HP Universal CMDB

Windows および Solaris オペレーティング・システム用

ソフトウェア・バージョン : 8.04

ディスカバリおよび依存関係マップ

ドキュメント・リリース日 : 2010 年 3 月（英語版）
ソフトウェア・リリース日 : 2010 年 3 月（英語版）

利用条件

保証

HP の製品およびサービスの保証は，かかる製品およびサービスに付属する明示的な保証の声明において定められている保証に限ります。本ドキュメントの内容は，追加の保証を構成するものではありません。HP は，本ドキュメントに技術的な間違いまたは編集上の間違い，あるいは欠落があった場合でも責任を負わないものとします。

本ドキュメントに含まれる情報は，事前の予告なく変更されることがあります。

制限事項

本コンピュータ・ソフトウェアは，機密性があります。これらを所有，使用，または複製するには，HP からの有効なライセンスが必要です。FAR 12.211 および 12.212 に従って，商用コンピュータ・ソフトウェア，コンピュータ・ソフトウェアのドキュメント，および商用アイテムの技術データは，HP の標準商用ライセンス条件に基づいて米国政府にライセンスされています。

著作権

商標

Adobe® および Acrobat® は，Adobe Systems Incorporated の商標です。

Intel®，Pentium® および Intel® Xeon™ は，米国およびその他の国における Intel Corporation またはその子会社の商標または登録商標です。

Java™ は，Sun Microsystems, Inc. の米国商標です。

Microsoft®，Windows®，Windows NT®，および Windows® XP は，Microsoft Corporation の米国登録商標です。

Oracle® は，カリフォルニア州レッドウッド市の Oracle Corporation の米国登録商標です。

Unix® は The Open Group の登録商標です。

文書の更新

本書のタイトル・ページには，次の識別情報が含まれています。

- ソフトウェアのバージョンを示すソフトウェア・バージョン番号
- ドキュメントが更新されるたびに更新されるドキュメント発行日
- 本バージョンのソフトウェアをリリースした日付を示す，ソフトウェア・リリース日付

最新のアップデートまたはドキュメントの最新版を使用していることを確認するには，次の URL にアクセスしてください。

http://h20230.www2.hp.com/selfsolve/manuals

このサイトでは，HP Passport に登録してサインインする必要があります。HP Passport ID の登録は，次の場所で行います。

http://h20229.www2.hp.com/passport-registration.html

または，HP Passport のログイン・ページの［**New users - please register**］リンクをクリックしてください。

適切な製品サポート・サービスに登録すると，更新情報や最新情報も入手できます。詳細については HP の営業担当にお問い合わせください。

サポート

HP ソフトウェアのサポート Web サイトは，次の場所にあります。

http://support.openview.hp.com

この Web サイトでは，連絡先情報と，HP ソフトウェアが提供する製品，サービス，およびサポートについての詳細が掲載されています。

HP ソフトウェア・オンライン・ソフトウェア・サポートでは，お客様にセルフ・ソルブ機能を提供しています。HP ソフトウェアのオンライン・サポートでは，対話型の技術支援ツールに効率的にアクセスできます。有償サポートをご利用のお客様は，サポート・サイトの以下の機能をご利用いただけます。迅速かつ効率的にアクセスできます。有償サポートをご利用のお客様は，サポート・サイトの以下の機能をご利用いただけます。

- 関心のある内容の技術情報の検索
- サポート・ケースおよび機能強化要求の提出および追跡
- ソフトウェア・パッチのダウンロード
- サポート契約の管理
- HP サポートの連絡先の表示
- 利用可能なサービスに関する情報の確認
- ほかのソフトウェア顧客との議論に参加
- ソフトウェアのトレーニングに関する調査と登録

ほとんどのサポート・エリアでは，HP Passport ユーザとして登録し，ログインする必要があります。また，多くの場合，サポート契約も必要です。HP Passport ID の登録は，次の場所で行います。

http://h20229.www2.hp.com/passport-registration.html

アクセス・レベルの詳細に関しては次を参照してください。

http://h20230.www2.hp.com/new_access_levels.jsp

目次

第 II 部：はじめに

第 III 部：管理

第 IV 部 : コンテンツ記述

第 V 部 : ディスカバリおよび依存関係マップ・セキュリティ

目次

ようこそ

本書では，ディスカバリおよび依存関係マップ (DDM) Probe をインストール方法と，DDM プロセスを管理して IT インフラストラクチャのリソースとその相互依存関係を自動的に検出してマップする方法を説明します。DDM では，アプリケーション，データベース，ネットワーク・デバイス，さまざまなタイプのサーバなどのリソースを検出できます。DDM に関する高度な知識がある方のために，コンテンツ記述の項もあります。

DDM コンテンツを使った作業の詳細については，『Discovery and Dependency Mapping Content Guide（英語版）』を参照してください。

本章の内容

本書の構成

本書は，次の各章で構成されています。

第 I 部　ディスカバリおよび依存関係マップ Probe

DDM Probe のインストール方法について説明し，Probe がどのように動作するかについても説明しています。

第 II 部　はじめに

ディスカバリ実行，ディスカバリ・プローブ設定，ディスカバリ・リソースの管理，ステータス・スナップショット表示の各アプリケーションの主概念，タスク，参照先について説明します。

第 III 部　管理

DDM アプリケーションと HP Discovery and Dependency Mapping Web Service API を使用したディスカバリの管理方法について説明します(ディスカバリおよび依存関係マップの高度な知識を持ったユーザを対象としています)。

第 IV 部　コンテンツ記述

新しいディスカバリおよび依存関係マップ（DDM）コンテンツの開発（パターン記述とも呼ばれます）の手法，方法論，および慣例について説明します。また，多言語ロケール機能についても説明します。

第 V 部　ディスカバリおよび依存関係マップ・セキュリティ

本項では，ディスカバリおよび依存関係マップと DDM Probe を強化する方法について説明します。

対象読者

本書は，次の HP Universal CMDB 利用者を対象としています。

- HP Universal CMDB 管理者
- HP Universal CMDB プラットフォーム管理者
- HP Universal CMDB アプリケーション管理者
- HP Universal CMDB データ・コレクタ管理者

本書の読者は，エンタープライズ・システム管理に精通し，ITIL の概念を理解していること，そして HP Universal CMDB についての知識を備えている必要があります。

詳細情報の入手

HP Universal CMDB に含まれているすべての全オンライン・ドキュメントの一覧，その他のオンライン・リソース，最新版のドキュメントの入手情報，本書で使用する表記規則については，『**HP Universal CMDB デプロイメント・ガイド**』(PDF) を参照してください。

第 I 部

ディスカバリおよび依存関係マップ Probe

第 1 章

DDM Probe のインストール

本章では, Windows プラットフォームでのディスカバリおよび依存関係マップ (DDM) Probe のインストールに必要な手順について説明します。

注: インストールを開始する前に,『**HP Universal CMDB デプロイメント・ガイド**』(PDF) の「HP Universal CMDB の紹介」をよく読んでおくことを強くお勧めします。

本章の内容

タスク

参照先

DDM Probe のインストール

次に DDM Probe のインストール方法について説明します。

Probe のインストールは，HP Universal CMDB サーバをインストールする前でもインストールした後でも可能です。ただし，Probe のインストール時にはサーバ名を指定する必要があります。したがって，Probe をインストールする前にサーバをインストールしておくことをお勧めします。

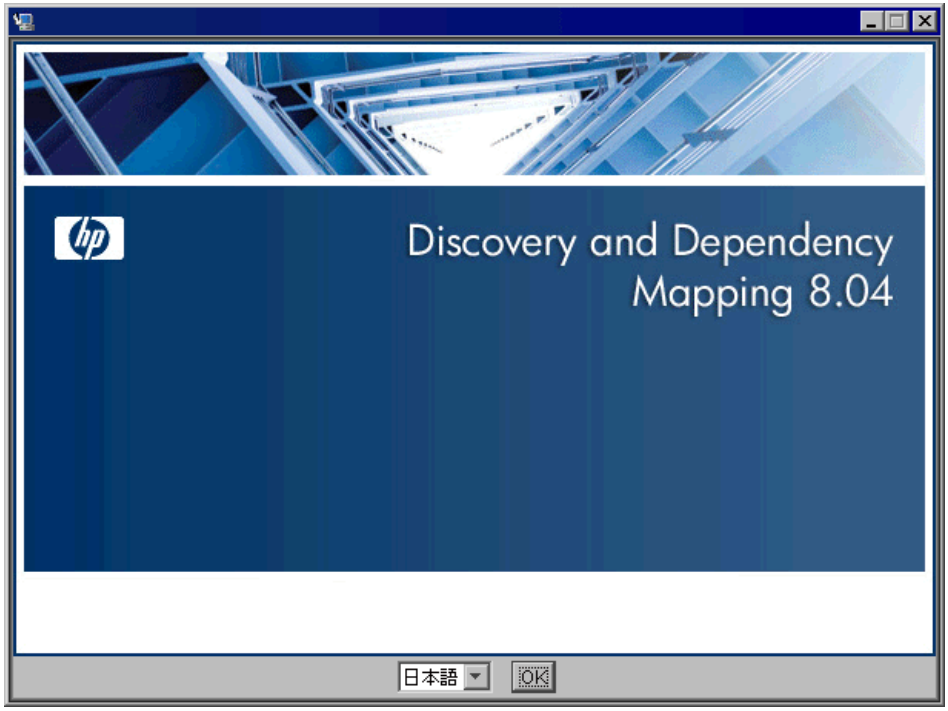
注 : ライセンスの詳細については，『**HP Universal CMDB デプロイメント・ガイド**』の「HP Universal CMDB のライセンス・モデル」を参照してください。

DDM Probe をインストールするには，次の手順を実行します。

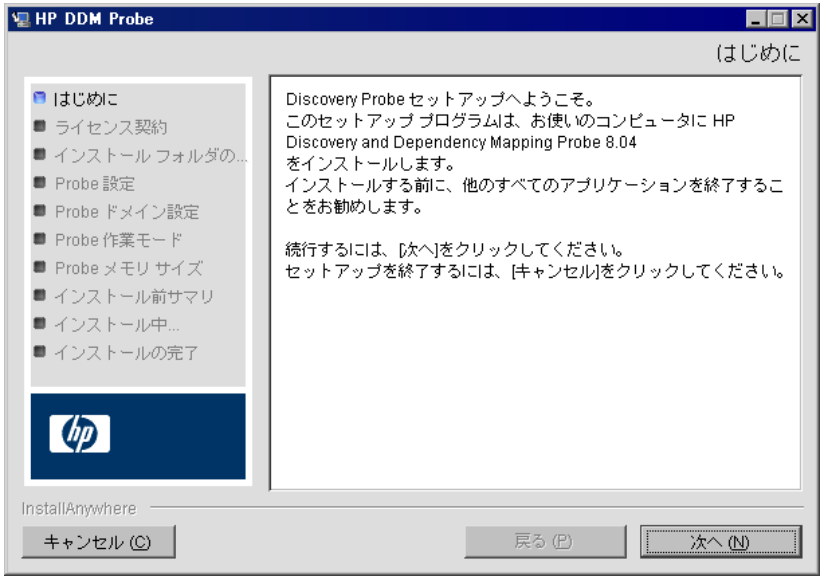
1 インストール元のドライブに **HP Universal CMDB 8.04 Setup Windows** DVD を挿入します。ネットワーク・ドライブからインストールする場合は，そのドライブに接続します。

2 **＜DVD ルート フォルダ＞¥UCMDB804¥HPDiscoveryProbe_v804_win32.exe** ファイルをダブルクリックします。

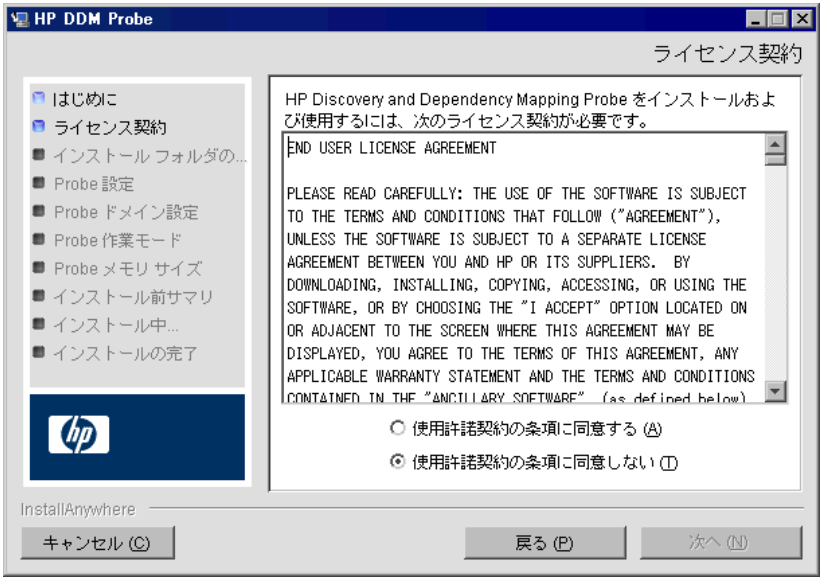
進行状況バーが表示されます。最初のプロセスが完了すると，スプラッシュ画面が開きます。



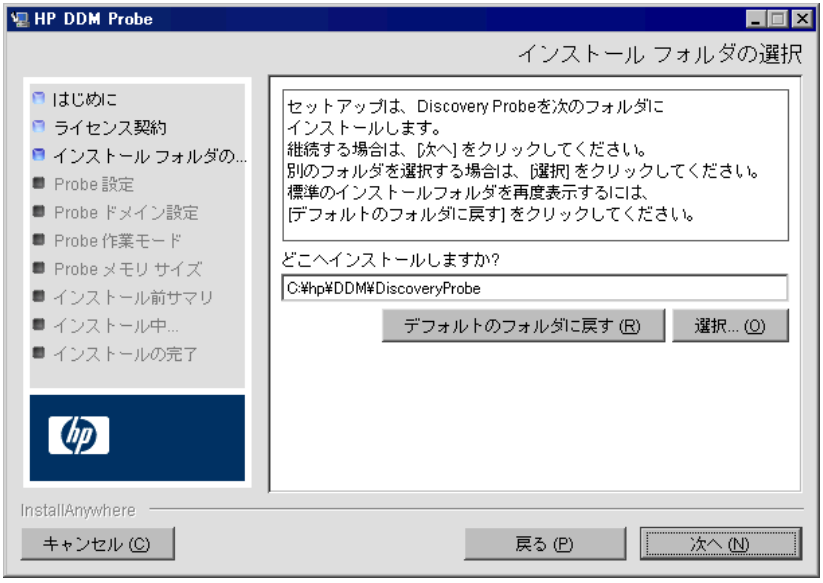
3 ロケール言語を選択し，[**OK**] をクリックします。[はじめに] ダイアログ・ボックスが開きます。



4 [**次へ**] をクリックすると [ライセンス契約] ダイアログ・ボックスが開きます。



5 契約の内容を受け入れて，［**次へ**］をクリックします。［インストール フォルダの選択］ダイアログ・ボックスが開きます。

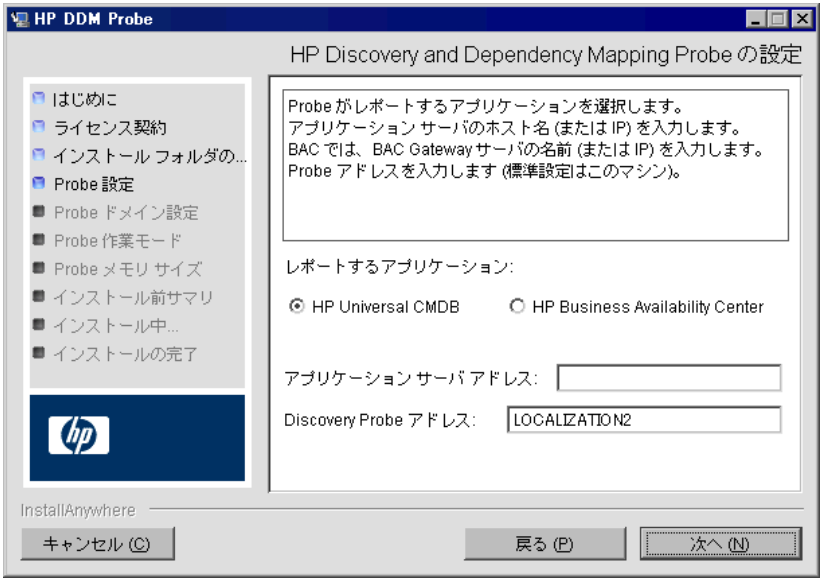


標準設定のインストール先をそのまま使用するか，［**選択**］をクリックして標準の［フォルダの参照］ダイアログ・ボックスを表示します。別のディレクトリにインストールするには，フォルダを参照してインストール・フォルダを選択します。

注：標準設定のインストール先ディレクトリに戻すには，［フォルダの参照］ダイアログ・ボックスでディレクトリを選択した後で［**デフォルトのフォルダに戻す**］をクリックします。

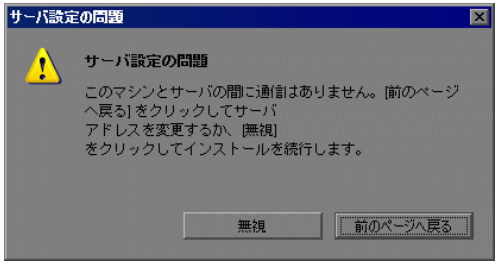
---

6 ［次へ］をクリックすると，［HP Discovery and Dependency Mapping Probe の設定］ダイアログ・ボックスが開きます。

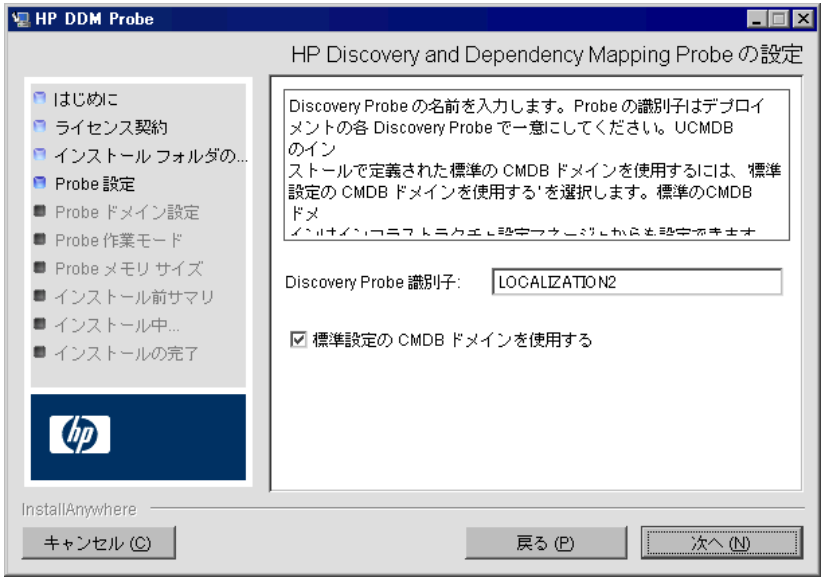


- ［**レポートするアプリケーション**］**:** 使用するアプリケーション・サーバを選択します。Probe は HP Universal CMDB または HP Business Availability Center とともに使用できます。
  - HPUniversal CMDB を選択した場合は，［**アプリケーション・サーバ・アドレス**］ボックスに，Probe の接続先の HPUniversal CMDB サーバの名前または IP アドレスを入力します。
  - HP Business Availability Center を選択した場合は，［**アプリケーション サーバ アドレス**］ボックスに，Gateway Server の IP または DNS 名を入力します。
- ［**Discovery Probe アドレス**］ボックスに，現在 Probe をインストールしているマシンの IP アドレスまたは DNS 名を入力するか，標準設定をそのまま使用します。

7 アプリケーション・サーバのアドレスを入力しないと，メッセージが表示されます。アドレスを入力せずに Probe のインストールを続行するか，前のページに戻ってアドレスを追加するかを選択できます。



8 ［次へ］をクリックすると，［HP Discovery and Dependency Mapping Probe の設定］ダイアログ・ボックスが開きます。

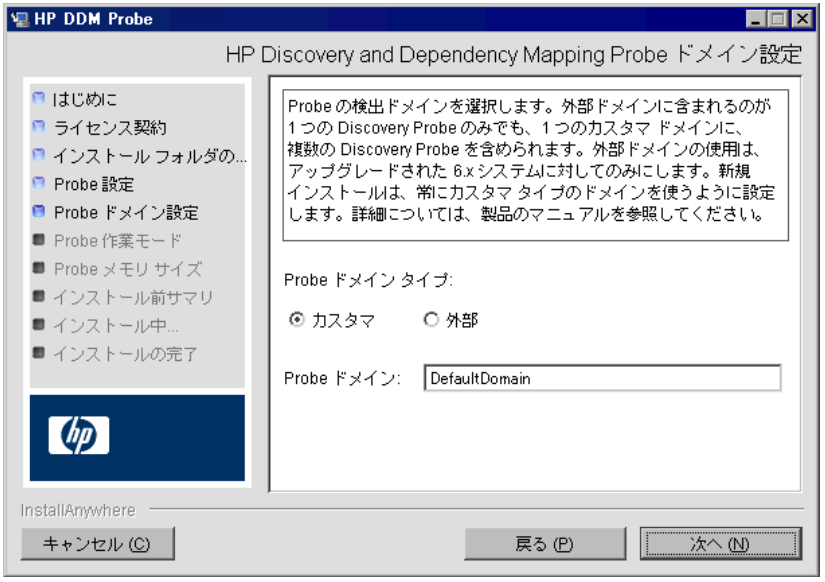


➤ ［**Discovery Probe 識別子**］ボックスに，お使いの環境で Probe を識別するための Probe 名を入力します。

重要：UCMDB Probe 識別子は，デプロイメント内の各 Probe に対して一意にする必要があります。

➤ UCMDB サーバのインストールで定義されている，標準設定の UCMDB の IP アドレスまたはマシン名を使用する場合は，［**標準設定の CMDB ドメインを使用する**］を選択します。標準設定の UCMDB ドメインは，インフラストラクチャ設定マネージャでも設定でき，HP Universal CMDB のインストール後に使用できます（［**Foundations**］＞［**CMDB**］＞［**CMDB クラス モデルの設定**］＞［**Default Domain Property Value**］）。

9 ［**次へ**］をクリックすると，［HP Discovery and Dependency Mapping Probe ドメイン設定］ダイアログ・ボックスが開きます。（このダイアログ・ボックスが表示されるのは，［HP Discovery and Dependency Mapping Probe の設定］ダイアログ・ボックスにある［**標準設定の CMDB ドメインを使用する**］ボックスをクリアした場合のみです）。

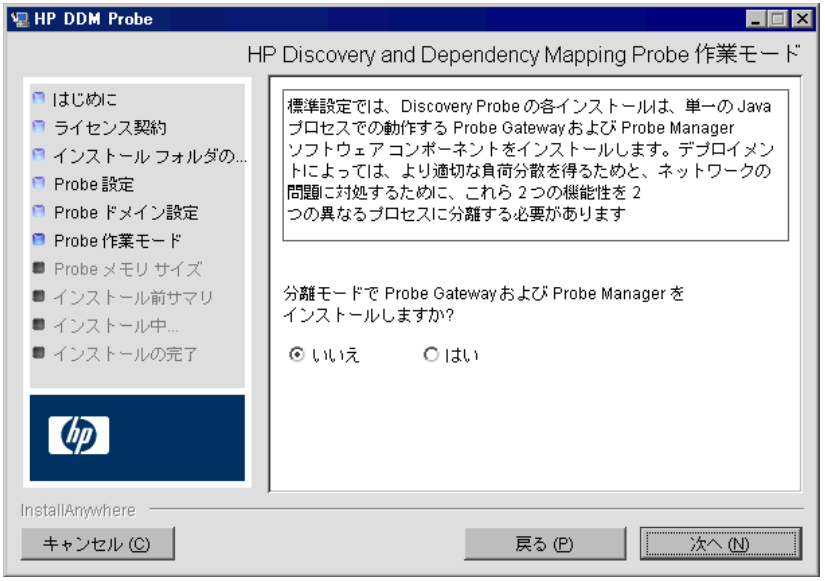


- Probe を実行するドメインのタイプに応じて［**カスタマ**］または［**外部**］を選択します。
  - **カスタマ：** デプロイメントに 1 つ以上の Probe をインストールする場合に選択します。

重要：新規インストールの場合は，常に［**カスタマ**］を選択します。

  - **外部：** バージョン 6.x システムからアップグレードする場合に選択します。
- **Probe ドメイン**：UCMDB サーバのドメインに Probe をインストールしない場合，ここにドメイン名を入力します。

10 ［**次へ**］をクリックすると，［HP Discovery and Dependency Mapping Probe 作業モード］ダイアログ・ボックスが開きます。



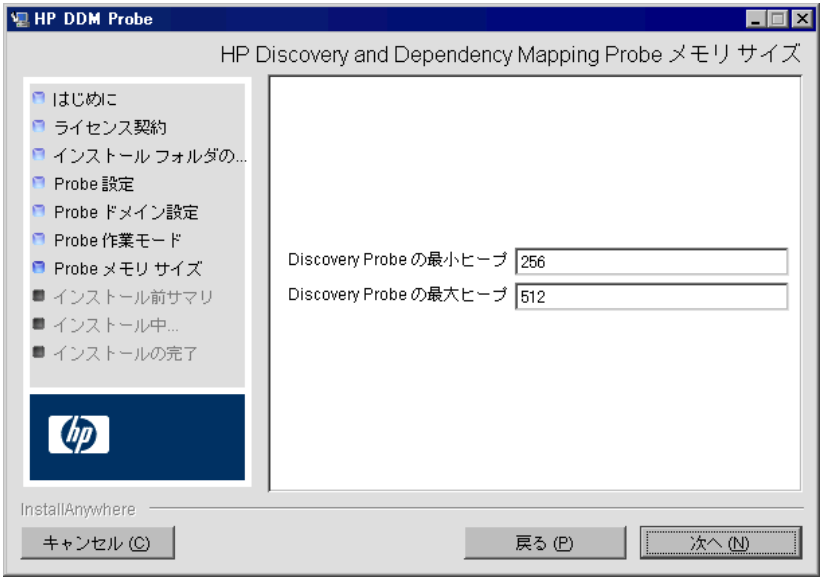
Probe Gateway と Probe Manager は，1 つの Java プロセスとしても別々のプロセスとしても実行できます。より効果的な負荷分散が必要で，ネットワーク問題を解消する必要があるデプロイメントでは，おそらく，別々のプロセスとして実行するのがよいでしょう。

Probe Gateway と Probe Manager を 1 つのプロセスとして実行するには，［**いいえ**］をクリックします。

Probe Gateway と Probe Manager を別々のプロセスとして実行するには，［**はい**］をクリックします。詳しい手順については，26 ページ「異なるマシンでの Probe Manager および Probe Gateway の実行」を参照してください。

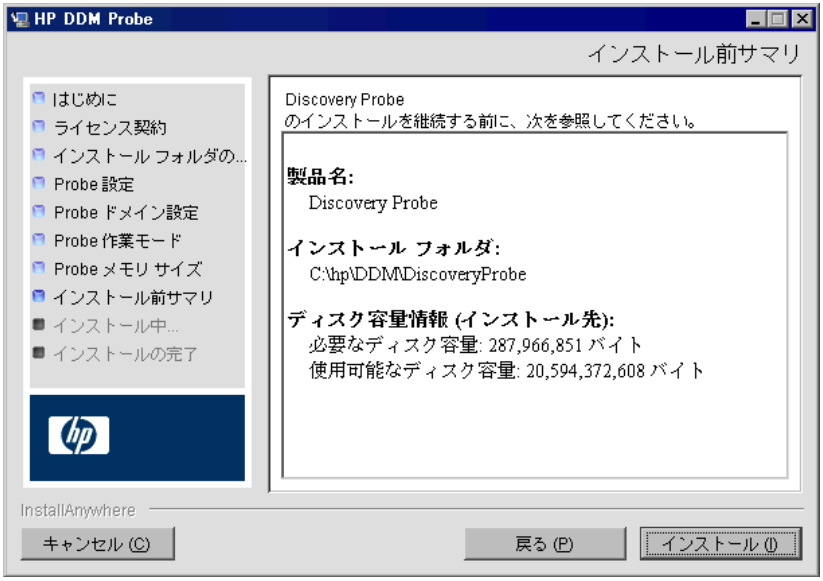
Probe Gateway および Probe Manager の詳細については，32 ページ「DDM Probe のタスク」を参照してください。

11 ［**次へ**］をクリックすると，［HP Discovery and Dependency Mapping Probe メモリ サイズ］ダイアログ・ボックスが開きます。



Probe に割り当てる最小および最大メモリ・サイズを定義します。値の単位はメガバイトです。

12 ［**次へ**］をクリックして［インストール前サマリ］ダイアログ・ボックスを開き，選択した内容を確認します。



13 ［**インストール**］をクリックして Probe のインストールを完了します。インストールが完了すると，［インストール完了］ページが表示されます。

注：インストール中に発生したエラーは **C:¥hp¥DDM¥DiscoveryProbe¥Discovery_Probe_InstallLog.log** ファイルに書き込まれます。

---

14 ［**完了**］をクリックします。Windows の［**スタート**］メニューに次のショート・カットが追加されます。

［**プログラム**］＞［**HP DDM**］＞［**DDM Probe**］

15 このショート・カットを選択して Probe をアクティブ化します。

HP Universal CMDB に Probe が表示されます。［**管理**］＞［**ディスカバリ**］＞［**ディスカバリ プローブ設定**］＞［**ドメインとプローブ**］にアクセスします。詳細については, 199 ページ「[ドメインとプローブ]表示枠」を参照してください。

Probe のアップグレード

本タスクでは，DDM Probe のアップグレード方法について説明します。

1 古い Probe のアンインストール

既存の Probe をすべてアンインストールします。Probe が実行中の場合は，停止してからアンインストールしてください。

2 新しい Probe のインストール

インストールの詳細については，16 ページ「DDM Probe のインストール」を参照してください。

注：

- 新しい Probe は同じ設定でインストールする必要があります。つまり，前回の Probe のインストールのときと同じ Probe ID，ドメイン名，サーバ名を使用します。
- 新しくインストールされた Probe が割り当てられたタスクを受信できるように，アップグレード後，アクティブなジョブを再アクティブ化する必要があります。

異なるマシンでの Probe Manager および Probe Gateway の実行

インストール時に，Probe Manager と Probe Gateway のプロセスを切り離して別々のマシンで実行されるように選択できます。それには，次の作業が必要になります。

1. 16 ページ「DDM Probe のインストール」の手順に従って，両方のマシンに Probe をインストールします。
2. 手順 23 ページ「10」で［**はい**］を選択します。
3. 27 ページ「Probe Manager コンポーネントおよび Probe Gateway コンポーネントの設定」の設定を行います。

Probe Manager コンポーネントおよび Probe Gateway コンポーネントの設定

本項では, Probe Manager と Probe Gateway を 2 つのマシンで別々のプロセスとして実行する場合の Probe の設定方法について説明します。

本項の内容

1 Probe Gateway マシンの設定

a 次のファイルを開きます。
C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥probeGateway¥probeMgrList.xml

b **<probeMgr ip>=** で始まる行を探し, Manager のマシン名を大文字で追加します。たとえば，次のようになります。

```
<probeMgr ip>=OLYMPICS08
```

c 次のファイルを開きます。
C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥DiscoveryProbe.properties

d **appilog.collectors.local.ip =** および **appilog.collectors.probe.ip =** で始まる行を探し，Gateway のマシン名を大文字で入力します。たとえば，次のようになります。

```
appilog.collectors.local.ip = STARS01
appilog.collectors.probe.ip = STARS01
```

2 Probe Manager マシンの設定

a 次の場所で設定します。
C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥DiscoveryProbe.properties appilog.collectors.local.ip = で始まる行を探し，Manager のマシン名を大文字で入力します。たとえば，次のようになります。

```
appilog.collectors.local.ip = OLYMPICS08
```

b **appilog.collectors.probe.ip =** で始まる行を探し，Gateway のマシン名を大文字で入力します。たとえば，次のようになります。

```
appilog.collectors.probe.ip = STARS01
```

3 サービスの開始

a Probe Manager マシンで Manager を起動します。［**スタート**］＞［**プログラム**］＞［**HP DDM**］＞［**DDM Manager**］を選択します。

b Probe Gateway マシンで Gateway を起動します。［**スタート**］＞［**プログラム**］＞［**HP DDM**］＞［**DDM Gateway**］を選択します。

Probe のインストール要件

ハードウェア要件

コンピュータ / プロセッサ	**Windows :** Pentium IV 2.4 GHz 以上のプロセッサ
メモリ	**Windows :** 1 GB 以上の RAM（推奨 : 2 GB RAM）
仮想メモリ（Windows デプロイメントの場合）	2 GB 以上 **注 :** 仮想メモリ・サイズは，必ず物理メモリ・サイズの 2 倍以上必要です。
ハード・ディスク・ドライブの空き領域	**Windows :** 4 GB 以上（データベース・ソフトウェアおよびデータ・ファイル用に 4 GB 以上）（推奨 : 20 GB ハード・ディスク）
ディスプレイ	**Windows :** 最低 256 色以上のカラー・パレット設定（32,000 色表示を推奨）
仮想マシンへのインストール	DDM Probe の仮想マシンへのインストールは，UCMDB バージョン 8.00 以降でサポートされています。

ソフトウェア要件

オペレーティング・システム	**Windows:** ➤ Windows 2000 Server/Advanced Server，Service Pack 4 以降 ➤ Windows 2003 Standard/Enterprise Edition，Service Pack 1，Service Pack 2
Java 実行環境	JRE 1.5.0（本製品とともにインストール）

第 2 章

DDM Probe の概要

本章では，DDM Probe について説明します。

本章の内容

概念

タスク

参照先

DDM Probe のタスク

本項では，DDM Probe でタスクを管理する方法について説明します。

DDM Probe は，Probe Gateway と Probe Manager という 2 つのコンポーネントで構成されています。

- Probe Gateway は，タスクをダウンロードしたり，タスク結果を返したりする処理を実行するときに，HTTP または HTTPS によってサーバと通信します。接続は常に Probe Gateway によって開始されます。
- Probe Manager は DDM プロセスそのものを実行します。

Probe Gateway は，RMI を使用して Probe Manager と通信します。

標準設定では，Probe Gateway と Probe Manager は単一プロセスとして動作しますが，別々のプロセスとして存在するように（インストール時に）設定することもできます。詳細については，「DDM Probe のインストール」の手順 23 ページの手順 10 を参照してください。また，複数の Probe Manager を 1 つの Probe Gateway に接続するように設定することもできます。これは，一定の DDM ジョブを特定の Probe Manager で処理する必要がある場合に便利です。

本項の内容

ステージ 1. Probe Gateway

HP Universal CMDB サーバが Probe 上のタスクを開始するわけではなく，必要なタスクの実行を要求するのは Probe の役目です。

ステージ 2. HP Universal CMDB サーバ

Probe は，サーバにタスクを要求するのと同時に，設定の最終更新時刻および最後に受信したタスク ID をサーバに送信します。サーバは次のいずれかを Probe に返します。

- 更新されたサーバ・データ（ Probe 上の設定が最新のものではない場合）。サーバ・データには Python スクリプト，パターン，ドメイン・スコープ・ドキュメント辞書ファイルなどが含まれます。詳細については，36 ページ「Probe の設定の更新」を参照してください。ドメイン・スコープ・ドキュメントを使用して DDM を強化する方法の詳細については，第 12 章:「DDM のセキュリティ強化」を参照してください。
- 最後に送信したタスク（ Probe とサーバの最後のタスク ID が一致しない場合）
- 実行すべき新規タスク（存在する場合）
  - ジョブが非アクティブ化されている場合，サーバは Probe に **delete job** メッセージを送信します。
  - ジョブがアクティブ化されている場合，サーバは Probe に **run new job** メッセージを送信します。
- HP Universal CMDB サーバは，新規タスク・データとともに Probe に応答（XML 形式）を送信します。各タスクには，ジョブおよびパターンの名前，関連するトリガ CI データが含まれます。

  1 つのタスクに対するトリガ CI の数には制限があります（標準設定で 100 個）。たとえば，アクティブ・ジョブに 1000 個の宛先が含まれている場合，そのジョブは，それぞれ 100 個のトリガ CI が含まれる 10 個のタスクとなって Probe に送信されます。

ステージ 3. Probe Gateway

- Probe Gateway は，HP Universal CMDB サーバからタスクを受信すると，そのタスクをローカル・データベース（MySQL）に保存します。
- Probe Gateway のスレッドは，定期的に，タスクがないかデータベースをスキャンして，タスクを Probe Manager に送信します。このプロセスにより，各 Probe Gateway に複数の Probe Manager が存在する場合でも，DDM の負荷分散が可能になります。

ステージ 4. Probe Manager

- Probe Manager のタスクは，Quartz というサード・パーティ・ライブラリを使用してスケジュール設定されます。タスクが完了すると，Probe Manager は結果（XML 形式）を Probe Gateway に送信します（Quartz の詳細については，http://www.opensymphony.com/quartz/ のドキュメントを参照してください）。
- Probe Manager は一連の結果オブジェクトを受信します。Probe Manager は，最初に結果に対して処理を行い（たとえば，結果のフィルタ処理や結果重複メカニズムの実行など），次に結果を Probe Gateway に送信する準備を行います。
- 結果は Probe Manager データベースに格納されます。
- スレッドは，Probe Gateway への送信準備が整った結果がないかデータベースをスキャンします。これらの結果は 1 つに結合されます。このときサイズは，discoveryProbe.properties ファイルで次のように定義される最大結果サイズ（現在は 20,000）を超えないように処理されます。

```
appilog.agent.local.maxTaskResultSize = 20000
```

  結果が Probe Gateway に到達すると，Probe Gateway は直ちに成功か失敗かの応答を返します。Probe Manager は，Probe Gateway からのこの受信確認に基づいてデータベース内の結果に **ack** とマークを付け，次のサイクルで結果が再び送信されないようにします。
- Probe Gateway で受信確認されたタスク結果は，週に 1 度の削除が行われるまで Probe Manager データベースに保持されます。
- 結果が Probe Gateway に到達したとき，その結果は直接サーバに送信されるのではなく，サーバがデータであふれるのを避けるために Probe Gateway データベースに格納されます。

ステージ 5. Probe Gateway

- Probe Gateway の専用スレッドは，データベースをスキャンして，サーバへの送信準備が整ったタスク結果を検索します。この結果は，Probe Gateway によって **sendResultsToServer()** API を介してサーバに送信されます。
- 送信する必要があるデータのサイズが非常に大きい場合は，チャンク（最大 50,000）で送信されます。それから，CMDB の情報が（作成，更新，削除によって）更新されます。

- Probe Gateway は最後に，サーバが結果の処理を完了したか確認し，（結果が再びサーバに送信されないように）結果をデータベースから削除します。結果がまだ存在する場合は，サーバへの結果の送信を続行します。

ステージ 6. Probe とサーバの同期プロセス

Probe は, 既定のいくつかのタスクの読み取り後, それらのタスクについてサーバと確認を行います（このプロセスにより，パターンまたはジョブを手動で再アクティブ化する必要がなくなります)。

- Probe は，アクティブ化されているすべてのジョブの名前，および，各ジョブのトリガ CI の数をサーバに送信します。
- サーバは，その数が CMDB の数と一致するか調べます。
  - Probe からジョブが失われている場合, サーバは Probe にジョブを再度送信します。
  - Probe の CI 数がサーバの CI 数より少ないまたは多い場合，サーバは問題のあるジョブの名前と CI を Probe に返します。
- Probe は問題のあるジョブのリストを調べます。サーバに存在しないジョブが Probe に含まれていた場合，Probe はすべての Probe Manager に **remove job** リモート・メソッド呼び出しを送信します。
- サーバに存在しない CI が Probe に含まれていた場合, Probe はすべての Probe Manager に **remove CI** リモート・メソッド呼び出しを送信します。
- Probe に存在しないトリガ CI がサーバに含まれていた場合，Probe はその CI をサーバに要求します。サーバはタスク（XML 形式）を返し，Probe はそのタスクを配布します。

Probe の設定の更新

- ディスカバリを実行するには，ドメイン・スコープ・ドキュメント辞書ファイルやスクリプトなどのリソース・データが Probe に必要となります。Probe は，これらのリソースによって自動的に更新されます。Probe Gateway は，Probe Gateway からサーバへの各タスク要求とともに，最新の更新リソースの最終（サーバ）更新時刻を送信します。
- サーバは，新しいタスクを返す前に，最近更新されたリソースがもうないかを確認します。存在する場合，サーバは，キューに入れられた Probe の通常のタスクを返す代わりに, Probe のリソースを更新するための巧みに作られた特別なタスクを返します。
- Probe は, このタスクを受信すると, **GetResouces()** 要求をサーバに送信します。この要求は，更新されていないリソースのリストを Probe に返すというものです。このようにして, Probe は常に, 最新のシステム構成ファイルで更新されます。

DDM Probe でのデータ検証

バージョン 7.0 から DDM Probe にも CIT モデルが存在するようになりました。これにより，サービスからのデータ受信時に Probe でデータ検証を行えるようになります。問題は，特定のトリガ CI に対して発生し，ユーザに表示されます。詳細については，132 ページ「[ディスカバリ ステータス] 表示枠」を参照してください。

Probe では次の検証が行われます。

- CI の CIT を CIT モデルの CIT と比較します。
- キー属性がすべて存在するか検証するために CI を調べます（CmdbObjectId 属性が定義されていない場合）。
- CI の属性がすべて CIT に定義されているか検証するために CI の属性を調べます。
- STRING タイプの CI の属性がサイズ制限を超えていないか検証するために，STRING タイプの CI の属性を調べます。属性がサイズ制限を超えている場合，DDM はその属性に AUTO_TRUNCATE 修飾子が定義されているかどうか調べます。この修飾子があると，値が切り捨てられ，Probe の error.log ファイルに警告メッセージが書き込まれます。

無効な属性はすべて CollectorsProcessException 例外を発生させます。この例外は特定の CI について報告します。CIT に関連する無効なデータを Probe が発見した場合, その CI について Probe が収集したすべてのデータは Probe によって削除され，サーバには送信されません。

属性の詳細については,『**モデル管理**』の「CI タイプの属性」を参照してください。

DDM Probe のセキュリティ強化

DDM Probe のセキュリティ強化の詳細については，第 13 章:「DDM Probe のセキュリティ強化」を参照してください。

ドメイン・スコープ・ドキュメントのセキュリティ強化の詳細については，411 ページ「domainScopeDocument ファイルの場所の制御」を参照してください。

DDM Probe で SSL を有効にする方法の詳細については, 399 ページ「UCMDB サーバと DDM Probe の間の相互認証による SSL の有効化」を参照してください。

結果のフィルタ処理

Probe から HP Universal CMDB サーバに送信される結果はフィルタ処理できます。おそらく，関係のないデータは，実運用実行時に定期的にフィルタ処理する必要が生じます。特に，限定的な環境についてテストするときはその必要があります。

フィルタ処理には，パターン・フィルタリングとグローバル・フィルタリングという 2 つのレベルがあります。

- **パターン・フィルタリング**: DDM は，特定のパターンの結果をフィルタ処理し，そのフィルタ処理された CI だけを CMDB に送信します。パターン・フィルタは，［パターン管理］タブの［結果管理］表示枠で定義します。詳細については，258 ページ「［パターン管理］タブ」を参照してください。
- **グローバル・フィルタリング**: DDM は，Probe で実行されたすべてのジョブの結果をフィルタ処理します。グローバル・フィルタは globalFiltering.xml ファイルで定義します。詳細については，236 ページ「globalFiltering.xml」を参照してください。

  フィルタ処理の順序は次のとおりです。実行時，DDM は最初にパターン・フィルタを検索し，そのフィルタを実行結果に適用します。パターン・フィルタがないとき，DDM はグローバル・フィルタを検索して結果に適用します。フィルタが何も見つからなかったときは，すべての結果がサーバに送信されます。

DDM Probe のインストール

本項では，DDM Probe をインストールおよび起動する方法について説明します。

注：管理対象環境はドメインの IP 範囲で定義します。ただし，いくつかのパターンを使用すれば，この動作をオーバライドし，Probe の範囲外の CI を検出できます。

このタスクには次の手順が含まれます。

- 39 ページ「Probe のインストール」
- 39 ページ「HP Universal CMDB の起動」
- 39 ページ「[スタート] メニューからの Probe の起動」
- 40 ページ「Probe のサービスとしての起動」
- 40 ページ「ディスカバリおよび依存関係マップの実行」
- 40 ページ「Probe の停止」

Probe のインストール

詳細については，第 1 章：「DDM Probe のインストール」を参照してください。

HP Universal CMDB の起動

詳細については，『**HP Universal CMDB デプロイメント・ガイド**』（PDF）の「HP Universal CMDB への最初のログイン」を参照してください。

[スタート] メニューからの Probe の起動

Probe を起動するには，Probe がインストールされているマシンで［**スタート**］＞［**プログラム**］＞［**HP DDM**］＞［**DDM Probe**］を選択します。コマンド・プロンプト・ウィンドウが開きます。Probe が正常に起動したことを確認するには，HP Universal CMDB で，［**管理**］＞［**ディスカバリ**］＞［**ディスカバリ プローブ設定**］を選択します。Probe を選択し，［詳細］表示枠でステータスが［**接続されました**］になっていることを確認します。

Probe の仕組みの詳細については，32 ページ「DDM Probe のタスク」を参照してください。

Probe のサービスとしての起動

Probe は，サービスとして自動的に開始するように設定できます。この場合，コマンド・プロンプト・ウィンドウは開きません。

Microsoft の［**サービス**］ウィンドウにアクセスし，［**DDM_Probe**］サービスを見つけます。［**DDM_Probe のプロパティ**］ダイアログ・ボックスを開き，サービスを開始します。必要に応じて［スタートアップの種類］を［**自動**］に変更してください。

注：Probe サービスを実行するユーザは，管理者グループのメンバである必要があります。

ディスカバリおよび依存関係マップの実行

詳細については，90 ページ「ディスカバリ実行 – 概要」を参照してください。

Probe の停止

- コマンド・プロンプト・ウィンドウで実行されているときに Probe を停止するには，CTRL キーを押しながら C キーを押し，次に **y** キーを押します。
- サービスとして実行されているときに Probe を停止するには，Microsoft の［サービス］ダイアログ・ボックスにアクセスします。DDM_Probe サービスを見つけて，［**サービスの停止**］リンクをクリックします。

