何もプロパティを追加しません。

Solaris_SGBoardPresenceChange インジケーション

説明

このプロセスインジケーションは Sun Fire 6800、4810、4800、および 3800 システムで使用され、CPU や入出力ボードがスロットに装着または取り外されたことをクライアントに通知します。

Solaris_SGBoardPresenceChange のプロパティ

表 4-21 Solaris_SGBoardPresenceChange のプロパティ

名前	データ型	説明
LogicalID	string	スロットの論理名。Sun Fire 6800、4810、4800、または 3800 システムには、最大 6 個のシステムボードと最大 4 個の入出力ボードを装着できます。システムボードのスロットは SB0、SB1、... SB5 と表記され、入出力ボードのスロットは IB6、IB7、IB8、および IB9 と表記されます。
ChassisSerialNumber	string	シャーシのシリアル番号は 8 けたの 16 進数文字列です (たとえば 10483D99)。
BoardType	uint32	スロットが空でない場合に、スロットに装着されているボードのタイプ。値: Unknown、Empty、CPU、IO、CPUWIB、IOWIB、SC、L2、Fan、Power Supply、または Logic Analyzer。現在は、タイプ CPU および IO のボードのみが通知されます。

Solaris_SGDomainACLChange インジケーション

説明

このプロセスインジケーションは Sun Fire 6800、4810、4800、および 3800 システムで使用され、使用可能構成要素リストが変更されたことをクライアントに通知します。

Solaris_SGDomainACLChange のプロパティ

表 4-22 Solaris_SGDomainACLChange のプロパティ

名前	データ型	説明
DomainID	uint32	ボードが割り当てられていた、またはボードが割り当て解除されたドメイン。値：A、B、C、または D。
AvailableBoards[]	boolean	DomainID プロパティで特定されるドメインで使用可能なスロットのリスト。値：SB0、SB1、SB2、SB3、SB4、SB5、IB6、IB7、IB8、および IB9。

Solaris_SGDomainStateChange インジケーション

説明

このプロセスインジケーションは Sun Fire 6800、4810、4800、および 3800 システムで使用され、ドメインの開始と停止、ドメインの自己診断失敗、またはドメインのキースイッチ状態の変更をクライアントに通知します。

Solaris_SGDomainStateChange のプロパティ

表 4-23 Solaris_SGDomainStateChange のプロパティ

名前	データ型	説明
DomainID	uint32	状態が変更されたドメイン。値 : A、B、C、または D。
KeyswitchPosition	uint32	仮想キースイッチの位置を特定します。値 : Unknown、Off、Standby、On、Diag、Secure、Off To Standby、Off To On、Off To Diag、Off To Secure、Standby To Off、Active To Off、Active To Standby、Reboot To On、Reboot To Diag、および Reboot To Secure。
State	uint32	ドメインの現在の状態。値 : Unknown、Running Post、Standby、Active、Powered Off、Domain Idle、Running OBP、Booting、Running Solaris、Halted、Reset、Panic、Debugger、または Hang Detected。

Solaris_SGSlotAssignmentChange インジケーション

説明

このプロセスインジケーションは Sun Fire 6800、4810、4800、および 3800 システムで使用され、スロットがドメインに割り当てられていた、またはドメインから割り当て解除されたことをクライアントに通知します。

Solaris_SGSlotAssignmentChange のプロパティ

表 4-24 Solaris_SGSlotAssignmentChange のプロパティ

名前	データ型	説明
LogicalID	string	スロットの論理名。Sun Fire 6800、4810、4800、または 3800 システムには、最大 6 個のシステムボードと最大 4 個の入出力ボードを装着できます。システムボードのスロットは SB0、SB1、... SB5 と表記され、入出力ボードのスロットは IB6、IB7、IB8、および IB9 と表記されます。
ChassisSerialNumber	string	シャーシのシリアル番号は 8 けたの 16 進数文字列です (たとえば 10483D99)。
AssignedDomain	sint32	スロットが割り当てられているドメイン (割り当てられている場合)。値: A、B、C、D、または None。
AssignmentState	uint32	現在スロットに割り当てられている状態。値: Unknown、Free、Assigned、または Active。

Solaris_SGBoardStateChange インジケーション

説明

このプロセスインジケーションは Sun Fire 6800、4810、4800、および 3800 システムで使用され、ボードの自己診断が完了したこと、またはボードの電源が投入または切断されたことをクライアントに通知します。

Solaris_SGBoardStateChange のプロパティ

表 4-25 Solaris_SGBoardStateChange のプロパティ

名前	データ型	説明
LogicalID	string	スロットの論理名。Sun Fire 6800、4810、4800、または 3800 システムには、最大 6 個のシステムボードと最大 4 個の入出力ボードを装着できます。システムボードのスロットは SB0、SB1、... SB5 と表記され、入出力ボードのスロットは IB6、IB7、IB8、および IB9 と表記されます。
ChassisSerialNumber	string	シャーシのシリアル番号は 8 けたの 16 進数文字列です (たとえば 10483D99)。
PowerState	uint32	ボードの電源状態。値: Unknown、On、Off、または Failed。
TestState	uint32	ボードのテスト状態。値: Unknown、Not Tested、Passed、Failed、Under Test、Start Test、Degraded、または Unusable。

Solaris_SGSlotAvailabilityChange インジケーション

説明

このプロセスインジケーションは Sun Fire 6800、4810、4800、および 3800 システムで使用され、スロットの可用性が変更されたことをクライアントに通知します。

Solaris_SGSlotAvailabilityChange のプロパティ

表 4-26 Solaris_SGSlotAvailabilityChange のプロパティ

名前	データ型	説明
LogicalID	string	スロットの論理名。Sun Fire 6800、4810、4800、または 3800 システムには、最大 6 個のシステムボードと最大 4 個の入出力ボードを装着できます。システムボードのスロットは SB0、SB1、... SB5 と表記され、入出力ボードのスロットは IB6、IB7、IB8、および IB9 と表記されます。
AssignedDomain	sint32	スロットが割り当てられていたが、現在は割り当て解除されているドメイン、またはスロットが現在も割り当てられているドメイン。 値：A、B、C、または D。
AssignmentState	uint32	現在スロットに割り当てられている状態。値：Unknown、Free、Assigned、または Active。

Solaris_XCSystemBoardConfigChange インジケーション

説明

このプロセスインジケーションは Sun Fire 15K および 12K システムで使用され、特定のドメインで、1 つまたは複数の Sun Fire 15K/12K ドメイン構成プロパティが変更されたことをクライアントに通知します。

Solaris_XCSystemBoardConfigChange のプロパティ

表 4-27 Solaris_XCSystemBoardConfigChange のプロパティ

名前	データ型	説明
LogicalID	string	構成データが変更されたシステムボードを特定します。

Solaris_XCEnvironmentalIndication インジケーション

説明

Sun Fire 15K および 12K システム上の環境インジケーションすべての共通の祖先として機能する抽象クラス。

Solaris_XCEnvironmentalIndication のプロパティ

Solaris_XCEnvironmentalIndication クラスでは、その基底クラスに以下のプロパティを追加します。

表 4-28 Solaris_XCEnvironmentalIndication のプロパティ

名前	データ型	説明
ComponentID	string	環境イベントが発生しているコンポーネント。
FRUID	uint32	コンポーネントがシステムボードの場合は、対応する Field Replaceable Unit 識別子が含まれ、それ以外の場合は NULL です。

Solaris_XCComponentRemove インジケーション

このクラスは Solaris_XCEnvironmentalIndication 抽象クラスから派生し、特定のホットプラグ対応コンポーネントが Sun Fire 15K または 12K システム上のスロットから取り外されたことをクライアントに通知します。

このクラスは基底クラスに何もプロパティを追加せず、直系の既知のサブクラスもありません。

Solaris_XCComponentInsert インジケーション

このクラスは `Solaris_XCEnvironmentalIndication` 抽象クラスから派生し、特定のホットプラグ対応コンポーネントが Sun Fire 15K または 12K システム上のスロットに挿入されたことをクライアントに通知します。

このクラスは基底クラスに何もプロパティを追加せず、直系の既知のサブクラスもありません。

Solaris_XCBoardPowerOn インジケーション

このクラスは `Solaris_XCEnvironmentalIndication` 抽象クラスから派生し、Sun Fire 15K または 12K システムのシステムボードの電源が投入されたことをクライアントに通知します。

このクラスは基底クラスに何もプロパティを追加せず、直系の既知のサブクラスもありません。

Solaris_XCBoardPowerOff インジケーション

このクラスは `Solaris_XCEnvironmentalIndication` 抽象クラスから派生し、Sun Fire 15K または 12K システムのシステムボードの電源が切断されたことをクライアントに通知します。

このクラスは基底クラスに何もプロパティを追加せず、直系の既知のサブクラスもありません。

Solaris_XCDomainIndication インジケーション

説明

この抽象クラスは `Solaris_XCEnvironmentalIndication` 抽象クラスから派生し、Sun Fire 15K および 12K システム上のすべてのドメインインジケーションの共通の祖先として機能します。

Solaris_XCDomainIndication のプロパティー

Solaris_XCDomainIndication クラスでは、その基底クラスに以下のプロパティーを追加します。

表 4-29 Solaris_XCDomainIndication のプロパティー

名前	データ型	説明
DomainID	uint32	イベントが発生しているドメインを特定します。

Solaris_XCDomainConfigChange インジケーション

このクラスは、Solaris_XCDomainIndication 抽象クラスから派生し、Sun Fire 15K または 12K システム上の特定のドメインで、1 つまたは複数の構成プロパティーが変更されたことをクライアントに通知します。

このクラスは基底クラスに何もプロパティーを追加せず、直系の既知のサブクラスもありません。

Solaris_XCDomainUp インジケーション

このクラスは Solaris_XCDomainIndication 抽象クラスから派生し、Sun Fire 15K または 12K システム上の特定のドメインが開始されたことをクライアントに通知します。ドメインが開始するのは、キースイッチが **On** に設定されているとき、またはドメイン監視デーモン (DSMD) が再起動してから、ドメインに割り当てられている IOSRAM がアクセス可能であることを検出したときです。

このクラスは基底クラスに何もプロパティーを追加せず、直系の既知のサブクラスもありません。

Solaris_XCDomainDown インジケーション

このクラスは Solaris_XCDomainIndication 抽象クラスから派生し、Sun Fire 15K または 12K システム上の特定のドメインが停止されたことをクライアントに通知します。キースイッチが **Off** または **Standby** に設定されると、ドメインは停止します。

このクラスは基底クラスに何もプロパティを追加せず、直系の既知のサブクラスもありません。

Solaris_XCDomainStop インジケーション

このクラスは Solaris_XCDomainIndication 抽象クラスから派生し、Sun Fire 15K または 12K システム上の特定のドメインがハードウェア状態ダンプを開始したことをクライアントに通知します。回復不能なハードウェア障害が発生したために、ドメインでハードウェア状態情報をダンプファイルに書き込めなくなった場合に、ハードウェア状態ダンプが行われます。

このクラスは基底クラスに何もプロパティを追加せず、直系の既知のサブクラスもありません。

Solaris_XCDomainStateChange インジケーション

説明

このクラスは Solaris_XCDomainIndication 抽象クラスから派生し、Sun Fire 15K または 12K システム上の特定のドメインの状態が変更されたことをクライアントに通知します。

Solaris_XCDomainStateChange のプロパティ

Solaris_XCDomainStateChange クラスでは、その基底クラスに以下のプロパティを追加します。

表 4-30 Solaris_XCDomainStateChange のプロパティ

名前	データ型	説明
Signature	uint32	Signature、State、および SubState プロパティを組み合わせる現在のドメインの状態を表します。
State	uint32	Signature、State、および SubState プロパティを組み合わせる現在のドメインの状態を表します。
SubState	uint32	Signature、State、および SubState プロパティを組み合わせる現在のドメインの状態を表します。

第5章

WDR のプログラミング手法

この章では、コーディング例を示しながら、WDR を使用してタスクを実行するときの手法を説明します。ただし、ここに挙げるコーディング例は、製品版の WDR アプリケーションでの使用を目的としたものではありません。

コーディング例では、以下のプロバイダの操作方法を説明しています。

- EventProvider
- InstanceProvider
- AssociatorProvider
- MethodProvider

システムの状態情報のキャッシュ

WDR のクライアントアプリケーションを開発する際に考慮すべき重要な事項は、管理対象プラットフォームのドメイン、接続点、およびスロットの現在の状態をクライアントで把握する方法には、ポーリングによる方法とキャッシュを使用する方法の、全く異なる 2 つのアプローチがあることです。

クライアントでは、対応する WDR クラスのインスタンスを列挙することにより、ドメイン、接続点、およびスロットの状態を定期的にポーリングすることができます。ただし、WDR を使用した操作の実行に要する時間はシステムの状態と作業負荷に依存し、変動することがあるため、このアプローチは推奨されません。これは、システムコントローラ (SC) とクライアントアプリケーション双方の性能に悪影響を及ぼすこととなります。

より適切なアプローチは、クライアントで、ドメイン、接続点、およびスロットの現在の状態のキャッシュを保持し、クライアントの状態情報のキャッシュを更新する必要があるときに、WDR プロセッシングケーションを使用して、更新を指示する方法です。詳細は、87 ページの「CIM プロセッシングケーションクラス」を参照してください。

EventProvider の操作

EventProvider を作成するには、次の作業を実行します。

- WDR インジケーションを選択して読み取る。
- イベントリスナーを実装する。
- イベントフィルタとイベントハンドラをバインドする。

▼ WDR インジケーションを選択して読み取る

以下のコードは、WDR イベントインジケーションの選択および読み取り方法を示しています。

```
/* Standard java packages */
import java.io.*;
/* Solaris WBEM packages */
import com.sun.wbem.cim.*;
import com.sun.wbem.client.*;
import com.sun.wbem.security.*;

public class IndicationReader
{
    public static void main(String args[]) throws CIMException
    {
        if (args.length != 3) {
            System.out.println("Usage: java IndicationReader " +
                "<hostname> <username> <password>");
            System.exit(1);
        }
        String hostName = args[0];
        UserPrincipal userName = new UserPrincipal(args[1]);
        PasswordCredential passWord = new PasswordCredential(args[2]);
        CIMNameSpace nameSpace = new CIMNameSpace();
        nameSpace.setHost(hostName);
        // Read all WDR Indications.
        final String filter = "SELECT * FROM Solaris_WDRIndication";
        IndicationSubscription subscription = null;
```

```

try
{
    // creates a CIMClient adding CIMListener to it.
    CIMClient cc = new CIMClient(nameSpace, userName,
        passWord);
    cc.addCIMListener(new EventListener());
    // subscribes to WDR Indications and waits
    subscription = new IndicationSubscription(cc, filter);
    System.out.println("Waiting for Indications...");
    waitForQuit();
}
catch (Exception e) {
    e.printStackTrace();
}
finally {
    if ( subscription != null ) {
        subscription.remove();
    }
}
System.exit(0);
}
/*
 * Exit when user types 'quit'
 */
private static void waitForQuit() throws IOException
{
    BufferedReader stdin =
        new BufferedReader( new InputStreamReader(System.in));
    String line = null;
    do {
        System.out.println("Type 'quit' followed by <CR> to exit");
        System.out.print("IR> ");
        line = stdin.readLine();
    } while ( ! line.startsWith("quit") );
}
}

```

▼ イベントリスナーを実装する

以下のコードでは、CIM イベントを待機できるように、CIMListener インタフェースを実装しています。CIM イベントのインジケーションに関する登録を行うには、CIMListener のインスタンスを追加する必要があります。

```
/* WBEM libraries */
import com.sun.wbem.client.*;

public class EventListener implements CIMListener
{
    public EventListener()
    {
    }
    /**
     * Prints indication of an event when the indication is available
     * for delivery.
     */
    public void indicationOccured(CIMEvent e)
    {
        System.out.println("Received " + e.getIndication());
    }
}
```

▼ イベントフィルタとイベントハンドラをバインドする

IndicationSubscription クラスを使用すると、クライアントは CIM イベントが通知されるように申請できます。以下のコードは、イベントフィルタをイベントハンドラにバインドします。

```
/* Standard Java packages */
import java.util.*;

/* Standard WBEM packages */
import com.sun.wbem.cim.*;
import com.sun.wbem.client.*;
import com.sun.wbem.security.UserPrincipal;
```

```

import com.sun.wbem.security.PasswordCredential;

public class IndicationSubscription
{
    static protected int m_FilterCnt = 0;

    protected CIMClient m_Client;
    protected CIMObjectPath m_Filter;
    protected CIMObjectPath m_Handler;
    protected CIMObjectPath m_Subscription;

    final String subscriptionClassName =
        "CIM_IndicationSubscription";
    final String filterClassName = "CIM_IndicationFilter";
    final String deliveryClassName = "Solaris_RMIDelivery";
    /**
    * Force construction through another constructor that is public.
    */
    protected IndicationSubscription() {
        m_Client = null;
        m_Filter = null;
        m_Handler = null;
        m_Subscription = null;
    }

    /**
    * Construct an IndicationSubscription that subscribed for
    * Indications as expressed by the specified filterExp. Three
    * CIM objects are created in the CIM repository as a
    * side-effect of calling this method, a CIM_IndicationFilter,
    * a CIM_IndicationHandler, and a CIM_IndicationSubscription.
    * These can be removed by calling the remove method.
    *
    * @param cc          a CIMClient instance
    * @param filterExp   The query string on which to filter
    *                    Indications
    * @exception CIMException
    */
}

```

```

*/
public IndicationSubscription(CIMClient cc, String filterExp)
throws CIMException
{
    m_Client = cc;
    m_Filter = createFilter(filterExp);
    m_Handler = createHandler();
    m_Subscription = createSubscription();
}
/**
 * Removes the otherwise persistent filter, handler and
 * subscription CIM objects from the CIM repository.
 * @exception CIMException if an attempt is made to delete a
 * non-existent CIM object.
 */
public void remove() throws CIMException {
    if ( m_Subscription != null ) {
        m_Subscription.setNameSpace("");
        m_Client.deleteInstance(m_Subscription);
        m_Subscription = null;
    }
    if ( m_Handler != null ) {
        m_Handler.setNameSpace("");
        m_Client.deleteInstance(m_Handler);
        m_Handler = null;
    }
    if ( m_Filter != null ) {
        m_Filter.setNameSpace("");
        m_Client.deleteInstance(m_Filter);
        m_Filter = null;
    }
}

/**
 * Create an IndicationFilter of the specified name and with the