DiscoveryProbe.properties ファイル

DDM プロセスでは，いくつかのパラメータをアクティブ化する必要があります。これらのパラメータによって，使用する方法（たとえば，失敗を宣言する前に 5 回 ping を行うなど），および方法を実行する対象の CI を指定します。ユーザがパラメータを定義していない場合は，**DiscoveryProbe.properties** ファイルに定義された標準設定のパラメータが使用されます。パラメータを編集するには，テキスト・エディタで **DiscoveryProbe.properties** を開きます。

DiscoveryProbe.properties ファイルは **C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors** にあります。

注：DiscoveryProbe.properties ファイルのパラメータを更新したときは，変更が反映されるように Probe を再起動する必要があります。

DiscoveryProbe.properties ファイルは次のセクションに分かれています。

- **Server Connection Definitions:** 使用するプロトコル，マシン名，標準設定の Probe およびドメインの名前，タイムアウト，基本認証など，サーバと Probe の接続を確立するのに必要なパラメータが含まれます。
- **DDM Probe Definitions:** ルート・フォルダの場所，ポート，Probe Manager および Probe Gateway のアドレスなど, Probe を定義するパラメータが含まれます。
- **Probe Gateway Configurations:** データを取得する時間間隔を定義するパラメータが含まれます。
- **Probe Manager Configurations:** スケジュール設定の間隔，結果のグループ化，チャンク，スレッド，タイムアウト，フィルタ処理など，Probe Manager 機能を定義するパラメータが含まれます。
- **I18N Parameters:** 言語設定を定義するパラメータが含まれます。
- **Internal Configurations:**（**注**：このパラメータは，ディスカバリおよび依存関係マップに関する高度な知識がないときは変更しないでください。）スレッド・プール・サイズなど，DDM を効率的に機能させることができるパラメータが含まれます。

トラブルシューティングと制限事項

問題 : あるドメインから別のドメインに DDM Probe を移すことは出来ません。Probe のドメインを定義した後は，範囲を変更できますが，ドメインは変更できません。

解決策 : Probe を再インストールします。

1 （任意）新規ドメインで同じ範囲を使用する場合は，既存の Probe を削除する前に範囲をエクスポートします。詳細については，197 ページ「［範囲］表示枠」を参照してください。

2 UCMDB から既存の Probe を削除します。詳細については，199 ページ「［ドメインとプローブ］表示枠」の［**ドメインまたはプローブの削除**］ボタンの説明を参照してください。

3 Probe をインストールします。詳細については，第 1 章 :「DDM Probe のインストール」を参照してください。

インストール時には必ず，古い Probe で使用したものとは異なる名前を Probe に付けてください。詳細については，21 ページの手順 8 を参照してください。

第 II 部

はじめに

第 3 章

ディスカバリおよび依存関係マップの紹介

本章の内容

概念

タスク

参照先

ディスカバリおよび依存関係マップ ー 概要

ディスカバリおよび依存関係マップ (DDM) プロセスは，IT インフラストラクチャのリソースとそれらの相互依存関係を検出することで，システムに関する情報を収集できるメカニズムです。DDM は自動的に，論理アプリケーション・アセットを検出して OSI（Open System Interconnection: 開放型システム間相互接続）モデルのレイヤ 2 ～ 7 にマップします。

DDM は，アプリケーション，データベース，ネットワーク・デバイス，サーバなどのリソースを検出します。また，業界標準 API やアプリケーション API との通信も行います。検出された各 IT リソースは，管理された CI としてリソースが表現される構成管理データベース (CMDB) に配信および保存されます。

DDM は，IT インフラストラクチャで発生した変更を絶え間なく検出し，それに応じて CMDB を更新する継続した自動プロセスです。検出対象のデバイスにエージェントをインストールする必要はありません。

インストール後，DDM Probe が置かれているネットワーク，Probe が存在するホスト，ホストの IP アドレスが自動的に検出され，そのオブジェクトごとに CI が作成されます。検出された CI は CMDB に置かれます。この CI は，DDM ジョブをアクティブ化するトリガの役割を果たします。ジョブは，アクティブ化されるたびに CI を検出します。そして次に，ほかのジョブのトリガとして使用されます。このプロセスは，IT インフラストラクチャ全体が検出されてマップされるまで続きます。

DDM が設定され，必要なパターンがアクティブ化されると，DDM がシステム上で動作し，システム・コンポーネントを検出して，CMDB に CI として保存します。新しいオブジェクトは手動でも自動でも検出できます。Probe のネットワークの外側にあるオブジェクトには，手動による追加の設定が必要です。

注: 本書は, DDM Probe が標準設定の場所 (**C:¥hp¥DDM¥DiscoveryProbe¥**) にインストールされていることを想定しています。

エージェントレス・テクノロジ

DDM は，カスタマのサイトに存在する専用 Probe を通して IT 環境のコンポーネントを検出する，エージェントレス・テクノロジです。たとえば，Netlinks ディスカバリ・モジュールは，受信した NetFlow データから TCP/IP 接続を検出します。

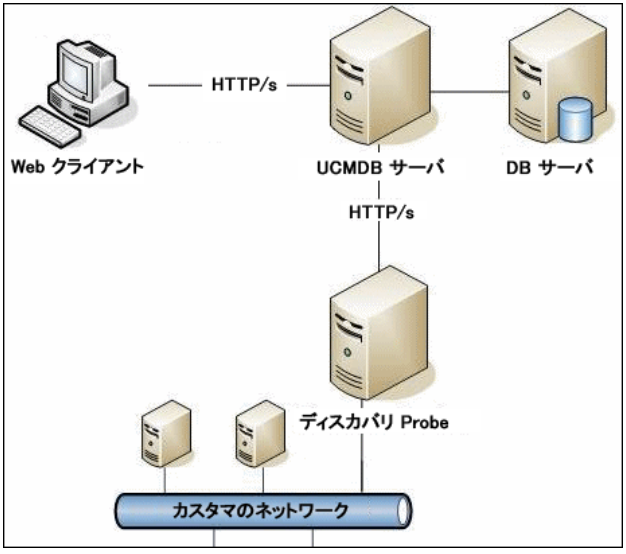
Probe は，http または https トラフィックを介して HP Universal CMDB に接続し，新規タスクの受信やタスク結果の送信などを行います。Probe のワークフローの詳細については, 32 ページ「DDM Probe のタスク」を参照してください。

DDM はエージェントレスであり，カスタマのマシンにエージェントをインストールする必要はありませんが，すでにインストールされている次のエージェントには依存します。

- **SNMP エージェント :** オペレーティング・システム，デバイスの種類，およびインストールされているソフトウェアに関する情報や，それ以外のシステム・リソース情報を提供します。SNMP エージェントは通常，管理のために拡張して新しい MIB をサポートし，より多くのデータを公開することができます。
- **WMI エージェント :** Microsoft 社のリモート管理エージェントです。通常，リモート管理者がアクセスできます。WMI エージェントも，汎用エージェントに WMI プロバイダを追加することによって拡張できます。
- **Telnet/SSH エージェント（またはデーモン）:** 主に UNIX システムで使用されます。リモートでマシンに接続し，さまざまなコマンドを実行してデータを取得します。
- **xCmd:** 機能の点で Telnet/SSH に似たリモート管理テクノロジで，Windows マシンに対してコンソール・コマンドを実行できます。xCmd は，正常に機能するために Administrative Shares & Remove Service Administration API に依存しています。
- **アプリケーション固有 :** このエージェントは，リモート・アプリケーションに依存しながらエージェントとして機能し，Probe のリモート・クエリ（データベースの検出，Web サーバの検出，SAP および Siebel の検出など）に適切に応答します。

ディスカバリおよび依存関係マップ のアーキテクチャ

ディスカバリおよび依存関係マップのアーキテクチャは次のようにデプロイされます。



DDM のユーザ・インタフェース

- DDM サーバは，HP Universal CMDB 表示システムとともに置かれます。
- DDM Probe は, カスタマのネットワークでデータ収集を行い, 動作するコンポーネントです。
- Probe は，http または https トラフィックを介して HP Universal CMDB サーバに接続します。単方向通信を強制することで，ファイアウォールのバイパスが可能になります。この http 要求または https 要求に応答するため，適切な場所に専用サーブレットがデプロイされます。
- DDM サーブレットは，DDM コンポーネント，表示システム，CMDB とともに HP Universal CMDB サーバに置かれます。

ディスカバリおよび依存関係マップ のコンポーネント

本項の内容

DDM Probe

Probe はメイン・コンポーネントであり，サーバからタスクを要求し，それを送信し，サーバを介して結果を CMDB に返します。インストールされている特定の Probe に，ネットワーク・アドレスの範囲を定義します。各 Probe は名前で識別されます。Probe の動作の詳細については，32 ページ「DDM Probe のタスク」を参照してください。

設定パラメータは DiscoveryProbe.properties ファイルに含まれています。このファイルは **C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors** にあります。詳細については，41 ページ「DiscoveryProbe.properties ファイル」を参照してください。

HP Universal CMDB サーバ

HP Universal CMDB サーバは，Probe への要求の配信，結果の受信，CMDB への収集データの格納を行います。

ディスカバリ・モジュール

モジュールとは，論理的にまとめられたジョブのグループであり，ひとまとまりとして操作と管理ができます。数多くのジョブを書き込む必要があるときにメイン・ビューを整理するのに役立ち，管理もしやすくなります。

ジョブを作成するときは，モジュールを選択するか，新しいモジュールを作成する必要があります。複数のジョブを作成する場合のベスト・プラクティスは，ジョブを論理グループに分け，それに従ってジョブをモジュールに割り当てるというものです。

ジョブ

ジョブは，異なる DDM プロセスに対してパターンの再利用ができます。ジョブによって，異なるトリガ CI セットに対して同じパターンをさまざまにスケジュール設定でき，各セットに異なるパラメータを設定することもできます（DDM をアクティブ化するには，パターンではなく，モジュールにまとめられたジョブをアクティブ化します）。

ジョブは，モジュールに次のようにまとめられます。

- **アプリケーション :** このモジュールは, Microsoft Exchange, Oracle E-Business Suite コンポーネント，Computer Center Management System (CCMS) に基づいた SAP 環境, Siebel 環境（Siebel トポロジやデータベースなど）, WebSphere MQ，UDDI レジストリなどの Web サービスを検出します。
- **クラスタ :** このモジュールは, Microsoft Cluster, ServiceGuard, および Veritas を検出します。
- **データベース:** DDM は, 最初にデータベースのインスタンスを探し, 次に各データベース・インスタンスのデータベース・リソース（ユーザ，テーブル，テーブル・スペースなど）のインスタンスを探します。HP Universal CMDB には，DB2，Oracle，Microsoft SQL Server のデータベースの事前定義された標準設定のビューが含まれています。
- **ディスカバリ・ツール :** このモジュールは，ドキュメント・ファイルやディレクトリの検出，ホストの検出，および外部ソースからのデータのインポートに必要なジョブを保持し，テンプレートの一例となります。
- **インテグレーション :** これらのモジュールは，UCMDB と NNM Layer 2，および Storage Essentials の統合に必要です。
- **J2EE:** このモジュールは, JBoss, Oracle Application Server, WebLogic, WebSphere のコンポーネントを検出します。

- **ネットワーク :** このモジュールは, ディスク情報, 実行中のプロセスやサービス, 負荷分散など，Windows および UNIX ホスト上のリソースを検出します。
- **仮想化 :** このモジュールは，VMware コンポーネントを検出します。
- **Web サーバ :** このモジュールは，Windows 上の Apache および Microsoft IIS, Solaris 上の SunOne，IBM HTTP Server を検出します。

ディスカバリおよび依存関係マップのウィザード

IP 範囲やネットワーク資格情報などの標準設定の値セットを使用する必要があるときは，（インフラストラクチャ，データベース，J2EE アプリケーションの検出のために）DDM ウィザードの 1 つを使用します。ウィザードを使用して DDM を実行する方法の詳細については，116 ページ「[ベーシック モード] ウィンドウ」を参照してください。

プロトコル

IT インフラストラクチャ・コンポーネントのディスカバリでは，SNMP，WMI，JMX，Telnet などのプロトコルが使用されます。詳細については，203 ページ「ドメイン資格情報リファレンス」を参照してください。

パターン

パターンは, ディスカバリおよび依存関係マップ・ジョブのリソースのうちの 1 つです。パターンには，標準設定の設定パラメータ，入力 TQL（可能性のある入力 CI を記述），DDM の実行方法を定義したスケジュール設定情報が含まれます。また，パターンには，ディスカバリに必要なスクリプトとその他のコードも含まれます。

ジョブは, (特定のトリガ CI セットを各パターンに関連付けることによって) 標準設定のパターン設定を上書きできますが，パターンに宣言されている内容を実行することもできます。

パターンの変更の詳細については, 256 ページ「[ディスカバリ リソースの管理] ウィンドウ」を参照してください。パターン記述の詳細については，第 10 章 : 「コンテンツ開発と記述」を参照してください。

構成ファイル

構成ファイルには, DDM パターンに関連するプロパティおよびパラメータが含まれます。たとえば，portNumberToPortName.xml ファイル（検出されたポートの番号をポート名にマップするファイル）には，ネットワークの検出時に DDM によって使用されるポートのリストが含まれています。ユーザが定義可能なファイルの詳細については，235 ページ「リソース・ファイル」を参照してください。

外部リソース

外部リソースには，Visual Basic ファイルや資格情報ファイルなど，DDM に必要で HP Universal CMDB の外部に存在するすべてのリソースが含まれます。

パッケージ

パッケージには，ネットワーク拡張，アプリケーション，データベースなどの IT インフラストラクチャ・リソースの検出を可能にするジョブ定義，パターン，リソース，ツールが含まれます。詳細については，『**モデル管理**』の「パッケージ マネージャ」を参照してください。

スクリプト

HP Universal CMDB では, パターン記述に Jython スクリプトが使用されます。たとえば，SNMP を使用してマシンへの接続を試みる SNMP_NET_Dis_Connection パターンでは，SNMP_Connection.py スクリプトが使用されます。Jython は，Python に基づき，Java によって強化された言語です。パターン記述の詳細については，第 10 章 :「コンテンツ開発と記述」を参照してください。

Jython の使用方法の詳細については，次の Web サイトを参照してください。

- http://www.jython.org
- http://www.python.org

ディスカバリおよび依存関係マップ のアプリケーション

ディスカバリおよび依存関係マップには，次のアプリケーションがあります。

- 53 ページ「ディスカバリ実行」
- 53 ページ「ディスカバリ・プローブ設定」
- 53 ページ「ディスカバリ・リソースの管理」
- 54 ページ「ステータス・スナップショット表示」

ディスカバリ実行

ディスカバリ実行アプリケーションでは，(特定の CI グループの検出に必要な) DDM モジュールおよびジョブを管理できます。プロセスを実行するには，ジョブをアクティブ化します。モジュール内のすべてのジョブをアクティブ化するか，一部のジョブをアクティブ化するかを選択できます。また，ジョブを編集したり，一定の時間にジョブが実行されるようにスケジュールを設定したりできます。

詳細については，第 5 章 :「ディスカバリ実行」を参照してください。

ディスカバリ・プローブ設定

ディスカバリ・プローブ設定では，システムへの Probe の追加，および，既存の Probe の編集を行うことができます。各 Probe がカバーするネットワーク範囲を定義します。

詳細については，第 6 章 :「ディスカバリ Probe の設定」を参照してください。

ディスカバリ・リソースの管理

注: ディスカバリおよび依存関係マップについて高度な知識がないユーザは，リソースの変更を行わないでください。

ディスカバリ・リソースの管理では，ディスカバリに必要なリソースを表示できます。パターン，スクリプト，構成ファイルの編集を行うことができます。また，DDM で必要な外部リソースを置換したり削除したりすることもできます。

詳細については，第 7 章 :「ディスカバリ・リソースの管理」を参照してください。

ステータス・スナップショット表示

ステータス・スナップショット表示では，特定のジョブのスケジュール設定に関する詳細，および，ジョブの統計情報を表示できます。

詳細については，第 8 章 :「ステータス・スナップショット表示」を参照してください。

トリガ CIT，トリガ CI，入力 TQL，トリガ TQL

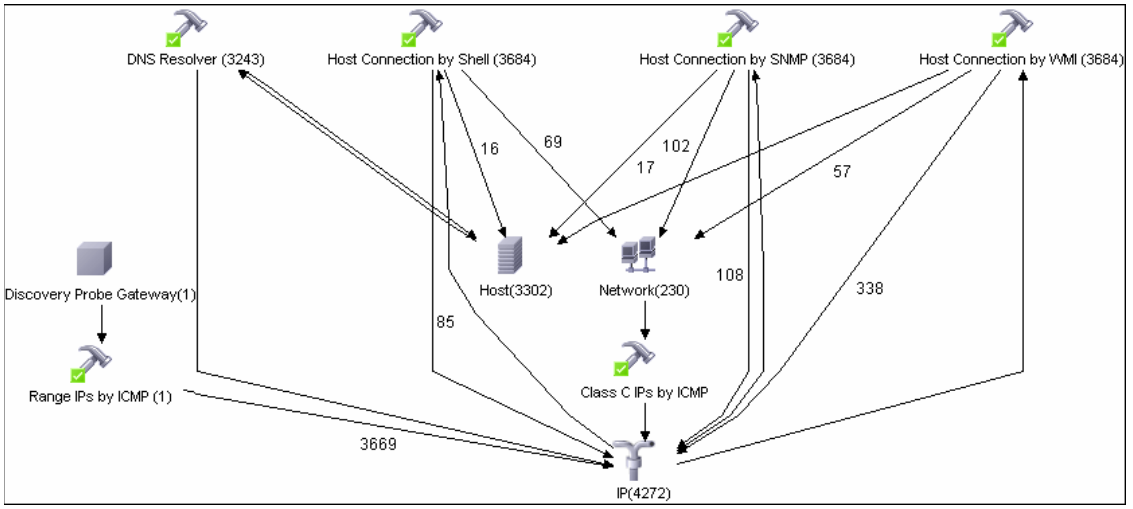
本項では，トリガ CIT，トリガ CI，入力 TQL，トリガ TQL の機能について説明します。これらのオブジェクトの詳細については，319 ページ「パターン入力 （トリガ CIT と入力 TQL）の定義」および第 10 章 :「コンテンツ開発と記述」を参照してください。TQL の詳細については，『**モデル管理**』の「トポロジ・クエリ言語」を参照してください。

トリガ CIT

トリガ CIT は，パターンに対する入力として使用される CIT を定義します。たとえば，IP を検出するパターンの場合，入力 CIT は Network となります。

例 ー ジョブをアクティブ化するトリガ CIT

Network CIT は，**Class C IPs by ICMP** ジョブをアクティブ化するトリガです。そして，**Class C IPs by ICMP** ジョブは IP アドレスのインスタンスを検出します。この検出された IP アドレス自体が，**Host Connection** ジョブをアクティブ化するトリガの役割を果たす CI となります。さらに，このジョブによって IP アドレスとネットワークの CI が検出されます。次の図のように，Probe に定義されている範囲に含まれるすべての IP アドレスが検出されたとき，プロセスが終了します。



トリガ CI

トリガ CI は，ジョブをアクティブ化する CMDB 内の CI です。ジョブは，アクティブ化されるたびに CI を検出します。そして次に，ほかのジョブのトリガとして使用されます。このプロセスは，IT インフラストラクチャ全体が検出されてマップされるまで続きます。

ジョブへのトリガ CI の追加の詳細については，132 ページ「[ディスカバリ ステータス] 表示枠」を参照してください。

入力 TQL

入力 TQL には次の 2 つの機能があります。

- **入力 TQL はパターンに関連付けられます。**入力 TQL は，そのパターンを実行するジョブに含まれているすべてのトリガ CI の最低限の要件セットを定義します（これは，ジョブにトリガ TQL が関連付けられていない場合でも当てはまります)。

  たとえば，入力 TQL は SNMP を実行中の IP を探すことができるとします。つまり，SNMP エージェントがインストールされている IP のみ，そのパターンをトリガできます。これにより，トリガとしてすべてのホストをパターンに追加するトリガ CI をユーザが手動で作成するというケースが回避されます。

- **入力 TQL は CMDB からデータ情報を取得する方法を定義します。**目的のデータ情報がトリガ CI に含まれていない場合でも，入力 TQL で取得できます。入力 TQL は，情報を取得する**方法**を定義します。

  たとえば，トリガ CI（**SOURCE** というノード名を持つノード）とターゲット CI の間に関係を定義し，その後，［トリガされた CI データ］表示枠でこのノード名によってターゲット CI を参照することができます。詳細については，270 ページ「［トリガされた CI データ］表示枠」を参照してください。

  パターン記述時の入力 TQL の使用方法に関する詳細については，318 ページ「手順 1: ディスカバリおよび依存関係マップのパターンの作成」を参照してください。

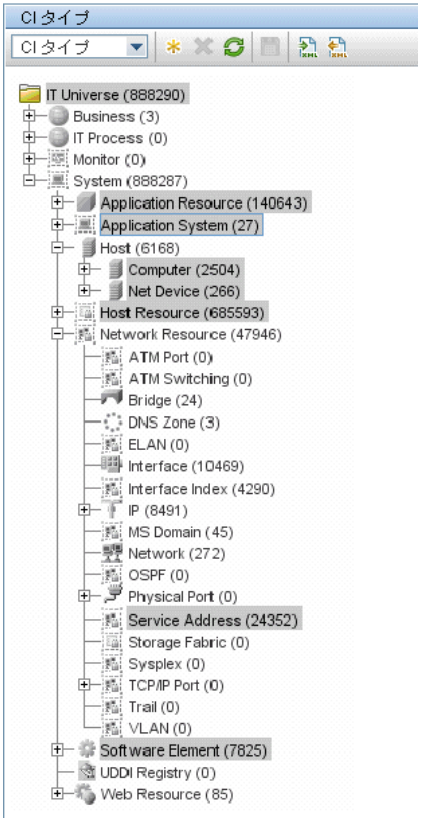
トリガ TQL

ジョブに関連付けられているトリガ TQL は，入力 TQL のサブセットであり，どの CI をジョブのトリガ CI にすべきかを定義します。つまり，入力 TQL が SNMP を実行中の IP を探す場合，トリガ TQL は，195.0.0.0 ～ 195.0.0.10 の範囲内の SNMP を実行中の IP を探します。

注 : トリガ TQL は，入力 TQL と同じオブジェクトを参照する必要があります。たとえば，パターンの入力 TQL が SNMP を実行中のに照会して IP を探す場合，ホストに接続されている IP を探すために照会する関連付けられたジョブにトリガ TQL を定義することはできません。これは，入力 TQL の要求のとおり，一部の IP が SNMP オブジェクトに接続されていない可能性があるためです。

クラス・モデル ー 概要

本項では，高度で重要な CI タイプに焦点を当てながら UCMDB クラス・モデルの概要を示します。



本項の内容

Host CIT

Host CIT は，環境のネットワーク・ノード（物理および仮想）を表します。

UCMDB には次の 2 つの Host CIT があります。

- **Computer:** 汎用オペレーティング・システム（UNIX，Windows，メインフレームなど）が実行される汎用デバイスです。
- **Network Device:** あらかじめ目的が決まっているネットワーク・デバイスです（つまり，ルータ，スイッチ，ロード・バランサ・デバイス，ファイアウォールなど）。

  UCMDB のホストは，IP アドレスまたは強固な ID 値（MAC アドレスやハードウェア ID など）で識別されます。IP アドレスで識別されたホストは不完全であると判断されます（**Host Is Complete** 属性の値が **false**）。強固な ID によって識別されたホストは，**complelte**（完全）とみなされます。

Host Resource CIT

Host Resource CIT はホストのリソースを表し，物理リソース（メモリや CPU など）とオペレーティング・システム・リソース（ファイル・システム，ファイル，ネットワーク共有，OS ユーザなど）の両方が含まれます。

Host Resource は常に Host CIT に含まれます（container link でリンクされます）。つまり，UCMDB の環境では，Host Resource CIT はホストのコンテキストの外部に存在できないということです。

Software Element CIT

Software Element CIT は，アプリケーション・ソリューションの一部である，ホストで実行されている 1 つのソフトウェアを表します。

DDM で検出される Software Element の例は次のとおりです : データベース (Oracle，MSSQL，DB2，Sybase)，Web サーバ，Microsoft Exchange Server，J2EE サーバ，負荷分散ソフトウェア，クラスタ・ソフトウェア。

Software Element CIT は，**強いタイプ**または**弱いタイプ**のどちらかであると判断されます。

- **強いタイプの Software Element CIT:** これらの CIT は特殊化された CIT であり（ソフトウェア要素 CIT から派生します），より多くの属性，さまざまな識別ルール，表示ラベルなどを含んでいます。
- **弱いタイプの Software Element CIT:** これらの CIT は，ソフトウェア要素 CIT 自体の CI インスタンスです。これらの CIT は，基本構成属性だけを含みます。また，モデル内には各ソフトウェア・コンポーネント・タイプの 1 つのインスタンスだけが存在できます。つまり，単一のホスト上で弱いタイプのソフトウェア要素を使って 2 つの異なる Oracle インスタンスをモデリングすることはできません。

  強いタイプのソフトウェア要素が検出されると，調整メカニズムによって，弱いタイプのソフトウェア要素が対応する強いタイプにマッピングされます。このメカニズムにより，DDM が 1 つのメソッドによってソフトウェアの存在を検出し，その後別のメソッドを使ってそのソフトウェアが検出されたときに，2 つの CI が最終的に UCMDB 内でマージされることを知って，そのソフトウェアに関する追加の詳細を報告することが可能になります。

  調整メカニズムは，弱いタイプと強いタイプの共有属性値である Name 属性 (**data_name**) に依存しています。Software Element はすべて（強いタイプも弱いタイプも），識別名付きで作成されます。同じ名前の弱いタイプの Software Element が含まれるホストで強いタイプの Software Element が検出された場合，弱いタイプの Software Element は強いタイプの Software Element と結合されます。

  弱いタイプの Software Element は通常，DDM のホスト・リソース・ディスカバリ・パターンによって検出されます。このパターンは，実行中のプロセスとそのネットワーク相互接続に基づいてキー・ソフトウェアを識別する構成ファイル (Application Signature) に依存しています。

Host Resource CIT と同様，Software Element CIT もホストに含まれます。

Application Resource CIT

Application Resource CIT は，Software Element の内部構成を表します。この CIT は，ある Software Element の複雑な構成を記述する場合に，属性では不十分なときに使用されます。そのようなリソースのグループの 1 つが，データベースの内部構成（データベース・スキーマ，データベース・データ・ファイル，データベース・テーブルなど）を記述する CI を含むデータベース・リソース・グループです。

さらなる汎用アプリケーション・リソース（構成ファイル，リソース・プール，Web サービスなど）は，単一の CIT に限定されるのではなく複数の Application Resource CIT に含まれます。

Application Resource CIT は，Software Element CIT に含まれます。

Application System CIT

Application System CIT は，アプリケーションのデプロイメントを表します。単一ホスト・コンテキストに制限される Software Element CIT とは異なり，Application System CIT は，複数のホストにまたがるアプリケーション構成を表すように設計されています。

Application System CI は通常，member 関係を介して Software Element CI にリンクされます。Application System CI は，Software Element CI とともに CMDB のアプリケーションを表します。

Application System CIT の例として，Clusters，SAP System，Siebel Enterprise System，MS Exchange System が挙げられます。

Service Address CIT

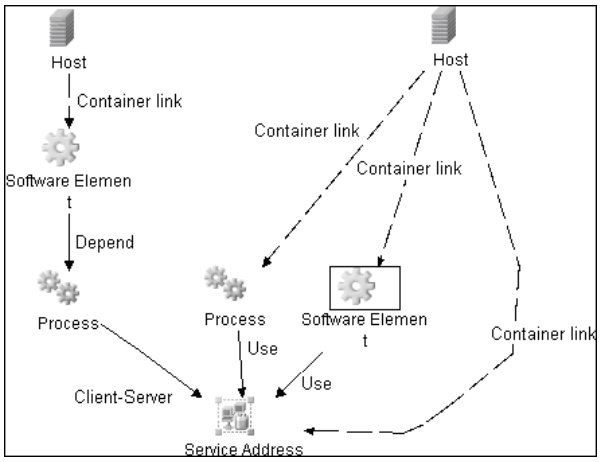
Service Address CIT は Network Resource CIT から派生しており，Software Element からサービスを取得する際にクライアント・ソフトウェアによって使用されるアドレスを表します。

Service Address CIT のキー属性は次のとおりです。

- **Address Type:** 現在，許容可能な値は **TCP/UDP** および **URL** です。
- **Service Address:**［Service Address］フィールドの内容はアドレス・タイプによって異なります。TCP/UDP アドレスでは，**＜ IP アドレス：ポート＞**という形式が使用されます。URL アドレスには，アクセス可能な URL 値が含まれます。

  ソフトウェアの依存関係のマッピングは，モデルの Service Address CIT に依存します。クライアントの Software Element，**depend** 関係を介して Client Process CI にリンクされます。Client Process CI には，リモートの Service Address CIT とのクライアント / サーバ関係があります。この Service Address CIT は，**use** 関係を通して，プロセス (Host Resource)，またはそのアドレスでサービスを提供する Software Element，あるいはその両方に接続されます。

Service Address CIT も Host CIT に含まれます。



クラス・モデルの変更

バージョン 8.00 のクラス・モデルの変更に関する詳細については，『**HP Universal CMDB デプロイメント・ガイド**』(PDF) の「クラス モデルの変更」を参照してください。また，57 ページ「クラス・モデル ― 概要」も参照してください。

DDM アップグレード情報

次の各項には，アップグレード情報が含まれています。

- 『**HP Universal CMDB デプロイメント・ガイド**』の「Discovery と Dependency Mapping API の変更点」。
- 『**HP Universal CMDB デプロイメント・ガイド**』の「ディスカバリ・モジュール」。
- 『**HP Universal CMDB デプロイメント・ガイド**』の「DDM DomainScopeDocument ファイルのアップグレード」。

手動によるジョブのアクティブ化

ジョブをアクティブ化するには，［ディスカバリ モジュール］表示枠の［**アクティブ化**］ボタンをクリックします。CI を手動でアクティブ化するには，TQL を無効にし，CI を追加します（TQL の無効化は［**TQL 出力用プローブ制限の編集**］ダイアログ・ボックスで行います。手動による CI の追加は［追加する CI の選択］ダイアログ・ボックスで行います）。ジョブは，再度ディスパッチされた CI のみを使用して実行されます。詳細については，142 ページ「［ディスカバリ モジュール］表示枠」を参照してください。

手動によるネットワーク CI の作成

Probe は，Probe が実行しているネットワークを検出するところから始めます。したがって，通常はユーザがネットワーク CI を作成する必要はありません。ただし，あるネットワークに Probe をインストールし，別のネットワークのオブジェクトを検出するように Probe を設定した場合，DDM はそのネットワークを検出できません。Probe での検出が必要なネットワークの Network CI を手動で作成する必要があります。

Network CI が存在していることを確認するには，ビュー マネージャにアクセスします（［**管理**］＞［**モデリング**］＞［**ビュー マネージャ**］）。［Network］フォルダを探し，そのフォルダに［Network Topology］ビューが含まれているか確認します。

手動によるネットワーク CI の作成の詳細については，『**モデル管理**』の「［新規 CI］ダイアログ・ボックス」を参照してください。

モジュールの実行スケジュール設定

ある決まった時刻に実行されるように，ジョブまたはモジュールのスケジュールを設定できます。詳細については，146 ページ「［ディスカバリ スケジューラ］ダイアログ・ボックス」を参照してください。

命名規則

DDM のエンティティに名前を付けるときに使用できる文字は次のとおりです：a ～ z，A ～ Z，0 ～ 9。IP アドレスを入力する際は，数字とアスタリスク (*) のみ使用します。

ログ・ファイル

本項では，DDM のログ・ファイルについて説明し，基本的なトラブルシューティングの方法について説明します。ログ・ファイルには，エラーを含む，ジョブのアクティブ化に関するメッセージが格納されます。問題管理の詳細については，93 ページ「エラー・レポートによる問題の管理」を参照してください。通信ログのオプションの設定に関する詳細については，260 ページ「[実行オプション］表示枠」を参照してください。

重大度レベル

各ログは，記録する情報が特定の重大度しきい値に対応するように設定されます。各種のログがさまざまな情報を追跡するのに使用されているため，ログはそれぞれ適切な標準レベルにあらかじめ設定されています。ログ・レベルの変更方法の詳細については，下の「ログ・レベルの変更」を参照してください。

次に，重大度レベルを範囲が最も狭いものから最も広いものの順に示します。

- **Fatal（致命的）:** このレベルでは，インフラストラクチャに関する問題，DLL ファイルの欠落，例外など，深刻なエラーが報告されます。
- **Debug（デバッグ）:** このレベルは，HP ソフトウェア・サポートが問題をトラブルシューティングするときに利用します。
- **Error（エラー）:** このレベルでは，DDM のデータ取得を妨げる問題が報告されます。このエラーは通常，何らかの対応（タイムアウトの延長，範囲の変更，パラメータの変更，ほかのユーザの資格情報の追加など）を必要とするため，よく調べるようにしてください。
- **Warning（警告）:** 実行は成功で，深刻ではないが認識しておくべき問題がある場合，重大度は**警告**となります。より詳細なデバッグ・セッションを開始する前に，CI を調べてデータが欠落していないか確認する必要があります。**警告**には，インストールされているエージェントやリモート・ホストの欠落に関するメッセージや，無効なデータが原因で属性が正しく計算されなかったというメッセージが含まれている可能性があります。
- **Info（情報）:** このログにはすべての活動が記録されます。通常は情報のほとんどは日常的なもので利用価値がなく，ログ・ファイルがすぐに一杯になります。
- **Success（成功）:** トリガ CI は正常に実行されました。

注：各ログ・レベルの名称は，サーバやプロシージャによって若干異なる場合があります。たとえば，「**Info**」は「**Always logged**」や「**Flow**」と呼ばれることがあります。

ログ・レベルの変更

HP ソフトウェア・サポートからの依頼で，ログの重大度しきい値レベルを変更しなければならない場合があります（たとえば，デバッグ・レベルなど，より詳細なレベルへの設定変更）。

重大度しきい値レベルを変更するには，次の手順を実行します。

1 テキスト・エディタでログ・プロパティ・ファイルを開きます。ログ・ファイルのプロパティは，次のディレクトリ内のファイルで定義します。

   a C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥probeManager¥probeMgrLog4j.properties

   b C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥probeGateway¥probeGwLog4j.properties

2 ログのパラメータを探します。次に例を示します。

```
log4j.appender.LOGFILE_Performance.Threshold=DEBUG
```

3 これを必要なレベルに変更します。次に例を示します。

```
log4j.appender.LOGFILE_Performance.Threshold=INFO
```

ログ・レベルの詳細については，63 ページ「重大度レベル」を参照してください。

4 ファイルを保存します。

本項の内容

➤ 65 ページ「サーバ・ログ」

➤ 66 ページ「Probe ログ」

サーバ・ログ

サーバ・ログ・ファイルは HP Universal CMDB サーバにあります。このログ・ファイルには，エラー・メッセージを含む，サーバ側で発生したサーバ動作に関する情報が格納されます。

次のログ・ファイルは，**C:¥hp¥UCMDB¥UCMDBServer¥j2f¥log** フォルダまたはサブフォルダにあります。

mamAutoDiscovery.log

サーバで実行されているタスクに関する情報が格納されます。サーバは，ジョブのアクティブ化，Probe からの結果の処理，Probe のタスクの作成などのサービスをユーザ・インタフェースまたは Probe Gateway に提供します。

レベル	説明
エラー :	サーバ側のすべての DDM プロセス・エラー。
情報 :	処理中の要求に関する情報。
デバッグ :	主にデバッグのためのログ。

基本的なトラブルシューティング : 調査を必要とする無効なユーザ・インタフェース応答またはエラーがあるとき，このログを調べます。このログには，問題の分析を可能にする情報が含まれています。

discoveryServlet.log

このログには，次のサーブレットから送信されたメッセージが記載されます。

- **Collectors Utilities Servlet:** ユーザ・インタフェースは，このサーブレットを介してサーバに接続します。
- **Collectors Servlet:** Probe は，このサーブレットを介してサーバに新しいタスクを要求します。
- **Collectors Results Servlet:** Probe は，このサーブレットを介して新しい結果を送信します。

➤ **Collectors Download Servlet:** Probe は，このサーブレットを介して新しいサーバ・データをダウンロードします。

レベル	説明
エラー :	サーブレットのすべてのエラー。
情報 :	ユーザの要求および Probe のタスク要求に関する情報。
デバッグ :	➤ ユーザの要求 ➤ DDM タスクを読み取るための Probe 要求 ➤ サーブレットへの Probe のアクセス

基本的なトラブルシューティング :

➤ ユーザ・インタフェースとサーバの間の通信の問題

➤ Probe とサーバの間の通信の問題

処理に関する一部の問題は，**mamAutoDiscovery.log** ではなくこのログに書き出される場合があります。

mamAutoDiscoveryUpgrade.log

アップグレード・プロセスに関する情報が格納されます。

mamAutoDiscoveryResultsStat.log

Probe から受信した結果の統計情報が格納されます。

Probe ログ

Probe ログには，Probe Gateway または Probe Manager で発生したジョブのアクティブ化に関する情報が格納されます。

本項のログは **C:¥hp¥DDM¥DiscoveryProbe¥root¥ にあります。**

一般ログ

wrapperProbe.log

Probe のすべてのコンソール出力が単一のログ・ファイルに記録されます。

レベル	説明
エラー :	Probe Gateway 内で発生したエラー。
情報 :	新規タスクの発生や削除など，重要な情報メッセージ。
デバッグ :	サーブレットへのすべての Probe アクセスの記録。

基本的なトラブルシューティング : Probe Gateway の問題に対してはこのファイルを使用し，Probe Gateway に関して随時発生したことや，重要な問題を確認します。

probe-error.log

Probe のエラーのサマリです。

レベル	説明
エラー :	Probe コンポーネントのすべてのエラー。
情報 :	該当なし。
デバッグ :	該当なし。

基本的なトラブルシューティング : Probe コンポーネントでエラーが発生したかを確認するには，このログを調べます。

probe-infra.log

すべてのインフラストラクチャ・メッセージのリストです。

レベル	説明
エラー :	すべてのインフラストラクチャ・エラー。
情報 :	インフラストラクチャ・アクションに関する情報。
デバッグ :	主にデバッグのためのメッセージ。

基本的なトラブルシューティング：Probe のインフラストラクチャからのメッセージのみ。

WrapperProbeMgr.log

レベル	説明
エラー :	Probe Manager 内で発生したエラー。
情報 :	受信タスク，タスクのアクティブ化，結果の転送など，重要な情報メッセージ。
デバッグ :	該当なし。

基本的なトラブルシューティング：Probe Manager の問題に対してはこのファイルを使用し，Probe Manager に伴って随時発生したことおよび重要な問題を確認します。

Probe Gateway ログ

probeGW-taskResults.log

このログには，Probe Gateway からサーバに送信されたすべてのタスク結果が記録されます。

レベル	説明
エラー :	該当なし。
情報 :	結果の詳細（タスク ID，ジョブ ID，削除または更新する CI の数）。
デバッグ :	サーバに送信される **ObjectStateHolderVector** 結果（XML 文字列形式）。

基本的なトラブルシューティング：

- サーバに到着した結果に問題がある場合は，このログを調べて，Probe Gateway がどの結果をサーバに送信したか確認します。
- このログの結果は，サーバへの送信後に書き込まれます。送信前は，Probe JMX コンソールに結果を表示できます（**ProbeGW Results Sender** MBean を使用します）。ユーザ名とパスワードで JMX コンソールにログインしなければならない場合もあります。

probeGW-tasks.log

このログには，Probe Gateway が受信したすべてのタスクが記録されます。

レベル	説明
エラー :	該当なし。
情報 :	該当なし。
デバッグ :	タスクの XML。

基本的なトラブルシューティング :

- Probe Gateway のタスクがサーバのタスクと同期化されていない場合は，このログを調べて，Probe Gateway が受信したタスクを確認します。
- 現在のタスクの状態は, JMX コンソールで表示できます（**Discovery Scheduler** MBean を使用します）。

Probe Manager ログ

probeMgr-services.log

Java サービスのデバッグ・メッセージです。

レベル	説明
エラー :	該当なし。
情報 :	該当なし。
デバッグ :	該当なし。

基本的なトラブルシューティング : Java サービスのデバッグ・メッセージを確認するには，このログを調べます。

probeMgr-performance.log

事前定義された期間ごとに収集されたパフォーマンス統計情報のダンプです。メモリ情報およびスレッド・プール・ステータスが含まれます。

レベル	説明
エラー :	該当なし。
情報 :	該当なし。
デバッグ :	該当なし。

基本的なトラブルシューティング :

- ある期間におけるメモリの問題を調べるには，このログを確認します。
- 標準設定では，統計情報は 1 分ごとにログ記録されます。

probeMgr-patternsDebug.log

このログには，パターンの問題のデバッグに使用されるメッセージが格納されます。

レベル	説明
エラー :	該当なし。
情報 :	該当なし。
デバッグ :	該当なし。

基本的なトラブルシューティング : パターンのデバッグにはこのログ・ファイルを使用します。

トラブルシューティングと制限事項

ログ・ファイルを使用して基本的なトラブルシューティングを行う方法の詳細については，63 ページ「ログ・ファイル」を参照してください。

ログインやインストールなどのトラブルシューティングの詳細については，『**参照情報**』の「トラブルシューティングおよび制限事項」を参照してください。

本項の内容

Probe Gateway と Probe Manager のアクティブ化

問題 : Probe Gateway または Probe Manager をアクティブ化できない。

症状 : Probe Gateway または Probe Manager をアクティブ化しようとすると，コンソールが開いてすぐに閉じてしまいます。

検証 : 例外メッセージを確認するには，
C:¥hp¥DDM¥DiscoveryProbe¥root¥logs にある次のファイルを開きます。

- Probe Gateway の場合 : **WrapperProbeGw.log**
- Probe Manager の場合 : **WrapperProbeMgr.log**

メッセージが表示されます。次のメッセージのいずれかが表示された場合，問題はメモリ・サイズ定義にあります。

```
Initial heap too small for new size specified.
Incompatible initial and maximum heap sizes specified.
The port number is being used.
```

この問題を解決するには，次の解決策の項を参照してください。ほかのメッセージが表示されていて問題を解決できない場合は，HP ソフトウェア・サポートにお問い合わせください。

解決策 : アクティブ化の問題には，次に示すいくつかの原因が考えられます。

- **メモリ・サイズが不適切 :** 各 CMDB コンポーネントには，最小および最大メモリ・サイズが割り当てられています。この定義は，batch.cmd ファイルの set memory sizes セクションで設定します。メモリ・サイズがお使いのワーク・ステーションには大きすぎる場合，あるいはメモリ・サイズが正しくない場合や互いに不適合である場合は，メモリ・サイズを変更し，ファイルを保存して，値を変更したコンポーネントを再起動する必要があります。
- **インストール・パスが長すぎる :** 長いパスのディレクトリに HP Universal CMDB をインストールした場合は，オペレーティング・システムまたは JVM で HP Universal CMDB 実行コマンドに関する問題が発生することがあります。短いパスの別のディレクトリに HP Universal CMDB を再インストールします。

Probe Gateway と Probe Manager の接続

問題 : Probe Gateway と Probe Manager の接続が確立できない。

症状 : DDM プロセスが正常に機能していません。

検証: Probe Gateway の場合, 次の例のようなエラー・メッセージが Probe Gateway ログ (**WrapperProbeGw.log**) に表示されます（このログは **C:¥hp¥DDM¥DiscoveryProbe¥root¥logs** にあります）。

```
Failed to connect to probe manager at <server>. Will retry later
```

Probe Manager の場合，次の例のようなエラー・メッセージが Probe Manager ログ (**probe-infra.log**) に表示されます（このログは **C:¥hp¥DDM¥DiscoveryProbe¥root¥logs** にあります）。

```
Connection attempt to service:jmx:rmi:///jndi/rmi://<Probe GW HOST>:1742/jmxrmi
failed, probe GW may be down
```

解決策 : 次の点を確認します。

- 適切なポート (1742) が定義されているか確認します。RMI 接続ポート・パラメータは appilog.collectors.rmi.port と呼ばれます。このパラメータは，**C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors** にある **DiscoveryProbe.properties** というファイルで定義されます。
- ほかのアプリケーションが Probe Manager ポートを使用していないか確認します。確認するには，Windows コマンド・インタープリタ (**cmd.exe**) で「netstat -na」と入力します。すると，現在使用中のポートのリストが表示されます。Probe Manager ポートが使用中の場合は, ほかのアプリケーションを閉じるか, **DiscoveryProbe.properties** ファイルでポート番号を変更します。

ホスト名を IP アドレスに解決できない

問題 : ホスト名を IP アドレスに解決できない。この問題が発生すると，ホストの検出ができず，パターンも実行されなくなる。

解決策 : Probe マシン上の Windows HOSTS ファイルにホスト・マシン名を追加します。

接続に失敗

問題: RMI または http の例外が原因で HP Universal CMDB サーバと Probe の接続が失敗した。

解決策 : ほかのプロセスによって Probe ポートが使用されないようにします。

DDM の結果がトポロジ・マップに表示されない

問題 : DDM プロセス中に検出されたはずのデータがトポロジ・マップに表示されない。

検証 : CMDB は，データの取得または TQL 結果の作成を行うことができません。［統計結果］表示枠を調べます。CI が作成されていなければ，問題は DDM プロセス中に発生しています。

解決策: C:¥hp¥DDM¥DiscoveryProbe¥root¥logs にある **probeMgr-services.log** ファイルのエラー・メッセージを調べます。

ネットワークと IP

問題 : ネットワークまたは IP がすべて検出されなかった。

症状 : トポロジ・マップの結果にネットワークまたは IP がすべて表示されません。

検証 :［ディスカバリ・プローブ設定］ウィンドウの IP アドレス範囲が，検出する必要があるネットワークまたは IP の範囲を網羅していません。

解決策 : DDM の範囲を変更します。

1 ［**管理**］＞［**ディスカバリ**］＞［**ディスカバリ プローブ設定**］を選択して［ディスカバリ プローブ設定］ウィンドウを開きます。

2 Probe と範囲を選択します。

3 必要に応じて［範囲］ボックスの IP アドレス範囲を変更します。

TCP ポート

問題 : TCP ポートがすべて検出されなかった。

症状 : トポロジ・マップの結果に TCP ポートがすべて表示されません。

検証 : portNumberToPortName.xml ファイルを開き（[**管理**] > [**ディスカバリ リソースの管理**] > [**Network**] > [**構成ファイル**] > [**portNumberToPortName.xml**]），欠落している TCP ポートを探します。

解決策 : 検出する必要があるポート番号を **portNumberToPortName.xml** ファイルに追加します。

Probe のステータスが「非接続」になっている

問題 : ディスカバリおよび依存関係マップで Probe のステータスが非接続になっている。

解決策 : Probe マシンで次の点を調べます。

- Probe が実行されているか。
- ネットワーク問題が発生していないか。

SSH/Telnet 資格情報

問題 : TTY（SSH/Telnet）エージェントへの接続に失敗した。

解決策 : TTY (SSH/Telnet) エージェントに関する接続問題のトラブルシューティングを行うには，TTY (SSH/Telnet) エージェントとの接続を検証できるユーティリティを使用します。クライアント・ツールの PuTTY はその一例です。

SNMP 資格情報

問題 : SNMP デバイスからの情報収集に失敗した。

- **解決策 1:** SNMP エージェントとの接続を検証できるユーティリティを使用して，実際に Network Management ステーションから情報にアクセスできるか確認します。そのようなユーティリティの一例として GetIf があります。
- **解決策 2:** SNMP プロトコルへの接続データが［プロトコル パラメータを追加］ダイアログ・ボックスで正しく定義されているか確認します。詳細については，201 ページ「[プロトコル パラメータ]］ダイアログ・ボックス」を参照してください。
- **解決策 3:** SNMP エージェントの MIB オブジェクトからデータを取得するのに必要なアクセス権限があるか確認します。

Solaris マシンでのプロセスのディスカバリおよびソフトウェアの実行

問題 : Solaris マシンでプロセスのディスカバリおよびソフトウェアの実行ができない。

解決策: **/usr/ucb/ps** ユーティリティがマシンにインストールされていることを確認します。

Windows XP マシンのリソースの検出

問題 : Windows プラットフォームで動作しているマシンのリソースの検出に失敗した。

- **解決策 1:**［**スタート**］>［**設定**］>［**コントロール パネル**］>［**システム**］を選択します。［リモート］タブの［**このコンピュータにユーザがリモートで接続することを許可する**］チェック・ボックスが選択されているか確認します。
- **解決策 2:**（Windows XP の場合）Windows エクスプローラで［**ツール**］>［**フォルダ オプション**］を選択します。［表示］タブの［**簡易ファイルの共有を使用する（推奨)**］チェック・ボックスをクリアします。

SAP のディスカバリに失敗

問題 : SAP のディスカバリに失敗し，次の Java メッセージが表示された。

> MSVCR71.dll が見つからなかったため，このアプリケーションを開始できませんでした。

解決策 : 2 つの .dll ファイルが欠落しています。解決するには, https://websmp205.sap-ag.de/~form/sapnet?_FRAME=CONTAINER&_OBJECT=012003146900000245872003 の Note #684106 を参照してください。

制限事項

- 英語以外のオペレーティング・システムに DDM をインストールした場合も, ジョブ名とモジュール名に使用できるのは英字のみとなります。

- Oracle Real Application Clusters（Oracle RAC）のディスカバリを行う場合，DDM がリモート・マシン（データベース・クライアント）へのリンクを検出できないときがあります。それは，検出されたデータベースが IP アドレスではなくホスト名によってクライアントと通信し，そのホスト名を IP アドレスに解決できないときです。この場合，リモート・クライアントを作成することはできません。
- 各 Content Pack のインストールでは，用意済みのリソースすべてがその Content Pack のコンテンツによって上書きされます。つまり，これらのリソースに対して行った変更はすべて失われることになります。この上書きが適用されるリソースは，TQL，ビュー，エンリッチメント，レポート，DDM Jython スクリプト，DDM パターン，DDM ジョブ，DDM リソース，DDM 構成ファイル，DDM モジュール，CI タイプ，および関係です（CI タイプおよび関係に追加される属性は上書きされません）。

  通常は，用意済みリソースに変更を加えないようにすることをお勧めします。変更する必要がある場合は，変更内容を追跡して Content Pack のインストール後に確実に再適用できるようにしてください。重要かつ全般的な修正点（個人の環境に固有ではない）については，CSO が分析して次の Content Pack の一部として含めることができるよう，CSO に報告してください。

第 4 章

クラス・モデルの改良点

注：本項は，Content Pack 3.00 以降に適用されます。

本章の内容

概念

参照先

クラス・モデルの改良点の概要

DDM Content Pack 3.00 では，クラス・モデルが改良されています。Content Pack のこれらの変更のために, アップグレード手順を実行する必要はありません。たとえば，変更は新しい CIT の追加や既存の CIT への新しい属性の追加に限定されており，キー属性は変更されていません。

クラス・モデルに行われた改良点は次のとおりです。

➤ 80 ページ「BIOS UUID 属性のサポート」

➤ 83 ページ「ソフトウェア製品コード属性のサポート」

➤ 85 ページ「アプリケーション・インスタンス名属性のサポート」

BIOS UUID 属性のサポート

BIOS UUID 属性 (host_biouuid) が Host CIT に追加されました。本項では，さまざまなプロトコルを使用した BIOS UUID のディスカバリが DDM でどのようにサポートされているのかについて説明します。

本項の内容

➤ 80 ページ「Windows: NTCMD プロトコル」

➤ 81 ページ「Windows: WMI プロトコル」

➤ 81 ページ「UNIX/Sun Solaris/FreeBSD SSH および Telnet プロトコル」

➤ 82 ページ「dmidecode ユーティリティの可用性」

➤ 82 ページ「トラブルシューティングおよび制限事項」

Windows: NTCMD プロトコル

DDM は，**wmic** コマンドを使用して BIOS UUID データを検出します。

WMIC クエリ：

```
wmic path win32_ComputerSystemProduct get uuid /format:list <
%SystemRoot%¥win.ini
```

制限事項 :

wmic ユーティリティがリモート・マシンにインストールされていないと，この属性は設定されません。Windows 2000 以前の場合，標準設定では **wmic** ユーティリティはインストールされていません。

Windows: WMI プロトコル

DDM は，**wmic** コマンドを使用して BIOS UUID データを検出します。

WMI クエリ :

```
SELECT UUID from Win32_ComputerSystemProduct
```

制限事項 :

WMI クラスは，BIOS UUID 属性値を不正確な順序で返します。

UNIX/Sun Solaris/FreeBSD SSH および Telnet プロトコル

ディスカバリは，DMI データに基づいています。このデータには **dmidecode** ユーティリティを使用してアクセスできます。

コマンド :

```
dmidecode | grep UUID
```

制限事項 :

dmidecode ユーティリティがない場合，DDM はデータを取得できません。

dmidecode ユーティリティの可用性

dmidecode ユーティリティを利用できる環境を次に示します。

- Linux i386，x86-64，ia64
- FreeBSD i386，x86-64
- NetBSD i386，x86-64
- OpenBSD i386
- BeOS i386
- Cygwin i386
- Solaris x86

トラブルシューティングおよび制限事項

- BIOS UUID 属性は，UCMDB バージョン 8.02 またはそれ以降でのみ使用できます。
- 次の場合，BIOS UUID 属性は，すべて **0** またはすべて **F** で構成される可能性があります。
  - マシンが VM で，その BIOS UUID 属性が手動で上書きされているか，クリアされている。
  - 内部 DMI（システム）プロバイダに問題があり，実際の UUID を決定できない。これは DMI の標準設定の動作です。
- VM マシンが，ある場所から別の場所にコピーされている場合，異なるマシンで BIOS UUID 属性が同じになる可能性があります。

ソフトウェア製品コード属性のサポート

ソフトウェア製品コード属性 (software_productcode) が Software Element CIT に追加されました。本項では，さまざまなプロトコルを使用したソフトウェア製品コードのディスカバリが DDM でどのようにサポートされているのかについて説明します。

ProductCode プロパティは, 特定の製品リリースに対応する一意の識別子です。たとえば，**「{12345678-1234-1234-1234-123456789012}」**のような GUID 文字列として表されます。この GUID で使用する文字は大文字にする必要があります。バージョンや言語が異なれば, それに応じて ID を変える必要があります。

まったく新しい製品に更新するアップグレードでは，製品コードも変える必要があります。32 ビットおよび 64 ビット・バージョンのアプリケーションのパッケージでは，異なる製品コードを割り当てる必要があります。

本項の内容

➤ 83 ページ「Windows: Shell プロトコル」

➤ 84 ページ「Windows: WMI プロトコル」

Windows: Shell プロトコル

注：製品コード属性は，Windows MSI インストーラの一部であるため，UNIX のようなディストリビュートでは関係ありません。

DDM は，製品コードを決定するために次のレジストリ・キーを解析します。

```
HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥Uninstall
¥{productCode}
```

DDM は，**reg.exe** または **reg_mam.exe** ツールを使用してレジストリへのアクセス権を取得します。

製品コードを決定するための正規表現を次に示します（Python 表記）。

```
"¥¥Uninstall¥¥[¥{¥[¥
( ]*([¥dabcdefABCDEF]{8}(¥-[¥dabcdefABCDEF]{4}){3}-[¥dabcdefABCDEF]{12}).*¥n"
```

Windows: WMI プロトコル

DDM は，製品コードを決定するために次のレジストリ・キーを解析します。

```
HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥Uninstall¥
{productCode}
```

DDM は，**¥¥root¥default WMI** 名前空間の **StdRegProv** クラスを使用してレジストリへのアクセス権を取得します。

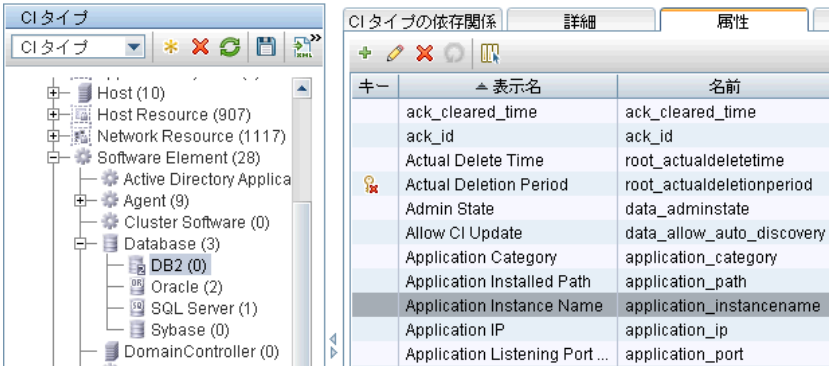
製品コードを決定するための正規表現を次に示します（Python 表記）。

```
"¥¥Uninstall¥¥[¥{¥[¥(
]*([¥dabcdefABCDEF]{8}(¥-[¥dabcdefABCDEF]{4}){3}-[¥dabcdefABCDEF]{12}).*¥n"
```

アプリケーション・インスタンス名属性のサポート

アプリケーション・インスタンス名属性 (application_instancename) が Software Element CIT に追加されました。これにより，DDM は 1 台のマシンに存在する同じアプリケーションの各インスタンスを識別できるようになります。

すべての子 Software Element CIT は，親 (Software Element) の属性を継承します。DDM は，ディスカバリ時に値を適切な SID に置換します。たとえば，J2EE Server アプリケーションでは，application_instancename が j2eeserver_servername に置換されます。データベース・アプリケーションでは，application_instancename が database_dbsid に置換されます。



トラブルシューティングおよび制限事項

レジストリにソフトウェアが複数ある場合，最初のキーからのみデータが取得されます。

第 III 部

管理

第 5 章

ディスカバリ実行

本章では, DDM を使ったシステム・コンポーネントの検出について説明します。

本章の内容

概念

タスク

参照先

ディスカバリ実行 – 概要

［ディスカバリ実行］ページでは，システムのコンポーネントを検出するジョブをアクティブ化できます。DDM をアクティブ化するには，次のいずれかの方法を使用します。

- ［**ベーシック モード**］を使用する方法。設定可能な標準設定のプリファレンスを使って，特定のコンポーネント（インフラストラクチャ，J2EE アプリケーション，データベースなど）に対して DDM を実行します。

  ワークフローの詳細については，94 ページ「ディスカバリ実行 – ベーシック・モードのワークフロー」を参照してください。

  ディスカバリ・ウィザードの詳細については，116 ページ「［ベーシック モード］ウィンドウ」を参照してください。

注 :［ディスカバリ実行］にアクセスすると，標準設定ではベーシック・モードが表示されます。

- ［**アドバンス モード**］を使って DDM を実行する方法。ジョブに変更を加えることで，実行をカスタマイズします。

  ワークフローの詳細については，95 ページ「ディスカバリ実行 – アドバンス・モードのワークフロー」を参照してください。

  ディスカバリ・ウィザードの詳細については，115 ページ「［アドバンス モード］ウィンドウ」を参照してください。

  特定のモジュールを実行する方法の詳細については，『**Discovery and Dependency Mapping Content Guide (英語版)** 』を参照してください。

注：［ディスカバリ実行］の各コンポーネントのヘルプについては，以下を参照してください。

- ［ディスカバリ モジュール］表示枠の詳細については，142 ページ「[ディスカバリ モジュール］表示枠」を参照してください。
- ［詳細]タブの詳細については, 130 ページ「[詳細]タブ」を参照してください。
- ［プロパティ］タブの詳細については，169 ページ「[プロパティ］タブ」を参照してください。
- ［依存関係マップ］タブの詳細については，128 ページ「［依存関係マップ］タブ」を参照してください。

ディスカバリ・ウィザード

ディスカバリ・ウィザードの作成には DDM に関する非常に高度な知識が必要なため，作成を始める前に HP ソフトウェア・サポートに問い合わせることをお勧めします。

ジョブ実行中の権限の表示

ジョブの実行中は，システム内のコンポーネントに接続するためにどの資格情報が使用されているかを知る必要が生じる場合があります。また，実行がネットワーク・パフォーマンスに与える影響（ジョブを日中ではなく夜間に実行すべきかどうかなど）を知る必要が生じる場合もあります。[権限を表示］には，次の図に示すように，ジョブの Jython スクリプト・コマンドのオブジェクトとパラメータが表示されます。

権限	操作	使用状況の詳細	オブジェクトとパラメータ
Shell	exec	Basic login	uname ver
Shell	exec	CPU Info	AIX: lsattr | grep "proc" AIX: prtconf | grep "proc" FreeBSD: dmesg | grep "cpu|Multiprocessor" FreeBSD: dmesg | grep -A 1 "CPU:" FreeBSD: sysctl hw.model hw.ncpu hw.clockrate HPUX: model Linux: cat /proc/cpuinfo SunOS: /usr/sbin/psrinfo -v SunOS: prtconf

注： ここで定義する情報は動的なものではありません。つまり，パターンが変更されても，このダイアログ・ボックス内の情報は更新されません。

詳細については，145 ページ「［ディスカバリの権限］ウィンドウ」を参照してください。

［ディスカバリの権限］ウィンドウの使用例

UNIX システム上で実行しているあるホストを検出するため，Host Connection by Shell ジョブを実行しています。［ディスカバリ ステータス］表示枠にエラー・メッセージが表示され，権限が拒否されたために DDM が SSH 経由でホストにアクセスできなかったことが示されました。［ディスカバリの権限］ウィンドウを表示したところ，そのホストにアクセスするためのコマンドには特定レベルの権限を持つユーザが必要であることがわかりました。SSH プロトコルのウィンドウを確認すると，そこで定義されているユーザに必要なレベルの権限がないことがわかりました。

この問題を解決するには，SSH プロトコルのユーザを変更するか，または外部システムの既存ユーザの権限を更新します。

権限ドキュメント

注： 本項では，Content Pack 4.00 以降で利用できる情報について説明します。

DDM ジョブ，プロトコル，およびジョブ・コンポーネントにアクセスするために必要な権限のリストを表示できます。たとえば, Host Resources by Shell ジョブを実行する場合の基本的なログインに必要な情報を表示できます。

このリストは，PDF ドキュメントで生成され，UCMDB サーバの次のフォルダにあります。**C:¥hp¥UCMDB¥UCMDBServer¥j2f¥AppServer¥webapps¥site.war¥amdocs¥eng¥pdfs¥Permissions.pdf**。

このリストは，モジュールごとに分類されており，次の情報で構成されています。

- モジュール
- ジョブ
- プロトコル
- 操作，使用状況の詳細，オブジェクトとパラメータ

権限ドキュメントのコンテンツの例
Database - Oracle: モジュール名です。

Oracle RAC Topology by Shell: ジョブ名です。

Discovers Oracle RAC Topology by Shell: ジョブ名の説明です。アプリケーションで説明が定義されていない場合，このセクションは除外されます。

Protocol: Shell: SQL，Shell，WMI，SNMP などのプロトコル名です。詳細なリストについては，203 ページ「ドメイン資格情報リファレンス」を参照してください。

操作	使用状況の詳細	オブジェクトとパラメータ
ファイルの読み取り	listener および tnsnames 構成ファイルの解析	cat $ORACLE_HOME¥network¥listener.ora cat $ORACLE_HOME¥network¥admin¥tnsnames.ora

エラー・レポートによる問題の管理

DDM の実行中は，接続障害，ハードウェアの問題，例外，タイムアウトなど，多くのエラーが検出される可能性があります。DDM では，ベーシック・モードとアドバンス・モードのどちらのディスカバリ実行でも，これらのメッセージが表示されます。問題の原因となったトリガ CI からドリルダウンして，エラー・メッセージ自体を表示できます。

DDM は，無視できるエラー（到達不可能なホストなど）と対処の必要なエラー（資格情報の問題，設定ファイルや DDL ファイルの欠落など）を区別します。さらに，その後の実行で同じエラーが発生してもエラーは 1 回しか報告されません。また，1 回しか発生しなかったエラーも報告されます。

重大度レベルの詳細については，63 ページ「重大度レベル」を参照してください。

データベース内のエラー・テーブル

DDM のすべてのエラーは，Probe Manager データベース・スキーマの［discovery_problems］テーブルに保存されています（エラー情報は，サーバへの配信を保証するため，Probe のメモリで処理されるのではなく，データベースに保存されます）。Probe には，各トリガ CI に関する問題の最新のリストが保持されます。各実行の後で，Probe は変化を確認し，それらを［ディスカバリ ステータス］表示枠に表示します。詳細については，132 ページ「［ディスカバリ ステータス］表示枠」を参照してください。

ディスカバリ実行 – ベーシック・モードのワークフロー

このタスクでは，ディスカバリ・ウィザードを使ってシステムとそのコンポーネントのマッピングを開始する方法について説明します。インフラストラクチャ，データベース，または J2EE のディスカバリで各コンポーネントの標準設定値を使用するには，このワークフローを実行します。

注： アドバンス・モードで DDM を実行する方法の詳細については，95 ページ「ディスカバリ実行 – アドバンス・モードのワークフロー」を参照してください。

このタスクには次の手順が含まれています。

1 前提条件

Probe がインストールされていることを確認してください。Probe のインストールの詳細については，16 ページ「DDM Probe のインストール」を参照してください。

ライセンスの詳細については，『**HP Universal CMDB デプロイメント・ガイド**』(PDF) の「HPUniversal CMDB のライセンス・モデル」を参照してください。

2 ディスカバリ ウィザードへのアクセス

詳細については，関連するウィザード（151 ページ「インフラストラクチャ・ウィザード」，159 ページ「J2EE ウィザード」，または 121 ページ「データベース・ウィザード」）を参照してください。

ディスカバリ実行 – アドバンス・モードのワークフロー

このタスクでは，システムとそのコンポーネントのマッピングを開始する方法について説明します。モジュールのコンポーネントをカスタマイズするには，このワークフローを使用します。

注： ベーシック・モードでディスカバリを実行する方法の詳細については，94 ページ「ディスカバリ実行 – ベーシック・モードのワークフロー」を参照してください。

このタスクには次の手順が含まれています。

1 前提条件

a Probe がインストールされていることを確認してください。Probe のインストールの詳細については，16 ページ「DDM Probe のインストール」を参照してください。

ライセンスの詳細については，『**HP Universal CMDB デプロイメント・ガイド**』の「HPUniversal CMDB のライセンス・モデル」を参照してください。

b 関連する package がデプロイされていることを確認してください。

詳細については，『**モデル管理**』の「パッケージ・マネージャ」を参照してください。

2 ネットワーク範囲の決定

検出するネットワークのネットワーク範囲を定義する必要があります。詳細については，188 ページ「[IP 範囲の追加 / 編集] ダイアログ・ボックス」を参照してください。

注 : パターンは，範囲内のすべての IP に接続しようとします。このため，範囲が広いと，ネットワークのパフォーマンスに影響する可能性があります。

3 関連する資格情報の設定

特定のプロトコル（NTCmd，SNMP，TTY，WMI など）を使って DDM からサーバまたはアプリケーションに接続するには，関連する資格情報を設定する必要があります。プロトコル・パラメータの詳細については，203 ページ「ドメイン資格情報リファレンス」を参照してください。［ディスカバリ プローブ設定］ウィンドウの［詳細］表示枠の詳細については，193 ページ「［詳細］タブ」を参照してください。

注 : DDM では，各資格情報を順に使ってホストへの接続が試行されます。成功した資格情報は DDM によって保存されます。このホストに次回接続するときは，最初に成功した資格情報を使って接続が試行されます。

4 関連するジョブのアクティブ化

ネットワーク範囲の定義と資格情報の設定が完了すると，特定のジョブでディスカバリを実行できます。詳細については，**『Discovery and Dependency Mapping Content Guide (英語版) 』**を参照してください。

ヒント : ジョブの詳細な説明を表示できます。モジュールを選択し，［ディスカバリ実行］の［プロパティ］タブの DDM パターン名の下にある［説明］表示枠を見つけます。

例 – SNMP 接続の検出

SNMP 接続を検出するすべてのジョブを検索できます。［**ディスカバリ実行**］>［**ディスカバリ モジュール**］表示枠で，［**ディスカバリ ジョブの検索**］アイコンをクリックします。［**ジョブの検索**］ダイアログ・ボックスで，［**名前**］ボックスに「**SNMP**」と入力し，［**すべて検索**］をクリックします。詳細については，150 ページ「［ジョブの検索］ダイアログ・ボックス」を参照してください。

5 関連するパターンの変更

パターンをカスタマイズして，希少なシステム・コンポーネントを検出できます。パターン記述の詳細については，第 10 章 :「コンテンツ開発と記述」を参照してください。

重要：HP ソフトウェア・サポートに相談せずに標準設定のパターンを変更しないでください。

---

6 DDM プロセスの監視

実行によって検出された CI を監視する方法の詳細については, 139 ページ「[統計結果］表示枠」を参照してください。

a TQL の定義

CMDB から CI および CIT に関する情報を取得する TQL クエリを作成します。詳細については,『**モデル管理**』の「TQL クエリの定義」を参照してください。

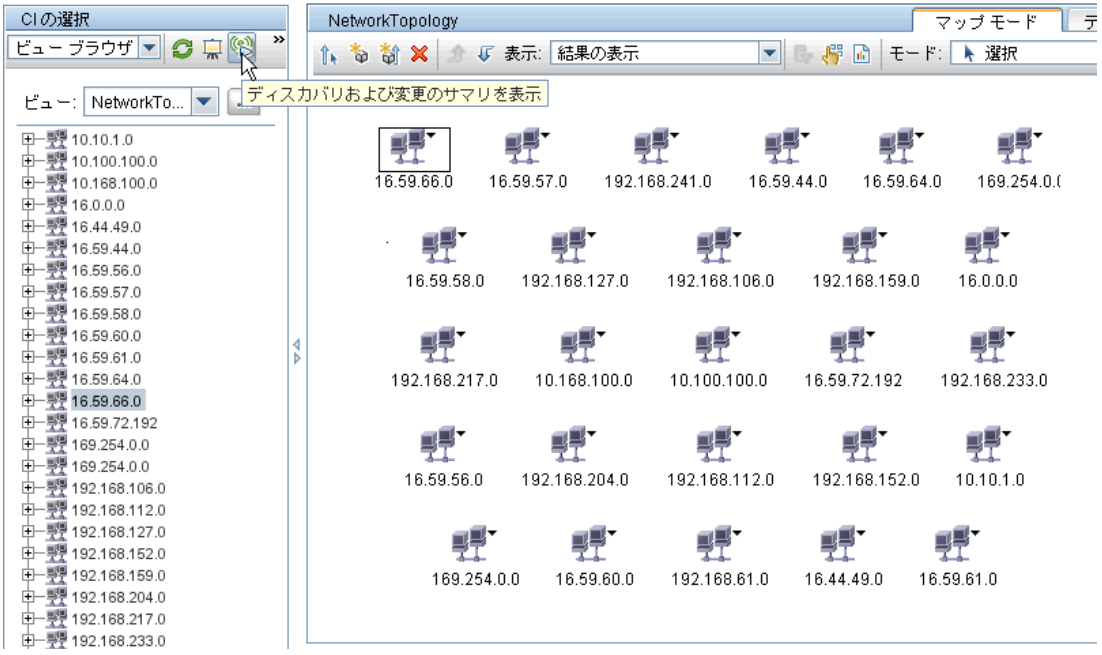
必要な場合は，TQL をトリガして手動でオブジェクトを検出できます。詳細については, 174 ページ「[トリガ TQL]表示枠」表示枠」を参照してください。

b TQL ごとのビューの構築

ビューでは，IT ユニバース・モデル全体のサブセットを構築して，特定のディスカバリに関連する CMDB 内の CI だけを保持できます。詳細については,『**モデル管理**』の「ビュー・マネージャ・ウィンドウ」を参照してください。

例 – 検出された CI インスタンスを表示するビューの作成

HP Universal CMDB によって検出されたインスタンスの数を表示するには，［**管理**］＞［**モデリング**］＞［**IT ユニバース マネージャ**］を選択します。作成したビューが次の図のように表示されます。



7 統計結果の表示

ジョブの全体的な統計情報を表示したり，時間範囲や Probe で結果をフィルタ処理したりできます。HP Universal CMDB にログインして［ディスカバリ実行］にアクセスするたびに，統計データが更新され，選択したモジュールまたはジョブの最新のデータが表示されます。

統計データを使った作業の詳細については，139 ページ「［統計結果］表示枠」を参照してください。

［ステータス スナップショット表示］表示枠にアクセスして，検出された CI を表示することもできます。詳細については，第 8 章 :「ステータス・スナップショット表示」を参照してください。

8 結果のトラブルシューティング

DDM の結果から，どのエラーが報告されているかを確認できます。詳細については，99 ページ「エラーの管理」を参照してください。

エラーの管理

このタスクでは，実行中に発生した問題を調べる方法について説明します。

注：重大度レベルなどの詳細については，93 ページ「エラー・レポートによる問題の管理」を参照してください。

このタスクには次の手順が含まれています。

- 99 ページ「前提条件」
- 99 ページ「ディスカバリ ウィザードの実行またはジョブの選択」
- 99 ページ「問題 CI の特定」
- 100 ページ「問題のトラブルシューティング」

1 前提条件

DDM をセットアップしてください。詳細については，94 ページ「ディスカバリ実行 – ベーシック・モードのワークフロー」または 95 ページ「ディスカバリ実行 – アドバンス・モードのワークフロー」を参照してください。

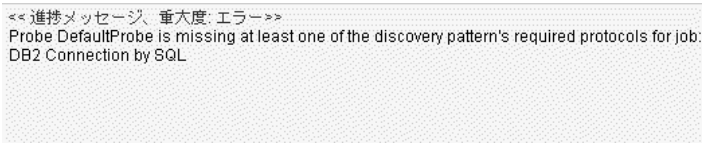
2 ディスカバリ ウィザードの実行またはジョブの選択

ベーシック・モードでは，標準設定のジョブ用のエラー・メッセージを表示できます。アドバンス・モードでは，1 つのジョブ，1 つのモジュール，またはすべてのモジュール用のエラー・メッセージを表示できます。ベーシック・モードでウィザードを実行する方法の詳細については，94 ページ「ディスカバリ実行 – ベーシック・モードのワークフロー」を参照してください。ジョブの実行の詳細については，95 ページ「ディスカバリ実行 – アドバンス・モードのワークフロー」を参照してください。

3 問題 CI の特定

[ディスカバリ ステータス] 表示枠を使って，エラー・メッセージまでドリルダウンできます。詳細については，132 ページ「[ディスカバリ ステータス] 表示枠」を参照してください。

例
DDM によって，次のエラー・メッセージが表示されています。



4 問題のトラブルシューティング

- 致命的なエラーの場合は，HP ソフトウェア・サポートまでご連絡ください。
- その他のエラーの場合は，CI を確認します。たとえば，Probe の範囲内にないトリガ CI によってエラーが表示されることがあります。
- 通信ログの設定の詳細については，260 ページ「[実行オプション] 表示枠」を参照してください。
- 問題の管理の詳細については，93 ページ「エラー・レポートによる問題の管理」を参照してください。

DDM Probe のジョブ情報の表示

このタスクでは，DDM Probe の MySql データベースに保存されたジョブ情報（ジョブ・スレッドやトリガ CI など）を呼び出す方法について説明します。JMX コンソールで作業します。

このタスクには次の手順が含まれています。

- 101 ページ「MBean 操作へのアクセス」
- 101 ページ「呼び出す操作の特定」
- 113 ページ「操作の実行」

1 MBean 操作へのアクセス

次の手順で，DDM Probe 上の JMX アプリケーションにアクセスし，JMX 操作を呼び出します。

a Web ブラウザを起動して次のアドレスを入力します。

```
http://< マシン名または IP アドレス >.< ドメイン名 >:1977/
```

＜マシン名または IP アドレス＞には，DDM Probe がインストールされているマシンを指定します。ユーザ名とパスワードでログインしなければならない場合もあります。

b ［**Local_ ＜マシン名または IP アドレス＞**］＞［**type=JobsInformation**］リンクをクリックします。

2 呼び出す操作の特定

［Mbean View］ページで次の操作を見つけます。

activateJob

ジョブの名前を入力してボタンをクリックすると，そのジョブが直ちにアクティブ化されます。この操作によって，「＜ジョブ名＞ was triggered」のようなメッセージが返されます。

注：次のメッセージは，ジョブがアクティブ化されず，ジョブに関する情報が Probe のデータベース内に存在しない場合に表示されます。

Job ' ＜ジョブ名＞ ' does not exist in the Jobs Execution table（job was not activated!）

---

activateJobOnDestination

ジョブとトリガ CI の名前を入力してボタンをクリックすると，特定のトリガ CI に対してジョブが直ちにアクティブ化されます。この操作によって，「The operation returned with the value: Job ＜ジョブ名＞ was triggered on destination ＜ CI 名＞」のようなメッセージが返されます。

注：［**JobID**］フィールドと［**triggerCI**］フィールドの両方が必須です。

start/stop

これらの操作は，JobsInformation サービスを開始および停止します。これらの操作を使用せずに，Probe 自体を再起動してください。

viewJobErrorsSummary

ジョブの名前を入力すると，そのジョブに関して報告されたエラー・メッセージのリストが返されます。これには，エラーの重大度，エラーが報告された最終日時，およびエラーが発生したトリガ CI の数が含まれます。

ジョブ操作パラメータの詳細については，111 ページ「ジョブ操作パラメータ」を参照してください。

［**Number of trigger CIs**］カラムのエントリをクリックすると，1 つのジョブの，エラーが発生したトリガ CI のリストが［**viewJobTriggeredCIsWithErrorId**］ページに表示されます。

viewJobExecHistory

ジョブの名前を入力すると，ジョブ呼び出しの履歴が取得されます。ジョブ呼び出しを示すメッセージが表示されます（最後の呼び出しが最初に表示されます）。

ジョブ操作パラメータの詳細については，111 ページ「ジョブ操作パラメータ」を参照してください。

呼び出しごとに，トリガされた CI の数と合計実行時間が表示されます。［Execution Details］カラムには，ジョブの実行回数が表示されます。ジョブの実行途中で Probe が終了し，その後実行が再開された場合や，ジョブの実行中に停電の期間があった場合は，複数の時間範囲が表示されます。

viewJobProblems

ジョブの名前またはトリガ CI の名前を入力すると，問題のあるトリガ CI のリストが取得されます。

注：少なくとも 1 つのフィールドに入力する必要があります。

ジョブ操作パラメータの詳細については，111 ページ「ジョブ操作パラメータ」を参照してください。

viewJobResultCiInstances

1 つ以上のパラメータを入力すると，ジョブによって検出された CI のリストが返されます。

ジョブ操作パラメータの詳細については，111 ページ「ジョブ操作パラメータ」を参照してください。

［Object State Holder］カラムに，CMDB で定義された CI または関係のコードが表示されます。一般的な CIT のオブジェクト状態ホルダを作成する方法の詳細については，359 ページ「Jython のライブラリとユーティリティ」の **modeling.py** を参照してください。ObjectStateHolder メソッドの詳細については，『**HP Discovery and Dependency Mapping API Reference（英語版）**』を参照してください。

viewJobResults

1 つ以上のパラメータを入力すると，ジョブによって検出された CI のリストが返されます。

ジョブ操作パラメータの詳細については，111 ページ「ジョブ操作パラメータ」を参照してください。

［**Hide Touched CIs Info**］が［**True**］に設定されているときは，結果ページに以下の情報が表示されます。

カラム	説明
Job Name	［**jobID**］フィールドを空のままにすると表示されます。 DDM に表示されるジョブ名。 ジョブをクリックすると，［**viewJobStatus**］ページに移動し，そのステータスとスケジュールの情報が表示されます。
CI Type	1 つの CIT の結果のみが表示されるようにリストをフィルタ処理するときにクリックします。
Total CI	クリックすると［**viewJobResultCiInstances**］ページに移動し，ジョブによって検出されたすべての CI のリストが表示されます。
Triggered CI	クリックすると［**viewJobTriggeredCIs**］ページに移動し，ジョブによって検出されたすべてのトリガ CI のリストが表示されます。
Last Discover Time	ジョブが呼び出された日時。

［**Hide Touched CIs Info**］が［**False**］に設定されているときは，結果ページに以下の情報が表示されます。

カラム	説明
Job Name	［**jobID**］フィールドを空のままにすると表示されます。 DDM に表示されるジョブ名。 ジョブをクリックすると，［**viewJobStatus**］ページに移動し，そのステータスとスケジュールの情報が表示されます。
CI Type	1 つの CIT の結果のみが表示されるようにリストをフィルタ処理するときにクリックします。
Touched CIs	クリックすると［**viewJobResultCiInstances**］ページに移動し，ジョブによって検出された「**検出済み CI**」である CI のリストが表示されます。詳細については，111 ページ「ジョブ操作パラメータ」を参照してください。

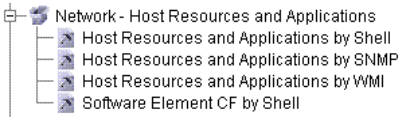
カラム	説明
Non Touched CIs	クリックすると［**viewJobResultCiInstances**］ページに移動し，ジョブによって検出された「検出済み CI」でない CI のリストが表示されます。
Triggered CIs for Touched CIs	クリックすると［**viewJobTriggeredCIs**］ページに移動し，ジョブに含まれる「検出済み CI」であるトリガ CI のリストが表示されます。
Triggered CIs for Non Touched CIs	クリックすると［**viewJobTriggeredCIs**］ページに移動し，ジョブに含まれる「検出済み CI」でないトリガ CI のリストが表示されます。
Last Discover Time	ジョブが呼び出された日時。

結果ページの結果をさらにフィルタするには，いずれかのフィールドにテキスト・フィルタを入力して［**Search**］ボタンをクリックします。

viewJobsStatuses

［**viewJobsStatuses**］ボタンをクリックすると，すべてのジョブのステータスとスケジュールの情報が返されます。結果をフィルタ処理することもできます。詳細については，111 ページ「ジョブ操作パラメータ」を参照してください。

結果ページに以下の情報が表示されます。

カラム	説明
No.	リスト内のジョブの番号。
Job Name	次のような，DDM に表示されるジョブ名。  ジョブをクリックすると，［**viewJobStatus**］ページに移動し，そのステータスとスケジュールの情報が表示されます。

カラム	説明
Status	Probe によって計算されたジョブのステータスの重大度。 **Blocked :** 使用されていません。 **Removed :** このジョブはアクティブでなくなりました。 **Running :** このジョブは現在実行中です。 **Scheduled :** このジョブの実行がスケジュール設定されています。ジョブのスケジュール設定の詳細については，146 ページ「［ディスカバリ スケジューラ］ダイアログ・ボックス」を参照してください。 赤色の背景は，スレッドが期待より長く実行されており，応答しなくなった可能性があることを示します。緑色の背景は，ジョブが期待どおりに実行されていることを示します。
Errors	特定のジョブに関するエラーの数。クリックすると［**viewJobErrorsSummary**］ページに移動し，このジョブに関して報告されたエラー・メッセージのリストが表示されます。
Triggered CI	このジョブによって実行されたトリガ CI。クリックすると，［**viewJobTriggeredCIs**］ページに移動します。
Last Invocation	ジョブが最後に実行された日時。
Next Invocation	ジョブが次に実行される日時。
Last Total run duration（seconds）	最後の呼び出しでジョブを実行するのにかかった合計時間。この結果を，ジョブの実行にかかる平均時間と比較してください。時間の違いは，おそらくジョブがほかのジョブの完了を待機する期間によるものです。
Avg run duration（seconds）	過去のすべての呼び出しから計算された，ジョブの平均実行時間。
Recurrence	ジョブが呼び出された回数。クリックすると［**viewJobExecHistory**］ページに移動し，ジョブ呼び出しの履歴が取得されます。
Results	このジョブによって検出された CIT の数。クリックすると［**viewJobResults**］ページに移動し，CIT が表示されます。

viewJobStatus

ジョブの名前を入力すると，そのステータスとスケジュールの情報が返されます。

ジョブ操作パラメータの詳細については，111 ページ「ジョブ操作パラメータ」を参照してください。

結果ページに以下の情報が表示されます。

カラム	説明
Threading info	呼び出しによって作成されたワーカー・スレッドの総数，空きワーカー・スレッド数，および応答しなくなったワーカー・スレッド数。
Total work time	Probe がこのジョブを実行するのにかかった時間。
Tasks waiting for execution	アクティブ化を待機しているトリガ CI の数を含む，ジョブのリスト。
Max. threads	このジョブに提供されているスレッドの数。
Progress	現在の（つまり，特定の実行がアクティブ化されてからの）実行のサマリ。 たとえば，「Progress: 2017 / 6851 destinations（29%）」は，6851 個の CI のうち，2017 個の CI がすでに実行されていることを意味します。

カラム	説明
Working Threads information	**Thread Name:** このジョブを現在実行しているスレッド。クリックすると，［**viewJobThreadDump**］ページに移動します。スレッドが長時間実行されているときは，このページを使って，長時間実行がスレッドの正常な動作によるものであり，問題の発生によるものではないことを確認する必要があります。 **Curr Dest. ID**: ジョブが実行されているホストの名前。 **Curr Dest. IP**: ジョブによる情報検出の対象となっている IP。 **Work Time（秒）:** このスレッドの実行時間。 **Communication Log :** クリックすると［**viewCommunicationLog**］ページに移動し，Probe とリモート・マシン間の接続を記録した XML ファイルが表示されます。詳細については，260 ページ「［実行オプション］表示枠」の［**通信ログの作成**］フィールドを参照してください。

カラム	説明
［Discovery Jobs Information］テーブル	**Status:** Probe によって計算されたジョブのステータスの重大度。詳細については，106 ページ「Status」を参照してください。 **Errors :** クリックすると［**viewJobErrorsSummary**］ページに移動し，このジョブに関して報告されたエラー・メッセージのリストが表示されます。 **Triggered CIs :** クリックすると［**viewJobTriggeredCIs**］ページに移動し，ジョブに含まれるトリガ CI のリストが表示されます。 **Last invocation :** ジョブが最後に実行された日時。 **Next invocation :** ジョブが次に実行される日時。 **Last Total run duration（seconds）:** 詳細については，106 ページ「Last Total run duration（seconds）」を参照してください。 **Avg run duration（seconds）:** 詳細については，106 ページ「Avg run duration（seconds）」を参照してください。 **Recurrence:** ジョブが呼び出された回数。クリックすると［**viewJobExecHistory**］ページに移動し，ジョブ呼び出しの履歴が表示されます。
Results	このジョブによって検出された CIT の数。クリックすると［**viewJobResults**］ページに移動し，CIT が表示されます。

viewJobTriggeredCIs

1 つ以上のパラメータを入力すると，ジョブに含まれるトリガ CI のリストが返されます。

ジョブ操作パラメータの詳細については，111 ページ「ジョブ操作パラメータ」を参照してください。

結果ページに以下の情報が表示されます。

カラム	説明
No.	リスト内のジョブの番号。
Triggered CI ID	このジョブによって検出された CI インスタンス。クリックすると［**viewJobResults**］ページに移動し，各インスタンスの CIT に関する情報が表示されます。
Last Execution	ジョブが最後に実行された日時。
Service Exec. Duration(ms)	最後の呼び出しでジョブを実行するのにかかった最大時間（ジョブが実行されなかった期間を含む）。この結果を合計実行継続時間と比較してください。 たとえば，複数のジョブを同時に実行するときに CPU が 1 つしかない場合は，あるジョブが別のジョブの完了を待機しなければならないことがあります。サービス継続時間にはこの待機時間が含まれますが，合計継続時間には含まれません。
Total Exec. Duration(ms)	最後の呼び出しでジョブを実行するのにかかった時間（ジョブが実行されなかった期間を除く）。
Last Run Status	最後の実行のステータス（実行が成功したか失敗したか）。失敗した場合は，クリックすると［**viewJobProblems**］ページに移動し，問題が発生したトリガ CI のリストが表示されます。
hostID	ジョブが実行されているマシンの名前。
ip_address	ジョブによる情報収集の対象となっている IP アドレス。 **注 :** Probe がホストの ID を検出しなかった場合（たとえば，特定の IP ではなく検出の範囲がジョブに指定された場合など）は，ホストに関する情報が見つからないことがあります。このような場合は，hostID，ip_address，および ip_domain フィールドに［(Empty)］という文字列が表示されます。
ip_domain	DDM に表示される Probe 名。

viewJobTriggeredCIsWithErrorId

注： この操作は，内部インタフェースの一部であり，ヘルパー関数として機能します。トリガ CI の情報を表示するときは，このページではなく［**viewJobTriggeredCIs**］ページを使用してください。

ジョブ操作パラメータ

次のリストは，ジョブ操作パラメータを示します。

- **ciType**: CI タイプの名前（ip，host など）。
- **data**: 検出されたオブジェクトに関する情報を含む DiscoveryResults テーブル内のテキスト・フィールド。次に例を示します。

```
<object class="ip">
<attribute name="ip_Probename" type="String">EBRUTER02</attribute>
<attribute name="ip_address" type="String">16.59.58.200</attribute>
<attribute name="ip_domain" type="String">DefaultDomain</attribute>
</object>
```

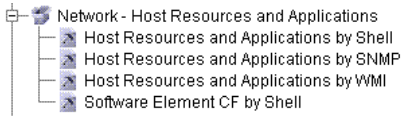
- **Error Id:** Jobs_Problems テーブルに表示されるエラー・メッセージのハッシュ文字列（エラー・ハッシュ ID）。
- **HideRemovedJobs: True** にすると，以前に実行され，現在の実行に関係しないジョブが表示されなくなります。
- **Hide Touched CIs Info**: 検出済み CI は，以前の呼び出しで検出された CI です。これらの CI に関する情報は DDM にすでに存在するため，Probe からサーバに情報を再度送信する必要はありません。これらの CI が必要なものであり，これらの CI に対してエージング・メカニズムを適用する必要がないことが，サーバによって確認されています。エージングの詳細については，『**モデル管理**』の「エージング・メカニズム ― 概要」を参照してください。

  True にすると，CIT ごとの CI の総数とトリガ CI の総数がテーブルに表示されます。**False** にすると，CI およびトリガ CI の総数が検出済み CI と未検出 CI に分けてテーブルに表示されます。

- **includeNonTouched**: 未検出 CI を表示するようにテーブルをフィルタ処理できます。次のように，未検出 CI のみの表示，すべての CI（検出済みと未検出の両方）の表示，または CI の表示なしを選択します。

	Non-touched CIs	All CIs	No CIs
(boolean)includeTouchedCis	○ True ◉ False	◉ True ○ False	○ True ◉ False
(boolean)includeNonTouchedCis	◉ True ○ False	◉ True ○ False	○ True ◉ False

- **includeNonTouchedCIs**: 「**includeNonTouched**」を参照してください。
- **includeTouched**: 検出済み CI を表示するようにテーブルをフィルタ処理できます。検出済み CI のみの表示，すべての CI（検出済みと未検出の両方）の表示，または CI の表示なしを選択します。
- **includeTouchedCIs**: 「**includeTouched**」を参照してください。
- **jobID**: ジョブの名前（**Host Resources and Applications by SNMP** など）。



- **maxRows**: 結果テーブルに表示される最大行数。標準設定値は 100 または 1000 です。
- **maxTriggeredCIs**: 「**maxRows**」を参照してください。
- **objectID**: CMDB オブジェクト ID。
- **showRemovedJobs**: 以前に実行されたが，現在は実行がスケジュール設定されていないジョブに関する情報を表示します。これらのジョブは **REMOVED** という状態になります。
- **showResults**: ［**Show Results**］カラムを表示するかどうかを指定します。［Show Results］カラムが表示されている場合は，［**viewJobsStatuses**］から［**viewJobResults**］に移動できます。
- **triggerCI**: ジョブのトリガの CMDB オブジェクト ID。
- **triggeredCiID**: 「**triggerCI**」を参照してください。

3 操作の実行

➤ ボタンをクリックして操作を実行します。操作の実行結果を示すメッセージが表示されます。

Reload: JMX インタフェースの自動再ロード間の秒数。**0** にすると，インタフェースは再ロードされません。（操作が追加または削除された場合に）現在のページを手動で再ロードするには，［**Reload**］をクリックします。