```

```

* specified filterExp as the query string. Register the filter
* by creating its instance in the repository. Only one filter
* may exist per IndicationSubscription object.
*
* @param filterExp The query string on which to filter
* Indications
* @return CIMObjectPath of the filter.
* @exception CIMException
*/
protected CIMObjectPath createFilter(String filterExp) throws
    CIMException
{
    CIMClass filterClass =
        m_Client.getClass(new CIMObjectPath(filterClassName),
            false, true, true, null);

    CIMInstance ci = filterClass.newInstance();

    ci.setProperty("Name", new CIMValue(generateFilterName()));
    ci.setProperty("Query", new CIMValue(filterExp));
    ci.setProperty("QueryLanguage", new CIMValue("WQL"));

    CIMObjectPath op = m_Client.createInstance(new
        CIMObjectPath(), ci);
    return ( op );
}
/**
* Generate a unique filter name for this Java VM.
*
* @return Name of the filter.
*/
protected String generateFilterName()
{
    String filterName = "WDRFilter"+ m_FilterCnt;
    m_FilterCnt = (m_FilterCnt + 1) % Integer.MAX_VALUE;
    return ( filterName );
}

```

```

/**
 * Create an indication handler.
 * Register the handler by creating its instance in the repository.
 *
 * @return CIMObjectPath of the handler.
 */
protected CIMObjectPath createHandler() throws CIMException
{
    CIMClass deliveryClass =
        m_Client.getClass(new CIMObjectPath(deliveryClassName),
            false, true, true, null);
    CIMInstance ci = deliveryClass.newInstance();

    CIMObjectPath op = m_Client.createInstance(new
        CIMObjectPath(), ci);
    return ( op );
}
/**
 * Create an indication subscription that binds filter to handler.
 * Register the subscription by creating its instance in the
    repository.
 *
 * @return CIMObjectPath of subscription.
 */
protected CIMObjectPath createSubscription() throws CIMException
{
    final String subscriptionClassName =
        "CIM_IndicationSubscription";
    CIMClass subscriptionClass =
        m_Client.getClass(new CIMObjectPath(subscriptionClassName),
            false, true, false, null);

    CIMInstance ci = subscriptionClass.newInstance();
    ci.setProperty("Filter", new CIMValue(m_Filter));
    ci.setProperty("Handler", new CIMValue(m_Handler));
}

```

```

m_Client.createInstance(new CIMObjectPath(), ci);

// we are looking for the subscription's reference because
// createInstance() returns a null reference for the
// subscription.
CIMObjectPath cop =
    new CIMObjectPath(subscriptionClassName,
        ci.getKeyValuePairs());
return ( cop );
}
}

```

InstanceProvider の操作

以下のコーディング例では、m_Client という CIMClient オブジェクトがすでに作成されて使用できる状態になっているものとします。

1. enumerateInstanceNames メソッドと getInstance メソッドを使用して、Solaris_XCDomain クラスのすべてのインスタンスを取得します。

```

// gets path to all instances
CIMObjectPath cop = new CIMObjectPath("Solaris_XCDomain");
Enumeration e = m_Client.enumerateInstanceNames(cop);

// gets instances from the instances' paths
while ( e.hasMoreElements() ) {
    cop = (CIMObjectPath) e.nextElement();
    CIMInstance ci = m_Client.getInstance(cop, true, false, false,
        null);
    System.out.println(ci.toString());
}

```

2. enumerateInstances メソッドを呼び出します。

```

CIMObjectPath cop = new CIMObjectPath("Solaris_XCDomain");
Enumeration e = m_Client.enumerateInstances(cop, true, false, false,
    null);

while ( e.hasMoreElements() ) {

```

```
CIMInstance ci = (CIMInstance) e.nextElement();
System.out.println(ci.toString());
}
```

AssociatorProvider の操作

以下のコーディング例では、`m_Client` という `CIMClient` オブジェクトがすでに作成されて使用できる状態になっているものとします。

1. `Solaris_SystemBoardHasProcessors` 関連により `Solaris_CHSystemBoard` クラスのインスタンスに関連付けられている `Solaris_CHCPU` クラスの各インスタンスを取得します。

```
// sbCOP is a CIMObjectPath of a system board.
String assocClass = "Solaris_SystemBoardHasProcessor";
String resultClass = "Solaris_CHCPU";
String role = "SystemBoard";
String resultRole = "Processor";
boolean includeQualifiers = true;
boolean includeClassOrigin = true;
String[] cpuProperty = null;

Enumeration e = m_Client.associators(sbCOP, assocClass, resultClass,
    role, resultRole, includeQualifiers, includeClassOrigin,
    cpuProperty);
while ( e.hasMoreElements() ) {
    CIMInstance ci = (CIMInstance) e.nextElement();
    System.out.println(ci.toString());
}
```

2. `SolarisCHSystemBoard` クラスと `Solaris_CHCPU` クラスのインスタンスを参照する関連オブジェクトを列挙します。

```
// cop is CIMObjectPath of the Solaris_CHSystemBoard instance
String resultClass = "Solaris_SystemBoardHasProcessors"
String role = "SystemBoard";
String includeQualifiers = true;
String includeClassOrigin = true;
String[] propertyList = "Processor";
```

```

Enumeration e = m_Client.references(cop, resultClass, role,
includeQualifiers, includeClassOrigin, propertyList);
while ( e.hasMoreElements() ) {
    CIMInstance assoc = (CIMInstance) e.nextElement();
    System.out.println(assoc.toString());
}

```

MethodProvider の操作

以下のコーディング例では、m_Client という CIMClient オブジェクトがすでに作成されて使用できる状態になっているものとします。

- 1 つのプロセッサを構成し、その構成処理中に発生したエラーメッセージがあれば、それを標準の出力デバイスに出力します。

```

// cop is CIMObjectPath of the processor
String method = "configure";
Vector inParams = new Vector(4);
Vector outParams = new Vector(2);

inParams.add(CIMValue.FALSE);           /* force */
inParams.add(new CIMValue(new String(""))); /* hwOptions */
inParams.add(new CIMValue(new Integer(3))); /* 3 retries */
inParams.add(new CIMValue(new Integer(5))); /* 5s delay */

CIMValue returnVal = m_Client.invokeMethod(cop, method, inParams,
outParams);
int status = ((Integer)(returnVal.getValue())).intValue();
if ( status != 0 && outParams.size() != 0 ) {
    Object obj = ((CIMValue)(outParams.elementAt(0))).getValue();
    String error = (String) obj;
    if ( error != null ) {
        System.out.println(error);
    }
}
}

```

2. システムボードをドメインに割り当てて、その割り当て処理中に発生したエラーメッセージがあれば、それを標準の出力デバイスに出力します。

```
// cop is the CIMObjectPath of a system board
String method = "Assign";
Vector inParams = new Vector(1);
Vector outParams = new Vector(2);
inParams.add(new CIMValue(new Integer(domainID))); /* domainID
CIMValue returnVal = m_Client.invokeMethod(cop, method, inParams,
    outParams);
int status = ((Integer)(returnVal.getValue())).intValue();
if ( status != 0 && outParams.size() != 0 ) {
    Object obj = ((CIMValue)(outParams.elementAt(0))).getValue();
    String error = (String) obj;

    if ( error != null ) {
        System.out.println(error);
    }
}
```

付録 A

MOF ファイル

MOF (Managed Object Format) ファイルには、WDR を使用して管理できるオブジェクトが記述されています。WDR には、3 つの MOF ファイルが同梱されています。

- WDR_Core1.0.mof ファイルには、WDR を実行できるすべてのシステムに共通のオブジェクトが記述されています。
- WDR_SG1,0.mof ファイルには、WDR を使用して管理できる Sun Fire 6800、4810、4800 および 3800 システムでのオブジェクトが記述されています。
- 3 つ目の MOF ファイルには、WDR を使用して管理できる Sun Fire 15K および 12K システムでのオブジェクトが記述されています。

以下に、上記 3 つの MOF ファイルの内容を記載します。

WDR_Core1.0.mof ファイル

```
// Copyright (c) 2001 by Sun Microsystems, Inc. All rights
// reserved.

// Title:          WBEM Dynamic Reconfiguration (DR) Common
                  Information Model (CIM) Schema

// Filename:      WDR_Core1.0.mof

// Author:        Sun Microsystems, Inc.

// Description:   This file contains CIM classes and CIM
                  // associations for the WBEM DR Common
                  // Information Model (CIM) Schema that are
                  // common to all platform implementations. The
                  // WBEM DR CIM Schema models DR related
                  // operations and resources for the SunFire
                  // 15K and SunFire 68x0, 48x0, and 3800
                  // platforms.
```

```

// @(#) WDR_Core1.0.mof 1.10@(#)
// =====
//      Pragmas:
// =====

#pragma namespace ("root/system")

instance of Solaris_ProviderPath {
    [Description("Describes the path to the JAR file
        containing the WBEM DR provider classes." ) ]
    pathurl = "file:///usr/sadm/lib/wbem/wdr.jar";
};

#pragma Locale ("en-US")

#pragma namespace ("root/cimv2")

// =====
//      CIM Solaris_WDRAttachmentPoint class
// =====

[Description("The CIM Solaris_WDRAttachmentPoint class
represents the core Configuration Administration (cfgadm)
information. This information is gathered using the
libcfgadm library."),
Provider("com.sun.wbem.wdr.AttachmentPointProvider"),
Version("1.0") ]

class Solaris_WDRAttachmentPoint : CIM_LogicalElement
{
    [Key, MaxLen(30), Description("The logical name of the
attachment point." ) ]
    string LogicalID;

    [Key, Description("The domain to which this attachment
point is assigned or available." ) ]
    uint32 DomainID;

    [MaxLen(1044), Description("The physical name of the
attachment point." ) ]
    string PhysicalID;

    [MaxLen(12), Description("The class of the attachment
point." ) ]
    string ClassName;

    [Description("The busy state indicator of the attachment
point." ) ]
    uint32 Busy;
}