Unregister: 使用しないでください（ビューから実行中のアプリケーションにアクセスできなくなります)。

［ディスカバリ実行］のユーザ・インタフェース

本項では，次の項目について説明します。

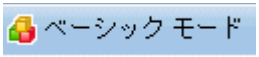
［アドバンス モード］ウィンドウ

説明	モジュールとジョブの表示と管理，ジョブのアクティブ化，およびジョブの進行状況の追跡ができます。 アドバンス・モードには以下の表示枠があります。 ➤ **［ディスカバリ モジュール］表示枠：**個々のモジュールにはジョブが含まれています。特定の CI グループを検出するには，モジュールまたはジョブをアクティブ化します。詳細については，142 ページ「［ディスカバリ モジュール］表示枠」を参照してください。 **注**：［ディスカバリ実行］にアクセスすると，標準設定ではベーシック・モードが表示されます。 ➤ **［詳細］タブ：**モジュールの CI の管理と CI の統計情報の表示ができます。詳細については，130 ページ「［詳細］タブ」を参照してください。 ➤ **［プロパティ］タブ：**モジュールとジョブのプロパティを表示して管理できます。詳細については，169 ページ「［プロパティ］タブ」を参照してください。 ➤ **依存関係マップ：**プロセスの進行状況をリアルタイムで視覚的に表示します。詳細については，128 ページ「［依存関係マップ］タブ」を参照してください。 **利用方法：**［**管理**］>［**ディスカバリ**］>［**ディスカバリ実行**］。
重要情報	［ディスカバリ実行］で行った個々の変更内容は，CMDB に配信および保管されます。変更内容は，そこから Probe に送信されます。変更内容が Probe に送信されたことを確認するには，**C:¥hp¥DDM¥DiscoveryProbe¥root¥logs¥** にある **wrapperProbe.log** ファイルを開き，次の行を検索します。 processing document domainScopeDocument.bin Processing document domainScopeDocument.bin is done. **注**：［ディスカバリ実行］にアクセスすると，標準設定ではベーシック・モードが表示されます。
ほかのタスク	95 ページ「ディスカバリ実行 – アドバンス・モードのワークフロー」

[ベーシック モード] ウィンドウ

説明	DDM ウィザードを使って，インフラストラクチャ，データベース，および J2EE アプリケーションを検出できます。 ベーシック・モードには以下の表示枠があります。 ➤ **ウィザードのリスト** : 実行するウィザードを選択できます。詳細については，151 ページ「インフラストラクチャ・ウィザード」，121 ページ「データベース・ウィザード」，または 159 ページ「J2EE ウィザード」を参照してください。 ➤ **[サマリ] 表示枠** : ウィザードを実行したり，DDM の実行を停止したりできます。詳細については，後述の 117 ページ「[サマリ] 表示枠」 ➤ [**ディスカバリ概要**] 表示枠。 ➤ 簡単な実行ステータスを表示し，問題のあるトリガ CI にドリルダウンします。エラーの管理の詳細については，132 ページ「[ディスカバリ ステータス] 表示枠」を参照してください。 ➤ 統計結果を表示します。詳細については，139 ページ「[統計結果] 表示枠」を参照してください。 この表示枠は，コンポーネントに対してディスカバリを実行すると表示されます。 **利用方法 :** [**管理**] > [**ディスカバリ**] > [**ディスカバリ実行**]。
重要情報	**注** : [ディスカバリ実行] にアクセスすると，標準設定ではベーシック・モードが表示されます。 アドバンス・モードの詳細については，115 ページ「[アドバンス モード] ウィンドウ」を参照してください。
ほかのタスク	94 ページ「ディスカバリ実行 – ベーシック・モードのワークフロー」
関連リンク	90 ページ「ディスカバリ実行 – 概要」

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	クリックすると，ウィザードのリストが更新されます。
	（現在表示されています）クリックすると，設定可能な標準設定のプリファレンスを使って，特定のコンポーネント（インフラストラクチャ，J2EE アプリケーション，データベースなど）に対して DDM が実行されます。
	DDM の実行時に，ジョブ，パターン，その他に変更を加えて実行をカスタマイズする必要がある場合にクリックします。詳細については，115 ページ「[アドバンス モード] ウィンドウ」を参照してください。

［サマリ］表示枠

説明	ディスカバリ・ウィザードを実行できます。 **利用方法：**［**管理**］＞［**ディスカバリ**］＞［**ディスカバリ実行**］。
重要情報	ウィザードがすでに実行されているかどうかに応じて，［サマリ］表示枠に以下の情報が表示されます。 ➤ ウィザードがまだ実行されていない場合は，［サマリ］表示枠にウィザードで実行する手順と［**設定および実行**］ボタンが表示されます。 ➤ ウィザードがすでに実行されている場合は，［サマリ］表示枠に実行パラメータのサマリ，［**設定**］ボタン，および［**ディスカバリの停止**］ボタンが表示され，［ディスカバリの進行状況］表示枠に前の実行結果が表示されます。 ディスカバリを実行するには，左側の表示枠でウィザードを選択し，［**設定**］または［**設定および実行**］をクリックしてディスカバリ・ウィザードを開きます。 ディスカバリ実行を停止するには，［**ディスカバリの停止**］をクリックします。
ほかのタスク	94 ページ「ディスカバリ実行 – ベーシック・モードのワークフロー」

［追加する CI の選択］ダイアログ・ボックス

説明	選択したジョブで実行する CI を選択できます。 **利用方法 :** ➤［**ディスカバリ**］＞［**ディスカバリ実行**］。［詳細］タブで［**ディスカバリ ステータス**］表示枠を見つけます。［**CI の追加**］ボタンをクリックします。 ➤ データベース・ウィザードの［Oracle TNSName ファイルの検索］ページで［**CI の追加**］ボタンをクリックします。

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
［CI の追加］ボタン	**注 :** エラー・ステータスを持つ CI を選択してトリガ・リストに追加すると，［**追加**］ボタンをクリックしたときにメッセージが表示されます。
CI の検索	［検索結果］表示枠に表示される CI の数を制限できるフィルタが含まれています。 ➤ **ディスカバリ TQL で :** ディスカバリ TQL を選択して，その TQL と一致する CI を検索します。 ➤ **次を含む CI だけを表示 :** 特定のテキストを含む CI を検索するには，ここにそのテキストを入力します。 ➤ **厳密な一致 :** テキスト・ラベルが正確に一致する CI を検索するときに選択します （標準設定では，テキストの一部を入力して検索します。たとえば，IP CI の中から「**10**」を検索すると，アドレスに「10」が含まれるすべての IP が見つかります。しかし，「**10**」と入力して［**厳密な一致**］を選択すると，何も見つかりません）。 ➤ **検索 :** クリックすると，検索結果が表示されます。

GUI 要素	説明
検索結果	フィルタに設定された条件に対応するトリガされた CI のリストが表示されます。[トリガされた CI] 表示枠のリストに CI を追加するには，その CI を選択します。複数の選択を行うことができます。 ➤ **CIT**: 選択したトリガされた CI の CI タイプ。 ➤ **CI**: トリガされた CI のラベル。 ➤ **関連ホスト**：トリガされた CI に関連するホストのラベル。 ➤ **関連 IP**: 関連するホストの IP。 **ページ：**CI のリストは，複数のページに分割して表示されます。［ページ］ボックス内の数値は，現在表示されているページ番号を示します。ほかのページを表示するには，上向き矢印と下向き矢印を使用するか，またはページ番号を入力して **Enter** キーを押します。 ページに表示される CI の数を決めるには，上向きボタンまたは下向きボタンを右クリックし，必要な数を選択します。標準設定は 25 です。

［ディスカバリ TQL を選択してください］ダイアログ・ボックス

説明	ジョブにトリガ TQL を追加できます。 **利用方法：**［トリガ TQL］表示枠で［**TQL の追加**］ボタンをクリックします。

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
<ディスカバリ TQL 名>	選択した CIT を CMDB から検索できる TQL。
TQL のプレビュー	要素の上にカーソルを置くと，詳細が表示されます。

［プローブの選択］ダイアログ・ボックス

説明	Probe のリストをフィルタ処理できます。 **利用方法：**［**ディスカバリ実行**］＞［**詳細**］タブで，次のいずれかの［フィルタ］ボタンをクリックします。 ➤［**トリガされた CI**］表示枠の［フィルタ］ボタン。メニュー・オプションの詳細については，132 ページ「[ディスカバリ ステータス］表示枠」を参照してください。 ➤［**統計情報**］表示枠の［フィルタ］ボタン。メニュー・オプションの詳細については，139 ページ「［統計結果］表示枠」を参照してください。

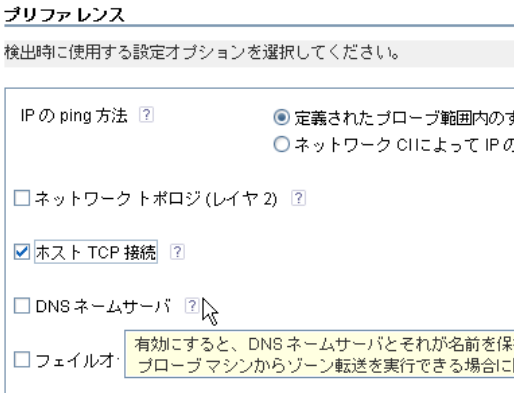
［構成アイテムのプロパティ］ダイアログ・ボックス

説明	CI のプロパティを表示できます。 **利用方法：**［**検出された CI**］ダイアログ・ボックスで，CI を右クリックして［**プロパティ**］を選択します。
重要情報	詳細については，『**モデル管理**』の「[構成アイテムのプロパティ］ダイアログ・ボックス」を参照してください。

［新規ディスカバリ ジョブの作成］ウィンドウ

説明	ジョブを作成できます。 **利用方法：**［ディスカバリ モジュール］表示枠でモジュールを右クリックし，［**ジョブの新規作成**］を選択します。
重要情報	➤ ジョブ名の最大長は 50 文字です。 ➤ ジョブ名の最初の文字を数値にすることはできません。
Useful Links	このウィンドウ内の表示枠の詳細については，以下を参照してください。 ➤ 131 ページ「[ディスカバリ ジョブの詳細］表示枠」 ➤ 173 ページ「[パラメータ］表示枠」 ➤ 174 ページ「[トリガ TQL］表示枠」 ➤ 268 ページ「[グローバル構成ファイル］表示枠」 ➤ 169 ページ「[ディスカバリ スケジューラ］表示枠」

データベース・ウィザード

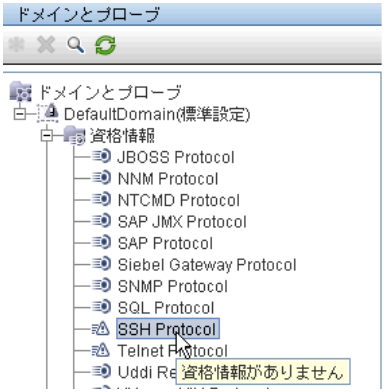
説明	データベース（DB2，Oracle，Microsoft SQL，Sybase など）を検出できます。 **利用方法：**［**管理**］＞［**ディスカバリ**］＞［**ディスカバリ実行**］＞［**ベーシック モード**］。左側の表示枠のリストからデータベース・ウィザードを選択します。［**設定および実行**］をクリックします。
重要情報	詳細については，以下のように疑問符アイコンの上にポインタを置いてください。 
ウィザード・マップ	データベース・ディスカバリ・ウィザードには，以下のページが含まれています。 **データベース・ウィザード** ＞ 資格情報の定義 ＞ データベース・ポートのスキャン ＞ カスタム JDBC ドライバ ＞ Oracle TNSName ファイルの検索 ＞ ディスカバリのスケジュール ＞ サマリ

資格情報の定義

説明	各プロトコルの接続データを設定できます。
重要情報	➤ プロトコルの設定は，何を検出する必要があるか，およびサイトのネットワークでどのプロトコルがサポートされているかによって異なります。 ➤ プロトコルのリストについては，203 ページ「ドメイン資格情報リファレンス」を参照してください。 ➤ このウィザードの一般的な情報については，121 ページ「データベース・ウィザード」を参照してください。
ウィザード・マップ	データベース・ディスカバリ・ウィザードには，以下のページが含まれています。 データベース・ウィザード ＞ **資格情報の定義** ＞ データベース・ポートのスキャン ＞ カスタム JDBC ドライバ ＞ Oracle TNSName ファイルの検索 ＞ ディスカバリのスケジュール ＞ サマリ

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します)。

GUI 要素	説明
	選択したプロトコル・タイプの新しい接続詳細を追加します。
	プロトコルを削除します。
	プロトコルを編集します。クリックすると，[プロトコル パラメータ] ダイアログ・ボックスが開きます。

GUI 要素	説明
	プロトコルを上下に移動します。DDM は，リスト内の先頭のプロトコルから順に，すべてのプロトコルを実行します。
プロトコル	クリックすると，ユーザの資格情報を含むプロトコルの詳細が表示されます。 **注**：資格情報が見つからない場合は，次の図に示すアイコン  で表されます。 

データベース・ポートのスキャン

説明	ポート自体の検出を可能にし，続けてデータベースを検出できるようにします
重要情報	このウィザードの一般的な情報については，121 ページ「データベース・ウィザード」を参照してください。
ウィザード・マップ	データベース・ディスカバリ・ウィザードには，以下のページが含まれています。 データベース・ウィザード > 資格情報の定義 > **データベース・ポートのスキャン** > カスタム JDBC ドライバ > Oracle TNSName ファイルの検索 > ディスカバリのスケジュール > サマリ

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	ポート・リストにポートを追加するときにクリックします。［新しいポートを追加］ダイアログ・ボックスが開きます。ポートを選択し，［**OK**］をクリックします。 既存のシステム・ポートを編集するには，［新しいポートを追加］ダイアログ・ボックスで［**既知のシステム ポートを編集**］をクリックします。［既知のシステム ポートを編集］ダイアログ・ボックスが開きます。ポートを選択して［**編集**］ボタンをクリックします。開いたダイアログ・ボックスで，エントリに変更を加え，［**OK**］をクリックします。 リストにポートを追加するには，［既知のシステム・ポートを編集］ダイアログ・ボックスで［**追加**］ボタンをクリックします。ポートの詳細（名前，番号，およびタイプ）を入力し，［**OK**］をクリックします。
	ポートを選択してこのボタンをクリックすると，リストからそのポートが削除されます。

カスタム JDBC ドライバ

説明	DB2 および Sybase JDBC ドライバの JAR ファイルを選択できます。
重要情報	このウィザードの一般的な情報については，121 ページ「データベース・ウィザード」を参照してください。
ウィザード・マップ	データベース・ディスカバリ・ウィザードには，以下のページが含まれています。 データベース・ウィザード ＞ 資格情報の定義 ＞ データベース・ポートのスキャン ＞ **カスタム JDBC ドライバ** ＞ Oracle TNSName ファイルの検索 ＞ ディスカバリのスケジュール ＞ サマリ

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
DB2 JDBC ドライバ	このチェック・ボックスを選択して［**ファイルをインポート ...**］をクリックすると，DB2 JDBC インストールに含まれる以下の適切な JAR ファイルが検索されます。 ➤ db2java.zip ➤ db2jcc.jar ➤ db2jcc_license_cu.jar ➤ db2jcc_license_cisuz.jar
Sybase JDBC ドライバ	このチェック・ボックスを選択して［**ファイルをインポート ...**］をクリックすると，Sybase JDBC インストールに含まれる jconnectXXX.jar JAR ファイルが検索されます。

Oracle TNSName ファイルの検索

説明	Oracle データベースを検出できます。Oracle データベースを検出するのに必要なデータベース情報（ポート，ホスト，SID など）が含まれている TNSNames.ora 設定ファイルの場所を指定します。
重要情報	このウィザードの一般的な情報については，121 ページ「データベース・ウィザード」を参照してください。
ウィザード・マップ	データベース・ディスカバリ・ウィザードには，以下のページが含まれています。 データベース・ウィザード ＞ 資格情報の定義 ＞ データベース・ポートのスキャン ＞ カスタム JDBC ドライバ ＞ **Oracle TNSName ファイルの検索** ＞ ディスカバリのスケジュール ＞ サマリ

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
サーバ・ホスト	TNSNames.ora ファイルがあるホストを選択します。[**CI の追加**] ボタンをクリックして，これらのホストを表すトリガ CI を選択します。詳細については，118 ページ「[追加する CI の選択] ダイアログ・ボックス」を参照してください。 ➤ **CIT**: 選択したトリガされた CI の CI タイプ。 ➤ **CI**: トリガされた CI のラベル。 ➤ **関連ホスト**: トリガ CI に関連するホストのラベル。 ➤ **関連 IP**: 関連するホストの IP。
TNSNames.ora ファイルの場所	サーバ・ホスト・システム内の TNSNames.ora ファイルの場所を入力します。複数の場所を（カンマで区切って）入力できます。パスの末尾を区切り文字（たとえば，c:¥temp¥）にすると，ファイル名は **tnsnames.ora** であると仮定されます。

ディスカバリのスケジュール

説明	特定のジョブのスケジュールを定義できます。
重要情報	このウィザードの一般的な情報については，121 ページ「データベース・ウィザード」を参照してください。
ウィザード・マップ	データベース・ディスカバリ・ウィザードには，以下のページが含まれています。 データベース・ウィザード ＞ 資格情報の定義 ＞ データベース・ポートのスキャン ＞ カスタム JDBC ドライバ ＞ Oracle TNSName ファイルの検索 ＞ **ディスカバリのスケジュール** ＞ サマリ

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	［プロパティ］タブの［ディスカバリ スケジューラ］表示枠で，時間テンプレートを定義します。詳細については，169 ページ「[ディスカバリ スケジューラ］表示枠」を参照してください。
次からのディスカバリの開始を許可	スケジュールをを実行する時間を選択します。
繰り返し間隔	スケジュールを実行する頻度を選択します。

サマリ

説明	ディスカバリを実行する前に，ウィザードの定義を確認できます。
重要情報	実行に変更を加えるには，［**戻る**］をクリックします。 このウィザードの一般的な情報については，121 ページ「データベース・ウィザード」を参照してください。
ウィザード・マップ	データベース・ディスカバリ・ウィザードには，以下のページが含まれています。 データベース･ウィザード ＞ 資格情報の定義 ＞ データベース･ポートのスキャン ＞ カスタム JDBC ドライバ ＞ Oracle TNSName ファイルの検索 ＞ ディスカバリのスケジュール ＞ **サマリ**

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
実行	このボタンをクリックすると，ディスカバリが実行されます。

［依存関係マップ］タブ

説明	ディスカバリ・プロセスの進行状況をリアルタイムで視覚的に表示します。このマップには以下の項目が表示されます。 ➤ ジョブによってトリガされた CI ➤ アクティブ化されたジョブの結果として検出された CI **利用方法 :** ［ディスカバリ実行］ウィンドウの［**依存関係マップ**］タブをクリックします。
重要情報	［依存関係マップ］タブに表示される情報は，［ディスカバリ モジュール］表示枠で選択したレベルによって異なります。 具体的な表示内容は次のとおりです。 ➤ ［**ディスカバリ モジュール］ルート**を選択し，［**アクティブなディスカバリ ジョブだけを表示**］チェック・ボックスを選択した場合は，アクティブなジョブとそれらの依存関係のみが［依存関係マップ］に表示されます。 ➤ ［**ディスカバリ モジュール**］ルートを選択し，［**アクティブなディスカバリ ジョブだけを表示**］チェック・ボックスをクリアした場合は，すべてのディスカバリ・ジョブとそれらの依存関係が［依存関係マップ］に表示されます。 ➤ **モジュール**を選択した場合は，そのモジュールのアクティブおよび非アクティブなジョブを示すトポロジ・マップが表示されます。 ➤ **ジョブ**を選択した場合は，モジュールのトポロジ・マップ内でそのジョブが強調表示されます。
関連リンク	141 ページ「［検出済み CIs］ダイアログ・ボックス」

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
<右クリック・メニュー>	右クリック・メニューを使って，ジョブ，CI，またはリンクの詳細（CMDB 内の（特定タイプの）CI インスタンスの数や，特定のジョブで作成された CI インスタンスの数など）を表示できます。 選択したオブジェクトに応じて，以下のメニュー・オプションが表示されます。 ➤ ジョブを選択した場合： **検出された CI の表示：**クリックすると，そのジョブによって検出された CI が表示されます。クエリをフィルタ処理するには，メニューから CIT を選択します。 **トリガされた CI の表示：**クリックすると，そのジョブをトリガした CI が表示されます。 ➤ CI を選択した場合： **すべてのインスタンスを表示：**クリックすると，その CI タイプのすべての CI が表示されます。 ➤ CI からジョブへのリンクを選択した場合： **ジョブにトリガされた CI の表示：**クリックすると，そのジョブをトリガした（選択したタイプの）CI が表示されます。 ➤ ジョブから CI へのリンクを選択した場合： **検出されたインスタンスの表示：**クリックすると，そのジョブによって検出された（選択したタイプの）CI が表示されます。
<ツールバー>	詳細については，『**モデル管理**』の「ツールバー・オプション」を参照してください。
<ツールチップ>	CI またはジョブの上にポインタを置いたままにすると，説明が表示されます。
アクティブなディスカバリ・ジョブだけを表示	［ディスカバリ モジュール］表示枠で［ディスカバリ モジュール］ルートを選択すると，このチェック・ボックスが表示されます。 選択すると，（任意のモジュールの）アクティブなジョブがすべて表示されます。

［詳細］タブ

説明	モジュールとジョブの表示と管理, DDM プロセスの進行状況の追跡，および検出時のエラーの管理ができます。 **利用方法 :** ［ディスカバリ実行］の［**詳細**］タブをクリックします。
重要情報	［詳細］タブに表示される情報は，［ディスカバリ モジュール］表示枠で選択したレベルによって異なります。 具体的な表示内容は次のとおりです。 ➤ ［**ディスカバリ モジュール**］ルートまたは**ディスカバリ・モジュール**を選択した場合は，［ディスカバリ ステータス］および［統計結果］表示枠が開き，すべてのアクティブなジョブと実行中に検出されたエラーに関する情報と統計値が表示されます。詳細については，132 ページ「［ディスカバリ ステータス］表示枠」および 139 ページ「［統計結果］表示枠」を参照してください。 ➤ **ジョブ**を選択した場合は，［ジョブの詳細］，［ディスカバリ ステータス］，および［統計結果］表示枠が表示されます。詳細については，131 ページ「［ディスカバリ ジョブの詳細］表示枠」，132 ページ「［ディスカバリ ステータス］表示枠」，および 139 ページ「［統計結果］表示枠」を参照してください。 ➤ 複数のジョブまたはモジュールを選択した場合は，［選択した項目］表示枠が表示されます。詳細については，138 ページ「［選択した項目］表示枠」を参照してください。
ほかのタスク	93 ページ「エラー・レポートによる問題の管理」

[ディスカバリ ジョブの詳細] 表示枠

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	クリックすると，[ディスカバリ リソース] ウィンドウ内のパターンに移動します。
	選択すると，パターンによって検出された CI とリンクのマップが（リストの代わりに）表示されます。このボタンをクリックすると，[検出されたクラスのマップ] ウィンドウが開きます。選択したパターンが，その CI と関係とともに表示されます。CIT の上にカーソルを置くと，ツールチップに説明が表示されます。
	クリックすると，特定のパターンに対して定義された権限が表示されます。詳細については，145 ページ「[ディスカバリの権限] ウィンドウ」を参照してください。 これらの権限を編集する方法の詳細については，272 ページ「[権限の編集] ダイアログ・ボックス」を参照してください。
ディスカバリ・パターン	ジョブで CI を検出するのに必要なパターン。
検出された CI	ジョブによって検出された CI。
CI タイプを入力	このジョブの CI をトリガする CIT。
ジョブ名	ジョブの名前と説明です。 **重要**：ジョブ名の最初の文字を数値にすることはできません。

［ディスカバリ ステータス］表示枠

説明	実行ステータスを表示して問題のあるトリガ CI にドリルダウンしたり，実行中の DDM に発生した問題（不正な資格情報など）を特定したりできます。 ［**ベーシック モード**］では，選択したジョブ・タイプ（インフラストラクチャ，データベース，または J2EE アプリケーション）の以前の実行結果を表示できます。 ［**アドバンス モード**］では，選択したモジュールまたはジョブ，またはすべてのモジュールの以前の実行結果を表示できます。 **利用方法 :** ➤ ［ベーシック モード］で［ディスカバリ概要］表示枠を見つけます。 ➤ ［アドバンス モード］で，モジュールまたはジョブを選択して［**詳細**］タブをクリックし，［ディスカバリ ステータス］表示枠を見つけます。
重要情報	➤ SHIFT キーおよび CTRL キーを使って，リスト内の複数の隣接する CI および隣接しない CI を選択できます。 ➤ ［ディスカバリ ステータス］表示枠には，［ディスカバリ モジュール］表示枠の［アドバンス モード］で選択したレベルに応じて，すべてのモジュール，特定のモジュール，または特定のジョブの情報が表示されます。 ➤ この表示枠の情報は 30 秒ごとに自動的に更新されます。
ほかのタスク	99 ページ「エラーの管理」 「アプリケーション・ディスカバリのステータス・チェック（ビューの再検出）」「ビュー・マネージャ」（『**モデル管理**』）
関連リンク	93 ページ「エラー・レポートによる問題の管理」

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	クリックすると，上部の表示枠に戻ります。
	クリックすると，問題のあるトリガ CI にドリルダウンします。 **注**：このアイコンが表示されるのは，エラーまたは警告のリンクからドリルダウンできるときだけです。
	クリックすると，ステータス表示が更新されます。
	クリックすると，新しく検出された CI が追加されます。詳細については，118 ページ「［追加する CI の選択］ダイアログ・ボックス」を参照してください。
	CI が必要なくなった場合に，その CI をリストから削除するときにクリックします。CI が該当するジョブから削除されます。

GUI 要素	説明
	クリックして，メニューから次のオプションを選択します。 ➤ **ステータス別 :** 次のいずれかのオプションに従ってトリガ CI のリストを表示します。 ➤ **すべて :** すべてのトリガ CI を表示します。 ➤ **プローブの待機中 :** ディスパッチできる状態にあり，Probe による受信を待っているトリガ CI を表示します。 ➤ **実行中 :** Probe 上で実行されているアクティブなトリガ CI を表示します。 ➤ **実行中（削除中）:** ［**トリガ CI**］リストから削除されようとしているトリガ CI を表示します。 ➤ ［**成功**］，［**失敗**］，［**警告**］: 選択したステータスを持つ CI のみを表示します。 ➤ **プローブで:** 選択した Probe によってトリガされた CI のみを表示します。クリックすると，［Probe の選択］ダイアログ・ボックスが開きます。 ➤ **ディスパッチ・タイプで :** 次のいずれかのオプションに従って CI のリストを表示します。 ➤ **すべて** : ジョブを手動でアクティブ化するときに使用する CI とジョブを自動的にアクティブ化するときに使用するディスカバリ TQL の両方を表示します。 ➤ **手動で追加しました** : ジョブを手動でアクティブ化するときに使用する CI を表示します。 ➤ **ディスカバリ TQL で**: ジョブを自動的にアクティブ化するときに使用する CI を表示します。 ➤ **リセット :** クリックすると，すべてのフィルタが削除されます。
	クリックすると，障害の説明を含むメッセージ・ボックスが表示されます（CI を右クリックし，［**エラー詳細の表示**］を選択してメッセージを表示することもできます）。
	クリックすると，［トリガされた CI］ダイアログ・ボックスが開き，その CI に関する追加情報が表示されます。詳細については，177 ページ「［トリガされた CI］ウィンドウ」ウィンドウ」を参照してください。

GUI 要素	説明
	➤ 特定のトリガ CI の統計情報が表示されます。
	クリックすると，ディスカバリが再実行されます。
	CI を検索します。
<ドリルダウン>	ジョブまたはモジュールからドリルダウンできます。 ➤ ジョブからドリルダウンすると，そのジョブに含まれるトリガ CI のリストが表示されます。 ➤ モジュールからドリルダウンすると，モジュール内のジョブと各ジョブによって返された CI の数のリストが表示されます。さらに，ジョブからそのトリガ CI にドリルダウンします。 **注 :** 同じトリガ CI を複数のジョブに含めることができます。

GUI 要素	説明
＜CI の右クリック・メニュー＞	CI を右クリックすると，以下のオプションが表示されます。 ➤ **エラー詳細の表示 :** この CI によって返された各種のエラーのリストが表示されます。重大度の詳細については，63 ページ「重大度レベル」を参照してください。 ➤ **[削除] :** 選択すると，ジョブから CI が削除されます。この CI は，複数のジョブに含まれる場合でも，そのジョブからのみ削除されます。 ➤ **検出の再実行 :** 特定の CI または CI のセットを実行するには，該当する CI を選択します。選択した CI は，Probe が実行する予定である CI のリスト（[**プローブの待機中**]）に追加されます。 ➤ **トリガされた CI の結果を表示 :** DDM は，一時的な要求を Probe に送信し，特定のトリガ CI に対して実行されたジョブの最新の結果（検出された CI の CIT 名と数）を取得します。 この一時的な要求は，ジョブを実行するものではなく，Probe のデータベースに格納されている以前のジョブ実行の結果を取得するものです。このトリガ CI に対してジョブがまだ実行されていない場合は，メッセージが表示されます。詳細については，175 ページ「[トリガされた CI の結果を表示] ダイアログ・ボックス」を参照してください。 Probe に通信ログが存在しない場合は，メッセージが表示されます。常に通信ログを作成するように DDM を設定できます。詳細については，260 ページ「［実行オプション］表示枠」を参照してください。 ➤ **デバッグ :** 次のいずれかを選択します。 ➤ **通信ログの表示 :** Probe とリモート・マシン間の通信に関する情報を含むログが開きます。条件として，[**通信ログの作成**] が［**常時**］または［**失敗時**］に設定されている必要があります。詳細については，260 ページ「[実行オプション］表示枠」を参照してください。 ➤ **パターンへ移動 :** ジョブに含まれているパターンが[ディスカバリ リソースの管理］に表示されます。 ➤ **ジョブへ移動 :** 当該 CI を含むジョブが表示されます。 ➤ **スクリプトの編集 :** スクリプト・エディタで開くスクリプトを選択します。 ➤ **アンディスパッチ :** トリガ CI を削除します。

GUI 要素	説明
失敗	Error または Fatal の重大度を返した CI を表示します。 ディスカバリを再実行するには, そのジョブを右クリックします。 エラー・メッセージを表示するには, そのジョブをダブルクリックします。 ジョブを非アクティブ化または再実行するには, そのエラーを右クリックします。
実行中	実行される順番を待っているトリガ CI の数が表示されます。クリックすると, 実行されるのを待っているジョブが表示されます。
検索対象	特定の Probe, 関連するホスト, または関連する IP を検索するには, このボックスに名前の一部を入力し, [**検索**] をクリックします。
進行状況	このインジケータには, 現在の（つまり, 特定の実行がアクティブ化されてからの）実行のサマリが表示されます。
成功	実行に成功した（エラーが発生しなかった）CI の数が表示されます。 クリックすると, 正常に完了したジョブ（および各ジョブに含まれる CI の数）が表示されます。 CI を選択し, CI の右クリック・メニューを使って情報を表示します。 **警告表示 :** クリックすると, 各ジョブの警告メッセージが表示されます。 メッセージをダブルクリックすると, 警告付きで正常に完了した CI が表示されます。 メッセージを右クリックすると, CI の右クリック・メニューが表示されます。
合計	ジョブのすべてのトリガ CI のステータスが表示されます。[**警告**] または [**エラー**] ステータスをダブルクリックすると, [メッセージ] ダイアログ・ボックスが表示されます。
プローブの待機中	Probe または実行を待機しているトリガ CI。

［選択した項目］表示枠

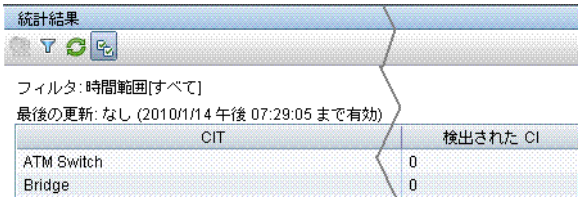
含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
<右クリック・メニュー>	**スケジューリングの編集**: クリックすると, 特定のジョブのスケジュールを定義する［ディスカバリ スケジューラ］が開きます。詳細については，169 ページ「[ディスカバリ スケジューラ］表示枠」を参照してください。
ただちに呼び出す	➤ チェック・マークは, トリガされた CI が Probe に到達するとすぐにディスカバリ・ジョブが実行されることを示します。その場合は,［プロパティ］タブの［**新たにトリガされた CI で直ちに呼び出し**］チェック・ボックスが選択されます。 ➤ このカラムにチェック・マークが付いていない場合は，スケジュール・マネージャで定義されたスケジュールに従ってジョブが実行されます。
ジョブ名	ジョブの名前です。
スケジュール情報	［ディスカバリ スケジューラ］で定義されたジョブのスケジュール情報。
トリガ TQL	ジョブをアクティブ化した TQL の名前。詳細については，174 ページ「［トリガ TQL］表示枠」を参照してください。

［統計結果］表示枠

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	CI を選択して［インスタンスの表示］ボタンをクリックすると，CI インスタンスとそれらの属性が表示されます。［検出された CI］ダイアログ・ボックスが開きます。 以下の場合は，メッセージが表示されます。 ➤ このジョブによって検出されたすべての CI が，別のジョブによってすでに検出されていた場合。 ➤ このジョブによって検出されたすべての CI が削除されていた場合。 ➤ CI インスタンスが以前のバージョンで検出された場合（バージョン 7.0 では，以前のバージョンで検出された CI のインスタンスを表示できません）。 **注**： ➤ 行をダブルクリックして CI インスタンスを表示することもできます。 ➤ インスタンス化されたインスタンスがない CIT も表示されます。
	CIT に関する統計情報を表示する時間範囲または Probe を選択します。 ➤ **時間範囲別**： ➤ **すべて：**すべてのジョブ実行の統計情報が表示されます。 ➤ **現在から/最新/直近の 1 時間 /直近の 1 日/直近の 1 週間**：CIT に関する統計情報を表示する期間を選択します。 ➤ **カスタム範囲：**クリックすると，［統計時間範囲のカスタマイズ］ダイアログ・ボックスが開きます。［**開始**］と［**終了**］に日付を入力するか，矢印をクリックしてカレンダーから日時を選択します。日付を削除するには，［**リセット**］をクリックします。 ➤ **プローブで**：特定の Probe の統計情報を表示するには，このオプションを選択して［プローブの選択］ダイアログ・ボックスを開きます。

GUI 要素	説明
	クリックすると，サーバの最新データが取得されます（［統計情報］表示枠内のジョブの結果は自動的に更新されません）。
	**宣言されているすべての CI タイプを表示します**：標準設定では，検出された CIT のみがテーブルに表示されます。つまり，検出された CI の数が 0 よりも多い場合，［検出された CI］カラムに CIT が表示されます。このボタンをクリックすると，［検出された CI］の値が 0 でもジョブによって検出できる CI がすべて表示されます。 
＜カラム・タイトル＞	CIT の順序を昇順から降順あるいは降順から昇順に変更するには，カラム・タイトルをクリックします。
＜タイトルの右クリック＞	次のオプションから選択できます。 ➤ **カラムを非表示**：選択すると，特定のカラムが非表示になります。 ➤ **全カラムを表示**：カラムが非表示のときに表示されます。 ➤ **カスタマイズ**：カラムの表示と非表示を切り替えたり，テーブル内のカラムの順序を変更したりするときに選択します。［カラム］ダイアログ・ボックスが開きます。 ➤ **自動サイズ変更カラム**：選択すると，内容の長さに合わせてカラムの幅が変更されます。 詳細については，『**参照情報**』の「［カラムの選択］ダイアログ・ボックス」
CIT	検出された CIT の名前。
作成済み	選択した期間または選択した Probe で作成された CIT インスタンスの数。

GUI 要素	説明
削除済み	選択した期間または選択した Probe で削除された CIT インスタンスの数。
検出された CI	CI タイプごとに検出された CI の数。
フィルタ	［時間範囲別］ボタンで設定された時間範囲。
前回更新	特定のジョブに関して統計情報テーブルが最後に更新された日時。
合計	各カラム内の CI の総数。
更新済み	選択した期間内に更新された CI インスタンスの数。

［検出済み CIs］ダイアログ・ボックス

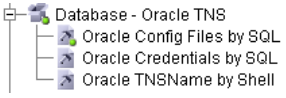
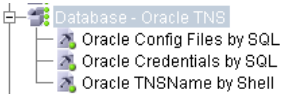
説明	あるジョブによって検出された CIT の CI インスタンスを表示できます。 **利用方法 :** ➤ ［統計結果］表示枠で，CIT を選択して［**インスタンスの表示**］ボタンをクリックします。 ➤ ［依存関係マップ］タブで，［**検出された CI の表示**］または［**すべてのインスタンスを表示**］を選択します。
重要情報	➤ ［＜ジョブ名＞で検出］ウィンドウには，［要素インスタンス］ウィンドウと同じ情報が含まれています。詳細については，『**モデル管理**』の「［要素インスタンス］ダイアログボックス」を参照してください。 ➤ ［依存関係マップ］で［**検出された CI の表示**］と［**すべてのインスタンスを表示**］のどちらを選択するかによって，選択したジョブによって検出されたすべての CI か，選択したタイプのすべての CI のどちらかが表示されます。

［ディスカバリ モジュール］表示枠

説明	モジュールとジョブを表示して管理できます。各モジュールには，特定の CI を検出するのに必要なジョブが含まれています。 **利用方法：**［**管理**］＞［**ディスカバリ**］＞［**ディスカバリ実行**］。［**ベーシック モード**］と呼ばれる標準設定のビューには，［ディスカバリ ウィザード］が表示されます。J2EE，データベース，またはインフラストラクチャのディスカバリを実行できます。すべてのモジュールを表示するには，［**アドバンス モード**］をクリックします。
重要情報	**注意：**モジュールの削除は，DDM プロセスを十分に理解している管理者のみが行ってください。 **廃止事項：**下位互換性とアップグレードのために残された不要なモジュールが複数含まれています。新しいインストールではこれらのモジュールを使用しないでください。 **モジュールなし：**ほかのどのモジュールにも含まれないジョブが含まれています。

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	**すべて更新：**モジュールを更新します。
	**ジョブの検索：**クリックすると，［ジョブの検索］ダイアログ・ボックスが開きます。たとえば，SNMP 接続を検出するすべてのジョブを検索するには，［フィルタ］アイコンをクリックします。［**ジョブの検索**］ダイアログ・ボックスで，［**名前**］ボックスに「**SNMP**」と入力し，［**すべて検索**］をクリックします。詳細については，150 ページ「［ジョブの検索］ダイアログ・ボックス」を参照してください。
	1 つ以上のモジュールで 1 つ以上のジョブを実行できます。 ジョブまたはモジュールを選択し，［**アクティブ化**］をクリックします。
	停止するジョブまたはモジュールを選択し，［**非アクティブ化**］をクリックします。

GUI 要素	説明
	モジュールのルートを表します。 モジュールを作成するには, このアイコンを右クリックして作成するモジュールの名前を入力します。 **注：**名前には大文字と小文字の区別があります。[ディスカバリ モジュール] リストでは，大文字で始まる名前が小文字で始まる名前よりも前に表示されます。
	モジュールを表します。
	ジョブを表します。クリックすると，そのジョブに関する情報が表示されます。パターンの説明を表示するには，ジョブの上にポインタを置いたままにします。 ジョブは, パターンやその他のリソースから派生された設定情報を含み, モジュールをアクティブ化または非アクティブ化するときになどにユーザによって制御されるエンティティです。 右クリック・メニューの詳細については，144 ページ「右クリック・メニュー」を参照してください。
	緑色の 1 つの点は，モジュールのジョブの一部がアクティブ化されていることを示します。 
	緑色の 3 つの点は，モジュールのすべてのジョブがアクティブ化されていることを示します。 

GUI 要素	説明
	感嘆符は，1 つ以上のジョブで DDM プロセスに影響を与える問題（プロトコル接続障害など）が発生していることを示します。 問題の理由を表示するには，［ディスカバリ ステータス］表示枠の［**(エラー表示)**］リンクをクリックします。詳細については，137 ページ「失敗」を参照してください。 **注**：［**すべて更新**］ボタンのクリックによって問題が解決すると，問題のインジケータは表示されなくなります。
	クリックすると，設定可能な標準設定のプリファレンスを使って，特定のコンポーネント（インフラストラクチャ，J2EE アプリケーション，データベースなど）に対して DDM が実行されます。詳細については，116 ページ「[ベーシック モード] ウィンドウ」を参照してください。
	（現在表示されています）DDM の実行時に，ジョブ，パターン，その他に変更を加えることで実行をカスタマイズする必要がある場合にクリックします。

右クリック・メニュー

GUI 要素	説明
アクティブ化	モジュールのすべてのジョブを実行するには，そのモジュールをクリックします。特定のジョブを実行するには，そのジョブを選択してアクティブ化します。 ディスカバリ・モジュールは，各ジョブに記述されているタイプの CIT と関係を検出し，CMDB に配置します。たとえば，Class C IPs by ICMP ジョブは Depend，IP，および Member の CIT と関係を検出します。
ジョブの新規作成	クリックすると，［新規ディスカバリ ジョブの作成］が開きます。詳細については，120 ページ「[新規ディスカバリ ジョブの作成] ウィンドウ」を参照してください。
モジュールの新規作成	クリックし，モジュールのルートの新しい名前を定義します。 注 : モジュール名の最大長は 50 文字です。
非アクティブ化	モジュールまたはジョブの実行を停止します。
すべてのジョブを非アクティブ化	このオプションを表示するには，［**ディスカバリ・モジュール**］をクリックします。
削除	クリックし，警告メッセージに対して［**はい**］を選択します。

GUI 要素	説明
ジョブの削除	クリックし，警告メッセージに対して［**はい**］を選択します。
パターンへ移動	クリックし，［ディスカバリ リソースの管理］ウィンドウでパターンを表示，編集します。
スケジュールの編集	クリックすると，特定のジョブのスケジュールを定義する［ディスカバリ スケジューラ］が開きます。
ジョブの名前の変更	クリックすると，［名前を入力してください］ダイアログ・ボックスが開きます。ジョブの新しい名前を入力します。 **注 :** アクティブなジョブの名前は変更できません。
今すぐ実行	クリックすると，選択したトリガ CI を使ってジョブが再実行されます。
名前を付けて保存 ...	ジョブを複製するときにクリックします。

［ディスカバリの権限］ウィンドウ

説明	ジョブの権限データを表示できます。 **利用方法 :**［**ディスカバリ実行**］＞［**アドバンス モード**］。ジョブを選択します。［**詳細**］タブで［**ディスカバリ ジョブの詳細**］表示枠を見つけます。［**権限を表示**］ボタンをクリックします。
関連リンク	➤ 91 ページ「ジョブ実行中の権限の表示」 ➤ 269 ページ「［必要な権限］表示枠」 ➤ 272 ページ「［権限の編集］ダイアログ・ボックス」

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	権限オブジェクトを Excel，PDF，RTF，CSV，または XML 形式 でエクスポートします。詳細については，『**モデル管理**』の「参照モード」を参照してください。
オブジェクトとパラメータ	該当する Jython スクリプトに含まれるコマンド。

GUI 要素	説明
操作	実行されるアクション。
権限	ジョブに定義されているプロトコルの名前。
使用状況の詳細	プロトコルの使用状況の説明。

[ディスカバリ スケジューラ] ダイアログ・ボックス

説明	特定のジョブのスケジュールを定義できます。たとえば，クラス C ネットワークに対する IP ping スイープの実行を毎日午前 6:00 に開始できます。 **利用方法 :** ➤ ジョブを右クリックして [**スケジュールの編集**] を選択します。 ➤ [ディスカバリ実行] ウィンドウの [プロパティ] タブの [ディスカバリ スケジューラ] 表示枠で，[**スケジューラの編集**] ボタンをクリックします。
重要情報	[ディスカバリ スケジューラ] ではディスカバリの頻度（毎日，毎月など）を定義しますが，時間テンプレートではジョブの実行時間（日中，夜間，週末のみなど）を定義します。同じスケジュールを異なる時間テンプレートとともに実行できます。たとえば，毎日実行するスケジュールを定義し，午前 1:00 から午前 5:00 までの間に実行する時間テンプレートを定義したとします。このように定義されたジョブは，毎日午前 1:00 から午前 5:00 までの間に実行されます。さらに，実行時間が異なる 2 つ目の時間テンプレートを定義し，その時間テンプレートを同じスケジュールとともに使用できます。 時間テンプレートの作成の詳細については，149 ページ「[時間テンプレートを編集] ダイアログ・ボックス」を参照してください。

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
式を検証	クリックすると，入力した cron 式が検証されます。
<頻度>	➤ **1 回 :** タスクを 1 回だけ実行するように定義します。 ➤ **間隔 :** 連続する実行の間隔を定義します。 ➤ **日ごと :** タスクを毎日実行します。 ➤ **週ごと :** タスクを毎週実行します。 ➤ **月ごと :** タスクを毎月実行します。 ➤ **Cron:** cron 式を正しい形式で入力します。
日数	（［**月ごと**］を選択すると表示されます）このボタンをクリックして，アクションを実行する日を選択します。［必要な日数を追加］ダイアログ・ボックスが開きます。該当するチェック・ボックスを選択して，必要な日を選択します。複数の日を選択できます。 ➤ **すべて選択** : すべての日を選択します。 ➤ **すべて選択解除 :** 選択したすべての日をクリアします。
曜日	（［**週ごと**］を選択すると表示されます）アクションを実行する曜日を 1 つまたは複数選択します。
終了時刻	アクションの実行を停止する日時を選択します。それには，［**終了時刻**］チェック・ボックスを選択し，カレンダーを開いて日時を選択し，［**OK**］をクリックします。 **注** : この手順はオプションです。終了日時を指定しない場合は，［**終了時刻**］チェック・ボックスを未選択のままにしてください。

GUI 要素	説明
呼び出し時間	［**日ごと**］，［**週ごと**］，または［**月ごと**］を選択すると表示されます）アクションをアクティブ化する時間を選択します。このボタンをクリックすると，［時間を選択］ダイアログ・ボックスが開きます。該当するチェック・ボックスを選択して，必要な時間を選択します。複数の時間を選択できます。 ➤ **すべて選択** : すべての時間を選択します。 ➤ **すべて選択解除** : 選択したすべての時間をクリアします。 **注** : ［**呼び出し時間**］ボックスに手動で時間を入力することもできます。複数の時間はカンマで区切り，時間の後には「AM」または「PM」を付けます。手動で入力するアクション時間は，1 時間や 30 分の単位に限定されません。時間と分を自由に組み合わせて指定できます。HH:MM AM の形式を使用します（8:15 AM, 11:59 PM など）。
呼び出し時間	（［**1 回**］を選択すると表示されます）アクションの実行を開始する日時を選択します。カレンダーを開いて日時を選択するか，標準設定値をそのまま使用します。
＜月＞	（［**月ごと**］を選択すると表示されます）アクションを実行する月を 1 つまたは複数選択します。
繰り返し間隔	（［**間隔**］を選択すると表示されます）連続する実行の間隔の値を入力し，必要な時間単位（分，時間，または日）を選択します。
開始時刻	アクションの実行を開始する日時を選択します。それには，［**開始時刻**］チェック・ボックスを選択し，カレンダーを開いて日時を選択し，［**OK**］をクリックします。
タイム・ゾーン	Probe によるジョブのスケジュール設定の基準となるタイムゾーンを選択します。 標準設定値は，**<< ディスカバリ Probe のタイムゾーン >>** です。この場合，Probe は固有のシステム定義のタイムゾーンを使用します。これにより，異なる地理的位置では異なる時間に実行するようにスケジュールを設定できます。 すべての Probe が同時に作業を開始するようにするには，特定のタイムゾーンを選択します（これは，Probe のシステムの日時とタイムゾーンが正しく設定されていることが前提です）。

［TQL 出力用 Probe 制限の編集］ダイアログ・ボックス

説明	トリガ TQL を実行する Probe を変更できます。Probe の選択の詳細については，202 ページ「Probe の選択」を参照してください。 **利用方法**：ジョブを選択して次のボタンをクリックします。［**ディスカバリ実行**］＞［**プロパティ**］タブ＞［**トリガ TQL**］表示枠＞［**プローブ制限**］ボックス。

［時間テンプレートを編集］ダイアログ・ボックス

説明	ジョブのスケジュールを設定するときに使用する時間テンプレートを定義できます。 **利用方法**：［時間テンプレート］ダイアログ・ボックスで［**追加**］ボタンをクリックします。
重要情報	時間テンプレートの名前は一意でなければなりません。
関連リンク	146 ページ「［ディスカバリ スケジューラ］ダイアログ・ボックス」

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
次の期間毎日 - 開始	ジョブを実行する日ごとのスケジュールを定義します。時刻を手動で入力することもできます。時間と分を自由に組み合わせて指定できます。
時間テンプレート名	一意の名前を入力します。
週	ジョブを実行する週ごとのスケジュールを定義します。クリックして時間を選択します。隣のセルを選択するには，クリックしてポインタをテーブルにドラッグします。時間をクリアするには，セルを再度クリックします。

［ジョブの検索］ダイアログ・ボックス

説明	特定の条件に合わせてジョブを検索できます。 検索結果は，［詳細］タブの［選択した項目］表示枠に表示されます。 **利用方法 :** ［ディスカバリ モジュール］表示枠で［**ディスカバリ ジョブの検索**］ボタンをクリックします。

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
方向	モジュールを順方向または逆方向に検索します。
すべて検索	検索条件に一致するすべてのジョブが強調表示されます。
検索対象 ディスカバリ・ジョブ	次のいずれかを選択します。 ➤ **名前 :** ジョブの名前または名前の一部を入力します。 ➤ **入力タイプ :** ジョブをトリガした CI。このボタンをクリックすると［構成アイテム タイプを選択してください］ダイアログ・ボックスが開きます。検索する CI タイプを特定します。 ➤ **出力タイプ :** アクティブ化されたジョブの結果として検出される CI。
次を検索	検索条件に一致する次のジョブが強調表示されます。

インフラストラクチャ・ウィザード

説明	システム内のネットワークに対してディスカバリを実行できます。 **利用方法：**［**管理**］＞［**ディスカバリ**］＞［**ディスカバリ実行**］＞［**ベーシック モード**］。左側の表示枠のリストから**インフラストラクチャ・ウィザード**を選択します。［**設定および実行**］をクリックします。
ウィザード・マップ	インフラストラクチャ・ディスカバリ・ウィザードには，以下のページが含まれています。 **インフラストラクチャ・ウィザード** ＞ IP 範囲の定義 ＞ 資格情報の定義 ＞ プリファレンス ＞ ディスカバリのスケジュール ＞ サマリ

IP 範囲の定義

説明	Probe ごとに検出するネットワークの範囲を設定できます。定義した範囲内のアドレスから結果が取得されます。範囲から除外する IP アドレスも定義できます。
重要情報	ここで行った変更は，グローバルな設定に影響します。 このウィザードの一般的な情報については，151 ページ「インフラストラクチャ・ウィザード」を参照してください。
ウィザード・マップ	インフラストラクチャ・ディスカバリ・ウィザードには，以下のページが含まれています。 インフラストラクチャ・ウィザード ＞ **IP 範囲の定義** ＞ 資格情報の定義 ＞ プリファレンス ＞ ディスカバリのスケジュール ＞ サマリ

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	詳細については，188 ページ「[IP 範囲の追加 / 編集］ダイアログ・ボックス」を参照してください。
	範囲を選択してこのボタンをクリックすると，リストからその範囲が削除されます。

GUI 要素	説明
	既存の範囲を選択してこのボタンをクリックすると，その範囲を編集できます。
アドレス範囲	➤ **範囲 :** 範囲を定義するルールの詳細については，190 ページ「範囲」を参照してください。 ➤ **除外された :** 範囲の一部を除外できます。範囲を選択して［**追加**］ボタンをクリックします。ダイアログ・ボックスの［詳細］ボタンをクリックします。詳細については，189 ページ「除外範囲」を参照してください。
ディスカバリ・プローブ	Probe の詳細を表示できます。これには範囲の情報が含まれます。Probe に範囲を追加したり，Probe から範囲を除外したりすることもできます。 Probe の定義の詳細については，199 ページ「［ドメインとプローブ］表示枠」を参照してください。

資格情報の定義

説明	プロトコルに設定された資格情報の追加，削除，および編集ができます。

重要情報	➤ 資格情報セットの設定は，何を検出する必要があるか，およびサイトのネットワークでどのプロトコルがサポートされているかによって異なります。 ➤ プロトコルのリストについては，203 ページ「ドメイン資格情報リファレンス」を参照してください。 ➤ このウィザードの一般的な情報については，151 ページ「インフラストラクチャ・ウィザード」を参照してください。
ウィザード・マップ	インフラストラクチャ・ディスカバリ・ウィザードには，以下のページが含まれています。 インフラストラクチャ・ウィザード ＞ IP 範囲の定義 ＞ **資格情報の定義** ＞ プリファレンス ＞ ディスカバリのスケジュール ＞ サマリ

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	選択したプロトコル・タイプの新しい接続詳細を追加します。
	プロトコルを削除します。
	プロトコルを編集します。クリックすると，［プロトコル パラメータ］ダイアログ・ボックスが開きます。
	ボタンをクリックすると，プロトコルが上下に移動し，資格情報セットの試行順序が設定されます。DDM は，リスト内の先頭のプロトコルから順に，すべてのプロトコルを実行します。
プロトコル	クリックすると，ユーザの資格情報を含むプロトコルの詳細が表示されます。

プリファレンス

説明	インフラストラクチャ・ディスカバリ・ウィザードによってアクティブ化されたディスカバリの実行中に使用される設定オプションを選択できます。
重要情報	このウィザードの一般的な情報については，151 ページ「インフラストラクチャ・ウィザード」を参照してください。
ウィザード・マップ	インフラストラクチャ・ディスカバリ・ウィザードには，以下のページが含まれています。 インフラストラクチャ・ウィザード ＞ IP 範囲の定義 ＞ 資格情報の定義 ＞ **プリファレンス** ＞ ディスカバリのスケジュール ＞ サマリ

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します)。

GUI 要素	説明
DNS ネームサーバ	DNS ネームサーバ・マシンと，それらのサーバに対応する名前が保持されている IP を検出します。 Probe マシンからネームサーバ・マシンにゾーン転送を実行できる場合（つまり，適切な権限が DNS ネームサーバ・マシンに存在する場合）にのみ選択してアクティブ化します。 **ネットワークへの影響 :** DDM は，DNS ネームサーバ・マシンへの接続を試行します。
フェイル・オーバ・クラスタ	HP Service Guard，Microsoft Cluster Service，および Veritas Cluster を含むフェイル・オーバ・クラスタを検出します。

GUI 要素	説明
ホスト情報	検出するホスト・リソースを選択します。ホスト・リソースとして, ホストの物理的な部分と論理的な部分のどちらを選択することもできます。 DDM は，ホストに接続してから以下のリソースを検出します。 ➤ SNMP エージェントの場合は, 適切な管理情報ベース (MIB)。 ➤ WMI エージェントの場合は，適切な WMI（Windows Management Instrumentation）Query Language（WQL）クエリ。 DDM は，マシン上でシェル・コマンドを実行することもできます。 **ネットワークへの影響:** ソフトウェアおよびサービスのネットワーク・リソースによって，大量のデータが転送されるため，ネットワーク・トラフィックが大幅に増大する可能性があります。このため, 標準設定ではこれらは検出されません。
ホスト TCP 接続	TCP 通信チャネルを検出し，ホスト間の依存関係をマップします。 このディスカバリでは，少なくとも 1 つのプロトコルに定義済みの資格情報セットが必要です。詳細については，前述の資格情報の定義の手順を参照してください。 **ネットワークへの影響 :** DDM は，マシン上でシェル・コマンドを実行して，開いているポートを検索します。

GUI 要素	説明
IP の ping 方法	環境内で IP を検出するための方法を選択します。 このディスカバリでは，前述の資格情報の定義の手順で SNMP プロトコルを設定する必要があります。 ➤ **定義されたプローブ範囲内のすべての IP に ping を行う:** ほとんどの IP アドレスが応答すること，ネットワークの範囲が小さいこと，および範囲内のほとんどの IP が対象になる（つまり，ネットワークに含まれている）ことがわかっている場合に，このオプションを選択します。 ➤ **ネットワーク CI によって IP の ping を行う :** 一部の IP アドレスが応答しないこと，およびネットワークの範囲が大きいことがわかっている場合に，このオプションを選択します。このような場合，DDM は最初にネットワークを検出し，次にそのネットワーク内で検出されたすべての IP に ping 要求を送信します。 **バージョンと制限事項 :** Probe とネットワーク内のいずれかのスイッチの間にあるすべてのマシンに正しい資格情報セットがあることを確認してください。
ネットワーク・トポロジ	アクティブ化すると，検出されたスイッチ（たとえば，ホスト）上で，ホストとその物理ポート間，およびホストとその論理レイアウト（VLAN，ELAN）間の接続が検出されます。 このディスカバリでは，少なくとも 1 つのプロトコルに定義済みの資格情報セットが必要です。詳細については，前述の資格情報の定義の手順を参照してください。

GUI 要素	説明
ポートのスキャン	［**ポート スキャンする TCP ポートを選択**］リストに表示されている TCP ポートがスキャンされ，開いているサーバ・ポートが検出されます。ポートのスキャンは，検出されたすべてのホストで行われます。 スキャン対象とする新しいポストを追加したり，リストから既存のポートを削除したりできます。 **リストに表示されていないポートを選択するには，次の手順を実行します。** 1 ［**ポートの追加**］ボタンをクリックして，［新しいポートを追加］ダイアログ・ボックスを開きます。 2 ［**ポートの追加**］ボタンをクリックして，ポートの名前と番号を入力します。 3 ［**OK**］をクリックします。 **ネットワークへの影響：** スキャン・プロセスは，ネットワークのパフォーマンスに影響することがあります。また，場合によっては対象マシンの所有者に，DDM がそのマシンに接続することを知らせる必要があります。
ソフトウェア要素	選択すると，検出されたホスト上で実行されているソフトウェア要素を検出します。ソフトウェア要素の検出の一部として，ソフトウェア要素に関連するプロセスとポートも検出されます。［ソフトウェア ライブラリ］ダイアログ・ボックスが開きます。詳細については，281 ページ「［ソフトウェア ライブラリ］ダイアログ・ボックス」を参照してください。 **ネットワークへの影響：** 検索パターンが一般的すぎると，パフォーマンスが低下する原因になります。たとえば，アスタリスク（*）だけから成るプロセス名を入力しないでください。その場合，フィルタはすべてのマシン上で実行されているすべてのプロセスを取得しようとします。

ディスカバリのスケジュール

説明	特定のジョブのスケジュールを定義できます。
重要情報	DDM のスケジュール設定の詳細については，146 ページ「[ディスカバリ スケジューラ] ダイアログ・ボックス」を参照してください。 このウィザードの一般的な情報については，151 ページ「インフラストラクチャ・ウィザード」を参照してください。
ウィザード・マップ	インフラストラクチャ・ディスカバリ・ウィザードには，以下のページが含まれています。 インフラストラクチャ・ウィザード > IP 範囲の定義 > 資格情報の定義 > プリファレンス > **ディスカバリのスケジュール** > サマリ

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	[プロパティ] タブの [ディスカバリ スケジューラ] 表示枠で，時間テンプレートを定義します。詳細については，169 ページ「[ディスカバリ スケジューラ] 表示枠」を参照してください。
次からのディスカバリの開始を許可	ジョブを実行する時間を選択します。
繰り返し間隔	ジョブを実行する頻度を選択します。

サマリ

説明	ディスカバリを実行する前に定義を確認できます。
重要情報	DDM を開始するには，[**実行**] をクリックします。 このウィザードの一般的な情報については，151 ページ「インフラストラクチャ・ウィザード」を参照してください。
ウィザード・マップ	インフラストラクチャ・ディスカバリ・ウィザードには，以下のページが含まれています。 インフラストラクチャ・ウィザード > IP 範囲の定義 > 資格情報の定義 > プリファレンス > ディスカバリのスケジュール > **サマリ**

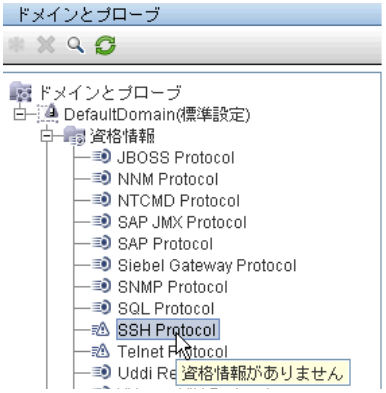
J2EE ウィザード

説明	J2EE アプリケーションに対してディスカバリを実行できます。 **利用方法：**［**管理**］＞［**ディスカバリ**］＞［**ディスカバリ実行**］＞［**ベーシック モード**］。左側の表示枠のリストから J2EE ウィザードを選択します。［**設定および実行**］をクリックします。
重要情報	詳細については，疑問符アイコンの上にポインタを置いてください。
ウィザード・マップ	J2EE ディスカバリ・ウィザードには，以下のページが含まれています。 **J2EE ウィザード** ＞ 資格情報の定義 ＞ J2EE ポートのスキャン＞ WebLogic ＞ WebSphere ＞ JBoss ＞ Oracle Application Server ＞ ディスカバリのスケジュール＞ サマリ

資格情報の定義

説明	各プロトコルの接続データを設定できます。
重要情報	➤ プロトコルの設定は，何を検出する必要があるか，およびサイトのネットワークでどのプロトコルがサポートされているかによって異なります。 ➤ プロトコルのリストについては，203 ページ「ドメイン資格情報リファレンス」を参照してください。 ➤ このウィザードの一般的な情報については，159 ページ「J2EE ウィザード」を参照してください。
ウィザード・マップ	J2EE ディスカバリ・ウィザードには，以下のページが含まれています。 J2EE ウィザード ＞ **資格情報の定義** ＞ J2EE ポートのスキャン ＞ WebLogic ＞ WebSphere ＞ JBoss ＞ Oracle Application Server ＞ ディスカバリのスケジュール ＞ サマリ

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します)。

GUI 要素	説明
	選択したプロトコル・タイプの新しい接続詳細を追加します。
	プロトコルを削除します。
	プロトコルを編集します。クリックすると，[プロトコル パラメータ] ダイアログ・ボックスが開きます。
	プロトコルを上下に移動します。DDM は，リスト内の先頭のプロトコルから順に，すべてのプロトコルを実行します。
プロトコル	クリックすると，ユーザの資格情報を含むプロトコルの詳細が表示されます。 **注**：資格情報が見つからない場合は，次の図に示すアイコン  で表されます。 

J2EE ポートのスキャン

説明	J2EE アプリケーションへの接続に使用するポート番号とポート・タイプを選択できます。
重要情報	このウィザードの一般的な情報については，159 ページ「J2EE ウィザード」を参照してください。
ウィザード・マップ	J2EE ディスカバリ・ウィザードには，以下のページが含まれています。 J2EE ウィザード ＞ 資格情報の定義 ＞ **J2EE ポートのスキャン** ＞ WebLogic ＞ WebSphere ＞ JBoss ＞ Oracle Application Server ＞ ディスカバリのスケジュール ＞ サマリ

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	ポート・リストにポートを追加するときにクリックします。［新しいポートを追加］ダイアログ・ボックスが開きます。ポートを選択し，［OK］をクリックします。 既存のシステム・ポートを編集するには，［新しいポートを追加］ダイアログ・ボックスで［**既知のシステム ポートを編集**］をクリックします。［既知のシステム ポートを編集］ダイアログ・ボックスが開きます。ポートを選択して［**編集**］ボタンをクリックします。開いたダイアログ・ボックスで，エントリに変更を加え，［**OK**］をクリックします。 リストにポートを追加するには，［既知のシステム ポートを編集］ダイアログ・ボックスで［**追加**］ボタンをクリックします。ポートの詳細（名前，番号，およびタイプ）を入力し，［**OK**］をクリックします。
	ポートを選択してこのボタンをクリックすると，リストからそのポートが削除されます。

WebLogic

説明	特定の WebLogic バージョン用の JAR ファイルを選択できます。
重要情報	DDM では，WebLogic のバージョン 6.x，7.x，8.x，9.x，および 10.x がサポートされます。 1 WebLogic を検出するには，以下のドライバを入手します。 ➤ **weblogic.jar**（バージョン 6.x，7.x，および 8.x のみ） ➤ **wlcipher.jar**（WebLogic が SSL 上で実行されている場合。すべてのバージョンが対象） ➤ **license.bea**（WebLogic が SSL 上で実行されている場合。ただし，バージョン 6.x，7.x，および 8.x のみが対象） ➤ **クライアントのトラスト・ストア JKS ファイル**（DemoTrust.jks など。ただし，WebLogic が SSL 上で実行されている場合のみ） ➤ **weblogic.jar**（バージョン 9.x および 10.x のみ） ➤ **wljmxclient.jar**（バージョン 9.x および 10.x のみ） 2 ドライバを以下の場所にある正しいバージョン・フォルダに置きます。 **C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥probeManager¥discoveryResources¥j2ee¥weblogic¥<version_folder>** 例： **C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥probeManager¥discoveryResources\j2ee¥weblogic¥6.x** 3 DDM ジョブを実行する前に，Probe コンソールを再起動します。 4 J2EE ウィザードの［WebLogic］ページで，検出するバージョンのチェック・ボックスを選択します。［**ファイルをインポート ...**］をクリックして参照ウィンドウを開きます。以下に示す適切な WebLogic JAR ファイルを参照します。 このウィザードの一般的な情報については，159 ページ「J2EE ウィザード」を参照してください。
ウィザード・マップ	J2EE ディスカバリ・ウィザードには，以下のページが含まれています。 J2EE ウィザード ＞ 資格情報の定義 ＞ J2EE ポートのスキャン ＞ **WebLogic** ＞ WebSphere ＞ JBoss ＞ Oracle Application Server ＞ ディスカバリのスケジュール ＞ サマリ

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
標準の JAR ファイルを使用してアクティブ化（バージョン 8.x のみ）	選択すると，特定バージョンの JAR ファイルを指定せずに検出できます。これは特定の環境でしか有効でないため，あまりお勧めしません。
WebLogic バージョン 6.x	➤ weblogic.jar ➤ SSL ベース・ディスカバリの場合は, wlcipher.jar, license.bea, および JKS ファイル（DemoTrust.jks など）を選択します。
WebLogic バージョン 7.x	➤ weblogic.jar ➤ SSL ベース・ディスカバリの場合は, wlcipher.jar, license.bea, およびクライアントのトラスト・ストア JKS ファイル（DemoTrust.jks など）を選択します。
WebLogic バージョン 8.x	➤ weblogic.jar ➤ SSL ベース・ディスカバリの場合は，wlcipher.jar, license.bea，およびクライアントのトラスト・ストア JKS ファイル（DemoTrust.jks など）を選択します。
WebLogic バージョン 9.x	➤ wlclient.jar ➤ wljmxclient.jar ➤ SSL ベース・ディスカバリの場合は，wlcipher.jar とクライアントのトラスト・ストア JKS ファイル（DemoTrust.jks など）を選択します。
WebLogic バージョン 10.x	➤ wlclient.jar ➤ wljmxclient.jar ➤ SSL ベース・ディスカバリの場合は，wlcipher.jar とクライアントのトラスト・ストア JKS ファイル（DemoTrust.jks など）を選択します。

WebSphere

説明	特定の WebSphere バージョン用の JAR ファイルを選択できます。
重要情報	DDM では，WebSphere のバージョン 5.x，6.0，および 6.1 がサポートされます。 ➤ WebSphere を検出するには，以下の証明書を入手します。 ➤ **クライアントのキー・ストア JKS ファイル**（WebSphere が SSL 上で実行されている場合は DummyClientKeyFile.jks。このファイルは必須です） ➤ **クライアントのトラスト JKS ファイル**（WebSphere が SSL 上で実行されている場合は DummyClientTrustFile.jks） 用意済みのドライバは，Probe マシン上の以下の場所にあります。 **C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥probeManager¥discoveryResources¥j2ee¥websphere** ➤ DDM ジョブを実行する前に，Probe コンソールを再起動します。 このウィザードの一般的な情報については，159 ページ「J2EE ウィザード」を参照してください。
ウィザード・マップ	J2EE ディスカバリ・ウィザードには，以下のページが含まれています。 J2EE ウィザード ＞ 資格情報の定義 ＞ J2EE ポートのスキャン ＞ WebLogic ＞ **WebSphere** ＞ JBoss ＞ Oracle Application Server ＞ ディスカバリのスケジュール ＞ サマリ

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
標準の JAR ファイルを使用してアクティブ化（バージョン 5.x, 6.x のみ）	選択すると，特定バージョンの JAR ファイルを指定せずに検出できます。これは特定の環境でしか有効でないため，あまりお勧めしません。
WebSphere	検出するバージョンのチェック・ボックスを選択します。［**ファイルをインポート ...**］をクリックして参照ウィンドウを開きます。以下に示す適切な WebSphere JAR ファイルを参照します。 ➤ admin.jar ➤ com.ibm.mq.pcf.jar ➤ ffdc.jar ➤ iwsorb.jar ➤ j2ee.jar ➤ jflt.jar ➤ jmxc.jar ➤ jmxx.jar ➤ log.jar ➤ mail.jar ➤ ras.jar ➤ sas.jar ➤ security.jar ➤ soap.jar ➤ utils.jar ➤ wasjmx.jar ➤ websphere_arm_util.jar ➤ wlmclient.jar ➤ wsexception.jar ➤ wssec.jar

JBoss

説明	特定の JBoss バージョン用の JAR ファイルを選択できます。
重要情報	DDM では，JBoss のバージョン 3.x および 4.x がサポートされます。 このウィザードの一般的な情報については，159 ページ「J2EE ウィザード」を参照してください。
ウィザード・マップ	J2EE ディスカバリ・ウィザードには，以下のページが含まれています。 J2EE ウィザード > 資格情報の定義 > J2EE ポートのスキャン > WebLogic > WebSphere > **JBoss** > Oracle Application Server > ディスカバリのスケジュール > サマリ

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します)。

GUI 要素	説明
標準の JAR ファイルを使用してアクティブ化（バージョン 3.x, 4.x のみ）	選択すると，特定バージョンの JAR ファイルを指定せずに検出できます。これは特定の環境でしか有効でないため，あまりお勧めしません。
JBoss バージョン 3.x / 4.x	検出するバージョンのチェック・ボックスを選択します。［**ファイルをインポート ...**］をクリックして参照ウィンドウを開きます。**jbossall-client.jar** JBoss JAR ファイルを参照します。

Oracle Application Server

説明	Oracle Application Server を検出できます。
重要情報	このウィザードの一般的な情報については，159 ページ「J2EE ウィザード」を参照してください。
ウィザード・マップ	J2EE ディスカバリ・ウィザードには，以下のページが含まれています。 J2EE ウィザード ＞ 資格情報の定義 ＞ J2EE ポートのスキャン ＞ WebLogic ＞ WebSphere ＞ JBoss ＞ **Oracle Application Server** ＞ ディスカバリのスケジュール ＞サマリ

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
Oracle Application Server（バージョン 10g）	選択すると，Oracle Application Server（バージョン 10g）に対して DDM が実行されます。

ディスカバリのスケジュール

説明	特定のジョブのスケジュールを定義できます。
重要情報	このウィザードの一般的な情報については，159 ページ「J2EE ウィザード」を参照してください。
ウィザード・マップ	J2EE ディスカバリ・ウィザードには，以下のページが含まれています。 J2EE ウィザード ＞ 資格情報の定義 ＞ J2EE ポートのスキャン ＞ WebLogic ＞ WebSphere ＞ JBoss ＞ Oracle Application Server ＞ **ディスカバリのスケジュール** ＞ サマリ

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	［プロパティ］タブの［ディスカバリ スケジューラ］表示枠で，時間テンプレートを定義します。詳細については，169 ページ「[ディスカバリ スケジューラ］表示枠」を参照してください。
次からのディスカバリの開始を許可	ジョブを実行する時間を選択します。
繰り返し間隔	ジョブを実行する頻度を選択します。

サマリ

説明	ディスカバリを実行する前に定義を確認できます。
重要情報	実行に変更を加えるには，［**戻る**］をクリックします。 このウィザードの一般的な情報については，159 ページ「J2EE ウィザード」を参照してください。
ウィザード・マップ	J2EE ディスカバリ・ウィザードには，以下のページが含まれています。 J2EE ウィザード ＞ 資格情報の定義 ＞ J2EE ポートのスキャン＞ WebLogic ＞ WebSphere ＞ JBoss ＞ Oracle Application Server ＞ ディスカバリのスケジュール ＞ **サマリ**

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
実行	クリックすると，DDM が実行されます。

［プロパティ］タブ

説明	モジュールとジョブのプロパティを表示して管理できます。 **利用方法：**［ディスカバリ実行］の［**プロパティ**］タブをクリックします。
重要情報	［プロパティ］タブに表示される情報は，［ディスカバリ モジュール］表示枠で選択したレベルによって異なります。 具体的な表示内容は次のとおりです。 ➤［**ディスカバリ モジュール**］ルートを選択すると，すべてのアクティブなジョブがスケジュール情報とともに表示されます。いずれかのカラムをクリックすると，そのカラムを基準にしてリストが並べ替えられます。ジョブのスケジュールを編集するには，そのジョブを右クリックします。詳細については，146 ページ「［ディスカバリ スケジューラ］ダイアログ・ボックス」を参照してください。 ➤ **ディスカバリ・モジュール**を選択すると，［説明］表示枠と［モジュール ジョブ］表示枠が表示されます。 説明を編集するには，［説明］表示枠で変更を行って［**OK**］をクリックします。 171 ページ「［モジュール ジョブ］表示枠」も参照してください。 ➤ **ジョブ**を選択すると，［パラメータ］，［トリガ TQL］，［グローバル構成ファイル］，および［ディスカバリ スケジューラ］表示枠が表示されます。詳細については，173 ページ「［パラメータ］表示枠」，174 ページ「［トリガ TQL］表示枠」，268 ページ「［グローバル構成ファイル］表示枠」，および 169 ページ「［ディスカバリ スケジューラ］表示枠」を参照してください。

［ディスカバリ スケジューラ］表示枠

説明	このジョブに設定されたスケジュールに関する情報を表示できます。 **利用方法：**［ディスカバリ実行］ウィンドウの［ディスカバリ モジュール］表示枠でジョブを選択します。

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します)。

GUI 要素	説明
	［**次からのディスカバリの開始を許可**］リストに時間を追加するときにクリックします。［時間テンプレート］ダイアログ・ボックスが開きます。リストに時間テンプレートを追加するには，［時間テンプレート］ダイアログ・ボックスで［**追加**］ボタンをクリックし，［時間テンプレートを編集］ダイアログ・ボックスを開きます。詳細については，149 ページ「[時間テンプレートを編集］ダイアログ・ボックス」を参照してください。
スケジューラの編集	クリックすると，［ディスカバリ スケジューラ］が開きます。詳細については，146 ページ「[ディスカバリ スケジューラ］ダイアログ・ボックス」を参照してください。
新たにトリガされた CI で直ちに呼び出し	チェック・マークは，トリガ CI が Probe に到達するとすぐにジョブが実行されることを示します。このカラムにチェック・マークが付いていない場合は，スケジュール・マネージャで定義されたスケジュールに従ってジョブが実行されます。
時間テンプレート	ジョブを実行する日時を含むテンプレートを選択します。

［グローバル構成ファイル］表示枠

詳細については，268 ページ「[グローバル構成ファイル］表示枠」を参照してください。

［モジュール ジョブ］表示枠

説明	特定のモジュールのアクティブなジョブを表示できます。 **利用方法：**［ディスカバリ実行］ウィンドウの［ディスカバリ モジュール］表示枠でモジュールを選択します。

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	**ディスカバリ・ジョブをモジュールに追加：**［ディスカバリ ジョブの選択］ダイアログ・ボックスが開き，複数の zip ファイルからジョブを選択できます （複数のジョブを選択するには，SHIFT キーまたは CTRL キーを使用します）。
	**モジュールから選択したディスカバリ・ジョブを削除：**ジョブを選択してこのボタンをクリックします （メッセージは表示されません。ジョブを復元するには，［**キャンセル**］ボタンをクリックします）。
	**結果をマップで表示**：選択すると，パターンによって検出された CI とリンクのマップが（リストの代わりに）表示されます。このボタンをクリックすると，［検出されたクラスのマップ］ウィンドウが開きます。選択したパターンが，その CI と関係とともに表示されます。CIT の上にカーソルを置くと，ツールチップに説明が表示されます。
<カラムのタイトル>	カラムのタイトルをクリックすると，CIT の順序が昇順から降順（またはその逆）に変更されます。
<ジョブのリスト>	モジュールに含まれるすべてのジョブ。 （［ディスカバリ モジュール］表示枠で特定のモジュールを選択したときに表示されます。）

GUI 要素	説明
<右クリック・メニュー>	行を右クリックすると，選択したジョブの［ディスカバリ スケジューラ］が開きます。詳細については，146 ページ「[ディスカバリ スケジューラ］ダイアログ・ボックス」を参照してください。 カラムのタイトルを右クリックすると，テーブルをカスタマイズできます。次のオプションから選択できます。 ➤ **カラムを非表示 :** 特定のカラムを非表示にするときに選択します。 ➤ **全カラムを表示 :** カラムが非表示になっているときに表示されます。 ➤ **カラムの選択 :** カラムの表示 / 非表示の切り替え，または，テーブル内のカラムの順序変更を行うときに選択します。［カラム］ダイアログ・ボックスが開きます。 ➤ **自動サイズ変更カラム :** 選択すると，内容の長さに合わせてカラムの幅が変更されます。詳細については，『**参照情報**』の「[カラムの選択］ダイアログ・ボックス」を参照してください。
直ちに呼び出す	➤ チェック・マークは，トリガされた CI が Probe に到達するとすぐにディスカバリ・ジョブが実行されることを示します。その場合は，［プロパティ］タブの［**新たにトリガされた CI で直ちに呼び出し**］チェック・ボックスが選択されます。 ➤ このカラムにチェック・マークが付いていない場合は，スケジュール・マネージャで定義されたスケジュールに従ってジョブが実行されます。
ジョブ名	ジョブとそれを含むパッケージの名前。 （［ディスカバリ モジュール］表示枠でジョブを選択したときに表示されます。）
スケジュール情報	［ディスカバリ スケジューラ］で定義されたジョブのスケジュール情報。
トリガ TQL	ジョブをアクティブ化した TQL の名前。

［パラメータ］表示枠

説明	パターンの動作を上書きできます。 説明を表示するには, パラメータの上にポインタを置いたままにします。 **利用方法：**［ディスカバリ実行］ウィンドウの［ディスカバリ モジュール］表示枠でジョブを選択します。
重要情報	特定のジョブの標準設定のパターン・パラメータを，標準設定値に影響を与えずに上書きできます。

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します)。

GUI 要素	説明
名前	パターンに設定された名前。
上書き	選択すると，パターンのパラメータ値が上書きされます。 このチェック・ボックスを選択すると，標準設定値を上書きできます。たとえば，protocolType パラメータを変更するには，［**上書き**］チェック・ボックスを選択し，「MicrosoftSQLServer」を新しい値に変更します。［プロパティ］タブで［**OK**］をクリックして変更内容を保存します。  ［ディスカバリ パターン パラメータ］表示枠のパラメータの編集の詳細については，267 ページ「［ディスカバリ パターン パラメータ］表示枠」を参照してください。
値	パターンに定義された値。

［トリガ TQL］表示枠

説明	選択したジョブをアクティブ化するためのトリガとして使用する 1 つ以上の TQL クエリを定義できます。 **利用方法：** ➤ ［ディスカバリ実行］ウィンドウの［ディスカバリ モジュール］表示枠でジョブを選択します。 ➤ ジョブを作成します（［ディスカバリ モジュール］表示枠でモジュールを右クリックし，［**ジョブの新規作成**］を選択します）。

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	**TQL の追加：** 選択したジョブをアクティブ化するためのトリガとして使用する 1 つ以上の標準設定でない TQL クエリを追加できます。クリックすると，［**ディスカバリ TQL を選択してください**］ダイアロング・ボックスが開きます。
	**TQL の削除：** TQL を選択してこのボタンをクリックします（メッセージは表示されません。TQL を復元するには，［**キャンセル**］ボタンをクリックします）。 **注：** アクティブなジョブの TQL クエリを削除すると，DDM はその TQL クエリから送られる新しい CI を受け取らなくなります。ただし，TQL クエリから最初に送られていた既存のトリガ CI は削除されません。
	特定の TQL の Probe を追加または削除するときにクリックします。詳細については，149 ページ「［TQL 出力用 Probe 制限の編集］ダイアログ・ボックス」を参照してください。
	クリックすると，［TQL エディタのトリガ］が開きます。詳細については，177 ページ「［TQL エディタのトリガ］ウィンドウ」を参照してください。
	クリックすると，クエリ・マネージャが開きます。詳細については，『**モデル管理**』の「クエリ・マネージャ」を参照してください。
プローブ制限	ディスカバリ・プロセスで使用される Probe。Probe を追加または削除するには，このボタンをクリックします。
TQL 名	ジョブをアクティブにするトリガ TQL クエリの名前。

［関連 CI］ウィンドウ

説明	選択した CI に関連する CI をマップ形式で表示できます。 **利用方法：**［**検出された CI**］ダイアログ・ボックスで，CIT を右クリックして［**関連 CI を取得**］を選択します。
重要情報	関連 CI とは, 既存の CI の親, 子, または兄弟である CI です。

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
<右クリック・メニュー>	詳細については,『**モデル管理**』の「IT ユニバース・マネージャのショートカット・メニュー」を参照してください。
<メニュー>	詳細については,『**モデル管理**』の「ツールバー・オプション」を参照してください。
<トポロジ・マップ>	詳細については,『**モデル管理**』の「トポロジ・マップの概要」を参照してください。

［トリガされた CI の結果を表示］ダイアログ・ボックス

説明	Probe に対する一時的な要求の実行結果を表示できます。DDM は，選択されたトリガ CI に対してジョブを実行して結果を取得します。エラー発生時にはメッセージが表示されます。 **利用方法：**［ディスカバリ実行］で，モジュールまたはジョブを選択し，［詳細］タブを選択します。［ディスカバリ ステータス］表示枠で，CI をドリルダウンし，その CI を右クリックして［**トリガされた CI の結果を表示**］を選択します。

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	CIT を選択してクリックすると，［トリガされた CI の結果を表示］ダイアログ・ボックスに追加情報が表示されます。詳細については，175 ページ「［トリガされた CI の結果を表示］ダイアログ・ボックス」を参照してください。
	クリックすると，トリガされた CI の結果マップを表示するトポロジ・マップが開きます。CIT を右クリックすると，そのプロパティが表示されます。

［ソース CI］ダイアログ・ボックス

［ソース CI］ダイアログ・ボックスには，［**検出済み CIs**］ダイアログ・ボックスと同じコンポーネントが含まれています。詳細については，141 ページ「［検出済み CIs］ダイアログ・ボックス」を参照してください。

［時間テンプレート］ダイアログ・ボックス

説明	選択したジョブを実行する日ごとまたは週ごとのスケジュールを定義できます。 **利用方法：**［**ディスカバリ実行**］＞［**プロパティ**］タブ＞［**ディスカバリ スケジューラ**］表示枠＞［**時間テンプレート**］アイコン。

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	時間テンプレートを追加するときにクリックします。［テンプレートの編集］ダイアログ・ボックスが開きます。
	時間テンプレートを選択し，このボタンをクリックして削除します。
	時間テンプレートを選択し，このボタンをクリックして編集します。［テンプレートの編集］ダイアログ・ボックスが開きます。

［トリガされた CI］ウィンドウ

説明	選択した TQL ノードで検出されたすべての CI インスタンスを表示できます。 **利用方法：** ➤［**ディスカバリ実行**］＞［**依存関係マップ**］タブ。CIT を右クリックして［**トリガされた CI の表示**］を選択します。 ➤［ディスカバリ ステータス］表示枠で，［**追加データの表示**］ボタンをクリックします。
重要情報	［トリガされた CI］ウィンドウには，［要素インスタンス］ウィンドウと同じ情報が含まれています。詳細については，『**モデル管理**』の「［要素インスタンス］ダイアログ・ボックス」を参照してください。

［TQL エディタのトリガ］ウィンドウ

説明	ジョブをトリガするように定義された TQL を編集できます。 **利用方法：**［**ディスカバリ実行**］＞［**プロパティ**］タブ＞［**トリガ TQL**］表示枠で，TQL を選択して［**TQL エディタを開く**］ボタンをクリックします。
重要情報	ジョブに関連付けられているトリガ TQL は，入力 TQL のサブセットであり，どの CI をジョブのトリガ CI にすべきかを定義します。つまり，入力 TQL が SNMP を実行中の IP を探す場合，トリガ TQL は，195.0.0.0 ～ 195.0.0.10 の範囲内の SNMP を実行中の IP を探します。
関連リンク	➤ 54 ページ「トリガ CIT，トリガ CI，入力 TQL，トリガ TQL」 ➤ 251 ページ「［TQL エディタの入力］ウィンドウ」

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します)。

GUI 要素	説明
<表示枠>	➤［CI タイプ セレクタ］表示枠 ➤ 編集表示枠 ➤ 情報ページ
TQL 名	ジョブをアクティブ化するトリガ TQL クエリの名前。

［CI タイプ セレクタ］表示枠

説明	CMDB にある CI タイプの階層ツリー構造を表示します。詳細については,『**モデル管理**』の「CI タイプ・マネージャのユーザ・インタフェース」を参照してください。 **注**: 各 CIT の右側に，CMDB 内の各 CIT のインスタンス数が表示されます。 TQL クエリを作成または変更するには，ノードをクリックして編集表示枠にドラッグし，ノード間の関係を定義します。変更が CMDB に保存されます。詳細については,『**モデル管理**』の「ノードと関係を TQL クエリに追加」を参照してください。
ほかのタスク	➤「TQL クエリの定義」(『**モデル管理**』) ➤「パターン・ビューの作成」(『**モデル管理**』)

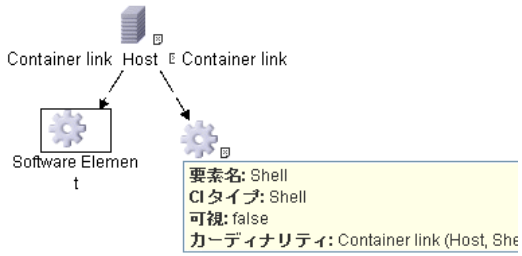
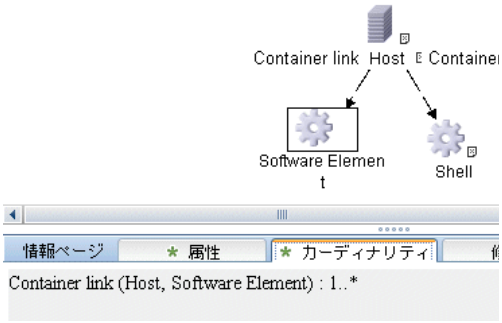
編集表示枠

説明	［トリガ TQL］表示枠で選択したノードを編集できます。

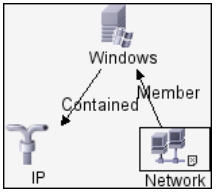
含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
<ノード>	クリックすると，そのノードに関する情報が情報表示枠に表示されます。
<右クリック・メニュー>	詳細については，『**モデル管理**』の「ショートカット・メニューのオプション」を参照してください。
<ツールバー>	詳細については，『**モデル管理**』の「ツールバー・オプション」を参照してください。

情報ページ

説明	選択したノードおよび関係のプロパティ，条件，およびカーディナリティが表示されます。
重要情報	ノードの上にポインタを置いたままにすると，次のように情報が表示されます。  情報を含むタブの横に，次のように小さい緑色のインジケータが表示されます。 

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します)。

GUI 要素	説明
編集	情報を表示するには，編集表示枠内のノードまたは関係を選択し，情報表示枠内のタブを選択して［**編集**］ボタンをクリックします。［ノード条件］ダイアログ・ボックスの詳細については，『**モデル管理**』の「［ノード / 関係のプロパティ］ダイアログ・ボックス」を参照してください。
属性	ノードまたは関係に対して定義された属性条件が表示されます。詳細については，『**モデル管理**』の「［属性］タブ」を参照してください。
カーディナリティ	カーディナリティは，関係のもう一方の端にあることが期待されるノードの数を定義します。たとえば，ホストと IP の関係でカーディナリティが 1:3 である場合，TQL は 1 ～ 3 個の IP に接続されているホストのみを取得します。詳細については，『**モデル管理**』の「［カーディナリティ］タブ」を参照してください。
詳細	➤ **CI タイプ :** 選択したノード / 関係の CIT です。 ➤ **可視 :** チェック・マークは，選択したノードまたは関係がトポロジ・マップに表示されることを示します。ノード / 関係が表示されない場合は，ボックスが編集表示枠の選択したノード / 関係の右側に が表示されます。  ➤ **サブタイプを含める ：選択した** CI とその子孫の両方がトポロジ・マップに表示されます。
修飾子	ノードまたは関係に対して定義された修飾子条件が表示されます。詳細については，『**モデル管理**』の「［修飾子］タブ」を参照してください。

GUI 要素	説明
選択された ID	TQL 結果に含める必要があるものを定義するために使用される要素インスタンスが表示されます。詳細については,『**モデル管理**』の「[ID] タブ」を参照してください。

第 6 章

ディスカバリ Probe の設定

本章では，ディスカバリおよび依存関係マップ（DDM）Probe の設定に関する情報を提供します。

本章の内容

概念

- ジョブ実行ポリシー（183 ページ）

タスク

- Probe の追加（186 ページ）

参照先

- ディスカバリ・プローブ設定ユーザ・インタフェース（187 ページ）
- ドメイン資格情報リファレンス（203 ページ）

ジョブ実行ポリシー

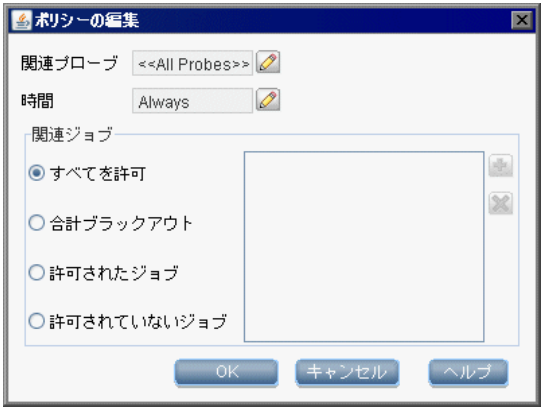
Probe が実行されてはいけない期間の時間を定義できます。また，すべての Probe 上の特定のジョブの実行を無効化したり，特定の Probe 上のすべてのジョブの実行を無効化したりできます。さらに，ジョブ実行ポリシーからジョブを除外して，それらのジョブが通常どおりに実行され続けるようにすることもできます。

ジョブ実行ポリシーの定義方法については，190 ページ「[ポリシーの追加 / 編集] ダイアログ・ボックス」を参照してください。

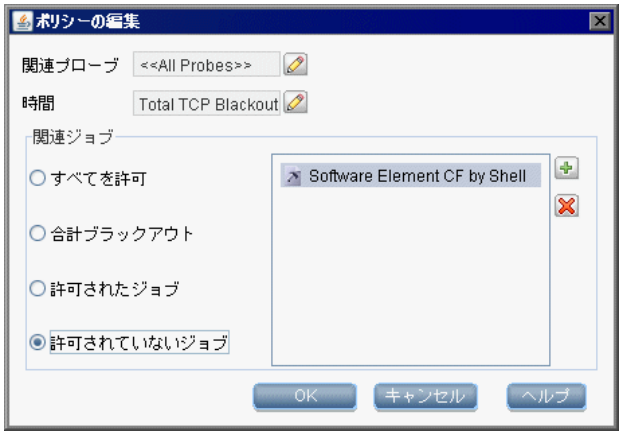
ポリシーの順序の例

Total TCP Blackout と **Always** という 2 つのポリシーがあります。**Total TCP Blackout** は，いかなる TCP ディスカバリ・ジョブの実行も許可しません。ポリシーは次のようにリストに表示されます。

ジョブ（Class C IPs by ICMP）が起動されます。ジョブは，ポリシー・リストの一番上から順にポリシーをチェックしていきます。まず最初に **Total TCP Blackout** がチェックされます。このジョブはこのポリシーの対象になっていないため，リスト内で次にある **Always** がチェックされます。このジョブはこのポリシーの対象になっているため（［ポリシーの編集］ダイアログ・ボックスで［**すべてを許可**］が選択されています），ジョブが実行されます。