```

```

[Description( "The receptacle state of the attachment point.
There are four possible states for the receptacle of an
attachment point: "None, Empty, Disconnected, Connected"),
ValueMap{"0", "1", "2", "3"}, Values{"None", "Empty",
"Disconnected", "Connected"} ]
    uint32 ReceptacleState;

[Description( "The occupant state of the attachment point.
There are three possible states for the occupant attachment
point: "None, Unconfigured, Configured"),
ValueMap{"0", "4", "5"},
Values{"None", "Unconfigured", "Configured"} ]
    uint32 OccupantState;

[Description("The condition state of the attachment point.
There are five different condition states for the attachment
point: "Unknown, OK, Failing, Failed, Unusable"),
ValueMap{"0", "1", "2", "3", "4"},
Values{"Unknown", "OK", "Failing", "Failed", "Unusable"} ]
    uint32 Condition;

[MaxLen(12), Description("The attachment point type.") ]
    string Type;

[MaxLen(4096), Description("The driver specific information.
This field contains the info string that the driver sets.
This property will be populated with a set of name-value
pairs.") ]
    string MiscInfo;

[MaxLen(4096), Description("The is the time at which the
Attachment Point was last updated.") ]
    datetime StatusTime;

[Override("InstallDate"), Description("This property's value
will always be NULL.") ]
    datetime InstallDate;

[Override("Name"), Description("This property's value will
always be NULL.") ]
    string Name;

[Override("Status"), Description("This property's value
will always be NULL.") ]
    string Status;

[Override("Caption"), Description("This property's value
will always be NULL.") ]
    string Caption;

```

```

[Override("Description"), Description("This property's
value will always be NULL.") ]
    string Description;

[Description ( "This method is used to bring the hardware
resources contained on, or attached to, an occupant into
the realm of Solaris, allowing use of the occupant's
hardware resources by the system.") ]
    sint32 Configure([IN] boolean force, [IN] string
hardwareOpts, [IN] uint32 retries, [IN] uint32
retryDelay, [OUT] string error);

[Description ("This method is used to remove the
hardware resources contained on, or attached to, an
occupant from the realm of Solaris, disallowing further use
of the occupant's hardware resources by the system.") ]
    sint32 Unconfigure([IN] boolean force, [IN] string
hardwareOpts, [IN] uint32 retries, [IN] uint32
retryDelay, [OUT] string error);

[Description("Change the receptacle state to connected.") ]
    sint32 Connect([IN] boolean force, [IN] string
hardwareOpts, [IN] uint32 retries, [IN] uint32
retryDelay, [OUT] string error);

[Description("This method is used to disable normal
communication to or from an occupant in a receptacle.") ]
    sint32 Disconnect([IN] boolean force, [IN] string
hardwareOpts, [IN] uint32 retries, [IN] uint32
retryDelay, [OUT] string error);

[Description ("This method is used to test an attachment
point. The test, used to evaluate the condition of the
attachment point, checks for hard faults. Note that the
receptacle state of the attachment point must be
disconnected to be tested. See cfgadm(1M).") ]
    sint32 Test( [IN] boolean verbose, [IN] string
hardwareOpts, [OUT] string error);

};

// =====
//     CIM Solaris_WDRDomain class
// =====

[Abstract, Description("This CIM Solaris_WDRDomain
represents a domain superclass for Starcat and Serengeti
domains."), Version("1.0") ]

```

```

class Solaris_WDRDomain : CIM_CollectionOfMSEs
{
    [Key, Description("This is the domain unique identifier on
the Starcat and Serengeti platforms. The domain identifier
will be a positive integer between 0 and 17 on the Starcat
and 0 and 4 on the Serengeti.") ]
        uint32 Id;

    [Override("CIM_ManagedElement.Caption"), Description("This
property's value will always be NULL.") ]
        string Caption;

    [Override("CIM_ManagedElement.Description"),
Description("This property's value will always be NULL.") ]
        string Description;

    [Override("CollectionID"), Description("This property's
value will always be NULL.") ]
        string CollectionID;
};

// =====
//      CIM Solaris_CHSystemBoard class
// =====

[Description("The CIM Solaris_CHSystemBoard class describes
the system board information on the Sun's enterprise system
that supports the NextGeneration Dynamic Reconfiguration
(NextGen DR)."),
Provider("com.sun.wbem.wdr.CHSystemBoardProvider"),
Version("1.0") ]

class Solaris_CHSystemBoard : Solaris_WDRAttachmentPoint
{
    [Description("Board assigned to the domain") ]
        boolean Assigned;

    [Description( "Board is powered-on") ]
        boolean PoweredOn;

    [Description( "Assign an available board to the domain.
This command requires the receptacle state of the board to
be Disconnected or Empty.") ]
        sint32 Assign([IN] boolean force, [IN] string
hardwareOpts, [OUT] string error);

```

```

[Description("Power off the board.  The receptacle state of
the board must be Disconnected.") ]
    sint32 PowerOff([IN] boolean force, [IN] string
    hardwareOpts, [OUT] string error);

[Description("Power on the board.  The receptacle state of
the board must be Disconnected.") ]
    sint32 PowerOn([IN] boolean force, [IN] string
    hardwareOpts, [OUT] string error);

[Description("Unassign a board from the domain.  An active,
(i.e. connected, or configured board may be unassigned.") ]
    sint32 Unassign([IN] boolean force, [IN] string
    hardwareOpts, [OUT] string error);

};

// =====
//     CIM Solaris_CHCPU class
// =====

[Description("The CIM Solaris_CHCPU class describes the
processor information available on Sun's enterprise
systems."), Provider("com.sun.wbem.wdr.CHCPUProvider"),
Version("1.0") ]

class Solaris_CHCPU : Solaris_WDRAttachmentPoint
{

[Description("The processor identifier.") ]
    uint32 ID;

[Units("MegaHertz"), Description("The speed of the
processor.") ]
    uint32 Speed;

[Units("MegaBytes"), Description("The ECache memory size of
the processor.") ]
    uint32 ECache;

};

// =====
//     CIM Solaris_CHMemory class
// =====

[Description("The CIM Solaris_CHMemory class describes the
memory information configured on Sun's enterprise systems
that supports the NextGeneration Dynamic Reconfiguration

```

```

        (NextGen DR)."),
        Provider("com.sun.wbem.wdr.CHMemoryProvider"),
        Version("1.0") ]

class Solaris_CHMemory : Solaris_WDRAttachmentPoint
{
    [Units("MegaBytes"), Description("When the memory drain is
in progress, this property stores the amount of already
deleted memory.") ]
        uint32 Deleted;

    [Description("True if the board is participating in
interleaving with other boards.") ]
        boolean Interleaved;

    [Units("KiloBytes"), Description("The property stores the
size of non-pageable memory in the board's memory.") ]
        uint32 Permanent;

    [octetstring, Description("The base physical address of
memory on the board.") ]
        uint64 PhysicalAddress;

    [Units("MegaBytes"),Description("When the memory drain is in
progress, this property stores the remaining memory needed
to be drained.") ]
        uint32 Remaining;

    [Units("MegaBytes"), Description("The board memory size.") ]
        uint32 Size;

    [Description( "When the memory drain is in progress, this
property stores the source system board attachment point
identifier.") ]
        string Source;

    [Description( "When the memory drain is in progress, this
property stores the target system board attachment point
identifier.") ]
        string Target;

    [Description("True if the operating system has been
configured to disallow this memory from being
unconfigured.") ]
        boolean Unconfigurable;
};

```

```

// =====
//      CIM Solaris_CHController class
// =====

[Description("The CIM CIM Solaris_CHController class models
the controller information configured in the Sun's
enterprise systems that supports the NextGeneration Dynamic
Configuration (NextGen DR)."),
Provider("com.sun.wbem.wdr.CHControllerProvider"),
Version("1.0") ]

class Solaris_CHController : Solaris_WDRAttachmentPoint
{
    [Description("The physical path of the IO component in
/devices.") ]
        string Device;

    [Description("True if the I/O component is referenced.") ]
        boolean Referenced;

};

// =====
//      CIM Solaris_WDRSlot class
// =====

[Abstract, Description("The CIM Solaris_WDRSlot is a
superclass class for the platform specific slot classes,
Solaris_XCSlot and Solaris_SGSlot. "),
Version("1.0") ]

class Solaris_WDRSlot : CIM_LogicalElement
{
    [Key, MaxLen(30), Description("The logical name of the slot
attachment point, (e.g SB0, IO15 for the Starcat or SB5, and
IB9 for the Serengeti).") ]
        string LogicalID;

    [Description( "Indicates whether this slot contains a board
or not. A NULL value for this property indicates the Empty
state of slot is unknown.") ]
        boolean Empty;

    [Override("InstallDate"),Description("This property's value
will always be NULL.") ]
        datetime InstallDate;
}

```

```

[Override("Name"), Description("This property's value will
always be NULL.") ]
    string Name;

[Override("Status"), Description("This property's value will
always be NULL.") ]
    string Status;

[Override("Caption"), Description("This property's value
will always be NULL.") ]
    string Caption;

[Override("Description"), Description("This property's value
will always be NULL.") ]
    string Description;

[Description("Assign the Slot to the specified domain.") ]
    sint32 Assign([IN] uint32 domainID, [OUT] string
error);

[Description("Unassign a board from the domain. This Slot
must not be active, (i.e. connected, or configured), in a
domain.") ]
    sint32 Unassign([IN] uint32 domainID, [OUT] string
error);

};

// =====
//      Associations
// =====

[Association, Aggregation,

Description("This CIM Relationship class is an aggregation
relationship between the CIM Solaris_CHSystemBoard instance
and the CIM Solaris_CHCPU instance."),
Provider("com.sun.wbem.wdr.BoardHasPartsProvider"),
Version("1.0") ]

class Solaris_SystemBoardHasProcessors : CIM_Component
{

[Override("GroupComponent"), Aggregate, Min(1), Max(1),
Description("This property references to the parent of the
relationship.") ]
    Solaris_CHSystemBoard ref GroupComponent;