次のジョブ（Software Element CF by Shell）が起動されます。ジョブは，ポリシー・リストの一番上から順にポリシーをチェックしていきます。最初に **Total TCP Blackout** がチェックされます。このジョブはこのポリシーの対象になっているため（［**ポリシーの編集**］ダイアログ・ボックスで［**許可されていないジョブ**］が選択されています），実行されません。



重要：どのポリシーにも接続されていないジョブは，実行されません。このようなジョブを実行するには，リストの最後のポリシーを［**すべてを許可**］に設定します。

ジョブ実行ポリシーが実行中のジョブの実行

Probe がジョブを実行しているときにポリシーが動作し始めると，そのジョブは一時停止します。ポリシーが完了すると，一時停止した箇所からジョブの実行が再開されます。たとえば，10,000 個のトリガ CI を持つジョブがあるとします。ジョブがそのうち 7,000 個の処理を完了し，その後ポリシーが起動されたとします。そのジョブの実行が再開されるときには（ポリシーが完了した後で），最初からやり直すのではなく，残りの 3,000 個のトリガ CI が処理されます。

Probe の追加

このタスクでは，Probe を DDM に追加する方法を説明します。

このタスクには次のステップが含まれます。

1 前提条件

Probe がインストールされていることを確認し，その IP アドレスをメモしておきます。

2 DDM にドメインを追加する

このステップでは，新しい Probe 用のドメインを作成します。

a ［ディスカバリ プローブ設定］ウィンドウにアクセスします（［**管理**］＞［**ディスカバリ**］＞［**ディスカバリ プローブ設定**］）。

b ［**ドメインとプローブ**］を選択し，［**ドメインまたは Probe の追加**］ボタンをクリックして［**新しいドメインの追加**］ダイアログ・ボックスを開きます。詳細については，191 ページ「［新しいドメインの追加］ダイアログ・ボックス」を参照してください。

3 Probe を新しいドメインに追加する

このステップでは，Probe とその範囲を定義します。

a 新しいドメインをダブルクリックし，**Probes** フォルダを選択します。

b ［**ドメインまたは Probe の追加**］ボタンをクリックして［新しい Probe の追加］ダイアログ・ボックスを開きます。詳細については，192 ページ「［新しいプローブの追加］ダイアログ・ボックス」を参照してください。

c 新しい Probe を選択し，その IP 範囲を定義します。詳細については，188 ページ「［IP 範囲の追加 / 編集］ダイアログ・ボックス」を参照してください。

4 ドメインにさらに Probe を追加する（任意指定）

このドメインに，さらに Probe を追加できます。詳細については，前のステップを参照してください。

5 資格情報の定義

プロトコルの設定は，何を検出する必要があるか，およびサイトのネットワークでどのプロトコルがサポートされているかによって異なります。

詳細については，194 ページ「資格情報」を参照してください。プロトコルのリストについては，203 ページ「ドメイン資格情報リファレンス」を参照してください。

ディスカバリ・プローブ設定ユーザ・インタフェース

本項の内容

- ［IP 範囲の追加 / 編集］ダイアログ・ボックス（188 ページ）
- ［ポリシーの追加 / 編集］ダイアログ・ボックス（190 ページ）
- ［新しいドメインの追加］ダイアログ・ボックス（191 ページ）
- ［新しいプローブの追加］ダイアログ・ボックス（192 ページ）
- ［ディスカバリ ジョブの選択］ダイアログ・ボックス（193 ページ）
- ［詳細］タブ（193 ページ）
- ［ドメインとプローブ］表示枠（199 ページ）
- ［関連プローブの編集］ダイアログ・ボックス（200 ページ）
- ［時間表の編集］ダイアログ・ボックス（200 ページ）
- ［プロトコル パラメータ]］ダイアログ・ボックス（201 ページ）
- ［対象定義］ダイアログ・ボックス（201 ページ）
- ［ディスカバリ プローブ設定］ウィンドウ］（202 ページ）
- Probe の選択（202 ページ）

［IP 範囲の追加 / 編集］ダイアログ・ボックス

説明	ディスカバリのネットワーク範囲を設定できます。ここで定義した範囲内のアドレスから結果が取得されます。範囲から除外する IP アドレスも定義できます。 **利用方法 :**［範囲］表示枠で［**IP 範囲を追加**］ボタンをクリックします（［**管理**］>［**ディスカバリ**］>［**ディスカバリ プローブ設定**］>［**詳細**］表示枠）。
重要情報	Probe がインストールされているネットワークの範囲外の IP 範囲を定義すると，HP Universal CMDB が Probe が動作する IP 範囲を自動的に定義します。その Probe 範囲が Probe に含まれていないことを知らせるメッセージが表示されます。その IP アドレスを範囲に含める場合は，［**はい**］を選択してください。
ほかのタスク	95 ページ「ディスカバリ実行 – アドバンス・モードのワークフロー」

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	ディスカバリから IP 範囲を除外するには，［**IP 範囲の追加**］ボタンをクリックします。
	IP 範囲の除外した部分を削除するには，除外した範囲を選択して［**IP 範囲の削除**］ボタンをクリックします。
	IP 範囲の除外した部分を編集するには，除外した範囲を選択して［**IP 範囲の編集**］ボタンをクリックします。詳細については，**除外範囲**を参照してください。

GUI 要素	説明
除外範囲	範囲の一部を除外するには，[**IP 範囲の追加**] ボタンまたは [**IP 範囲の編集**] ボタンをクリックします。[除外 IP 範囲] ダイアログ・ボックスで，除外する範囲を入力します。 **注 :** ➤ 除外範囲を入力する前に，まず範囲を入力する必要があります（[IP 範囲の追加 / 編集] ダイアログ・ボックスで）。 ➤ 除外範囲の入力に関する規則は，範囲を入力する場合と同じです。詳細については，**範囲**を参照してください。 ➤ この機能を使って，ネットワーク範囲をいくつかの部分範囲に分割できます。たとえば，次のような範囲があるとします。 10.0.64.0 － 10.0.64.255 ここで，次の 3 つの除外範囲を定義します。: 10.0.64.45 – 10.0.64.50 10.0.64.65 – 10.0.64.70 10.0.64.89 – 10.0.64.95 すると，次の 4 つの範囲が検出されることになります。: 10.0.64.0 – 10.0.64.44 10.0.64.51 – 10.0.64.64 10.0.64.71 – 10.0.64.88 10.0.64.96 – 10.0.64.255

GUI 要素	説明
範囲	IP アドレス範囲を定義する際の規則を次に示します。 ➤ IP アドレス範囲は次の形式で定義する必要があります。 ＜開始 IP アドレス＞ – ＜終了 IP アドレス＞ 次に例を示します。10.0.64.0 - 10.0.64.57 ➤ 範囲には, 0 ～ 255 の範囲の任意の数値を表すアスタリスク (*) を含めることができます。 ➤ アスタリスクを使用する場合は, 終了 IP アドレスを入力する必要はありません。たとえば, 範囲パターンとして開始 IP アドレスに「10.0.48.*」という値を入力すると，10.0.48.0 から 10.0.48.255 までの範囲が指定されます。 ➤ アスタリスクは，IP 範囲パターンの開始 IP アドレスでのみ使用できます。（アスタリスクを開始 IP アドレスで使用し，終了 IP アドレスも入力した場合には，終了 IP アドレスの指定は無視されます。） ➤ IP アドレスの指定では，アスタリスク（*）を複数使用できます（連続している必要があります）。アスタリスクを 2 つの数値の間に置いたり，数値の 1 桁目の代わりとして使用したりすることはできません。たとえば,「10.0.*.*」と入力することはできますが,「10.*.64.*」と入力することはできません。 ➤ 同じドメイン内の 2 つの Probe がそれぞれの範囲内に同じ IP アドレスを含むことはできません。

［ポリシーの追加 / 編集］ダイアログ・ボックス

説明	ジョブ実行ポリシーを追加し，特定の時間にジョブの実行を無効にすることができます。 **利用方法 :** ［**管理**］＞［**ディスカバリ**］＞［**ディスカバリ プローブ設定**］＞［**詳細**］表示枠＞［**ジョブ実行ポリシー**］セクション。既存のポリシーを選択して［**編集**］をクリックするか，［**追加**］ボタンをクリックします。
関連リンク	183 ページ「ジョブ実行ポリシー」 196 ページ「［ジョブ実行ポリシー］表示枠」 203 ページ「ドメイン資格情報リファレンス」

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
関連ジョブ	➤ **すべてを許可 :** ジョブ実行ポリシーをすべてのジョブに対して実行します。 ➤ **合計ブラックアウト :** ポリシーはどのジョブに対しても実行されません。 ➤ **許可されたジョブ :** 設定されたブラックアウト時間中も実行するジョブを選択します。 ➤ **許可されていないジョブ :** 設定されたブラックアウト時間中に実行しないジョブを選択します。 許可されたジョブと許可されていないジョブについて，［**追加ジョブ**］ボタンをクリックしてポリシーに含めるジョブを選択するか，［**ジョブの削除**］ボタンをクリックしてポリシーから除外するジョブを選択します。［追加ジョブ］ボタンをクリックした場合は，［**ディスカバリ ジョブの選択**］ダイアログ・ボックスが表示されます。
関連プローブ 	ポリシーの実行対象となる Probe。このボタンをクリックすると，［**関連 Probe の編集**］ダイアログ・ボックスが開き，ポリシーに含める Probe を定義できます。
時間 	ポリシーがアクティブになる日付と時間。このボタンをクリックすると，［**時間表の編集**］ダイアログ・ボックスが開きます。

［新しいドメインの追加］ダイアログ・ボックス

説明	ドメインを追加できます。 **利用方法 :** ［ドメインとプローブ］表示枠で［**ドメインまたは Probe の追加**］ボタンをクリックします。
重要情報	バージョン 6.x からアップグレードされたバージョン 8.01 以降の環境で，従来のバージョンと同様にデータをモデル化するためには，Probe を**カスタマ** ドメインではなく**外部**ドメインに所属するものとして定義する必要があります。

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
Description	［ディスカバリ プローブ設定］ウィンドウの［詳細］表示枠に表示される説明を入力します。
Domain Type	➤ **カスタマ :** 当該サイト用のプライベート・ドメイン。複数のドメインを定義し，各ドメインに複数の Probe を含めることができます。各 Probe には IP 範囲を含めることができますが，カスタマ・ドメイン自体について範囲を定義することはできません。 ➤ **外部 :** インターネット / パブリック・ドメイン。範囲付きで定義されたドメイン。外部ドメインには，ドメイン名と同じ名前の 1 つの Probe しか含められません。ただし，システム内に複数の外部ドメインを定義できます。
Name	ドメインの一意の名前を入力します。

［新しいプローブの追加］ダイアログ・ボックス

説明	Probe を追加できます。 **利用方法 :**［ドメインとプローブ］表示枠で［**ドメインまたは Probe の追加**］ボタンをクリックします。
重要情報	➤ 既存のドメインに Probe を追加するには，［ドメインとプローブ］表示枠で［**Probes**］を選択し，［**ドメインまたは Probe の追加**］ボタンをクリックします。 ➤ 新規のドメインに Probe を追加するには，ドメインを作成してから，そのドメインに Probe を追加します。 ➤ 同じドメイン内の 2 つの Probe がそれぞれの範囲内に同じ IP アドレスを含むことはできません。 ➤ Probe は，アクティブ化されると自動的に追加され，そのステータスが「**接続**」に変わります。詳細については，39 ページ「［スタート］メニューからの Probe の起動」または 40 ページ「Probe のサービスとしての起動」を参照してください。

［ディスカバリ ジョブの選択］ダイアログ・ボックス

説明	ジョブ実行ポリシーに追加するジョブ，またはジョブ実行ポリシーから除外するジョブを選択できます。 **利用方法 :**［ポリシーの編集］ダイアログ・ボックスで［**許可されたジョブ**］または［**許可されていないジョブ**］を選択し，ボタンをクリックします  。

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
<インストールされているパッケージ>	ポリシーに含めるジョブ，またはポリシーから除外するジョブを見つけます。（複数のパッケージを選択するには，SHIFT キーまたは CTRL キーを使用します）。

［詳細］タブ

説明	すべてのドメインで実行中の Probe を表示し，実行ポリシーをジョブに追加することができます（つまり，ジョブを実行しない期間をスケジュールできます）。 **利用方法 :**［ドメインとプローブ］表示枠でオブジェクトをクリックします。
重要情報	［ドメインとプローブ］表示枠で何を選択したかによって，［詳細］タブに表示される情報は異なります。詳細については，次の項の 194 ページ「表示される情報」を参照してください。

表示される情報

選択したもの	表示される情報
	**ドメインと Probe。**すべての Probe の詳細表示や，ジョブ実行ポリシーの定義および編集を行うことができます。詳細については，196 ページ「［ディスカバリ プローブ］表示枠」および 196 ページ「［ジョブ実行ポリシー］表示枠」を参照してください。
	**特定のドメイン。**説明を追加したり，そのドメイン内で実行される Probe のリストを表示したりできます。詳細については，196 ページ「［ディスカバリ プローブ］表示枠」および 195 ページ「［説明］表示枠」を参照してください。
	**特定のプロトコル。**プロトコル・パラメータを追加したり，ユーザの資格情報を含むプロトコルの詳細を表示することができます。詳細については，194 ページ「資格情報」と 203 ページ「ドメイン資格情報リファレンス」を参照してください。
	**特定の Probe。**範囲情報を含む Probe の詳細を表示できます。Probe に範囲を追加したり，Probe から範囲を除外したり，UCMDB から Probe を削除したりすることもできます。詳細については，197 ページ「［範囲］表示枠」，196 ページ「［ディスカバリ プローブ］表示枠」，および 195 ページ「［詳細］表示枠」を参照してください。

資格情報

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	選択したプロトコル・タイプの新しい接続詳細を追加します。
	プロトコルを削除します。
	クリックすると，プロトコルを編集できます。詳細については，201 ページ「［プロトコル パラメータ］］ダイアログ・ボックス」を参照してください。

GUI 要素	説明
	ボタンをクリックすると，プロトコルが上下に移動し，資格情報セットの試行順序が設定されます。DDM は，リスト内の先頭のプロトコルから順に，すべてのプロトコルを実行します。
プロトコル	クリックすると，ユーザの資格情報を含むプロトコルの詳細が表示されます。

［説明］表示枠

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
説明	ドメインの作成時に入力された説明。
ドメイン・タイプ	詳細については，191 ページ「［新しいドメインの追加］ダイアログ・ボックス」のドメイン・タイプを参照してください。

［詳細］表示枠

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
最後にアクセスされたプローブ	サーバ・マシン上で Probe が最後にアクセスされた日時。
プローブ IP	Probe マシンの IP。
ステータス	➤ **接続 :** Probe は正常にサーバに接続されました（Probe は数秒ごとに接続します）。 ➤ **非接続 :** Probe はサーバに接続されていません。

[ディスカバリ プローブ] 表示枠

説明	サーバに接続されたすべての Probe のリストを表示できます。 **利用方法 :** [ドメインとプローブ] 表示枠で Probe をクリックします。

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します)。

GUI 要素	説明
IP	Probe の作成時に定義された IP 範囲。
最終アクセス日時	Probe が最後にサーバにタスクを要求した日時。
名前	DDM に表示される Probe 名。
ステータス	「**接続**」または「**非接続**」

[ジョブ実行ポリシー] 表示枠

説明	ジョブを実行してはならない期間を設定できます。 **利用方法 :** [**管理**] > [**ディスカバリ**] > [**ディスカバリ プローブ設定**]。[**ドメインとプローブ**] を選択します。
重要情報	リスニング機能を持つジョブ（つまり，ディスカバリを実行せず，たとえば SNMP トラップをリッスンするジョブ）は，ポリシーには含まれません。
関連リンク	183 ページ「ジョブ実行ポリシー」 203 ページ「ドメイン資格情報リファレンス」

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	ポリシーを上下に移動します。DDM は，リスト内にあり，優先権がある最初のポリシーをすべて実行します。つまり，あるジョブが 2 つのポリシーに含まれている場合，DDM はそのジョブを含んでいる最初のポリシーだけを実行します。
	ポリシーを追加します。
	ポリシーを削除します。
	ポリシーを編集できます。クリックすると［**ポリシーの編集**］ダイアログ・ボックスが開きます。
ジョブ	ポリシーの影響を受けるジョブ。
プローブ	ポリシーの影響を受ける Probe。
時間	ポリシーのスケジュール

［範囲］表示枠

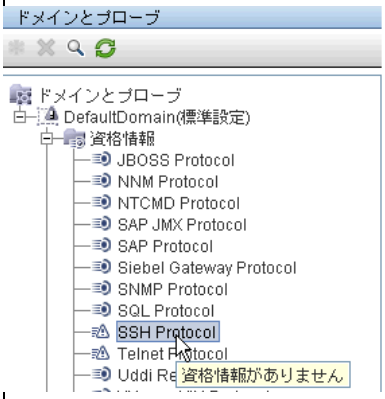
説明	Probe が使用する範囲を追加および削除できます。 **利用方法 :** ［ドメインとプローブ］表示枠で Probe をクリックします。
重要情報	特定の範囲の検索の詳細については，199 ページ「［ドメインとプローブ］表示枠」の［**IP ごとにプローブ範囲を検索**］ボタンを見つけます。

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	クリックすると［**IP 範囲を追加**］ダイアログ・ボックスが開きます。
	範囲をクリックしてこのボタンをクリックすると，リストから範囲を削除できます。

GUI 要素	説明
	クリックすると［**IP 範囲を編集**］ダイアログ・ボックスが開きます。
	権限オブジェクトを Excel，PDF，RTF，CSV，または XML 形式 でエクスポートします。詳細については，『**モデル管理**』の「参照モード」を参照してください。
	クリックすると，CSV ファイルから範囲をインポートできます。この機能を使用する前に，インポート・ファイルが有効な CSV ファイルであることと，そのファイル内の範囲が既存の範囲と競合しない（つまり重複する範囲や上書きする範囲がない）ことを確認してください。
除外された	Probe が CI を検出するために使用する範囲から除外された IP アドレスが表示されます。詳細については，188 ページ「[IP 範囲の追加/編集]ダイアログ・ボックス」を参照してください。
範囲	Probe が CI を検出するために使用するネットワーク IP アドレス。詳細については，188 ページ「[IP 範囲の追加 / 編集] ダイアログ・ボックス」を参照してください。

［ドメインとプローブ］表示枠

説明	ドメイン，Probe，または Probe の資格情報の表示，定義，または編集ができます。 **利用方法 :**［**管理**］>［**ディスカバリ**］>［**ディスカバリ プローブ設定**］。
重要情報	失われている資格情報は次のアイコンで表されます。 
関連リンク	183 ページ「ジョブ実行ポリシー」

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	選択されたものに応じて，ドメインまたは Probe を追加します。
	選択されたものに応じて，ドメインまたは Probe を削除します。
	［**IP ごとにプローブ範囲を検索**］ボタン: Probe に多くの範囲が定義されている場合，Probe を選択してこのボタンをクリックすることで特定の範囲を検索できます。［プローブ範囲の検索］ダイアログ・ボックスで，IP アドレスを入力して［**検索**］ボタンをクリックします。DDM により［範囲］表示枠で範囲が強調表示されます。
	すべてのドメインおよび Probe 情報を更新します。

[関連プローブの編集] ダイアログ・ボックス

説明	特定の Probe を選択できます。 **利用方法 :** [ポリシーの編集] ダイアログ・ボックス内の [**関連プローブ**] ボタン をクリックします。
関連リンク	183 ページ「ジョブ実行ポリシー」

[時間表の編集] ダイアログ・ボックス

説明	Probe がジョブ実行ポリシーを実行するべき時間を設定できます。 **利用方法 :** [ポリシーの編集] ダイアログ・ボックスで [**編集**] ボタン をクリックします。
関連リンク	190 ページ「[ポリシーの追加 / 編集] ダイアログ・ボックス」

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します)。

GUI 要素	説明
説明	特定のポリシーの説明を追加します。このフィールドは必須です。 **ヒント :** ここで入力したテキストは，[ジョブ実行ポリシー] 表示枠の [時間] ボックスに表示されるので，わかりやすい説明を入力してください。 
時間定義	ポリシーに含める日時のセルをクリックします。複数の時間単位を追加するには，それらのセル上でポインタをドラッグします。 **注 :** 時間単位をクリアするには，そのセルをもう 1 回クリックします。

［プロトコル パラメータ］］ダイアログ・ボックス

説明	プロトコルについて定義できる属性が表示されます。 **利用方法 :**［**ディスカバリ プローブ設定**］>［**ドメインとプローブ**］>［**ドメイン**］>［**資格情報**］を選択し，プロトコルを選択して［**追加**］または［**編集**］ボタンをクリックします。
重要情報	各プロトコルの詳細については，203 ページ「ドメイン資格情報リファレンス」を参照してください。

［対象定義］ダイアログ・ボックス

説明	プロトコルのディスカバリ対象にする範囲を設定できます。 **利用方法 :**［プロトコル パラメータ］ダイアログ・ボックスで［**編集**］ボタンをクリックします。

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
選択されたプローブ	IP 範囲を変更する特定の Probe を選択するには，［Edit］をクリックします。詳細については，120 ページ「［プローブの選択］ダイアログ・ボックス」を参照してください。
選択された範囲	➤ **すべて :** プロトコルはドメインのすべての範囲に対してディスカバリを実行します。 ➤ **選択された範囲 :** プロトコルによるディスカバリ実行対象となる特定の範囲を選択する手順と，除外される範囲を定義する手順については，188 ページ「［IP 範囲の追加 / 編集］ダイアログ・ボックス」を参照してください。

［ディスカバリ プローブ設定］ウィンドウ

説明	新しいドメインを定義したり，既存のドメインに対して新しい Probe を定義したりできます。また，各プロトコルの接続データを定義することもできます。 **利用方法 :** ［**管理**］＞［**ディスカバリ**］＞［**ディスカバリ プローブ設定**］
重要情報	➤［ドメインとプローブ］表示枠の詳細については，199 ページ「［ドメインとプローブ］表示枠」を参照してください。 ➤［詳細］表示枠の詳細については，193 ページ「［詳細］タブ」を参照してください。
関連リンク	203 ページ「ドメイン資格情報リファレンス」

Probe の選択

［プローブの選択］，［TQL 出力用プローブ制限の編集］，および［関連プローブの編集］ダイアログ・ボックスには，次の要素が含まれています（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	**選択した Probe を追加 :** クリックすると，［選択されたプローブ］カラムに Probe を追加できます。
	**選択した Probe を削除 :** クリックすると，［選択されたプローブ］カラムから Probe を削除できます。
すべてのディスカバリ Probe	➤ これを選択すると，［未選択のプローブ］リスト内のすべての Probe を追加できます。 ➤［未選択のプローブ］リストから特定の Probe を追加するには，このオプションの選択を解除します。
未選択のプローブ	ポリシー / フィルタ / 制限に含まれていない Probe。
選択されたプローブ	ポリシー / フィルタ / 制限に含まれている Probe。

ドメイン資格情報リファレンス

本項では，プロトコルの資格情報について説明します。資格情報の属性を編集できます。詳細については，201 ページ「[プロトコル パラメータ]] ダイアログ・ボックス」を参照してください。

注：次の情報はバージョンにより異なる可能性があります。内容の実装への変更によりプロトコルの属性が更新される場合があります。

[ドメインとプローブ] 表示枠でプロトコルが選択されている場合は，次の要素が表示されます（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	新しい接続詳細を追加するには，このボタンをクリックして [**プロトコル パラメータを追加**] ダイアログ・ボックスを開きます。
	プロトコルを選択してこのボタンをクリックすると，接続詳細が削除されます。削除する場合は，確認メッセージに対して [**OK**] をクリックします。
	接続詳細を編集するには，[**プロトコル パラメータ**] ダイアログ・ボックスでプロトコルを選択してこのボタンをクリックします。
	プロトコルのインスタンスを上下に移動するには，そのプロトコルを選択して矢印をクリックします。 どのポリシーが最初にチェックされるかは，リスト内でのポリシーの順序によって決まります。つまり，起動されたジョブはポリシー・リストを一番上から下へ向かってチェックしていきます。いずれかのポリシー内にそのジョブの名前が存在していたら，そのジョブは実行されます。プロトコルへのジョブの追加方法については，190 ページ「[ポリシーの追加 / 編集] ダイアログ・ボックス」を参照してください。ジョブ実行ポリシーの詳細については，184 ページ「ポリシーの順序の例」を参照してください。

GUI 要素	説明
<資格情報を右クリックする>	次のオプションから選択できます。 ➤ **編集:** リモート・マシン上にあるアプリケーションに DDM が接続できるようにするプロトコル・パラメータ（ユーザ名やパスワードなど）を入力するには，このオプションを選択します。 ➤ **旧インタフェースを使って編集 :** このオプションは次の場合に選択します。 ➤ 旧バージョンの HP Universal CMDB で，本バージョンに存在しないパラメータをこのプロトコルに追加した場合。 ➤ 本バージョンでの値を削除できない場合。たとえば，本バージョンではポート番号が空白の SQL プロトコル資格情報は設定できません。このオプションを選択して旧バージョンの［プロトコル パラメータを編集］ダイアログ・ボックスを開くと，ポート番号を削除できます。 ➤ **資格情報のチェック :** 開いたボックスで，プロトコルが実行されるリモート・マシンの IP アドレスを入力します。Probe はその IP アドレスへの接続を試み，接続が成功したかどうかを知らせる回答を返します。
<タイトルの右クリック>	次のオプションから選択します。 ➤ **カラムを非表示 :** カラムが表示されているときに表示されます。 ➤ **全カラムを表示 :** カラムが表示されていないときに表示されます。 ➤ **カラムの選択 :** このオプションを選択すると，カラムの表示順を変更できます。 ➤ **自動サイズ変更カラム :** 内容の長さに合わせてカラムの幅を変更する場合に選択します。

どのプロトコル資格情報にも，次のパラメータが含まれています。

パラメータ	説明
インデックス	プロトコルのインスタンスが接続の試行に使用される順序を示します。インデックスの値が小さくなるほど優先度が高くなります。 標準設定 : **9999**。標準設定を変更しなかった場合，このプロトコル・インスタンスは最後に使用されます。
ネットワーク・スコープ	プロトコルによるディスカバリの対象となる範囲を変更したり，Probe を選択したりするには，［**編集**］をクリックします。詳細については，201 ページ「［対象定義］ダイアログ・ボックス」を参照してください。 標準設定 : **すべて**。
ユーザ・ラベル	特定のプロトコル資格情報を後で識別するのに役立つラベルを入力します。入力できる文字数は最大 50 字までです。

本項の内容

➤ 221 ページ「WebSphere プロトコル」
➤ 222 ページ「WMI プロトコル」

JBoss プロトコル

パラメータ	説明
ポート番号	ポート番号。
接続タイムアウト	タイムアウトまでの時間です（ミリ秒単位）。この時間が経過すると，Probe は JBoss アプリケーション・サーバへの接続の試みを中止します。
ユーザ名	アプリケーションに接続するために必要なユーザの名前。
[パスワード]	アプリケーションに接続するために必要なユーザのパスワード。

LDAP プロトコル

パラメータ	説明
Connection Timeout	タイムアウトまでの時間です（ミリ秒単位）。この時間が経過すると，Probe は LDAP アプリケーション・サーバへの接続の試みを中止します。
LDAP Authentication Method	Simple: サポートされている認証方法です。
Port Number	ポート番号。
Trust Store File	LDAP トラスト･ストア･ファイルへのフル･パスを入力します。 トラスト・ストア・ファイルを使用するには，次のどちらかを行います。 ➤ ファイルの名前（拡張子を含む）を入力し，そのファイルをディスカバリ・リソース・フォルダ < **ディスカバリ Probe のインストール・ディレクトリ** >**¥root¥lib¥collectors¥probeManager¥discoveryResources¥** に置きます。 ➤ トラスト・ストア・ファイルのフル・パスを挿入します。
Trust Store Password	LDAP トラスト・ストアのパスワード。

パラメータ	説明
User Name	アプリケーションに接続するために必要なユーザの名前。
Password	アプリケーションに接続するために必要なユーザのパスワード。

NNM プロトコル

パラメータ	説明
Connection Timeout	タイムアウトまでの時間です（ミリ秒単位）。この時間が経過すると，Probe は NNM サーバへの接続の試みを中止します。
NNM Password	NNM Web サービス用のパスワード（たとえば **Openview** など）。
NNM User name	NNM Web サービスのユーザ名（たとえば **system** など）。
NNM Webservice Port	NNM サーバの Web サービス・ポート番号（たとえば **80**）。
NNM Webservice Protocol	NNMi Web サービス用のプロトコル（標準設定値は **http**）。
UCMDB Password	UCMDB Web サービス用のパスワード（標準設定値は **admin** など）。
UCMDB Username	UCMDB Web サービスのユーザ名（標準設定値は **admin** など）。
UCMDB Webservice Port	UCMDB Web サービスのポート番号（標準設定値は **8080**）。
UCMDB Webservice Protocol	UCMDB Web サービス用のプロトコル（標準設定値は **http** など）。

NTCMD プロトコル

パラメータ	説明
タイムアウト	タイムアウトまでの時間です（ミリ秒単位）。この時間が経過すると，Probe は NTCMD サーバへの接続の試みを中止します。
ユーザ名	ホストに管理者として接続するために必要なユーザの名前。
パスワード	ホストに管理者として接続するために必要なユーザのパスワード。
Windows ドメイン	資格情報が定義されている Windows ドメイン。たとえば，**user:mylab¥admin, password: xxx** は，ユーザ **admin** とパスワード **xxx** が **mylab** ドメインで定義されていることを意味します。

SAP JMX プロトコル

パラメータ	説明
ポート番号	SAP JMX ポート番号です。SAP JMX ポートの構造は，通常は 5 ＜システム番号＞ 04 という形式になっています。たとえば，システム番号が 00 の場合，このポートは 50004 となります。 検出された SAP JMX ポートへの接続を試みる場合は，このフィールドを空白のままにしておきます。SAP JMX ポート番号は，**portNumberToPortName.xml** 構成ファイルで定義されています。
接続タイムアウト	タイムアウトまでの時間です（ミリ秒単位）。この時間が経過すると，Probe は SAP JMX コンソールへの接続の試みを中止します。
ユーザ名	アプリケーションに管理者として接続するために必要なユーザの名前。
パスワード	アプリケーションに管理者として接続するために必要なユーザのパスワード。

SAP プロトコル

<table>
<tr><th>パラメータ</th><th colspan="2">説明</th></tr>
<tr><td>タイムアウト</td><td colspan="2">タイムアウトまでの時間です（ミリ秒単位）。この時間が経過すると，Probe は SAP コンソールへの接続の試みを中止します。</td></tr>
<tr><td rowspan="4">ユーザ名</td><td colspan="2">SAP システムにログインするために必要なユーザの名前。
このユーザは，次の権限を持っている必要があります。</td></tr>
<tr><td>認証オブジェクト</td><td>認証</td></tr>
<tr><td>S_RFC</td><td>S_RFC オブジェクトの場合，RFC1，SALX，SBDC，SDIF，SDIFRUNTIME，SDTX，SLST，SRFC，STUB，SUTL，SXMB，SXMI，SYST，SYSU，SEU_COMPONENT の権限を取得します。</td></tr>
<tr><td>S_XMI_PROD</td><td>EXTCOMPANY=MERCURY;EXTPRODUCT=DARM;INTERFACE=XAL</td></tr>
<tr><td></td><td>S_TABU_DIS</td><td>DICBERCLS=SS; DICBERCLS=SC</td></tr>
<tr><td>パスワード</td><td colspan="2">SAP システムにログインするために必要なユーザのパスワード。</td></tr>
<tr><td>SAP クライアント番号</td><td colspan="2">標準設定値（800）を使用することをお勧めします。</td></tr>
<tr><td>SAP システム番号</td><td colspan="2">標準設定値（00）を使用することをお勧めします。</td></tr>
<tr><td>SAP ルータ文字列</td><td colspan="2">ルータ文字列は，1 つ以上の SAProuter プログラムを使用している 2 つのホスト間で必要な接続を記述します。これらの SAP ルータ・プログラムは各自のルート許可テーブル（http://help.sap.com/saphelp_nw04/helpdata/en/4f/992dfe446d11d189700000e8322d00/content.htm）をチェックして，その先行ホストと後続ホストとの間の接続が許可されているかどうかを調べます。許可されている場合，SAProuter はその接続を設定します。</td></tr>
</table>

Siebel ゲートウェイ・プロトコル

パラメータ	説明
接続タイムアウト	タイムアウトまでの時間です（ミリ秒単位）。この時間が経過すると，Probe は Siebel ゲートウェイ・コンソールへの接続の試みを中止します。
ユーザ名	Siebel エンタープライズにログインするために必要なユーザの名前。
パスワード	Siebel エンタープライズにログインするために必要なユーザのパスワード。
Siebel サイト名	Siebel エンタープライズの名前。
Siebel クライアントへのパス	Probe サーバ上の srvrmgr がコピーされた場所。詳細については，『**Discovery and Dependency Mapping Content Guide (英語版)** 』の「Prerequisites – Copy the driver Tool to the Probe Server」を参照してください。 **注：** srvrmgr のバージョンが異なる複数のプロトコル・エントリが存在する場合は，新しいバージョンのエントリが古いバージョンのエントリより前に表示されます。たとえば，Siebel 7.5.3 と Siebel 7.7 を検出するには，Siebel 7.7 に関するプロトコル・パラメータを定義してから，Siebel 7.5.3 に関するプロトコル・パラメータを定義します。

SNMP プロトコル

パラメータ	説明
ポート	（SNMP バージョン v1，v2，および v3 の場合）SNMP エージェントがリッスンするポート番号。
タイムアウト	タイムアウトまでの時間です（ミリ秒単位）。この時間が経過すると，Probe は SNMP エージェントへの接続の試みを中止します。
再試行	Probe が SNMP エージェントへの接続を試行する回数。この回数を超えると，Probe は接続の試みを中止します。

パラメータ	説明
バージョン 1, 2	**コミュニティ**: SNMP サービス・コミュニティ（SNMP サービスの設定時に定義した，読み取り専用のコミュニティや読み取り / 書き込みのコミュニティなど）に接続するときに使用した認証パスワードを入力します。
バージョン 3	**認証方法**：管理情報へのアクセスのセキュリティを確保するために，次のオプションの 1 つを選択します。 ➤ **NoAuthNoPriv:** このオプションを使用すると，セキュリティ，機密保護，プライバシー保護はいっさい提供されません。このオプションは, 開発やデバッグなど特定のアプリケーション用にセキュリティをオフにするのに役立ちます。このオプションでは，認証のために必要なのはユーザ名だけです（v1 および v2 の要件と同様に）。 ➤ **AuthNoPriv:** 管理アプリケーションにログオンするユーザは，SNMP v3 エンティティによって認証された後, エージェント上の MIB オブジェクト内の値へのアクセスを許可されます。このオプションを使用した場合は，ユーザ名，パスワード，および認証アルゴリズム（HMAC MD5 または HMAC SHA アルゴリズム）が必要です。 ➤ **AuthPriv:** 管理アプリケーションにログオンするユーザは，SNMP v3 エンティティによって認証された後，エージェント上の MIB オブジェクト内の値へのアクセスを許可されます。さらに，管理アプリケーションから SNMP v3 エンティティへのすべての要求と応答は暗号化され，すべてのデータのセキュリティが確保されます。このオプションを使用した場合は，ユーザ名，パスワード，および認証アルゴリズム（HMAC MD5 または HMAC SHA）が必要です。 **ユーザ名**：管理アプリケーションへのログオンを許可されたユーザの名前。 **パスワード**：管理アプリケーションにログオンするために使用されるパスワード。 **認証アルゴリズム**: MD5 および SHA アルゴリズムがサポートされています。 **プライバシー・キー**: SNMP v3 メッセージ内の範囲設定された PDU 部分を暗号化するために使用される秘密鍵。 **プライバシー・アルゴリズム**: DES アルゴリズムがサポートされています。

SQL プロトコル

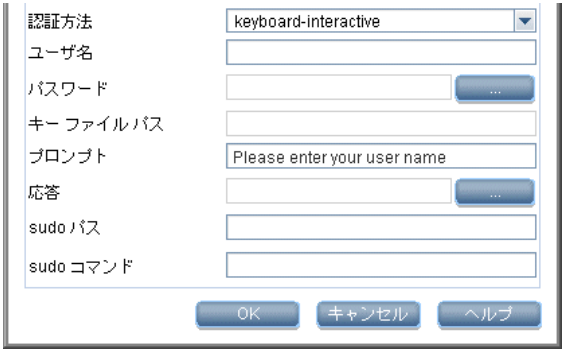
パラメータ	説明
データベース・タイプ	データベース・タイプ。ボックスから適切なタイプを選択します。
ポート	データベース・サーバがリッスンするポート番号です。 ➤ ポート番号を入力すると，DDM はそのポート番号を使って SQL データベースに接続しようと試みます。 ➤ **Oracle データベースの場合**：環境内に多くの Oracle データベースがあり，個々のデータベース・ポートそれぞれについて新しい資格情報を作成しないで済ませる場合は，［ポート番号］フィールドを空のままにしておくことができます。DDM は Oracle データベースにアクセスするときに，portNumberToPortName.xml ファイルを参照し，各 Oracle データベース・ポートについて正しいポート番号を取得します。 **注**：次の場合に，ポート番号を空白のままにしておくことができます。 ➤ すべての Oracle データベース・インスタンスが portNumberToPortName.xml ファイルに追加されている。詳細については，228 ページ「portNumberToPortName.xml ファイル」を参照してください。 ➤ すべての Oracle データベース・インスタンスへのアクセスに，同じユーザ名とパスワードが必要。
タイムアウト	タイムアウトまでの時間です（ミリ秒単位）。この時間が経過すると，Probe はデータベースへの接続の試みを中止します。
ユーザ名	データベースに接続するために必要なユーザの名前。
パスワード	データベースに接続するために必要なユーザのパスワード。
インスタンス名	データベースの名前。

SSH プロトコル

F-Secure アプリケーションが SSH サーバで実行中の Windows マシンを検出する場合の F-Secure の設定の詳細については，『**Discovery and Dependency Mapping Content Guide (英語版)** 』の「Host Connection by Shell: Discover Windows Running F-Secure」を参照してください。

パラメータ	説明
ポート	標準設定では，SSH エージェントはポート 22 を使用します。SSH 用に別のポートを使用している場合は，そのポート番号を入力してください。
タイムアウト	タイムアウトまでの時間です（ミリ秒単位）。この時間が経過すると，Probe はリモート・マシンへの接続の試みを中止します。 UNIX プラットフォームの場合 : サーバが遅い場合は，［タイムアウト］の値を 40000 に変更することをお勧めします。
バージョン	**SSH2**。SSH-2 のみを通じて接続します。 **SSH1**。SSH-1 のみを通じて接続します。 **SSH2 または SSH1**。SSH-2 を通じて接続し，エラーが発生した場合は（そのサーバが SSH-2 をサポートしていない場合は）SSH-1 を通じて接続を試みます。
Shell コマンド・セパレータ	シェル内の異なるコマンド間を区切る（同じ行で複数のコマンドを実行できるようにするために）文字。 たとえば UNIX では，標準設定のシェル・コマンド・セパレータはセミコロン（**;**）です。 Windows では，シェル・コマンド・セパレータはアンパサンド（**&**）です。
認証方法	SSH にアクセスするための次のオプションから 1 つを選択します。 ➤ **password**: ユーザ名とパスワードを入力します。 ➤ **publickey**: ユーザ名と，クライアントを認証する鍵ファイルへのパスを入力します。 ➤ **keyboard-interactive**: 質問と答を入力します。詳細については，215 ページ「プロンプトと応答」を参照してください。
ユーザ名	SSH ネットワーク・プロトコルを使ってホストに接続するために必要なユーザの名前。

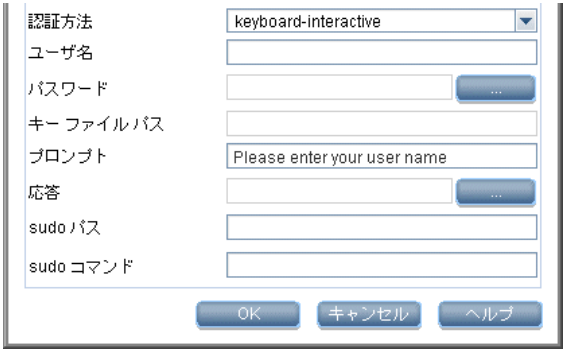
パラメータ	説明
パスワード	ホストに接続するために必要なユーザのパスワード。
キー・ファイル・パス	(publickey 認証方式を選択した場合にのみ有効) 認証鍵の位置です。(一部の環境では，SSH エージェントに接続するために鍵のフル・パスが必要です)。 **注** : Probe マシン上の鍵ファイルへの完全なフル・パスを入力してください。

パラメータ	説明
プロンプトと応答	(**keyboard-interactive** 認証方式を選択した場合にのみ有効) サーバが情報入力のために 1 つ以上のプロンプトを送信し，クライアントがそれらを表示してユーザがキーボードから入力した応答を返信する方法。 プロンプトと予期される応答の例を次に示します。 **プロンプト** : Please enter your user name. **応答** : Shelly-Ann **プロンプト** : What is your age? **応答** : 21 **プロンプト** : This computer is HP property. Press y to enter. **応答** : y これらのプロンプトと応答を作成するには，以下の文字列をカンマで区切って各フィールドに入力します。 **プロンプト** : user,age,enter **応答** : Shelly-Ann,21,y SSH プロンプトに表示されるとおりに完全な文字列を入力することができます。次に例を示します。  または，**user** などのキーワードを入力することもできます。DDM はこのキーワードを正しいプロンプトにマッピングします。

パラメータ	説明
sudo パス	sudo コマンドへのフル・パス。パスはカンマで区切ります。
sudo コマンド	sudo コマンドで実行できるコマンドのリスト。コマンドはカンマで区切ります。すべてのコマンドを sudo コマンドで実行することを指定するには，このフィールドにアスタリスク（*****）を追加します。

Telnet プロトコル

パラメータ	説明
ポート	ポート番号。標準設定では，Telnet エージェントはポート 23 を使用します。環境内で Telnet 用に別のポートを使用している場合は，必要なポート番号を入力してください。
タイムアウト	タイムアウトまでの時間です（ミリ秒単位）。この時間が経過すると，Probe はリモート・マシンへの接続の試みを中止します。 UNIX プラットフォームの場合 : サーバが遅い場合は，［接続タイムアウト］の値を 40000 に変更することをお勧めします。
認証方法	Telnet にアクセスするための次のオプションから 1 つを選択します。 ➤ **password**: ユーザ名とパスワードを入力します。 ➤ **keyboard-interactive**: 質問と答を入力します。詳細については，215 ページ「プロンプトと応答」を参照してください。
ユーザ名	ホストに接続するために必要なユーザの名前。
パスワード	ホストに接続するために必要なユーザのパスワード。

パラメータ	説明
プロンプトと応答	(keyboard-interactive 認証方式を選択した場合にのみ有効) サーバが情報入力のために 1 つ以上のプロンプトを送信し，クライアントがそれらを表示してユーザがキーボードから入力した応答を返信する方法。 プロンプトと予期される応答の例を次に示します。 **プロンプト** : Please enter your user name. **応答** : Shelly-Ann **プロンプト** : What is your age? **応答** : 21 **プロンプト** : This computer is HP property. Press y to enter. **応答** : y これらのプロンプトと応答を作成するには, 以下の文字列をカンマで区切って各フィールドに入力します。 **プロンプト** : user,age,enter **応答** : Shelly-Ann,21,y Telnet プロンプトに表示されるとおりに完全な文字列を入力することができます。次に例を示します。  または，**user** などのキーワードを入力することもできます。DDM はこのキーワードを正しいプロンプトにマッピングします。

パラメータ	説明
sudo パス	sudo コマンドへのフル・パス。パスはカンマで区切ります。
sudo コマンド	sudo コマンドで実行できるコマンドのリスト。コマンドはカンマで区切ります。すべてのコマンドを sudo コマンドで実行することを指定するには，このフィールドにアスタリスク（*****）を追加します。

UDDI プロトコル

パラメータ	説明
タイムアウト	タイムアウトまでの時間です（ミリ秒単位）。この時間が経過すると，Probe は UDDI レジストリへの接続の試みを中止します。
UDDI レジストリ URL	UDDI レジストリがある場所の URL。

VIM（VMware Infrastructure Management）プロトコル

パラメータ	説明
Connection Timeout	タイムアウトまでの時間です（ミリ秒単位）。この時間が経過すると，Probe は VMware インフラストラクチャへの接続の試みを中止します。
Port Number	DDM は，Network ー VMware ジョブの 1 つを処理するときに，ここで定義された番号を使用します。 ポート番号を空白のままにすると，DDM は WMI クエリを実行してレジストリからポート番号を抽出します。DDM は **HKLM¥SOFTWARE¥VMware, Inc.¥VMware VirtualCenter** に対してクエリを実行し，**HttpsProxyPort** または **HttpProxyPort** 属性を探します。 ➤ **HttpsProxyPort** 属性が見つかると，DDM はその値をポート番号として使用し，プレフィックスを **HTTPS** に設定します。 ➤ **HttpProxyPort** 属性が見つかると，DDM はその値をポート番号として使用し，プレフィックスを **HTTP** に設定します。
Use SSL	**true**: DDM は SSL（Secure Sockets Layer）プロトコルを使って VMware インフラストラクチャにアクセスし，プレフィックスを **HTTPS** に設定します。 **false**: DDM は http プロトコルを使用します。

パラメータ	説明
User Name	VMware インフラストラクチャに接続するために必要なユーザの名前。
User Password	VMware インフラストラクチャに接続するために必要なユーザのパスワード。

WebLogic プロトコル

パラメータ	説明
ポート番号	ポート番号を入力すると，DDM はそのポート番号を使って WebLogic に接続しようと試みます。 ただし，環境内に多くの WebLogic マシンがあることがわかっていて，マシンごとに新しい資格情報を作成したくない場合には，［ポート番号］フィールドを空白のままにしておきます。DDM は WebLogic マシンにアクセスするときに，そのマシン上ですでに検出されている（Network Connection − Active Discovery モジュールを使った TCP スキャンによって）WebLogic ポート（portNumberToPortName.xml で定義されている）を参照します。 **注**：次の場合に，ポート番号を空白のままにしておくことができます。 ➤ すべての WebLogic ポートが portNumberToPortName.xml ファイルに追加されている。詳細については，228 ページ「portNumberToPortName.xml ファイル」を参照してください。 ➤ すべての WebLogic インスタンスへのアクセスに，同じユーザ名とパスワードが必要。
接続タイムアウト	タイムアウトまでの時間です（ミリ秒単位）。この時間が経過すると，Probe は WebLogic アプリケーション・サーバへの接続の試みを中止します。
ユーザ名	アプリケーションに接続するために必要なユーザの名前。
パスワード	アプリケーションに接続するために必要なユーザのパスワード。
プロトコル	DDM がセキュリティを確保してサーバに接続するべきかどうかを決定する，アプリケーション・レベルのプロトコル。**http** か **https** を入力します。