```

```

        [Override("PartComponent"), Description("This property
        references the child of the relationship.") ]
            Solaris_CHCPU ref PartComponent;
    };

    [Association, Aggregation, Description("This CIM
    Relationship class is an aggregation relationship between
    the CIM Solaris_CHSystemBoard instance and the CIM
    Solaris_CHMemory instance."),
    Provider("com.sun.wbem.wdr.BoardHasPartsProvider"),
    Version("1.0") ]

class Solaris_SystemBoardHasMemory : CIM_Component
{
    [Override("GroupComponent"), Aggregate, Min(1), Max(1),
    Description("This property references to the parent of the
    relationship.") ]
        Solaris_CHSystemBoard ref GroupComponent;

    [Override("PartComponent"), Max(1), Description("This
    property references the child of the relationship.") ]
        Solaris_CHMemory ref PartComponent;
};

    [Association, Aggregation, Description("This CIM
    Relationship class is an aggregation relationship between
    the CIM Solaris_CHSystemBoard instance and the CIM
    Solaris_CHController instance."),
    Provider("com.sun.wbem.wdr.BoardHasPartsProvider"),
    Version("1.0") ]

class Solaris_SystemBoardHasControllers : CIM_Component
{
    [Override("GroupComponent"), Aggregate, Min(1), Max(1),
    Description("This property references to the parent of the
    relationship.") ]
        Solaris_CHSystemBoard ref GroupComponent;

    [Override("PartComponent"), Description("This property
    references the child of the relationship.") ]
        Solaris_CHController ref PartComponent;
};

```

```

[Association, Description("This CIM Relationship class is an
association relationship between the CIM Solaris_WDRSlot
instance and the CIM Solaris_CHSystemBoard instance."),
Provider("com.sun.wbem.wdr.SlotHasBoardProvider"),
Version("1.0") ]

class Solaris_SlotHasSystemBoard : CIM_Dependency
{
    [Override("Antecedent"), Min(1), Max(1), Description("This
property references to the parent of the relationship.") ]
        Solaris_WDRSlot REF Antecedent;

    [Override("Dependent"), Min(1), Max(1),Description("This
property references the child of the relationship.") ]
        Solaris_CHSystemBoard REF Dependent;
};

[Association, Aggregation, Description("This CIM
Relationship class is an aggregation relationship between a
CIM Solaris_WDRDomain instance and a set of CIM
Solaris_WDRSlots instances."),
Provider("com.sun.wbem.wdr.DomainHasSlotsProvider"),
Version("1.0") ]

class Solaris_DomainHasSlots: CIM_CollectedMSEs
{
    [Override("Collection"), Aggregate, Min(1), Max(1),
Description("This property references to the parent of the
relationship.") ]
        Solaris_WDRDomain REF Collection;

    [Override("Member"), Min(1), Max(1), Description("This
property references the child of the relationship.") ]
        Solaris_WDRSlot REF Member;
};

[Association, Aggregation, Description("This CIM
Relationship class is an aggregation relationship between a
CIM Solaris_WDRDomain instance and a set of CIM
Solaris_WDRAttachmentPoints instances."),
Provider("com.sun.wbem.wdr.DomainHasAttachmentPointsProvider
"),
Version("1.0") ]

class Solaris_DomainHasAttachmentPoints: CIM_CollectedMSEs
{

```

```

[Override("Collection"), Aggregate, Min(1), Max(1),
Description("This property references to the parent of the
relationship.") ]
    Solaris_WDRDomain REF Collection;

[Override("Member"), Min(1), Max(1), Description("This
property references the child of the relationship.") ]
    Solaris_WDRAttachmentPoint REF Member;

};

// =====
//      Indications
// =====
//      Solaris_WDRIndication indication
// =====

[Abstract, Indication, Description ("This indication class
serves as a common ancestor to all WBEM DR Indications. A
client can construct a filter using this class to subscribe
to all WBEM DR Indications."),
Version("1.0") ]

class Solaris_WDRIndication: CIM_ProcessIndication
{

};

```

WDR_SG1.0.mof ファイル

```

// =====

// Copyright (c) 2001 by Sun Microsystems, Inc. All rights
// reserved.

// Title:           WBEM Dynamic Reconfiguration (DR) Common
//                  // Information Model (CIM) Schema for the
//                  // SunFire 68x0, 48x0, and 3800

// Filename:       WDR_SG1.0.mof

// Author:         Sun Microsystems, Inc.

// Description:    This file contains CIM class and
//                  // association definitions for the WBEM
//                  // Dynamic Reconfiguration Model (CIM)

```

```

// Schema that are specific to the SunFire
// 68x0, 48x0, and 3800 platforms
// implementation. The WBEM DR CIM Schema
// models DR related operations and
// resources for the SunFire
// 15K/12K and SunFire 68x0, 48x0, and 3800
// platforms. The WDR_Core1.0.mof must be
// compiled before this file.

// @(#) WDR_SG1.0.mof 1.12@(#)

// =====
//     Pragmas
// =====

#pragma Locale ("en-US")

#pragma namespace ("root/cimv2")

// =====
//     CIM Solaris_SGDomain class
// =====

[Description("This CIM Solaris_SGDomain represents the
domain on the Serengeti platform."),
Provider("com.sun.wbem.wdr.SGDomainProvider"),Version("1.0")
]

class Solaris_SGDomain : Solaris_WDRDomain
{
    [Description("This property defines how a board is related
to this domain. The first 6 array positions relate to SB0
through SB5. The next 4 positions relate to IB6 through
IB9. Note that this applies for the Sun Fire 6800. The
Sun Fire 4810, 4800, and 3800 have only 5 system board
slots, (3 CPU boards and 2 I/O boards). For these models
the values of the array at indices SB3, SB4, SB5, IB8, and
IB9 will be 0, (i.e. Nonexistent Slot)."),
ValueMap {"0", "1", "2", "3", "4"},
Values {"Nonexistent Slot", "Not Available", "Available",
"Assigned", "Active"},
BitMap {"1", "2", "3", "4", "5", "6", "7", "8", "9", "10"},
BitValues {"SB0", "SB1", "SB2", "SB3", "SB4", "SB5", "IB6",
"IB7", "IB8", "IB9"} ]
    sint32 BoardRelationship[];

```

```

[Description("This property identifies the keyswitch
position of the virtual keyswitch.  The possible values and
their encodings are enumerated in the ValueMap and Values
qualifiers respectively."),
MappingStrings {"MIB.IETF | SUN-SC-MIB.domainKeySwitch"},
ValueMap {"1", "2", "3", "4", "5", "6", "7", "8", "9",
"10","11", "12", "13", "14", "15", "16"},
Values {"Unknown", "Off", "Standby", "On", "Diag",
"Secure", "Off To Standby", "Off To On", "Off To Diag",
"Off To Secure", "Standby To Off", "Active To Off", "Active
To Standby", "Reboot To On", "Reboot To Diag", "Reboot To
Secure"} ]
    uint32 KeyswitchPosition;

[Description("This is the current state of the domain."),
MappingStrings {"MIB.IETF | SUN-SC-MIB.domainStatus"},
ValueMap {"1", "2", "3", "4", "5", "6", "7", "8", "9",
"10", "11", "12", "13", "14"},
Values {"Unknown", "Running Post", "Standby", "Active",
"Powered Off", "Domain Idle", "Running OBP", "Booting",
"Running Solaris", "Halted", "Reset", "Panic", "Debugger",
"Hang Detected"} ]
    uint32 State;
};

// =====
//   CIM Solaris_SGSslot class
// =====

[Description("The CIM Solaris_SGSslot class represents the
expander board slots on a Serengeti platform which may or
may not contain various L1 system boards."),
Provider("com.sun.wbem.wdr.SGSslotProvider"),
Version("1.0") ]

class Solaris_SGSslot : Solaris_WDRslot
{
    [Description("The Domain to which this slot is assigned if
indeed it is assigned."),
MappingStrings {"MIB.IETF | SUN-SC-MIB.slotDomain"},
ValueMap {"-1", "0", "1", "2", "3"},
Values {"None", "A", "B", "C", "D"} ]
    sint32 AssignedDomain;
}

```

```

[Description("The current assignment state of the slot."),
MappingStrings {"MIB.IETF | SUN-SC-MIB.slotState"},
ValueMap {"1", "2", "3", "4"},
Values {"Unknown", "Free", "Assigned", "Active"} ]
    uint32 AssignmentState;

[Description("The type of board occupying the slot if the
slot is not empty."),
MappingStrings {"MIB.IETF | SUN-SC-MIB.slotBoardType"},
ValueMap {"1", "2", "3", "4", "5", "6", "7", "8", "9", "10",
"11"},
Values {"Unknown", "Empty", "CPU", "IO", "CPUWIB", "IOWIB",
"SC", "L2", "Fan", "Power Supply", "Logic Analyzer"} ]
    uint32 BoardType;

[Description("The power state of a board."),
MappingStrings {"MIB.IETF | SUN-SC-MIB.slotPowerStatus"},
ValueMap {"1", "2", "3", "4"},
Values {"Unknown", "On", "Off", "Failed"} ]
    uint32 PowerState;

[Description("The test state of a board."),
MappingStrings {"MIB.IETF | SUN-SC-MIB.slotTestStatus"},
ValueMap {"1", "2", "3", "4", "5", "6", "7", "8"},
Values {"Unknown", "Not Tested", "Passed", "Failed", "Under
Test", "Start Test", "Degraded", "Unusable"} ]
    uint32 TestState;

};

// =====
//      Indications
// =====

// Adapted from SC MIB Traps in SUN_SC_MIB.mib These
// indications are derived from a subset of SunFire SC SNMP
// Traps. Only SNMP traps of interest to WBEM DR are reported.
// Information is compiled from the SNMP trap and additional
// MIB queries.

// =====
//      Solaris_SGBoardPresenceChange indication
// =====

[Indication, Description ("CPU or IO Board becomes
present/absent from slot. Sent to platform and all domains
that have this slot in their Available Control List (ACL).
From SNMP Enterprise Trap sunFireEvents 6.1. Variables