パラメータ	説明
トラスト・ストア・ファイルのパス	SSL トラスト・ストア・ファイルのフル・パスを入力します。 トラスト・ストア・ファイルを使用するには，次のどちらかを行います。 ➤ ファイルの名前（拡張子を含む）を入力し，そのファイルをディスカバリ・リソース・フォルダ **< ディスカバリ Probe のインストール・ディレクトリ >¥root¥lib¥collectors¥probeManager¥discoveryResources¥j2ee¥weblogic¥<WebLogic のバージョン >** に置きます。 ➤ トラスト・ストア・ファイルのフル・パスを挿入します。
トラスト・ストア・パスワード	SSL トラスト・ストアのパスワード。
キー・ストア・ファイルのパス	SSL キー・ストア・ファイルのフル・パスを入力します。 キー・ストア・ファイルを使用するには，次のどちらかを行います。 ➤ ファイルの名前（拡張子を含む）を入力し，そのファイルをディスカバリ・リソース・フォルダ **< ディスカバリ Probe のインストール・ディレクトリ >¥root¥lib¥collectors¥probeManager¥discoveryResources¥j2ee¥weblogic¥<WebLogic のバージョン>** に置きます。 ➤ キー・ストア・ファイルのフル・パスを挿入します。
キー・ストア・パスワード	キー・ストア・ファイルのパスワード。

WebSphere プロトコル

パラメータ	説明
ポート	WebSphere システム管理者から提供されたプロトコル・ポート番号。 WebSphere システム管理者から提供されたユーザ名とパスワードを使って管理コンソールに接続することによって，プロトコル・ポート番号を取得することもできます。 ブラウザで，**http:/<ホスト>:9060/admin** という形式で URL を入力します。 ➤ **<ホスト>**は，WebSphere プロトコルを実行しているホストの IP アドレスです。 ➤ **9060** は，WebSphere コンソールに接続するために使用されるポートです。 必要なポート番号を取得するには，［**Servers**］＞［**Application Servers**］＞［**Ports**］＞［**SOAP_CONNECTOR_ADDRESS**］にアクセスします。
タイムアウト	タイムアウトまでの時間です（ミリ秒単位）。この時間が経過すると，Probe は WebSphere サーバへの接続の試みを中止します。
ユーザ名	アプリケーションに接続するために必要なユーザの名前。
パスワード	アプリケーションに接続するために必要なユーザのパスワード。
トラスト・ストア・ファイルのパス	SSL トラスト・ストア・ファイルのフルパスを入力します。 トラスト・ストア・ファイルを使用するには，次のどちらかを行います。 ➤ ファイルの名前（拡張子を含む）を入力し，そのファイルをディスカバリ・リソース・フォルダ **C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥probeManager¥discoveryResources¥j2ee\websphere** に置きます。 ➤ トラスト・ストア・ファイルのフル・パスを挿入します。
トラスト・ストア・パスワード	SSL トラスト・ストアのパスワード。

パラメータ	説明
キーストア・ファイルのパス	SSL キー・ストア・ファイルのフル・パスを入力します。 キー・ストア・ファイルを使用するには，次のどちらかを行います。 ➤ ファイルの名前（拡張子を含む）を入力し，そのファイルをディスカバリ・リソース・フォルダ **C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥probeManager¥discoveryResources¥j2ee¥websphere** に置きます。 ➤ キー・ストア・ファイルのフル・パスを挿入します。
キー ・ストア・パスワード	キー・ストア・ファイルのパスワード。

WMI プロトコル

パラメータ	説明
ユーザ名	ホストに接続するために必要なユーザの名前。
パスワード	ホストに接続するために必要なユーザのパスワード。
Windows ドメイン	資格情報が定義されている Windows ドメイン。たとえば，**user:mylab¥admin, password: xxx** は，ユーザ **admin** とパスワード **xxx** が **mylab** ドメインで定義されていることを意味します。

第 7 章

ディスカバリ・リソースの管理

本章では，パターンや構成ファイルなどのディスカバリおよび依存関係マップのリソースの管理について説明します。

本章の内容

概念

タスク

参照先

自動的に削除されるシステム・コンポーネント

DDM Probe はディスカバリの際に，前回成功したディスカバリで見つかった CI を現在のディスカバリで見つかった CI と比較します。欠落しているコンポーネント（ディスクやソフトウェアなど）がある場合，そのコンポーネントはシステムから削除されたものと想定され，そのコンポーネントの CI は Probe のデータベースから削除されます。

DDM Probe は，エージング・メカニズムによる計算の実行を待たずに，即座に削除要求をサーバに送信します。エージングの詳細については，『**モデル管理**』の「エージング・メカニズム ― 概要」を参照してください。

特定のジョブに関して，CI インスタンスを削除するように定義できます。詳細については，228 ページ「自動的に CI を削除するように DDM Probe を設定するワークフロー」を参照してください。

標準設定では，DDM Probe は次の CIT の CI インスタンスを削除します。

ジョブ	パターン	CIT/ リンク・タイプ
Host Connection By SNMP	SNMP_NET_Dis_Connection	Contained，SNMP
Host Connection By Shell	Host_Connection_By_Shell	Contained，Shell
Host Connection By WMI	WMI_NET_Dis_Connection	Contained，WMI
Host Resources and Applications by SNMP	SNMP_HR_All	Disk，Service，Software，OS User
Host Resources and Applications by WMI	WMI_HR_All	Disk，Service，Software，OS User
Host Resources and Applications by Shell	TTY_HR_All	Disk，Service，Software，OS User，dir

注：

- 変更はジョブのパターン上で定義されます。
- ディスカバリが失敗してエラーが発生した場合は，削除のために送信されるオブジェクトはありません。
- 削除候補にする CI は慎重に選んでください。たとえば，プロセス CIT は頻繁にシャットダウンと起動を繰り返すため，ディスカバリが呼び出されるたびに削除されるおそれがあるので，自動削除の候補には適していません。
- この手順を使って，関係を削除することもできます。たとえば，ホストと IP アドレスの間では contained 関係が使用されます。頻繁に別の IP アドレスを割り当てられるラップトップ・マシンがある場合は，関係を削除すれば，そのホストに割り当てられる古い IP アドレスが累積するのを防止できます。

自動削除の例

前回の呼び出しで，DDM Probe は **Host Resources and Applications by WMI** ジョブを実行し，ディスク a，b，c，および d を持つホストを検出しました。現在の呼び出しで，Probe はディスク a，b，c を検出し，これを前回の結果と比較して，ディスク d に対応する CI を削除します。

追加情報

- 削除された CI は，Probe ログと，［統計結果］表示枠の［削除済み］カラムで確認できます。詳細については，66 ページ「Probe ログ」と 139 ページ「［統計結果］表示枠」を参照してください。
- エージングの詳細については，『**モデル管理**』の「エージング・メカニズム ― 概要」を参照してください。
- 呼び出しはスケジューラに従って実行されます。詳細については，146 ページ「［ディスカバリ スケジューラ］ダイアログ・ボックス」を参照してください。

ソフトウェア要素の検出

環境内で実行されているソフトウェア（たとえば Oracle データベース）を検出できます。

本項の内容

ディスカバリ・プロセス

ディスカバリ・プロセスは次のように実行されます。

- ソフトウェア要素ジョブがアクティブになります。
- DDM が環境内のマシン上でプロセスを検索します。
- DDM がプロセス・データ（開いているポートやコマンド・ラインの情報を含む）を Probe データベースに保存します。
- Probe データベース内のこのデータを使ってジョブが実行され, データベース内のデータに従って新しいソフトウェア要素 CI が構築されます。そしてプロセス・データからキー属性が抽出されます。ジョブはその CI を UCMDB サーバに送信します。

ソフトウェア要素の標準設定のビュー

アプリケーション間の関係のマッピングを表示する標準設定のビューとして, [**Application Components**] ビューと [**Application Components Dependencies**] ビューがあります。ビューを利用するには, [**管理**] > [**モデリング**] > [**ビュー・マネージャ**] > [**Root**] > [**Application**] > [**Deployed Software**] を選択します。

ソフトウェア要素を検出するように DDM を設定できます。詳細については, 229 ページ「ソフトウェア要素を検出する ー シナリオ」を参照してください。

ソフトウェア要素プロセスの識別

キー・プロセスを選択して，DDM で次の操作を行えるようにすることができます。

- 検出するアプリケーションの識別。
- 適切なソフトウェア要素 CI の作成。

キー・プロセスを定義しない場合，DDM は他のプロセスを検索して次の操作を行います。

- プロセスがまったく見つからない場合，DDM は CI を作成しません。
- プロセスが少なくとも 1 つ見つかる場合は，ソフトウェア要素 CI も検出される場合に限り（つまり，プロセスがソフトウェア要素 CI とリンクされている場合に限り），DDM はプロセス CI を作成します。

たとえば，DDM がアプリケーション **A** およびアプリケーション **B** という，次のような 2 つのアプリケーションの署名を検出したとします。

- アプリケーション **A:** プロセス wrapper.exe，mainA.exe を含みます。
- アプリケーション **B:** プロセス wrapper.exe，mainB.exe を含みます。

キー・プロセスが定義されておらず，**wrapper.exe** が見つかった場合は，アプリケーション **A** および **B** のソフトウェア要素が両方とも作成されます。

キー プロセス（この例では mainA.exe または mainB.exe）が定義されている場合は，DDM は該当するキー プロセスを検出します。DDM が mainA.exe を検出した場合は，アプリケーション **A** のソフトウェア要素 CI が作成され，mainB.exe を検出した場合は，アプリケーション **B** のソフトウェア要素 CI が作成されます。

DDM が wrapper.exe, も検出した場合は，プロセス CI，およびソフトウェア要素 CI からプロセス CI へのリンクが作成されます。

［ソフトウェア識別ルール エディタ］ダイアログ・ボックスの主要なフィールドの詳細については，280 ページ「プロセスの識別中」を参照してください。

portNumberToPortName.xml ファイル

portNumberToPortName.xml ファイルは，ポート番号を意味のあるポート名にマップしてポート CI を作成する際のディクショナリとして，DDM によって使用されます。ポートが検出されると，Probe はポート番号を抽出し，portNumberToPortName.xml ファイル内でそのポート番号に対応するポート名を検索して，そのポート名でポート CI を作成します。ファイル内にポート名がない場合は，ポート番号をポート名として使用します。

検出するポートの新規追加の詳細については，233 ページ「新規ポートの定義」を参照してください。

注 : トポロジ・マップでは，Network Connections – Active ディスカバリの実行結果が，ポート番号ではなくポート名とともに表示されます（ポートのタイトルは CIT で定義されたポート名属性の値です）。詳細については，『**モデル管理**』の「属性の追加 / 属性の編集] ダイアログ・ボックス」を参照してください。

自動的に CI を削除するように DDM Probe を設定するワークフロー

このタスクでは，特定の CIT の CI インスタンスが自動的に削除されるようにジョブを設定する方法を説明します。DDM Probe による CI の削除方法の詳細については，224 ページ「自動的に削除されるシステム・コンポーネント」を参照してください。

このタスクには次の手順が含まれます。

1 前提条件

［パターン管理］タブの［結果管理］表示枠で，［**未変更の結果をフィルタ処理**］チェック・ボックスが選択されていることを確認してください。詳細については，265 ページ「［結果管理］表示枠」を参照してください。

2 削除する CI を選択する

a ［**削除済み CI の自動検出を有効化**］チェック・ボックスを選択します。

b ［**追加**］ボタンをクリックして［検出されたクラスを選択］ダイアログ・ボックスを開きます。詳細については，241 ページ「［検出されたクラスを選択］ダイアログ・ボックス」を参照してください。

c ページ下部にある，［**保存**］ボタンをクリックします。

3 結果

削除済みの CI を表示するには，［統計結果］表示枠の［削除済み］カラムにアクセスします。詳細については，139 ページ「［統計結果］表示枠」を参照してください。

ソフトウェア要素を検出する－シナリオ

このシナリオでは，各データベース・インスタンスを検出するために特定の資格情報のセットを入力する必要がないように，Oracle データベースのディスカバリを設定する方法を説明します。DDM は，データベース名属性を取得する extract コマンドを実行します。

このシナリオでは，Oracle コマンド・ラインで次の構文が使用されるものと想定しています。

```
c:¥ora10¥bin¥oracle.exe UCMDB
```

このタスクには次の手順が含まれています。

- 230 ページ「前提条件」
- 230 ページ「コマンド・ライン・ルールを作成する」
- 231 ページ「属性の値を定義する」
- 232 ページ「ジョブをアクティブにする」

1 前提条件

［ソフトウェア要素属性の割り当てルール］ダイアログ・ボックスを表示します。

a ［**管理**］ > ［**ディスカバリ**］ > ［**ディスカバリ実行**］を選択します。［**ディスカバリ・モジュール**］表示枠で，［**Host Resources and Applications**］ > ［**Software Element CF by Shell**］を選択します。［プロパティ］タブで，［**グローバル構成ファイル**］ > **applicationSignature.xml** を選択します。詳細については，268 ページ「［グローバル構成ファイル］表示枠」を参照してください。

b ［**編集**］ボタンをクリックして［ソフトウェア ライブラリ］ダイアログ・ボックスを開きます。詳細については，281 ページ「［ソフトウェア ライブラリ］ダイアログ・ボックス」を参照してください。

c ［**ソフトウェア要素カテゴリ**］>［**Database**］>［**Oracle by oracle.exe process**］を選択します。［**編集**］ボタンをクリックして［ソフトウェア識別ルール エディタ］ダイアログ・ボックスを開きます。詳細については，279 ページ「［ソフトウェア識別ルール エディタ］ダイアログ・ボックス」を参照してください。

d ［**属性の設定**］ボタンをクリックして［ソフトウェア要素属性の割り当てルール］ダイアログ・ボックスを開きます。詳細については，278 ページ「［ソフトウェア要素属性の割り当てルール］ダイアログ・ボックス」を参照してください。

2 コマンド・ライン・ルールを作成する

コマンド・ライン・ルールは，検出されるプロセスを識別するテキストです（例 : oracle.exe c:¥ora10¥bin¥oracle.exe UCMDB）。テキスト・エントリを正規表現で置き換えて，ディスカバリをより柔軟にすることができます。たとえば，名前に関係なくすべての Oracle データベースを検出するルールを設定できます。

その場合，DDM は正規表現によって検出されたコマンド・ライン内の情報を使用して，CI の data_name 属性にデータベース名を設定します。

a 正規表現を含んだコマンド・ライン・ルールを作成するには，［ソフトウェア要素属性の割り当てルール］ダイアログ・ボックスの［解析ルール］表示枠で［**追加**］をクリックします。詳細については，257 ページ「［解析ルール エディタ］ダイアログ・ボックス」を参照してください。

b ［解析ルール エディタ］ダイアログ・ボックスで，ルールを作成します。

- ［ルール ID］フィールドに一意の名前 **r1** を入力します。
- ［**プロセス属性**］フィールドで［コマンド行］を選択します。

➤ ［正規表現］フィールドに，**.+¥s+(¥w+)$** という正規表現を入力します。

この正規表現は，任意の文字（**.**）の後に 1 個以上のスペース (**+¥s+**) があり，その後に 1 個以上の単語（**(¥w+)**）が続き，その単語が行の最後にある (**$**) というテキストを検索します。文字には a ～ z，A ～ Z，または 0 ～ 9 を使用することができます。コマンド・ライン c:¥ora10¥bin¥oracle.exe UCMDB は，この正規表現を満たします。

3 属性の値を定義する

このステップでは，DDM がどの属性を使って Oracle データベースを検出するかと，その属性の値を定義します。

a ［ソフトウェア要素属性の割り当てルール］ダイアログ・ボックスの［ソフトウェア要素属性の割り当て］表示枠で，属性を選択するために［**追加**］ボタンをクリックします。

b ［属性エディタ］ダイアログ・ボックスで，

➤ Oracle CIT 属性のリストからデータベース名を指定する属性を選択します。この場合は，［**The Database instance name**］です。

➤ **${＜ルール ID 名＞(＜グループ番号＞)}** という形式を使って値を入力します（この場合は **${r1(1)}**）。

このダイアログ・ボックスは，DDM によってコマンド・ライン内のデータベース名属性の値（**${r1(1)}**）が正規表現内の最初のグループ（**(¥w+)$**）と等しい Oracle データベースが検索されるように，設定されます（詳細については，230 ページ「b」の「コマンド・ライン・ルールを作成する」を参照してください）。

つまり，DDM はディスカバリの実行時に，構成ファイル内で行の最後に 1 つ以上の単語があるコマンド・ラインを探します。たとえば，次のコマンド・ラインはこの正規表現と一致します。c:¥ora10¥bin¥oracle.exe UCMDB

4 ジョブをアクティブにする

詳細については, 62 ページ「手動によるジョブのアクティブ化」および 142 ページ「[ディスカバリ モジュール] 表示枠」を参照してください。

新規ポートの定義

portNumberToPortName.xml ファイルを編集して新規ポートを定義するには，次の手順を実行します。

1 ［ディスカバリ リソースの管理］ウィンドウ（［**管理**］＞［**ディスカバリ**］＞［**ディスカバリ リソースの管理**］を選択）で［**リソースの検索**］ボタンをクリックし，［**名前**］ボックスに「**portNumber**」と入力して，portNumberToPortName.xml ファイルを検索します。［**次を検索**］をクリックして［**閉じる**］をクリックします。

portNumberToPortName.xml ファイルの説明については，228 ページ「portNumberToPortName.xml ファイル」を参照してください。

2 ファイルに 1 行追加して，パラメータに変更を加えます。

```
<portInfo portProtocol="xxx" portNumber="xxx" portName="xxx" discover="0"/>
```

- **portProtocol:** ディスカバリに使用されるネットワーク・プロトコル（udp または tcp）。
- **portNumber:** 検出されるポート番号。
- **portName:** このポートについて表示される名前。
- **discover: 1**- このポートをディスカバリの対象にします。0 - このポートをディスカバリの対象にしません。
- **cpVersion**. パッケージ・マネージャを使用して **portNumberToPortName.xml** ファイルを別の UCMDB システムにエクスポートする場合は，このパラメータを使用します。他のシステムの **portNumberToPortName.xml** ファイルにこのアプリケーション用のポートが含まれていても，追加する新規ポートが含まれていない場合は，**cpVersion** 属性により，新規ポート情報が他のシステムのファイルに確実にコピーされるようにします。

  cpVersion 値は，**portNumberToPortName.xml** ファイルのルートの値より大きくする必要があります。

  たとえば，ルート **cpVersion** 値が **3** の場合は次のようになります。

```
<portList
parserClassName="com.hp.ucmdb.discovery.library.communication.downloader.cfgfiles.KnownPortsConfigFile" cpVersion="3">
```

新規ポート・エントリには，**4** が **cpVersion** 値として含まれている必要があります。

```
<portInfo portProtocol="udp" portNumber="1" portName="A1" discover="0"
cpVersion="4"/>
```

注： ルート **cpVersion** 値が欠落している場合は，負以外の任意の数値を新規ポート・エントリに追加できます。

このパラメータは, DDM Content Pack のアップグレード時にも必要になります。詳細については ,234 ページ「cpVersion 属性を使用したコンテンツの更新の検証」を参照してください。

cpVersion 属性を使用したコンテンツの更新の検証

cpVersion 属性は portNumberToPortName.xml ファイルに含まれており, ポートが検出された DDM Content Pack リリースを示します。たとえば，次のコードでは，DDM Content Pack 5.00 で LDAP ポート 389 が検出されたことを定義しています。

```
<portInfo portProtocol="tcp" portNumber="389" portName="ldap" discover="1"
cpVersion="5"/>
```

Content Pack のアップグレード中，DDM はこの属性を使用して，既存の portNumberToPortName.xml ファイル（ユーザ定義ポートを含むと考えられる）と新規ファイルの間でスマート・マージを実行します。ユーザが以前に追加したエントリは削除されず，ユーザが以前に削除したエントリは追加されません。

portNumberToPortName.xml ファイルの説明については，228 ページ「portNumberToPortName.xml ファイル」を参照してください。

DDM Content Pack が正常にデプロイされたことを検証するには，次の手順を実行します。

1 最新のサービス・パック・リリースをインストールします。

2 UCMDB サーバを開始します。

3 全てのサービスが実行されていることを確認します。詳細については,『**HP Universal CMDB デプロイメント・ガイド**』(PDF) の「サーバのステータスの表示」を参照してください。

4 Content Pack の最新リリースをインストールおよびデプロイします。詳細については，Content Pack のインストール・ガイドを参照してください。

5 portNumberToPortName.xml ファイルにアクセスします（[**ディスカバリ・リソースの管理**] > [**ディスカバリ・パッケージ**] > [**Network**] > [**構成ファイル**] > [**portNumberToPortName.xml**]）。

6 ユーザ定義ポートがどれも削除されておらず，ユーザが削除したポートが追加されていないことを確認します。

リソース・ファイル

以下のファイルに変更を加えて，標準設定以外のシステムで DDM を有効にすることができます。これらのファイルにアクセスするには，[**ディスカバリ リソースの管理**] > [**Network**] > [**構成ファイル**] を選択します。

本項の内容

➤ 235 ページ「oidToHostClass.xml」

➤ 236 ページ「globalFiltering.xml」

oidToHostClass.xml

oidToHostClass.xml ファイルには，システム内にある ID を持つすべての CI の OID（ディスカバリおよび依存関係マップ・ジョブ）番号のリストが含まれています。このリストは，CI を 正しい CIT にマッピングするためと，検出されたオペレーティング・システムまたはデバイスの OID 番号を文字列データに変換するために必要です。

oidToHostClass.xml ファイルにアクセスするには,[ディスカバリ リソースの管理] ウィンドウで [**リソースの検索**] ボタンをクリックして [**名前**] ボックスに「**oidto**」と入力し，このファイルを検索します。[**次を検索**] をクリックして [**閉じる**] をクリックします。

このファイルが [ディスカバリ リソース] 表示枠で選択され，ファイルの内容が表示枠に表示されます。

注 : OID が検出され，その詳細が oidToHostClass.xml ファイルに含まれていない場合，その CIT は CMDB に host として登録されています。

oidToHostClass.xml ファイルには以下のパラメータが含まれています。

- **class**: 検出された OID の変換された CIT 名。CMDB と HP Universal CMDB では，この名前の下に，オペレーティング・システムまたはデバイスが表示されます。
- **vendor**: オペレーティング・システムまたはデバイスのベンダ。
- **os**: 特定のオペレーティング・システム（たとえば Linux など）。このパラメータはオプションです。
- **model**: 特定のモデル（たとえば JETDIRECT,JD30 など）。このパラメータはオプションです。
- **oid**: 検出された OID。

globalFiltering.xml

このファイルを使用して，すべてのパターンについて Probe の結果をフィルタ処理し，興味のある結果だけが HP Universal CMDB サーバに送られるようにすることができます（特定のパターンだけをフィルタ処理することもできます。詳細については，258 ページ「[パターン管理] タブ」を参照してください）。

グローバル・フィルタを追加するには，次の手順を実行します。

1. globalFiltering.xml ファイルにアクセスします。それには，[ディスカバリ リソースの管理] で Network フォルダを開いて構成ファイル・フォルダをクリックします。ファイルを選択すると，コードが表示枠に表示されます。
2. <includeFilter> マーカと <excludeFilter> マーカを見つけます。
   - **<includeFilter>:** vector マーカがこのフィルタに追加されると，フィルタに一致しない CI はすべて削除されます。このマーカを空白のままにすると，すべての結果がサーバに送られます。
   - **<excludeFilter>:** vector マーカがこのフィルタに追加されると，フィルタと一致する CI はすべて削除されます。このマーカを空白のままにすると，すべての結果がサーバに送られます。

次の例は，アドレス属性とドメイン属性を持つ ip CI を示しています。

```
<vector>
    <object class="ip">
        <attribute name="ip_address" type="String">192¥.168¥.82¥.17.*</attribute>
        <attribute name="ip_domain" type="String">DefaultProbe</attribute>
    </object>
</vector>
```

この vector が **<includefilter>** の中で定義されると，フィルタと一致しない結果はすべて削除されます。サーバに送られる結果は，ip_address が正規表現 **192¥.168¥.82¥.17.*** と一致し，ip_domain が **DefaultProbe** である結果です。

この vector が **<excludefilter>** の中で定義されると，フィルタと一致した結果はすべて削除されます。サーバに送られる結果は，ip_address が正規表現 **192¥.168¥.82¥.17.*** と一致せず, ip_domain が **DefaultProbe.** ではない結果です。

次の例は，属性を持たない network CI を示しています。すべてのネットワーク結果がサーバに送られます。

```
<vector>
    <object class="network">
    </object>
</vector>
```

注：

- フィルタ内で使用できる属性は string 型だけです。属性のデータ型の詳細については，『**モデル管理**』の「[属性] ページ」を参照してください。
- 結果が一致しているとみなされるのは，フィルタのすべての属性が CI 内の属性と同じ値である場合だけです（CI の属性のうちの 1 つがフィルタ内で指定されていない場合，その属性についてはすべての結果がフィルタと一致したことになります)。
- 1 つの CI が複数のフィルタと一致する場合もあります。CI が削除されるか残されるかは，その CI が含まれているフィルタによって決まります。
- DDM は，まず <includeFilter> に従ってフィルタ処理を行い，その後 <includeFilter> の結果に対して <excludeFilter> を適用します。

内部構成ファイル

次のファイルは内部でのみ使用されます。これらのファイルの変更は，コンテンツ記述の高度な知識を持ったユーザのみが行ってください。

- **discoveryPolicy.xml:** Probe がいつタスクを実行しないかを指定したスケジュールが含まれています。詳細については，190 ページ「[ポリシーの追加 / 編集] ダイアログ・ボックス」を参照してください。[**ディスカバリ リソースの管理**] > [**ディスカバリ パッケージ**] > [**AutoDiscoveryInfra**] > [**構成ファイル**] にあります。
- **jythonGlobalLibs.xml:** DDM がスクリプトを実行する前にロードする標準設定の Jython グローバル・ライブラリのリストです。[**ディスカバリ・リソースの管理**] > [**ディスカバリ・パッケージ**] > [**AutoDiscoveryContent**] > [**構成ファイル**] にあります。

ディスカバリ・リソース管理ユーザ・インタフェース

本項では，次の項目について説明します。

［属性エディタ］ダイアログ・ボックス

説明	属性に従って CIT を検出するルールを定義できます。属性は，正規表現に従って定義されます。 **利用方法 :**［**ソフトウェア要素属性の割り当てルール**］ダイアログ・ボックスの［ **ソフトウェア要素属性の割り当て**］表示枠で［**追加**］ボタンをクリックします。
ほかのタスク	229 ページ「ソフトウェア要素を検出する ー シナリオ」
関連リンク	257 ページ「［解析ルール エディタ］ダイアログ・ボックス」

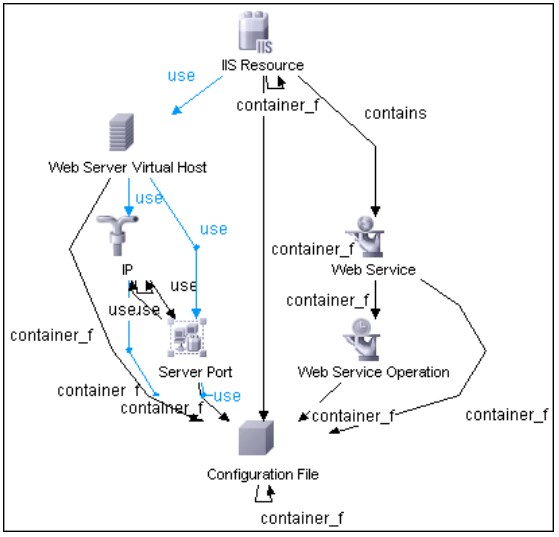
含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します)。

GUI 要素	説明
名前	ソフトウェア要素エディタで選択された CIT の属性のリストから選択します。この属性名は，正規表現によって検索された値で置き換えられます。
タイプ	属性に対して定義されている操作のタイプ（ブール，文字列，日付など）
値	［解析ルール エディタ］ダイアログ・ボックスの［ルール ID］フィールドに指定された名前と置き換えられる値。 この値には次の構文を使用します。 **${ ＜ルール ID 名＞（＜グループ番号＞）}** たとえば，**${DB_SID(1)}** は，DDM が **DB_SID** という名前のルール ID を探してその正規表現を取得するということを意味します。 DDM はその後，最初のグループ（**1**）のコードを取得しなければなりません。たとえば，正規表現 **.+¥s+(¥w+)$** の場合，最初のグループは **(¥w+)$**（つまり，行の最後にある 1 つまたは複数の単語）です。

［検出されたクラスを選択］ダイアログ・ボックス

説明	選択したパターンによって検出する CIT を選択し，特定の CIT に接続した場合にのみマッピングされるようにリンクを制限することができます。 **利用方法：** ➤［**管理**］＞［**ディスカバリ**］＞［**ディスカバリ リソースの管理**］を選択します。［ディスカバリ リソース］表示枠でパターンを選択します。［**パターン シグネチャ**］タブの［**検出された CIT**］表示枠で，［**検出された CIT の追加**］ボタンをクリックします。 ➤［**管理**］＞［**ディスカバリ**］＞［**ディスカバリ リソースの管理**］を選択します。［ディスカバリ リソース］表示枠でパターンを選択します。［**パターン管理**］タブの［**自動削除**］表示枠で，［**削除済み CI の自動検出を有効化**］チェック・ボックスを選択し，［**追加**］ボタンをクリックします。

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
リンク	リストからリンク・タイプを選択し，［エンド 1］および［エンド 2］ボックス内で  ボタンをクリックして，［構成アイテム タイプを選択してください］ダイアログ・ボックスを開きます。選択したリンク・タイプによってリンクされるときに DDM によってマッピングされるべき CIT を選択します。 **注：**DDM は CI 間のリンクを自動的に認識し，検出された CIT のマップにそれらのリンクを追加します。ただし，パターンを作成するときに，特定の CIT 間のリンクを除外する必要が生じる場合があります。たとえば，ホストと IP，およびホストとポートは，両方とも **use** によってリンクされます。**use** リンクによって接続されたホストと IP（ホストとポートではなく）についてのみ結果を受け取る必要が生じる可能性があります。パターンから受信される結果は End1 リンクと End2 リンクによって決定され，その結果は次の例が示すようにマップに反映されます。 
オブジェクト	パターンが検出する CIT のリストに追加する CIT を選択します。［パターン シグネチャ］表示枠の下部にある［**保存**］ボタンをクリックして，変更内容を保存します。

［構成ファイル］表示枠

説明	パッケージに含まれる特定の構成ファイルを編集できます。たとえば，**portNumberToPortName.xml** ファイルを編集して，特定のポート番号，名前，またはタイプが検出されるようにすることができます。 **利用方法：**［ディスカバリ リソース］表示枠で目的の構成ファイルをクリックします。
重要情報	次のファイルは内部でのみ使用されます。これらのファイルの変更は，パターン作成の高度な知識を持ったユーザのみが行ってください。 ➤ discoveryPolicy.xml ➤ jythonGlobalLibs.xml 詳細については，235 ページ「リソース・ファイル」と 238 ページ「内部構成ファイル」を参照してください。

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	構成ファイル内の特定のテキストを検索します。詳細については，251 ページ「[テキスト検索］ダイアログ・ボックス」を参照してください。
	クリックすると，構成ファイル内の特定の行に移動します。[次の行に移動］ダイアログ・ボックスで行番号を入力します。
	クリックするとファイルが外部エディタで開きます。エディタはユーザのプロファイルの一部として定義されます。詳細については，『**モデル管理**』の「［ユーザ プロファイル］ダイアログ・ボックス」を参照してください。

GUI 要素	説明
	クリックすると，外部エディタのプリファレンスを編集できます。エディタを実行するには，フラグをパスに追加します。 次の例で，  **:file** は，フラグとの関連でファイルの場所を設定します。ユーザがファイル名を設定することはできません。
	XML ファイルについて，そのコードが有効であることを示します。
	XML ファイルについて，そのコードが無効であることを示します。

［ディスカバリ パターン ソース エディタ］ウィンドウ

説明	パターン・スクリプトを編集できます。 **利用方法 :** ［ディスカバリ リソース］表示枠でパターンを右クリックし，［**パターン ソースを編集**］を選択します。
関連リンク	246 ページ「［ディスカバリ リソース］表示枠」

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	パターン・スクリプト内の特定のテキストを検索します。詳細については，251 ページ「[テキスト検索] ダイアログ・ボックス」を参照してください。
	クリックするとパターン・スクリプト内の特定の行に移動できます。[**次の行に移動**] ダイアログ・ボックスで行番号を入力します。
	クリックするとパターン・スクリプトが外部テキスト・エディタで開きます。どのエディタを使用するかは，[ユーザ プロファイル] ダイアログ・ボックスで定義します。詳細については，『**モデル管理**』の「[ユーザ プロファイル] ダイアログ・ボックス」を参照してください。
	クリックすると，外部エディタのプリファレンスを編集できます。エディタを実行するには，フラグをパスに追加します。 次の例で，  **:file** は，フラグとの関連でファイルの場所を設定します。ユーザがファイル名を設定することはできません。
	コードが有効であることを示します。
	コードが無効であることを示します。

［ディスカバリ リソース］表示枠

説明	特定のパッケージ，パターン，スクリプト，構成ファイル，または外部リソースを見つけることができます。 また，パターン，Jython スクリプト，構成ファイル，またはディスカバリ・ウィザードを作成することや，外部リソースをインポートすることもできます。 **利用方法 :** ［**管理**］＞［**ディスカバリ**］＞［**ディスカバリ リソースの管理**］を選択します。
重要情報	［ディスカバリ リソース］表示枠でどのレベルを選択したかによって，表示枠に表示される情報は異なります。 具体的な表示内容は次のとおりです。 ➤ 次のいずれかのフォルダの場合: ディスカバリ・パッケージ・ルート，特定のパッケージ，パターン，スクリプト，構成ファイル，または外部リソース : そのフォルダ内のリソースのリストが表示されます。リソースに直接アクセスするには，表示枠でそのリソースをダブルクリックします。 ➤ 特定のパターン : ［パターン シグネチャ］表示枠と［パターン管理］表示枠が表示されます。詳細については，266 ページ「［パターン シグネチャ］タブ」および 258 ページ「［パターン管理］タブ」を参照してください。 ➤ スクリプトまたは構成ファイル : スクリプト・エディタが表示されます。詳細については，276 ページ「スクリプト表示枠」を参照してください。 ➤ 外部リソース : ファイルに関する情報が表示されます。
関連リンク	『**モデル管理**』の「パッケージ マネージャのユーザ・インタフェース」

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	このボタンをクリックしてできる操作： ➤ **パターンの作成：** パターン名を入力して［**OK**］をクリックします。新しいパターンが << パッケージなし >> フォルダに追加されます。パターンを編集してください。詳細については，266 ページ「［パターン シグネチャ］タブ」および258 ページ「［パターン管理］タブ」を参照してください。パターンをパッケージに移動する方法については，『**モデル管理**』の「［カスタム パッケージ作成］ウィザードおよび［パッケージを編集］ウィザード」を参照してください。 ➤ **Jython スクリプトの作成：** スクリプト名を入力します。詳細については，276 ページ「スクリプト表示枠」を参照してください。 ➤ **構成ファイルの作成：** 構成ファイルの名前を入力します。標準設定では，構成ファイルの拡張子は .xml です。ファイルに別の拡張子を付けるには（たとえば *.properties のように），ファイルに拡張子も含めた名前を付けます。適切な XML コードやほかの内容を追加します。XML ファイルの場合は，コードが有効な場合にのみファイルを保存できます。詳細については，243 ページ「［構成ファイル］表示枠」を参照してください。 ➤ **外部リソースのインポート：** 開いたブラウザで，インポートするリソースを見つけて［**開く**］をクリックします。 ➤ **ディスカバリ・ウィザードの作成：** 新しいウィザードに名前を付けます。標準設定では，構成ファイルの拡張子は .xml です。新しいファイルが <<パッケージなし>> フォルダの［ディスカバリ ウィザード］フォルダに追加されます。このファイルはテンプレート形式です。
	クリックするとリソースを削除できます。
	クリックするとパッケージのリストを更新できます。
	クリックすると［ディスカバリ リソースの検索］ダイアログ・ボックスが開きます。フィルタ処理の詳細については，38 ページ「 結果のフィルタ処理」を参照してください。
	ディスカバリ・パッケージのルート。すべてのパッケージのリストを表示します。

GUI 要素	説明
	パッケージのルート。パッケージに含まれているすべてのリソースのリストを表示します。これらのリソースはどれでも，［ディスカバリ リソース］表示枠でそのリソースをクリックすると表示できます。
<スクリプト・ファイル>	右クリックすると次のオプションを選択できます。 ➤ **名前を付けて保存 :** ファイルを新しい名前で保存します。このオプションは，既存のファイルを複製するために使用します。新しいファイルには，元の既存ファイルの属性がすべて含まれています。新しいファイルに必要な変更を加えて保存してください。 ➤ **フレームで開く :** このオプションを選択すると，ファイルが新しいウィンドウで開きます。
<外部リソース・ファイル>	外部リソースは，DDM がディスカバリを実行するために必要とするファイルです。たとえば，資格情報なしのディスカバリには nmap.exe ファイルが必要です。 ➤ 右クリックすると次のオプションを選択できます。 ➤ **名前を付けて保存 :** リソースを新しい名前で保存します。このオプションは，既存のリソースを複製するために使用します。新しいリソースは既存のリソースの属性をすべて含んでおり，ファイル・システム内の既存リソースと同じ場所に保存されます。新しいリソースに必要な変更を加えて保存してください。 ➤ ファイルを選択すると，情報が表示枠に表示されます。 外部リソースを開いたり，エクスポートしたりできます。

GUI 要素	説明
<パターン・ファイル>	右クリックすると次のオプションを選択できます。 ➤ **名前を付けて保存 :** パターンを新しい名前で保存します。このオプションは，既存のパターンを複製するために使用します。新しいパターンには，元の既存パターンの属性がすべて含まれています。新しいパターンに名前を付けて，属性に必要な変更を加えてください。 ➤ **ディスカバリ・ジョブに移動 :** このオプションが有効になっているときにクリックすると，当該のジョブが選択された状態で［ディスカバリの実行］ウィンドウが開きます。 このオプションは，パターンがジョブに含まれている場合に有効になります。 ➤ **パターン・ソースを編集 :** パターン・ソース・エディタが開き，パターンに変更を加えられます。詳細については，244 ページ「［ディスカバリ パターン ソース エディタ］ウィンドウ」を参照してください。
<スクリプト・ファイル>	右クリックすると次のオプションを選択できます。 ➤ **名前を付けて保存:** スクリプトを新しい名前で保存します。このオプションは，既存のスクリプトを複製するために使用します。新しいスクリプトには，元の既存スクリプトの属性がすべて含まれています。新しいスクリプトに必要な変更を加えて保存してください。 ➤ **フレームで開く :** このオプションを選択すると，スクリプトが新しいウィンドウで開きます。スクリプトの編集の詳細については，244 ページ「［ディスカバリ パターン ソース エディタ］ウィンドウ」を参照してください。

［ディスカバリ リソースの検索］ダイアログ・ボックス

説明	特定のリソースまたはジョブを見つけるための検索クエリを構築できます。 **利用方法 :** ➤ ［ディスカバリ モジュール］表示枠でジョブを選択して，［**ディスカバリ ジョブの検索**］ボタンをクリックします。 ➤ ［ディスカバリ リソース］表示枠でリソースを選択して，［**リソースの検索**］フィルタ・ボタンをクリックします。
関連リンク	246 ページ「［ディスカバリ リソース］表示枠」

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します)。

GUI 要素	説明
	クリックして，開いたダイアログ・ボックスから CIT を選択します。［**OK**］をクリックすると［ディスカバリ リソースの検索］ダイアログ・ボックスに戻ります。 **注 :** ［**名前**］が選択されている場合は，このボタンにはアクセスできません。
方向	前方または後方に向かって各パッケージ内を検索します。
すべて検索	検索条件と一致するリソースがすべて，［ディスカバリ リソース］表示枠内で強調表示されます。
検索対象 ディスカバリ・リソース	次のいずれかを選択します。 ➤ **名前 :** リソースの名前または名前の一部を入力します。 ➤ **パターンの入力タイプ :** ジョブをトリガする CI。ボタンをクリックすると［構成アイテム タイプを選択してください］ダイアログ・ボックスが開きます。検索対象の CI タイプを見つけてください。 ➤ **パターンの出力タイプ :** アクティブにされたジョブの結果として検出される CI。
次を検索	検索条件と一致する次のリソースが，［ディスカバリ・リソース］表示枠内で強調表示されます。

［テキスト検索］ダイアログ・ボックス

説明	スクリプト内または構成ファイル内のテキストを検索できます。 **利用方法 :** スクリプトまたは構成ファイルを選択して，ファイルの表示枠で［**テキスト検索**］ボタンをクリックします。

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
方向	スクリプトまたは構成ファイル内を順方向または逆方向に検索します。
検索対象	検索するテキストを入力するか，下向き矢印をクリックして以前の検索条件から選択します。 隣の矢印をクリックすると，ワイルドカードまたは正規表現による検索で使用可能な記号のリストが表示されます。
オプション	検索の対象を絞り込む場合に選択します。
ターゲット	➤ **グローバル** : ファイル全体を検索します。 ➤ **選択されたテキスト** : 選択されたテキスト内を検索します。

［TQL エディタの入力］ウィンドウ

説明	特定のパターンを実行するジョブのトリガ CI としてどの CI を使用するかを定義できます。 **利用方法 :**［**ディスカバリ リソースの管理**］でパターンを選択し，［**パターン シグネチャ**］タブで［入力 TQL］ボックスの横にある［**編集**］ボタンをクリックします。
関連リンク	➤ 54 ページ「トリガ CIT，トリガ CI，入力 TQL，トリガ TQL」 ➤ 177 ページ「[TQL エディタのトリガ］ウィンドウ」

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します)。

GUI 要素	説明
<表示枠>	➤［CI タイプ・セレクタ］表示枠 ➤ 編集表示枠 ➤ 情報ページ
TQL 名	ジョブをアクティブにするトリガ TQL クエリの名前。

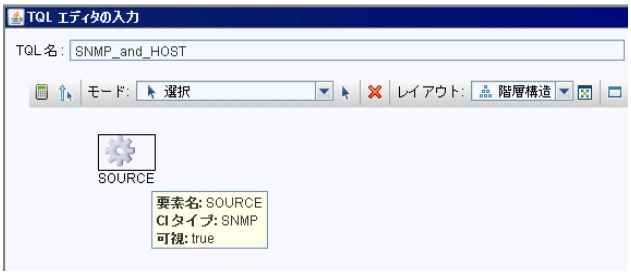
［CI タイプ・セレクタ］表示枠

説明	CMDB にある CI タイプの階層ツリー構造を表示します。詳細については,『**モデル管理**』の「CI タイプ・マネージャのユーザ・インタフェース」を参照してください。 **注**: 各 CIT の右側に，CMDB 内の各 CIT のインスタンス数が表示されます。 TQL クエリを作成または変更するには，ノードをクリックして編集表示枠にドラッグし，ノード間の関係を定義します。変更が CMDB に保存されます。詳細については,『**モデル管理**』の「ノードと関係を TQL クエリに追加」を参照してください。
ほかのタスク	➤『**モデル管理**』の「TQL クエリの定義」 ➤『**モデル管理**』の「パターン・ビューの作成」

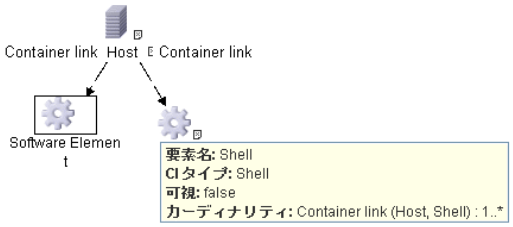
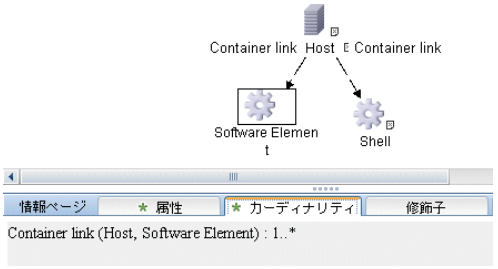
編集表示枠

説明	ノードを編集できます。

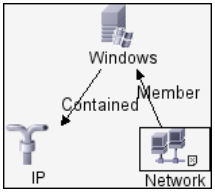
含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
<ノード>	ノードにカーソルを合わせると，そのノードの情報が表示されます。 
<右クリック・メニュー>	詳細は，『**モデル管理**』の「ショートカット・メニュー・オプション」を参照してください。
<ツールバー>	詳細は，『**モデル管理**』の「ツールバー・オプション」を参照してください。

情報ページ

説明	選択したノードおよび関係のプロパティ，条件，およびカーディナリティが表示されます。
重要情報	ノードにポインタを合わせると，情報が表示されます。  情報を含んでいるタブの横には，小さな緑色のマークが表示されます。 

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	［**編集**］をクリックすると，選択したタブに関連するダイアログ・ボックスが開きます。
属性	ノードまたは関係に対して定義された属性条件が表示されます。詳細については，『**モデル管理**』の「［属性］タブ」を参照してください。
カーディナリティ	カーディナリティは，関係のもう一方の端にあることが期待されるノードの数を定義します。たとえば，ホストと IP の関係でカーディナリティが 1:3 である場合，TQL は 1 ～ 3 個の IP に接続されているホストのみを取得します。詳細については，『**モデル管理**』の「［カーディナリティ］タブ」を参照してください。
詳細	➤ ［ノードのプロパティ］または［関係のプロパティ］ダイアログ・ボックスを開くには，［Editing］表示枠でノードまたは関係を選択して［編集］ボタンをクリックします。詳細については，『**モデル管理**』の「［ノード / 関係のプロパティ］ダイアログ・ボックス」を参照してください。 ➤ **CI タイプ**: 選択したノード / 関係の CIT です。 ➤ **可視**: 選択されたノード / 関係がトポロジ・マップ内に表示される場合は，可視であることを示すマークが付いています。ノード / 関係が表示されない場合は，［Editing］表示枠内の選択されたノード / 関係の右にボックス が表示されます。  ➤ **サブタイプを含める**: 選択された CI とその子孫を両方ともトポロジ・マップに表示します。
修飾子	ノードまたは関係に対して定義された修飾子条件が表示されます。詳細については，『**モデル管理**』の「［修飾子］タブ」を参照してください。