```

```

        slotChassisIndex, slotIndex, slotBoardType."),
        Provider("com.sun.wbem.wdr.SGEventProvider"),
        Version("1.0") ]

class Solaris_SGBoardPresenceChange : Solaris_WDRIndication
{
    [MaxLen(30), Description("The logical name of the slot
attachment point, (e.g SB5, and IB9).") ]
        string LogicalID;

    [MaxLen(8), Description("The serial number of the chassis,
which is an eight-digit hex string. E.g., 10483D99.") ]
        string ChassisSerialNumber;

    [Description("The type of board occupying the slot if the
slot is not empty. Presently among boards only CPU and IO
boards are reported."),
MappingStrings {"MIB.IETF | SUN-SC-MIB.slotBoardType"},
ValueMap {"1", "2", "3", "4", "5", "6", "7", "8", "9",
"10", "11"},
Values {"Unknown", "Empty", "CPU", "IO", "CPUWIB", "IOWIB",
"SC", "L2", "Fan", "Power Supply", "Logic Analyzer"} ]
        uint32 BoardType;
};

// =====
// Solaris_SGSlotAssignmentChange indication
// =====

[Indication, Description ("A slot is assigned/unassigned to
this domain. Sent to the new domain in the event of an
assignment. Sent to the old domain in the event of an
unassignment. From SNMP Enterprise Trap sunFireEvents 6.2.
Variables domainIndex, slotChassisIndex, slotIndex,
slotState."),
Provider("com.sun.wbem.wdr.SGEventProvider"),
Version("1.0") ]

class Solaris_SGSlotAssignmentChange : Solaris_WDRIndication
{
    [MaxLen(30), Description("The logical name of the slot
attachment point, (e.g SB5, and IB9).") ]
        string LogicalID;

    [MaxLen(8), Description("The serial number of the chassis,
which is an eight-digit hex string. E.g., 10483D99.") ]
        string ChassisSerialNumber;
};

```

```

[Description("The Domain to which this slot is assigned if
indeed it is assigned."),
ValueMap {"-1", "0", "1", "2", "3"},
Values {"None", "A", "B", "C", "D"} ]
    sint32 AssignedDomain;

[Description("The current assignment state of the slot."),
MappingStrings {"MIB.IETF | SUN-SC-MIB.slotState"},
ValueMap {"1", "2", "3", "4"},
Values {"Unknown", "Free", "Assigned", "Active"} ]
    uint32 AssignmentState;
};

// =====
// Solaris_SGDomainAclChange indication
// =====

[Indication, Description ("The Available Control List (ACL)
for this domain has changed. Sent to the domain with the
ACL that changed. From SNMP Enterprise Trap sunFireEvents
6.3. Variables domainIndex, domainACLDescr."),
Provider("com.sun.wbem.wdr.SGEventProvider"),
Version("1.0") ]

class Solaris_SGDomainAclChange : Solaris_WDRIndication
{
    [Description("The domain the slot was assigned to or
unassigned from."),
ValueMap {"0", "1", "2", "3"},
Values {"A", "B", "C", "D"} ]
        uint32 DomainID;

    [Description("The list of slots available to the domain
identified by DomainID."),
Bitmap {"1", "2", "3", "4", "5", "6", "7", "8", "9", "10"},
BitValues {"SB0", "SB1", "SB2", "SB3", "SB4", "SB5", "IB6",
"IB7", "IB8", "IB9"} ]
        boolean AvailableBoards[];
};

// =====
// Solaris_SGBoardStateChange indication
// =====

[Indication, Description ("Indicates (i) if a board self
test has completed or (ii) if a board was powered on/off.
Sent to the platform and the domain that owns the board

```

```

    if any. From SNMP Enterprise Trap sunFireEvents 6.4.
    Variables slotChassisIndex, slotIndex, slotTestStatus,
    slotPowerStatus."),
    Provider("com.sun.wbem.wdr.SGEventProvider"),
    Version("1.0") ]

class Solaris_SGBoardStateChange : Solaris_WDRIndication
{
    [MaxLen(30), Description("The logical name of the slot
    attachment point, (e.g SB5, and IB9).") ]
    string LogicalID;

    [MaxLen(8), Description("The serial number of the chassis,
    which is an eight-digit hex string. E.g., 10483D99.") ]
    string ChassisSerialNumber;

    [Description("The power status of a board."),
    MappingStrings {"MIB.IETF | SUN-SC-MIB.slotPowerStatus"},
    ValueMap {"1", "2", "3", "4"},
    Values {"Unknown", "On", "Off", "Failed"} ]
    uint32 PowerState;

    [Description("The test status of a board."),
    MappingStrings {"MIB.IETF | SUN-SC-MIB.slotTestStatus"},
    ValueMap {"1", "2", "3", "4", "5", "6", "7", "8"},
    Values {"Unknown", "Not Tested", "Passed", "Failed", "Under
    Test", "Start Test", "Degraded", "Unusable"} ]
    uint32 TestState;
};

// =====
// Solaris_SGDomainStateChange indication
// =====

[Indication, Description ("Indicates when (i) domain goes up
or down, (ii) domain self test fails or (iii) the keyswitch
state of a domain has changed. Sent to the platform and the
domain who changed state. From SNMP Enterprise Trap
sunFireEvents 6.9. Variables domainIndex, domainStatus,
domainKeySwitch."),
Provider("com.sun.wbem.wdr.SGEventProvider"),
Version("1.0") ]

class Solaris_SGDomainStateChange : Solaris_WDRIndication
{

```

```

[Description("The domain which underwent a state change."),
ValueMap {"0", "1", "2", "3"},
Values {"A", "B", "C", "D"} ]
    uint32 DomainID;

[Description ("This property identifies the keyswitch
position of the virtual keyswitch. The possible values and
their encodings are enumerated in the ValueMap and Values
qualifiers, respectively."),
MappingStrings {"MIB.IETF | SUN-SC-MIB.domainKeySwitch"},
ValueMap {"1", "2", "3", "4", "5", "6", "7", "8", "9", "10",
"11", "12", "13", "14", "15", "16"},
Values{"Unknown", "Off", "Standby", "On", "Diag", "Secure",
"Off To Standby", "Off To On", "Off To Diag", "Off To
Secure", "Standby To Off", "Active To Off", "Active To
Standby", "Reboot To On", "Reboot To Diag", "Reboot To
Secure"} ]
    uint32 KeyswitchPosition;

[Description("This is the current state of the domain."),
MappingStrings {"MIB.IETF | SUN-SC-MIB.domainStatus"},
ValueMap {"1", "2", "3", "4", "5", "6", "7", "8", "9", "10",
"11", "12", "13", "14"},
Values {"Unknown", "Running Post", "Standby", "Active",
"Powered Off", "Domain Idle", "Running OBP",
"Booting", "Running Solaris", "Halted", "Reset", "Panic",
"Debugger", "Hang Detected"} ]
    uint32 State;
};

// =====
// Solaris_SGSslotAvailabilityChange indication
// =====

[Indication, Description("This trap indicates that the
availability for a slot has changed. Not currently sent.
From SNMP Enterprise Trap sunFireEvents 6.19. Variables
domainIndex, slotChassisIndex, slotIndex, slotState."),
Provider("com.sun.wbem.wdr.SGEventProvider"),
Version("1.0") ]

class Solaris_SGSslotAvailabilityChange : Solaris_WDRIndication
{

[MaxLen(30), Description ("The logical name of the slot
attachment point, (e.g SB5, and IB9).") ]
    string LogicalID;

```

```

[Description("The Domain to which this slot was assigned and
is now unassigned from or to which it is newly assigned."),
ValueMap {"-1", "0", "1", "2", "3"},
Values {"None", "A", "B", "C", "D"} ]
    sint32 AssignedDomain;

[Description("The current assignment state of the slot."),
MappingStrings {"MIB.IETF | SUN-SC-MIB.slotState"},
ValueMap {"1", "2", "3", "4"},
Values {"Unknown", "Free", "Assigned", "Active"} ]
    uint32 AssignmentState;

};

```

WDR_XC1.0.mof ファイル

```

=====
// Copyright (c) 2002 by Sun Microsystems, Inc.
// All rights reserved.

// Title:           WBEM Dynamic Reconfiguration (DR) Common
//                  // Information Model (CIM) Schema for the
//                  // SunFire 15K/12K

// Filename:        WDR_XC1.0.mof

// Author:          Sun Microsystems, Inc.

// Description: This file contains CIM class and association
//               // definitions for the WBEM Dynamic
//               // Reconfiguration Model (CIM) Schema that
//               // are specific to the SunFire 15K/12K
//               // platform implementation. The WBEM DR CIM
//               // Schema models DR related operations and
//               // resources for the Starcat and Serengeti
//               // platforms. The WDR_Core1.0.mof must be
//               // compiled before this file.

// @(#) WDR_XC1.0.mof 1.12@(#)

// =====
//      Pragmas
// =====

#pragma Locale ("en-US")

```

```

#pragma namespace ("root/cimv2")

// =====
//     CIM Solaris_XCDomain class
// =====

[Description("This CIM Solaris_XCDomain represents the
domain on the Starcat platform."),
Provider("com.sun.wbem.wdr.XCDomainProvider"),
Version("1.0") ]

class Solaris_XCDomain : Solaris_WDRDomain
{
    [Description("This property specifies which IO board has the
active ethernet for the internal SC network.") ]
    string ActiveEthernetBoard;

    [Description("This property contains the UNIX group name
assigned to the Domain Administrator Group.") ]
    string AdminGroup;

    [Description("This property defines how a board is related
to this domain. The first 18 array positions relate to SB0
through SB17. The next 18 positions relate to IB0 through
IB17."),
ValueMap {"1", "2", "3", "4"},
Values {"Not Available", "Available", "Assigned", "Active"},
BitMap {"1", "2", "3", "4", "5", "6", "7", "8", "9", "10",
"11", "12", "13", "14", "15", "16", "17", "18", "19", "20",
"21", "22", "23", "24", "25", "26", "27", "28", "29", "30",
"31", "32", "33", "34", "35", "36"},
BitValues {"SB0", "SB1", "SB2", "SB3", "SB4", "SB5", "SB6",
"SB7", "SB8", "SB9", "SB10", "SB11", "SB12", "SB13", "SB14",
"SB15", "SB16", "SB17", "IO0", "IO1", "IO2", "IO3", "IO4",
"IO5", "IO6", "IO7", "IO8", "IO9", "IO10", "IO11", "IO12",
"IO13", "IO14", "IO15", "IO16", "IO17"} ]
    sint32 BoardRelationship[];

    [Description("This property identifies the keyswitch
position of the virtual keyswitch. The possible values and
their encodings are enumerated in the ValueMap and Values
qualifiers, respectively."),
ValueMap {"0", "1", "2", "3", "4", "5"},
Values {"On", "Standby", "Off", "Diag", "Secure", "Unknown"}
]
    uint32 KeyswitchPosition;
}

```

```

    [Description("This is the current state of the domain."),
    ValueMap {"0", "1", "2", "3", "4", "5", "6", "7", "8", "9",
    "10", "11", "12", "13", "14", "15", "16", "17", "18", "19",
    "20", "21", "22", "23", "24", "25", "26", "27", "28", "29",
    "30", "31", "32", "33", "34", "35", "36"},
    Values {"Unknown", "Powered Off", "Keyswitch Standby",
    "Running Domain Post", "Running Board Post", "Layout OBP",
    "Loading OBP", "OBP Booting", "OBP Running", "OBP
    Callback", "OBP Loading Solaris", "OBP Booting Solaris",
    "OBP Domain Exited", "OBP Failed", "OBP In Sync Callback",
    "OBP Exited", "OBP Error Reset", "OBP Domain Halt", "OBP
    Environmental Domain Halt", "OBP Booting Solaris Failed",
    "OBP Loading Solaris Failed", "OBP Debug", "OS Running
    Solaris", "OS Quiesce In Progress", "OS Quiesced", "OS
    Resume In Progress", "OS Panic", "OS Panic Debug", "OS
    Panic Continue", "OS Panic Dump", "OS Halt", "OS Panic
    Exit", "OS Environmental Exit", "OS Debug", "OS Exit",
    "Domain Down", "Domain In Recovery"} ]
        uint32 State;

    [Description("This is the UNIX group ID assigned to
    represent the Domain Reconfiguration privileges.") ]
        string ReconfigGroup;
};

// =====
//     CIM Solaris_XCSlot class
// =====

    [Description("The CIM Solaris_XCSlot class represents the
    expander board slots on the Starcat platform which may or
    may not contain various L1 system boards."),
    Provider("com.sun.wbem.wdr.XCSlotProvider"),
    Version("1.0") ]

class Solaris_XCSlot : Solaris_WDRSlot
{
    [Description("The Domain to which this slot is assigned if
    indeed it is assigned."),
    ValueMap {"-1", "0", "1", "2", "3", "4", "5", "6", "7",
    "8", "9", "10", "11", "12", "13", "14", "15", "16", "17",
    "18"},
    Values {"None", "A", "B", "C", "D", "E", "F", "G", "H",
    "I", "J", "K", "L", "M", "N", "O", "N", "P", "Q", "R"} ]
        sint32 AssignedDomain;
};

```

```

        [Description("The current assignment state of the slot."),
        ValueMap {"0", "1", "2", "3"},
        Values {"Unknown", "Free", "Assigned", "Active"} ]
        uint32 AssignmentState;

        [Description("The type of board occupying the slot if the
        slot is not empty."),
        ValueMap {"0", "1", "2", "3", "4", "5", "6", "7", "8"},
        Values {"CPU", "WIB", "HPCI", "CPCI", "MCPU", "WPCI",
        "SPCI", "HPCIX", "Unknown"} ]
        uint32 BoardType;

        [Description("The power state of a board."),
        ValueMap {"0", "1", "2", "3"},
        Values {"Off", "On", "Minimal", "Unknown"} ]
        uint32 PowerState;

        [Description("The test state of a board."),
        ValueMap {"0", "1", "2", "3", "4"},
        Values {"Unknown", "iPOST", "Passed", "Degraded", "Failed"}
        ]
        uint32 TestState;

};

// =====
// Indications
// =====

// =====
// Solaris_XCSystemBoardConfigChange indication
// =====

[Indication, Description ("Indications of this type notify
the client that some SunFire 15K/12K system board
configuration property (or properties) has changed for a
specific system board."),
Provider("com.sun.wbem.wdr.XCEventProvider"),
Version("1.0") ]

class Solaris_XCSystemBoardConfigChange: Solaris_WDRIndication
{

        [Description("The system board whose configuration data has
        changed.") ]
        string LogicalID;

};

```

```

// =====
// Solaris_XCEnvironmentalIndication indication
// =====

[Abstract, Indication, Description("This abstract class
serves as a common ancestor to all environmental
indications on the SunFire 15K/12K."),
Version("1.0") ]

class Solaris_XCEnvironmentalIndication: Solaris_WDRIndication
{

    [Description("The component experiencing the environmental
event.") ]
        string ComponentID;

    [Description("If the component is an L1 board, (i.e. a
system board), this property will contain the corresponding
Field Replaceable Unit identifier, otherwise it will be
NULL.") ]
        uint32 FRUID;

};

// =====
// Solaris_XCComponentRemove indication
// =====

[Indication, Description("Indications of this type notify
the client when a specific hot-pluggable component is
removed from its slot on a SunFire 15K/12K."),
Provider("com.sun.wbem.wdr.XCEventProvider"),
Version("1.0") ]

class Solaris_XCComponentRemove :
Solaris_XCEnvironmentalIndication
{

};

// =====
// Solaris_XCComponentInsert indication
// =====

[Indication, Description("Indications of this type notify
the client when a specific hot-pluggable component is
inserted into its slot on a SunFire 15K/12K."),
Provider("com.sun.wbem.wdr.XCEventProvider"),
Version("1.0") ]

```

```

class Solaris_XCComponentInsert:
Solaris_XCEnvironmentalIndication
{
};

// =====
//   Solaris_XCBoardPowerOn indication
// =====

[Indication, Description("Indications of this type notify
the client when a specific system board is powered on."),
Provider("com.sun.wbem.wdr.XCEventProvider"),
Version("1.0") ]

class Solaris_XCBoardPowerOn: Solaris_XCEnvironmentalIndication
{
};

// =====
//   Solaris_XCBoardPowerOff indication
// =====

[Indication, Description("Indications of this type notify
the client when a specific system board is powered off."),
Provider("com.sun.wbem.wdr.XCEventProvider"),
Version("1.0") ]

class Solaris_XCBoardPowerOff:
Solaris_XCEnvironmentalIndication
{
};

// =====
//   Solaris_XCDomainIndication indication
// =====

[Abstract, Indication,Description("This abstract class
serves as a common ancestor to all domain indications on the
SunFire 15K/12K."),
Version("1.0") ]

class Solaris_XCDomainIndication: Solaris_WDRIndication
{

[Description("The domain experiencing the event.") ]
uint32 DomainID;

};

```

```

// =====
//   Solaris_XCDomainConfigChange indication
// =====

[Indication, Description("Indications of this type notify
the client that some SunFire 15K/12K domain configuration
property (or properties) has changed for a specific
domain."),
Provider("com.sun.wbem.wdr.XCEventProvider"),
Version("1.0") ]

class Solaris_XCDomainConfigChange: Solaris_XCDomainIndication
{
};

// =====
//   Solaris_XCDomainUp indication
// =====

[Indication, Description("Indications of this type notify
the client when a specific domain goes up. This occurs
when a domain is keyswitched on, or after the domain
monitoring daemon, DSMD, is restarted and finds that the
IOSRAM assigned to this domain is accessible."),
Provider("com.sun.wbem.wdr.XCEventProvider"),
Version("1.0") ]

class Solaris_XCDomainUp: Solaris_XCDomainIndication
{
};

// =====
//   Solaris_XCDomainDown indication
// =====

[Indication, Description("Indications of this type notify
the client when a specific domain goes down. This occurs
as when a domain is keyswitched to off or standby."),
Provider("com.sun.wbem.wdr.XCEventProvider"),
Version("1.0") ]

class Solaris_XCDomainDown: Solaris_XCDomainIndication
{
};

```

```

// =====
//   Solaris_XCDomainStop indication
// =====

[Indication, Description("Indications of this type notify
the client when a specific domain begins a hardware state
dump.  This occurs as a result of some non-recoverable
hardware failure and as a consequence the domain dumps its
state information to a dump file."),
Provider("com.sun.wbem.wdr.XCEventProvider"),
Version("1.0") ]

class Solaris_XCDomainStop: Solaris_XCDomainIndication
{
};

// =====
//   Solaris_XCDomainStateChange indication
// =====

[Indication, Description("Indications of this type notify
the client when a specific domain's state changes."),
Provider("com.sun.wbem.wdr.XCEventProvider"),
Version("1.0") ]

class Solaris_XCDomainStateChange: Solaris_XCDomainIndication
{
    [Description("The triple (Signature, State, SubState)
combine to describe the current state of the domain.") ]
    uint32 Signature;
    uint32 State;
    uint32 SubState;
};

```


索引

A

ACL (アクセス制御リスト)

Solaris_NamespaceAcl クラス, 27

Solaris_UserAcl クラス, 25

WBEM, 12, 20

API

アクセス権の設定, 24

AssociatorProvider

作成

例, 99, 108

C

CIM (Common Information Model (CIM))

リスナー

追加, 46

CIM (Common Information Model), 3, 8, 11

イベントモデル, 43

インジケーション

生成, 45

インジケーションクラス

CIM_IndicationFilter クラス, 47

CIM_IndicationHandler クラス, 49

関連, 82, 84

クラス

CIM_IndicationSubscription クラス, 51

集約, 82, 83, 85, 86

スロットクラス, 68

Solaris_SGSlot クラス, 74

Solaris_WDRSlot クラス, 68

Solaris_XCSlot クラス, 72

接続点クラス, 55

CIM Solaris_CHController クラス, 67

CIM Solaris_CHCPU クラス, 64

CIM Solaris_CHMemory クラス, 65

CIM Solaris_CHSystemBoard クラス, 60

CIM Solaris_WDRAttachmentPoint クラス, 55

ドメインクラス, 76

Solaris_SGDomain クラス, 80

Solaris_WDRDomain クラス, 76

Solaris_XCDomain クラス, 77

プロセスインジケーション, 87

クラス階層図, 54

CIMOM (CIM Object Manager), 8

D

DTMF (Distributed Management Task Force), 2, 3

E

EventProvider

作成

例, 99

I

InstanceProvider

作成

例, 99, 107

M

Managed Object Format (MOF)

ファイルのコンパイル, 15

MethodProvider

作成

例, 99, 109

Midframe Service Processor (MSP), 1

MOF (Managed Object Format)

CIM オブジェクト, 11

コンパイラ, 12, 13

ファイル, 4

mofcomp コマンドでコンパイル, 13

WDR, 111

WDR_Core1.0.mof, 111

WDR_SG1.0.mof, 122

WDR_XC1.0.mof, 130

mofcomp コマンド, 13

引数, 14

MSP, 1

R

Remote Method Invocation (RMI), 8

RMI (Remote Method Invocation), 8

S

SMC (Solaris Management Console)

WBEM ログビューア, 12, 13

SMC (Solaris Management Console) ユーザーツール,

12, 28

Solaris

インジケーションクラス階層, 88

Solaris RBAC (役割によるアクセス制御), 12

Solaris WBEM SDK (ソフトウェア開発キット), 9

Solaris WBEM Services, 7

Web サイト, 11

概要, 11

層, 12

ログファイル, 30

規則, 30

形式, 31

Solaris WBEM ログイングクラス, 31

Solaris_LogRecord クラス, 32

Solaris_LogService クラス, 32

Solaris WBEM ログイングサービス, 29

Solaris WBEM ログイングプロパティ

設定, 39

Solaris WBEM ログビューア, 41

Solaris WBEM ログファイル

データの読み取り, 36

Solaris_LogServiceProperties クラス, 39

Solaris_NamespaceAcl クラス

セキュリティ, 27

Solaris_UserAcl クラス, 25

ユーザーへのアクセス制御の設定に使用, 26

Sun Fire システム

WBEM DR をサポートするモデル, 1

Sun WBEM User Manager, 12, 21

起動, 22

W

WBEM

ACL (アクセス制御リスト), 12, 20

プロバイダ, 8

WBEM (Web-based Enterprise Management)

コンポーネント, 3

WBEM DR

サポートする Sun Fire システム, 1

WDR (WBEM dynamic reconfiguration)

サポート対象のシステム, 1

実行される操作, 4

ソフトウェア要件, 2

あ

アクセス制御リスト (ACL)

Solaris_NamespaceAcl クラス, 27

Solaris_UserAcl クラス, 25

WBEM, 12

アプリケーションプログラムインタフェース (API)

WBEM DR, 1

い

イベント, 43, 87

受信の申請, 45

待機, 46

ハンドラ

作成, 49

フィルタ

イベントハンドラへのバインド, 51

作成, 47

インジケーション, 43, 87, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98

クラス階層, 88

生成, 45

インジケーションクラス

CIM_IndicationFilter クラス, 47

CIM_IndicationHandler クラス, 49

CIM_IndicationSubscription クラス, 51

か

開発ツール

WBEM DR クライアントの開発に使用されるタイプ, ix

関連, 82, 84

く

クラス

Solaris インジケーション, 88

インジケーション

Solaris_SGBoardPresenceChange インジケーション, 89

Solaris_SGBoardStatusChange インジケーション, 92

Solaris_SGDomainACLChange インジケーション, 89

Solaris_SGDomainStateChange インジケーション, 90

Solaris_SGSlotAssignmentChange インジケーション, 91

Solaris_SGSlotAvailabilityChange インジケーション, 93

Solaris_XCBoardPowerOff インジケーション, 96

Solaris_XCBoardPowerOn インジケーション

, 96

Solaris_XCComponentInsert インジケーション, 96

Solaris_XCComponentRemove インジケーション, 95

Solaris_XCDomainConfigChange インジケーション, 97

Solaris_XCDomainDown インジケーション, 97

Solaris_XCDomainIndication インジケーション, 96

Solaris_XCDomainStateChange インジケーション, 98

Solaris_XCDomainStop インジケーション, 98

Solaris_XCDomainUp インジケーション, 97

Solaris_XCEnvironmentalIndication インジケーション, 95

Solaris_XCSystemBoardConfigChange インジケーション, 94

関連, 82

Solaris_SlotHasSystemBoard 関連, 84

集約, 82

Solaris_DomainHasSlots 集約, 83

Solaris_SystemBoardHasControllers 集約, 86

Solaris_SystemBoardHasMemory 集約, 85

Solaris_SystemBoardHasProcessors 集約, 85

スロット, 68

Solaris_SGSlot クラス, 74

Solaris_WDRSlot クラス, 68

Solaris_XCSlot クラス, 72

接続点, 55

CIM Solaris_CHController クラス, 67

CIM Solaris_CHCPU クラス, 64

CIM Solaris_CHMemory クラス, 65

CIM Solaris_CHSystemBoard クラス, 60

CIM Solaris_WDRAttachmentPoint クラス, 55

ドメイン, 76

Solaris_SGDomain クラス, 80

Solaris_WDRDomain クラス, 76

Solaris_XCDomain クラス, 77

こ

コントローラ, 86

コンポーネント

使用可能, 89

さ

サブスクリプション
イベント, 45

し

システムアーキテクチャー
プラットフォーム間の違い, 3

システムボード
情報の表示, 5
ドメインからの削除, 4
ドメイン間での移動, 4
ドメインへの追加, 4

集約, 82, 83, 85, 86

使用可能構成要素リスト, 6, 7, 82, 89

す

スロット, 83, 84, 91, 93
クラス, 68
Solaris_SGSlot クラス, 74
Solaris_Slot クラス, 68
Solaris_XCSlot クラス, 72

せ

セキュリティー, 12, 20, 25
Solaris_NamespaceAcl クラス, 27
Sun Fire 15K/12K および 6800/4810/4800/3800
システム, 5
アクセス制御の設定, 24
ネームスペースでのアクセス制御の設定, 27
ネームスペースへのアクセス権の削除, 24
ネームスペースへのアクセス権の設定, 23
ユーザーごとのアクセス制御の設定, 26
ユーザーのアクセス権の削除, 23
ユーザーのアクセス権の変更, 23
ユーザーへのデフォルトのアクセス権の許可
, 22

接続点

クラス, 55
CIM Solaris_AttachmentPoint クラス, 55
CIM Solaris_CHController クラス, 67
CIM Solaris_CHCPU クラス, 64
CIM Solaris_CHMemory クラス, 65
CIM Solaris_CHSystemBoard クラス, 60
ドメイン内のすべての一覧表示, 4

と

ドメイン, 83, 89, 90
クラス, 76
Solaris_SGDomain クラス, 80
Solaris_WDRDomain クラス, 76
Solaris_XCDomain クラス, 77

ね

ネームスペース
アクセス制御の設定, 27

は

ハンドラ
イベント
イベントフィルタへのバインド, 51
作成, 49

ふ

フィルタ
イベント
イベントハンドラへのバインド, 51
作成, 47
プログラミング手法, 99
プロセスインジケーション, 43
プロセッサ, 85

ほ

ボード, 84, 85, 86, 89, 92, 94, 95, 96, 97, 98

め

メモリー, 85

メモリー構成
情報の取り出し, 5

り

リスナー

CIM

追加, 46

ろ

ロギングサービス, 29, 30, 31, 32

Solaris WBEM ログビューア, 41

プロパティの設定, 39

ログファイルからのデータの読み取り, 36