GUI 要素	説明
選択された ID	TQL 結果に含める必要があるものを定義するために使用される要素インスタンスが表示されます。詳細については,『**モデル管理**』の「[ID] タブ」を参照してください。

[ディスカバリ リソースの管理] ウィンドウ

説明	DDM プロセスに使用される標準設定のパラメータ値の表示と編集ができます。 **利用方法 :** [**管理**] > [**ディスカバリ**] > [**ディスカバリ リソースの管理**] を選択するか, [ディスカバリの実行] ウィンドウでジョブを右クリックします。
重要情報	**注 :** リソース (パターン, スクリプト, または構成ファイル) の横に付いているアスタリスク (*) は, そのリソースを含んでいるパッケージがデプロイされた後にそのリソースが変更されたことを示します。元のパッケージが再デプロイされると, その変更はリソースから削除されます。変更を保存するには, リソースを新しいパッケージに移動してから, そのパッケージをデプロイします (アスタリスクが消えます)。 **注意 :** パッケージの削除は, DDM プロセスの専門知識を持つ管理者が行ってください。
関連リンク	➤ 266 ページ「[パターン シグネチャ] タブ」 ➤ 268 ページ「[グローバル構成ファイル] 表示枠」 ➤ 258 ページ「[パターン管理] タブ」 ➤ 276 ページ「スクリプト表示枠」 ➤ 246 ページ「[ディスカバリ リソース] 表示枠」 ➤ 243 ページ「[構成ファイル] 表示枠」 ➤『**Discovery and Dependency Mapping Content Guide (英語版)** 』

［解析ルール エディタ］ダイアログ・ボックス

説明	属性をプロセス関連情報（IP，ポート，コマンド・ライン，および所有者）と照合するルールを作成できます。 **利用方法 :**［ソフトウェア識別ルール エディタ］ダイアログ・ボックスで［**属性の設定**］をクリックします。
重要情報	ルールの変更は，正規表現の知識を持ったユーザが行ってください。
ほかのタスク	229 ページ「ソフトウェア要素を検出する ― シナリオ」
関連リンク	➤ 240 ページ「[属性エディタ］ダイアログ・ボックス」 ➤ 279 ページ「[ソフトウェア識別ルール エディタ］ダイアログ・ボックス」

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
プロセス属性	プロセス関連情報として，［**ポート**］，［**IP**］，［**コマンド ライン**］，または［**所有者**］を選択します。ルールは，ここで選択された属性に対して起動されます。
正規表現	このソフトウェア要素を定義するプロセスを少なくとも 1 つは見つける動的な表現を作成できます。正規表現は，［プロセス属性］フィールドの値に対して実行されます。 たとえば，コマンド・ライン・プロセスに次の正規表現が含まれているとします。 **.+¥s+(¥w+)$** この正規表現は，任意の文字の後に 1 個以上のスペースがあり，その後に 1 個以上の単語（a ～ z または A ～ Z または 0 ～ 9）が続き，その単語が行の最後にあるというテキストを検索します。 次のコマンド・ラインはこの正規表現と一致します。 **c:¥ora10¥bin¥oracle.exe UCMDB**

GUI 要素	説明
ルール ID	ルールの一意の名前を入力します。ルール ID は，ソフトウェア要素の追加属性値の中でルールを識別するために必要です。詳細については，279 ページ「＜ソフトウェア要素追加属性＞」を参照してください。

［パターン管理］タブ

説明	DDM が結果をどのようにフィルタ処理するかを定義できます。 **利用方法：**［ディスカバリ リソース］表示枠で特定のパターンを選択します。
重要情報	［**保存**］ボタンをクリックして変更内容を保存します。
関連リンク	41 ページ「DiscoveryProbe.properties ファイル」

［自動削除］表示枠

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します)。

GUI 要素	説明
削除済み CI の自動検出を有効化	このチェック・ボックスを選択すると，DDM Probe が次の呼び出しで検出しなかった場合にデータベースから削除されるべき CI が自動的に削除されます。 CI のリストに CIT を追加するには，［**追加**］ボタンをクリックします。［構成アイテム タイプを選択してください］ダイアログ・ボックスで，自動的に削除する CIT を選択します。 ここで加えた変更は，パターン・ファイルに追加されます。次に例を示します。 <resultMechanism isEnabled="true"> <autoDeleteCITs isEnabled="true"> <CIT>networkshare</CIT> </autoDeleteCITs> </resultMechanism> DDM Probe が CI の削除をどのように処理するかについては，224 ページ「自動的に削除されるシステム・コンポーネント」を参照してください。 **注 :** このチェック・ボックスは，［結果管理］表示枠で［**未変更の結果をフィルタ処理**］チェック・ボックスが選択されている場合にのみ有効です。

［実行オプション］表示枠

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
通信ログの作成	Probe とリモート・マシンの間の接続を記録するログ・ファイルを作成するには，このオプションを選択します。 ➤ **常時**：このセッションの通信ログが作成されます。 ➤ **しない**：このセッションの通信ログは作成されません。 ➤ **失敗時**：実行に失敗した場合にのみ，このセッションの通信ログが作成されます。 つまり，DDM はエラーを報告します（警告が報告されても通信ログは作成されません）。このオプションは，最も所要時間が長いのはどのクエリまたは操作なのかを分析したり，分析用のデータを別の場所から送信したりする必要がある場合などに役立ちます。 ジョブが正常に完了した場合には，ログは作成されません。 ［ディスカバリ・ステータス］表示枠で要求されると，DDM は Probe から取得したログを表示します（ログが作成されている場合）。詳細については，132 ページ「［ディスカバリ ステータス］表示枠」を参照してください。 **注**：［**通信ログの作成**］が［**失敗時**］に設定されている場合でも，最後の 10 回の実行に関する通信ログをいつでもデバッグ目的で取得できます。 通信ログ・ファイルは，Probe マネージャの **C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥probeManager¥record** 通信ログがどのように機能するかについては，343 ページ「DDM コードの記録」を参照してください。

GUI 要素	説明
自動削除を有効化	次のいずれかを選択します。 ➤ **Always**: ディスカバリが成功したか，失敗したかに関わりなく，自動削除が常に有効になります。 ➤ **On Success or Warnings**: ディスカバリが成功ステータスまたは警告ステータスで完了した場合にのみ自動削除が有効になります。ディスカバリ・エラーが発生した場合は，何も削除されません。 ➤ **Only on Success**: ディスカバリが成功ステータスで完了した場合にのみ自動削除が有効になります。ディスカバリ・エラーまたは警告が発生した場合は，何も削除されません（これが標準設定です)。 この要素は，［自動削除］表示枠の［**削除済み CI の自動検出を有効化**］チェック・ボックスと連携して働きます。 ディスカバリ・ステータスの詳細については, 132 ページ「［ディスカバリ ステータス］表示枠」を参照してください。
通信ログに結果を含める	検出された結果が作成済みの通信ログに記録されるようにするには，このオプションを選択します。これらの検出された結果は，ディスカバリのさまざまな問題を調べるのに役立ちます。
最大実行時間	1 つのトリガ CI に対するパターンの実行にかけることができる最大時間。

GUI 要素	説明
最大スレッド数	各ジョブは複数のスレッドを使って実行されます。ジョブを実行するときに同時に使用できるスレッドの最大数を定義できます。このボックスを空のままにすると，Probe の標準設定のスレッド数（8）が使用されます。 標準設定値は，defaultMaxJobThreads パラメータの DiscoveryProbe.properties で定義されます。 ➤ **regularPoolThreads:** マルチスレッド・アクティビティに割り当てられるワーカ・スレッドの最大数（標準設定は 50）。 ➤ **priorityPoolThreads:** 優先ワーカ・スレッドの最大数（標準設定は 20）。 **注:** 実際のスレッド数は, regularPoolThreads + priorityPoolThreads を超えてはなりません。 **注 : Network – Host Resources and Applications** モジュールのジョブでは，Probe の内部データベースに永続的に接続する必要があります。そのため, これらのジョブの最大同時実行スレッド数は 20（内部データベースに対して許可される最大同時接続数）に制限されています。詳細については，『**Discovery and Dependency Mapping Content Guide (英語版)** 』と「Host Resources and Applications」を参照してください。

［一般オプション］表示枠

次の要素が含まれています。

GUI 要素	説明
検出データの収集を有効にする	➤ **選択した場合**：DDM は，パターンの実行結果に関するデータを収集します。そのデータは，CI の再検出を可能にするために使用されます。このデータは，IT ユニバースの［ディスカバリ］タブが正しく機能するために必要です。また，ビュー・ベースのディスカバリ・ステータス機能でも，特定のビューについて完全なディスカバリ・ステータスを集計するために，このデータが使用されます。 ➤ **クリアした場合**：DDM は，このデータを収集しません。再検出が役に立たないパターンの場合は，このチェック・ボックスをクリアする必要があります。たとえば，Range IPs by ICMP ジョブの場合はこのチェック・ボックスが標準設定でクリアされています。これは，このジョブのトリガ CI が Probe Gateway であり，このジョブによって検出される CI はすべて同じトリガ CI を持っているためです。このチェック・ボックスをクリアしなかった場合は，単一の IP を含むビューでの再検出の試みが発生し，カスタマ・ネットワーク全体に対して ping が実行されます。明らかに，これは望ましい動作ではありません。 このパターンのジョブの結果は，このチェック・ボックスが選択されている場合にのみ［Discovery for View］ダイアログ・ボックスに表示されます。詳細については，『**モデル管理**』の「アプリケーション・ディスカバリのステータス・チェック（ビューの再検出）」と「ディスカバリおよび変更サマリ・ダイアログ・ボックス」を参照してください。

［Probe 選択］表示枠

説明	どの Probe をパターンとともに使用するかを指定できます。 **利用方法：**［ディスカバリ リソース］表示枠で特定のパターンを選択します。
重要情報	標準設定では, DDM はトリガ CI の Probe をその CI の関連ホストに従って自動的に選択します。CI の関連ホストが取得された後，ホストの IP の 1 つが選択され，Probe のネットワーク範囲定義に従って Probe が選択されます。 この動作は次の場合に失敗することがあります。 ➤ トリガ CI が関連ホストを持っていない場合（**network** CIT など)。 ➤ トリガ CI のホストが複数の IP を持ち，それぞれが異なる Probe に属している場合。 これらの問題を解決するために，パターンで使用する Probe を次の手順で指定できます。 ➤［Probe 選択］セクションで，［**標準の probe 選択範囲を上書き**］を選択します。 ➤ タスクに使用する Probe を，［プローブ］ボックスに入力します。

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します)。

GUI 要素	説明
標準の probe 選択範囲を上書き	次のような計算値を使用できます。 **${Network.network_domain}** この値では，［パターン シグネチャ］タブの［トリガされた CI データ］で使用された構文を使用します。詳細については, 270 ページ「［トリガされた CI データ］表示枠」を参照してください。

[結果のグループ化] 表示枠

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
グループ化間隔（秒）	Probe 内の結果をサーバに送信する前にグループ化するために，結果をサーバに送信するまでに Probe 内に保存しておく期間を指定する値を入力します。 **注 :** 両方のボックスに値を入力した場合は，どちらか先に発生した方の値が適用されます。
グループの最大 CI 数	CI をサーバに転送する前に Probe に蓄積しておく CI の数を指定します。

[結果管理] 表示枠

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
エージング可能	CI が検出されてから DDM がその CI を期限切れとみなして削除するまでの期間を指定するエージング・メカニズムを実行する場合は，このチェック・ボックスを選択します。エージングの詳細については，『**モデル管理**』の「エージング・メカニズム ― 概要」を参照してください。
未変更の結果をフィルタ処理	最後に結果がサーバに送られて以来変更されていない，フィルタ条件と一致する CI だけが CMDB に送られるようにする場合は，このチェック・ボックスを選択します。フィルタ処理の詳細については，38 ページ「 結果のフィルタ処理」を参照してください。

[パターン シグネチャ] タブ

説明	以下を指定して，パターンを定義できます。 ➤ パターンで検出する CIT ➤ ディスカバリを実行するために必要なプロトコル **利用方法：**［ディスカバリ リソース］表示枠で特定のパターンを選択します。
ほかのタスク	318 ページ「パターンの実装」

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します)。

GUI 要素	説明
説明	パターンの詳細。
入力 TQL:	このパターンを実行するジョブのトリガ CI としてどの CI を使用できるかを定義します。 **注：**このフィールドへの入力は任意であるため，すべてのパターンが入力 TQL を含んでいるわけではありません。「**NA**」は「**該当なし**」を意味します。つまり，このパターンは現在入力 TQL を持っていません。 ➤［入力 TQL の編集］ボタンをクリックすると  ，［TQL エディタの入力］ウィンドウが開きます。 ➤［入力 TQL の削除］ボタンをクリックすると  ，パターンから入力 TQL が削除されます。 詳細については，251 ページ「［TQL エディタの入力］ウィンドウ」を参照してください。 解説については，54 ページ「トリガ CIT，トリガ CI，入力 TQL，トリガ TQL」を参照してください。 一例として，320 ページ「入力 TQL 定義の例」を参照してください。
トリガ CIT 	選択したパターンをアクティブにするトリガとして使用される CIT。トリガ CIT はパターン入力として使用されます。詳細については，319 ページ「パターン入力（トリガ CIT と入力 TQL）の定義」を参照してください。 このボタンをクリックすると，トリガとして使用する CIT を選択できます。

［検出された CIT］表示枠

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	クリックすると［**検出されたクラスを選択**］ダイアログ・ボックスが開き，パターンを使って検出する CIT を選択できます。詳細については，241 ページ「［検出されたクラスを選択］ダイアログ・ボックス」を参照してください。
	クリックすると，パターンによって検出される CIT のリストから CIT を削除できます。
	パターンによって検出された CIT とリンクのマップ（リストではなく）を表示できます。このボタンをクリックすると，［検出されたクラスのマップ］ウィンドウが開きます。パターンによって検出された CI と関係リンクが表示されます。
CIT	パターンによって検出された CIT のリスト。

［ディスカバリ パターン パラメータ］表示枠

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	クリックするとパラメータ・エディタが開きます。パラメータの詳細を入力します。ここで入力した値が属性に割り当てられます。
	パラメータを選択してこのボタンをクリックすると，パラメータ・エディタが開き，パラメータの変更ができます。
	クリックすると，パラメータを削除できます。
名前	1 つの行が 1 つのパラメータの定義を表します。
値	値と値の間はカンマで区切ります。

［グローバル構成ファイル］表示枠

説明	標準設定の構成ファイルと，パターンに必要な特定の構成ファイルを，パターンに追加できます。 **利用方法：** ➤［ディスカバリ リソースの管理］で，パターンを選択して［パターン シグネチャ］タブを選択します。 ➤［ディスカバリ実行］で，ジョブを選択して［プロパティ］タブを選択します。
重要情報	構成ファイル applicationsSignature.xml は［ソフトウェア ライブラリ］ダイアログ・ボックスを開きます。詳細については，281 ページ「［ソフトウェア ライブラリ］ダイアログ・ボックス」を参照してください。 applicationsSignature.xml ファイルには，DDM が環境内で検出しようと試みるすべてのアプリケーションのリストが格納されています。
ほかのタスク	229 ページ「ソフトウェア要素を検出する ― シナリオ」

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します)。

GUI 要素	説明
	クリックすると［**グローバル構成ファイル**］ダイアログ・ボックスが開き，パターンに必要な構成ファイルを選択できます。
	クリックすると，選択した構成ファイルを削除できます。
	構成ファイルを選択してこのボタンをクリックすると，対応するエディタが開きます。たとえば，msServerTypes.xml を選択するとスクリプト・エディタが開きます。

［必要な権限］表示枠

説明	パターンについて設定した権限を表示できます。 **利用方法：**［**ディスカバリ リソースの管理**］でパターンを選択し，［**パターン シグネチャ**］タブで［**必要な権限**］表示枠を選択します。
重要情報	➤ ワークフロー： ➤［権限の編集］ダイアログ・ボックスで権限を設定します。 ➤ この表示枠で権限を表示します。 ➤［ディスカバリ実行］ウィンドウでジョブを操作するときに，特定のジョブについてこれらの権限を表示します。 ➤ この表示枠の詳細については，272 ページ「［権限の編集］ダイアログ・ボックス」を参照してください。
関連リンク	➤ 272 ページ「［権限の編集］ダイアログ・ボックス」 ➤ 145 ページ「［ディスカバリの権限］ウィンドウ」 ➤ 91 ページ「ジョブ実行中の権限の表示」

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	クリックすると，権限オブジェクトを追加できます。［権限の編集］ダイアログ・ボックスが開きます。詳細については，272 ページ「［権限の編集］ダイアログ・ボックス」を参照してください。
	権限オブジェクトを削除するには，その権限オブジェクトを選択してこのボタンをクリックします。
	権限オブジェクトを編集するには，その権限オブジェクトを選択してこのボタンをクリックします。詳細については，272 ページ「［権限の編集］ダイアログ・ボックス」を参照してください。
	権限オブジェクトを選択して上または下ボタンをクリックすることにより，権限の順序を変更できます。ここで設定した順序に従って，資格情報が検証されます。
	権限オブジェクトを Excel，PDF，RTF，CSV，または XML 形式 でエクスポートします。詳細については，『**モデル管理**』の「参照モード」を参照してください。

[必要なディスカバリ プロトコル] 表示枠

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	[**必要なプロトコルの選択**] ダイアログ・ボックスを開きます。
	クリックすると，既存のプロトコルを削除できます。
プロトコル	タスクのためにパターンが必要とするプロトコルのリスト。たとえば，DDM が Windows システムにアクセスするためには，NTCmd プロトコルと，ユーザ名，パスワード，およびその他のパラメータが必要です。

[トリガされた CI データ] 表示枠

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	トリガ CI のデータをパターンに追加します。
	トリガ CI のデータをパターンから削除します。
	トリガ CI のデータを [パラメータ エディタ] ダイアログ・ボックスで編集できます。

GUI 要素	説明
名前	特定の CI に対してタスクを実行するために必要な情報。この情報は，タスクでクエリの対象になる CI に渡されます。
値	属性値。変数は次の構文を使用して記述します。 ${VARIABLE_NAME.attributeName} **VARIABLE_NAME** には, 次の 3 つの定義済み変数の 1 つを指定できます。 ➤ **SOURCE**。タスクのトリガとして機能する CI。 ➤ **HOST**。トリガされた CI が含まれているホスト。 ➤ **PARAMETERS**。［**パラメータ**］セクションで定義されたパラメータ。 変数を作成できます。たとえば，${SOURCE.network_netaddr} は，トリガ CI がネットワークであることを示します。

［使用するスクリプト］表示枠

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します)。

GUI 要素	説明
	スクリプトの順序を変更します。DDM は，ここに表示されている順序でスクリプトを実行します。
	スクリプトをパターンに追加します。
	パターンからスクリプトを削除します。
	選択したスクリプトをスクリプト・エディタで編集できます。
スクリプト	パターンが使用する Jython スクリプトのリスト。太字で表示されている Jython スクリプトは，現在選択しているパターンで使用されているスクリプトです。

［権限の編集］ダイアログ・ボックス

説明	作成したパターンを，ユーザがジョブの権限を表示できるように設定することができます。 **利用方法 :**［**ディスカバリ リソースの管理**］でパターンを選択し，［**パターン シグネチャ**］タブ＞［**必要な権限**］表示枠を選択して，［**追加**］ボタンをクリックします。
重要情報	ここで定義する情報は動的ではありません。つまり，パターンが変更されても，このダイアログ・ボックス内の情報は更新されません。
関連リンク	➤ 145 ページ「［ディスカバリの権限］ウィンドウ」 ➤ 91 ページ「ジョブ実行中の権限の表示」 ➤ 269 ページ「［必要な権限］表示枠」 ➤ 131 ページ「［ディスカバリ ジョブの詳細］表示枠」

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します)。

GUI 要素	説明
操作	実行される操作。
権限	［必要な権限］表示枠に表示される，権限の名前を入力します。
使用状況の詳細	権限オブジェクトとそのパラメータの説明のために自由に入力できるテキスト。通常，このテキストは権限オブジェクトのタイプに関する一般的なコメントにします。一方で，より具体的なコメントも入力できます。たとえば，ここで「**ホスト・マシンの権限**」と入力することも，特定の行について「**Windows 上で動作するホスト・マシンの権限**」と入力することもできます。

［権限オブジェクトとパラメータ］ダイアログ・ボックス

GUI 要素	説明
	クリックすると［**権限オブジェクトとパラメータ**］ダイアログ・ボックスが開きます。それぞれの権限について，複数のオブジェクトやパラメータを入力できます。 このダイアログ・ボックスで入力した情報は，［必要な権限］表示枠の［オブジェクトとパラメータ］カラムに表示されます。
	クリックすると，権限オブジェクトを削除できます。
	クリックすると，既存の権限オブジェクトを編集できます。
コンテキスト	権限オブジェクトの環境に関する具体的な情報（たとえば，Windows か UNIX かなど）。
パラメータ	ジョブの実行時に必要なパラメータ。たとえば，UNIX 権限オブジェクト cat は，/etc/passwd パラメータを必要とします。
権限のオブジェクト	コマンド，テーブル，または Jython スクリプトのほかの内容の名前。

［プロセス データ］ダイアログ・ボックス

説明	特定のソフトウェア要素を識別可能なプロセスを追加できます。 **利用方法**：［**ソフトウェア識別ルール・エディタ**］ダイアログ・ボックスの［**プロセスの識別中**］表示枠で，［**追加**］ボタンをクリックします。
関連リンク	279 ページ「［ソフトウェア識別ルール エディタ］ダイアログ・ボックス」

含まれている要素は次のとおりです：

GUI 要素 (A–Z)	説明
コマンド・ライン	ソフトウェア要素は，プロセス名を使ってマッピングすることもできます。その場合は，ソフトウェアを一意に識別するプロセス名を含むプロセス・コマンド・ラインまたはその一部（たとえば）**c:¥ora10¥bin¥oracle.exe**
キー・プロセス	ディスカバリ時に，同じようなプロセス（IP，ポート，コマンド・ライン，または所有者）を実行する複数のアプリケーションを DDM が区別する必要がある場合，このチェック・ボックスを選択します。このチェック・ボックスの説明については，227 ページ「ソフトウェア要素プロセスの識別」を参照してください。
名前	プロセスの正確な名前（たとえば **java.exe**）を入力します。

GUI 要素 (A–Z)	説明
ポート	ポート番号を入力するか，［**追加**］ボタンをクリックして［グローバル ポート リスト］からポートを選択して，ポート番号またはポート名を追加します。 ➤ 追加するプロセスが特定のポートをリッスンする必要がある場合は，そのポートを指定しなければなりません。**8888,8081,8080,81,8000,82,80** など，複数のポートをカンマで区切って入力できます。 ➤ プロセスが特定のポートをリッスンする必要がない場合は(つまり，ソフトウェア要素が任意のポートを使用できる場合は)，このフィールドを空白のままにしてください。
ポートの一致は任意	➤ ［ポート］フィールドに入力されたポートをどれもリッスンしないプロセスの検出（つまり，プロセス名のみによる識別）を有効にするには，このチェック・ボックスを選択します。このチェック・ボックスを選択した場合，単語 **optional** が［ポート］フィールドに追加されます。 ➤ ［ポート］フィールドに入力されたプロセス名とポート番号に基づいたプロセスの検出を有効にするには，このチェック・ボックスをクリアします。

［スクリプト エディタ］ウィンドウ

説明	パッケージに含まれる特定のスクリプトを編集できます。 **利用方法：** ➤ ［ディスカバリ リソース］表示枠でスクリプトを右クリックし，［**フレームで開く**］を選択します。 ➤ ［グローバル構成ファイル］表示枠で構成ファイルを選択し，［**編集**］ボタンをクリックします。 詳細については，276 ページ「スクリプト表示枠」を参照してください。

スクリプト表示枠

説明	パッケージに含まれる特定のスクリプトを編集できます。 **利用方法 :** ［ディスカバリ リソース］表示枠で特定のスクリプトをクリックします。
重要情報	スクリプト表示枠のタイトル・バーには，スクリプトの実際の物理的な保存場所が含まれています。たとえば，次のスクリプトは **C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥probeManager¥discoveryScripts**（**または probeGateway¥discoveryScripts**）にあります。 
関連リンク	第 10 章 :「コンテンツ開発と記述」

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	スクリプト内の特定のテキストを検索します。詳細については，251 ページ「［テキスト検索］ダイアログ・ボックス」を参照してください。
	クリックすると，スクリプト内の特定の行に移動できます。［次の行に移動］ダイアログ・ボックスで行番号を入力します。
	クリックするとスクリプトが外部テキスト・エディタで開きます。どのエディタを使用するかは，［ユーザ プロファイル］ダイアログ・ボックスで定義します。詳細については，『**モデル管理**』の「［ユーザ プロファイル］ダイアログ・ボックス」を参照してください。

GUI 要素	説明
	クリックするとスクリプトがテキスト・エディタで表示されます。標準設定であるソース・エディタ表示に問題が発生した場合に使用します。  **:file** は，フラグとの関連でファイルの場所を設定します。ユーザがファイル名を設定することはできません。
	簡易なコード・エディタへ切替えます。高度なエディタでは問題が起こる場合，クリックするとテキスト・エディタが開きます。
	Jython ファイルについて，そのコードが有効であることを示します。
	Jython ファイルについて，そのコードが無効であることを示します。
	下の検証情報を参照してください。 **注** : このボタンは，スクリプトに Framework API エラーが含まれている場合に表示されます。

GUI 要素	説明
<スクリプト>	パッケージが使用する Jython スクリプト。Jython の使用に関する詳細については，329 ページ「手順 3: Jython コードの作成」を参照してください。
検証情報	スクリプトがバージョン 8.00 で有効でない場合，［検証情報］はスクリプト内のエラーを表示します。次に例を示します。 Script has failed validation. At line 48: Factory.getProtocolProperty(found. This is a problem - Usage of Factory is deprecated. Use Framework.getProtocolProperty instead. ［**検証エラーの修正**］をクリックして［**OK**］をクリックすると，スクリプトが更新されます。 フレームワーク・オブジェクトの API に加えられた変更が原因で，エラーが発生する場合があります。詳細については，『**HP Universal CMDB デプロイメント・ガイド**』の「Discovery and Dependency Mapping API の変更点」を参照してください。

［ソフトウェア要素属性の割り当てルール］ダイアログ・ボックス

説明	CIT の属性値に従って特定のソフトウェア要素を検出する正規表現を定義できます。 **利用方法 :** ［ソフトウェア識別ルール エディタ］ダイアログ・ボックスで［**属性の設定**］をクリックします。
ほかのタスク	229 ページ「ソフトウェア要素を検出する ー シナリオ」
関連リンク	➤ 257 ページ「［解析ルール エディタ］ダイアログ・ボックス」 ➤ 240 ページ「［属性エディタ］ダイアログ・ボックス」 ➤ 279 ページ「［ソフトウェア識別ルール エディタ］ダイアログ・ボックス」 ➤ 58 ページ「Software Element CIT」

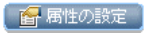
含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	クリックすると，検出する CI の属性を決定する正規表現の追加や，属性の追加ができます。
	クリックすると，正規表現または属性を削除できます。
	クリックすると，既存の正規表現または属性を編集できます。
解析ルール	詳細については，257 ページ「［解析ルール エディタ］ダイアログ・ボックス」を参照してください。
＜ソフトウェア要素追加属性＞	詳細については，240 ページ「［属性エディタ］ダイアログ・ボックス」を参照してください。

［ソフトウェア識別ルール エディタ］ダイアログ・ボックス

説明	新しいソフトウェア要素を定義できます。 **利用方法：**［ソフトウェア ライブラリ］ダイアログ・ボックスで，［**追加**］ボタンをクリックするか，既存の要素を選択して［**編集**］ボタンをクリックします。
重要情報	各解析ルールが，少なくとも 1 つのプロセスと一致する必要があります。
ほかのタスク	229 ページ「ソフトウェア要素を検出する ― シナリオ」
関連リンク	➤ 268 ページ「［グローバル構成ファイル］表示枠」 ➤ 58 ページ「Software Element CIT」

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	クリックすると，コンポーネントに属性を追加できます。詳細については，278 ページ「［ソフトウェア要素属性の割り当てルール］ダイアログ・ボックス」を参照してください。
	クリックすると，プロセスまたはソフトウェア構成ファイルを追加できます。
	プロセスまたはソフトウェア構成ファイルを選択してこのボタンをクリックすると，選択したものを削除できます。
	プロセスまたはソフトウェア構成ファイルを選択してこのボタンをクリックすると，選択したものを編集できます。
カテゴリ	次のことが行えます。 ➤ 新しいソフトウェア要素を表示するカテゴリを選択する。 ➤ 既存の要素のカテゴリを変更する。 ➤ このフィールドに名前を入力して新しいカテゴリを追加する。 ここで行った変更は，即座に［ソフトウェア ライブラリ］ダイアログ・ボックスに表示されます。
CI タイプ	検出する CIT を選択します。
構成ファイルの識別中	構成ファイルを追加するには，［**追加**］ボタンをクリックします。［構成ファイル名］ダイアログ ボックスが開きます。ソフトウェア要素の構成ファイルまでのフル・パスとファイル名を入力します。
プロセスの識別中	特定のソフトウェア要素を識別可能なプロセスを追加するには，［**追加**］ボタンをクリックします。［**プロセス データ**］ダイアログ・ボックスが開きます。詳細については，274 ページ「[プロセス データ］ダイアログ・ボックス」を参照してください。
作成済みソフトウェア名	この署名により作成されるソフトウェア要素の名前です。
ソフトウェア署名の名前	定義の名前。 **注 :** これは，ソフトウェア要素の名前ではなく，このディスカバリを類似のディスカバリから区別するために付ける名前です。

［ソフトウェア ライブラリ］ダイアログ・ボックス

説明	ソフトウェア要素の論理グループを表示できます。 **利用方法：** ➤［**ディスカバリ実行**］ウィンドウ＞［**Host Resources and Applications**］モジュール＞［**Software Element CF by Shell**］ジョブ。［プロパティ］タブで［**グローバル構成ファイル**］表示枠を見つけます。**applicationsSignature.xml** を選択して［**編集**］ボタンをクリックします。 ➤［**ディスカバリ リソースの管理**］ウィンドウ＞ **Host_Resources_Basic** パッケージ＞ **Dis_AppComponents_CF** パターンを選択します。［**パターン シグネチャ**］タブで［**グローバル構成ファイル**］表示枠を見つけます。**applicationsSignature.xml** を選択して［**編集**］ボタンをクリックします。 ➤ インフラストラクチャ・ディスカバリ・ウィザードの プリファレンス・ページで，［**検出するソフトウェア要素の選択**］ボックスを選択します。
重要情報	ソフトウェア要素は，論理的なカテゴリ別に編成されています。これらの要素の名前を変更したり，要素を別のカテゴリに移動したり，新しい要素とカテゴリを定義することができます。詳細については，279 ページ「［ソフトウェア識別ルール エディタ］ダイアログ・ボックス」の「**カテゴリ**」の項を参照してください。 このダイアログ・ボックスと［ソフトウェア識別ルール エディタ］ダイアログ・ボックスで定義したコードは，applicationsSignature.xml の中のコードを上書きします。
ほかのタスク	229 ページ「ソフトウェア要素を検出する ― シナリオ」
関連リンク	➤ 268 ページ「［グローバル構成ファイル］表示枠」 ➤ 58 ページ「Software Element CIT」

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します)。

GUI 要素	説明
	カテゴリまたはソフトウェア要素をディスカバリに含めるには，チェック・ボックスを選択します。 カテゴリまたは要素をディスカバリから除外するには，チェック・ボックスをクリアします。
	クリックすると，新しいソフトウェア要素を定義できます。詳細については，279 ページ「［ソフトウェア識別ルール エディタ］ダイアログ・ボックス」を参照してください。
	ソフトウェア要素を削除するには，その要素を選択してこのボタンをクリックします。
	ソフトウェア要素に変更を加えるには，その要素を選択してこのボタンをクリックします。詳細については，279 ページ「［ソフトウェア識別ルール エディタ］ダイアログ・ボックス」を参照してください。
＜ソフトウェア要素のリスト＞	ソフトウェア要素であるオブジェクトのリスト。

第 8 章

ステータス・スナップショット表示

本章では，DDM Probe で検出された CI の現在のステータスの表示に関する情報を提供します。

本章の内容

概念

タスク

参照先

ステータス・スナップショット表示 – 概要

Probe で検出された CI の現在のステータスを表示するには，ステータス・スナップショット表示を使用します。ステータス・スナップショット表示により，Probe からステータスが取得され，結果がビューに表示されます。

ビューは自動的に更新されません。ステータス・データを更新するには，[**Get snapshot**] ボタンをクリックします。

検出された CI の現在のステータスの表示

本タスクでは，検出された CI の現在のステータスの表示方法について説明します。

このタスクには次の手順が含まれます。

➤ 284 ページ「前提条件」

➤ 284 ページ「ステータス・スナップショット表示へのアクセス」

1 前提条件

Probe が有効になっており，HP Universal CMDB サーバに接続されていることを確認します。詳細については，39 ページ「DDM Probe のインストール」を参照してください。

2 ステータス・スナップショット表示へのアクセス

a ［**ディスカバリ**］＞［**ステータス スナップショット表示**］に移動します。

b 接続されているプローブを選択します。

c ［**Get Snapshot**］ボタンをクリックします。

d ［進行状況］リストからジョブを選択し，［**ジョブの進捗表示**］ボタンをクリックします。ジョブの進捗ウィンドウが開きます。

ステータス・スナップショット表示のユーザ・インタフェース

本項の内容

➤ ［<ジョブ名>］ダイアログ・ボックス（285 ページ）

➤ ［ステータス スナップショット表示］ウィンドウ（286 ページ）

［<ジョブ名>］ダイアログ・ボックス

説明	ジョブの詳細情報（スケジュール設定を含む）および統計情報を表示できます。 **利用方法 :** ［ステータス スナップショット表示］ウィンドウの［進行状況］表示枠でジョブを選択し，［**ジョブの進捗表示**］ボタンをクリックします。
重要情報	ジョブをダブルクリックすると，ジョブの詳細が表示されたダイアログ・ボックスが開きます。

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
ジョブの詳細	**ステータス :** ［**Scheduled**］（定義されたスケジュールに従ってジョブが実行される）または［**Running**］（現在ジョブが実行されている）となります。 **Last updated:** ジョブが最後に更新された時刻です。 **スレッド :** 現在このジョブに割り当てられているスレッドの数です。 **進行状況 :** ジョブのトリガ CI の数，および，Probe が処理を完了したトリガ CI の数です。
スケジュール	**前の呼び出し :** DDM が最後にジョブを実行した時刻です。 **次の呼び出し :** スケジュール設定されている，DDM が次にジョブを実行する時刻です。 **最終期間 :** 前の呼び出しでジョブの実行に要した時間です（単位：秒）。 **平均期間 :** Probe がこのジョブを実行するのに要した時間の平均期間です（単位：秒）。 **繰り返し :** 1 週間にジョブが実行される回数です。たとえば，ジョブが毎日実行されるようにスケジュール設定されている場合は，1 週間に 7 回実行されることになります。ジョブが週に 1 回実行されるようにスケジュール設定されている場合は，繰り返し = 1 となります。
統計結果	詳細については，288 ページ「［統計結果］表示枠」を参照してください。

［ステータス スナップショット表示］ウィンドウ

説明	検出された CI の現在のステータス，および，Probe で実行されているすべてのアクティブなジョブを表示できます。 **利用方法 :**［**管理**］＞［**ディスカバリ**］＞［**ステータス スナップショット表示**］を選択します。
重要情報	表示枠には，［ドメインとプローブ］表示枠での選択に応じて異なる情報が表示されます。 具体的な表示内容は次のとおりです。 ➤ ドメインを選択した場合は，ドメインの詳細と CIT 統計情報が表示されます。詳細については，193 ページ「［詳細］タブ」と 288 ページ「［統計結果］表示枠」を参照してください。 ➤ Probe を選択した場合は，Probe の詳細 (Probe IP など)，ジョブの進捗，CIT 統計情報が表示されます。詳細については，286 ページ「［詳細］表示枠」，287 ページ「［進行状況］表示枠」，288 ページ「［統計結果］表示枠」，および 289 ページ「表示枠」を参照してください。
ほかのタスク	284 ページ「検出された CI の現在のステータスの表示」
関連リンク	283 ページ「ステータス・スナップショット表示 – 概要」

［詳細］表示枠

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します)。

GUI 要素	説明
ドメイン・タイプ	**カスタマ :** 当該サイト用のプライベート・ドメイン。複数のドメインを定義し，各ドメインに複数の Probe を含めることができます。各 Probe には IP 範囲を含めることができますが，カスタマ・ドメイン自体について範囲を定義することはできません。 **外部 :** インターネット / パブリック・ドメイン。範囲付きで定義されたドメイン。外部ドメインには，ドメイン名と同じ名前の 1 つの Probe しか含められません。ただし，システム内に複数の外部ドメインを定義できます。 ドメインの定義の詳細については，191 ページ「［新しいドメインの追加］ダイアログ・ボックス」を参照してください。

［進行状況］表示枠

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

GUI 要素	説明
	CI を選択してこのアイコンをクリックすると，ジョブの詳細が表示されます。詳細については，285 ページ「［<ジョブ名>］ダイアログ・ボックス」を参照してください。
ジョブ	ジョブの名前です。 ジョブをダブルクリックすると，ダイアログ・ボックスが開いてジョブの詳細が表示されます。詳細については，285 ページ「［<ジョブ名>］ダイアログ・ボックス」を参照してください。
次の呼び出し	スケジュール設定されている，Probe が次に実行される時刻です。
前の呼び出し	Probe が最後に実行された時刻です。
進行状況	［Scheduled］または［Running］となります。ジョブが実行中の場合は，進行状況バーに完了した割合が表示されます。
スレッド数	現在このジョブに割り当てられているスレッドの数です。
トリガされた CI	ジョブでトリガされる CI の数です。

［統計結果］表示枠

説明	詳細情報と CIT 統計情報を表示できます。 **利用方法：**［ドメイン ブラウザ］表示枠で標準設定のドメインまたは Probe の名前をクリックします。

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します)。

GUI 要素	説明
	Probe から最新データを取得するときにクリックします（データは自動的には更新されません)。
	CIT に関する統計情報を表示する時間範囲を設定します。 ➤ **すべて：**すべてのジョブ実行の統計情報が表示されます。 ➤ **現在から/最新/直近の1時間/直近の1日/直近の1 週間：**CIT に関する統計情報を表示する期間を選択します。 ➤ **カスタム範囲：**クリックすると,［統計時間範囲のカスタマイズ］ダイアログ・ボックスが開きます。［**開始**］と［**終了**］に日付を入力するか，矢印をクリックしてカレンダーから日時を選択します。日付を削除するには，［**リセット**］をクリックします。
＜カラム・タイトル＞	CIT の順序を昇順から降順あるいは降順から昇順に変更するには，カラム・タイトルをクリックします。
＜タイトルの右クリック＞	次のオプションから選択できます。 ➤ **カラムを非表示：**特定のカラムを非表示にするときに選択します。 ➤ **全カラムを表示：**カラムが非表示になっているときに表示されます。 ➤ **カラムの選択：**カラムの表示/非表示の切り替え,または，テーブル内のカラムの順序変更を行うときに選択します。［カラム］ダイアログ・ボックスが開きます。 ➤ **自動サイズ変更カラム：**選択すると，内容の長さに合わせてカラムの幅が変更されます。 詳細については,『**参照情報**』の「[カラムの選択］ダイアログ・ボックス」を参照してください。

GUI 要素	説明
CIT	検出された CIT の名前です。
作成済み	Probe によって作成された CIT インスタンスの数です。
削除済み	Probe によって削除された CIT インスタンスの数です。
検出された CI	すべての呼び出しの全 CI の合計です。
フィルタ	［フィルタの設定］ボタンで設定された時間範囲です。
最後の更新	特定の Probe の統計情報テーブルが更新された日時です。
合計	各カラムの CI の総数です。
更新済み	更新された CIT インスタンスの数です。

表示枠

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します)。

GUI 要素	説明
	選択した Probe で検出された CI とジョブの現在のステータスを表示するときにクリックします。
前回更新	［Get snapshot］ボタンが最後にクリックされた日時 (つまり, ［ステータス スナップショット］に表示されたデータの日時）です。
プローブ IP	Probe に定義された IP アドレスです。
実行中のジョブ	Probe で実行されているジョブの数です。
予定されているジョブ	ディスカバリ・スケジューラの設定に従って実行が予定されているジョブの数です。詳細については，146 ページ「［ディスカバリ スケジューラ］ダイアログ・ボックス」を参照してください。
ステータス	Probe のステータスです（非接続または接続)。
スレッド	実行中のジョブに現在割り当てられている全スレッドの合計です。

第 9 章

DDM Web サービス API

本章の内容

概念

タスク

参照先

DDM

本章では, サードパーティ・ツールまたはカスタム・ツールで HP Discovery and Dependency Mapping Web Service を使って ディスカバリおよび依存関係マップ (DDM) をどのように管理できるかについて説明します。

使用できる操作の詳細については,『**HP Discovery and Dependency Mapping Schema Reference（英語版）**』を参照してください。ファイルは次のフォルダにあります。

C:¥hp¥UCMDB¥UCMDBServer¥j2f¥AppServer¥webapps¥site.war¥amdocs¥eng¥doc_lib¥Discovery_and_Dependency_Mapping¥DDM_Schema¥webframe.htm

表記規則

本章では，次の表記規則を使用します。

- Element : この書体は，その項目がデータベース内のエンティティか，スキーマ内で定義されている要素であることを示します。これには，メソッドに渡されたりメソッドから返されたりする構造体も含まれます。通常の書体は，その項目が一般的な説明であることを示します。
- DDM 要素およびメソッドの引数は，それらがスキーマ内で指定されているとおりに，大文字と小文字を区別して表記されます。これは通常，クラス名や，クラス・インスタンスに対する一般的な参照において，先頭の文字が大文字で表記されることを意味します。要素やメソッドへの引数の名前の先頭には大文字は使用されません。たとえば，credential は，メソッドに渡される Credential タイプの要素です。

HP Discovery and Dependency Mapping Web Service

HP Discovery and Dependency Mapping Web Service は，アプリケーションと HP Universal CMDB(UCMDB) を統合するために使用される API です。この API は，次の操作を行うためのメソッドを提供します。

- 資格情報の管理 : 表示，追加，更新，削除
- ジョブの管理 : ステータスの表示，アクティブ化，非アクティブ化
- プローブ範囲の管理 : 表示，追加，更新
- トリガの管理 : トリガ CI の追加または削除，および，トリガ TQL の追加，削除，または無効化
- ドメインおよびプローブに関する一般データの表示

HP Discovery and Dependency Mapping Web Service のユーザは，次のことを十分理解している必要があります。

- SOAP の仕様
- オブジェクト指向プログラミング言語（C++ や C#，Java など）
- HP Universal CMDB
- ディスカバリおよび依存関係マップ

権限

管理者により，Web サービスに接続するためのログイン資格情報が提供されます。必要な資格情報は，HP Universal CMDB のスタンドアロン・バージョンとともに DDM を使用しているか，それとも HP Business Availability Center から DDM を使用しているかによって異なります。

DDM を通して権限が割り当てられる場合の権限レベルは，表示，更新，および実行です。HP Business Availability Center を通して権限が割り当てられる場合の権限レベルは，表示と更新ですが，この更新には実行も含まれます。各操作に必要な権限を表示するには，『**HP Discovery and Dependency Mapping Schema Reference（英語版）**』で各操作の要求についての説明を参照してください。

権限を割り当てる方法：

- **DDM を HP Universal CMDB のスタンドアロン・バージョンとともに使用する場合：**ディスカバリ・リソースに対する権限を付与されている DDM ユーザの資格情報を使用してログインします。詳細については，『**モデル管理**』の「アクセス権の割り当て」を参照してください。
- **DDM が HP Business Availability Center に組み込まれている場合：** Business Availability Center ユーザの資格情報を使ってログインします。そのユーザは，UCMDB Web サービスのリソースに対する適切な権限を Business Availability Center によって付与されている必要があります。

Web サービスの呼び出し

HP Discovery and Dependency Mapping Web Service では，標準の SOAP プログラミング手法を使ってサーバ側のメソッドを呼び出せます。ステートメントを解析できなかった場合や，メソッドの呼び出しで問題が生じた場合，API メソッドは，SoapFault 例外を発生させます。SoapFault 例外が発生すると，サービスは 1 つ以上のエラー・メッセージ，エラー・コード，および例外メッセージ・フィールドに情報を書き込みます。エラーがない場合は，呼び出しの結果が返されます。

SOAP プログラマは，次の URL で WSDL にアクセスできます。

http://< サーバ >[: ポート]/axis2/services/DiscoveryService?wsdl

ポートの指定は，標準とは異なる設定でインストールされている場合にのみ必要です。正しいポート番号についてはシステム管理者に問い合わせてください。

サービスを呼び出すための URL は，次のとおりです。

http://<サーバ>[:ポート]/axis2/services/DiscoveryService

ディスカバリおよび依存関係マップ のメソッド

本項では，Web サービス操作とその使用方法の要約の一覧を示します。各操作の要求と応答の完全な説明については，『**HP Discovery and Dependency Mapping Schema Reference（英語版）**』を参照してください。

本項の内容

ディスカバリ・ジョブ・メソッドの管理

- **activateJob**

  指定されたジョブをアクティブにします。

- **deactivateJob**

  指定されたジョブを非アクティブにします。

- **dispatchAdHocJob**

  プローブに対してジョブを一時的にディスパッチします。ジョブはアクティブでなければならず，指定されたトリガ CI を含んでいる必要があります。

- **getDiscoveryJobsNames**

  ジョブ名のリストを返します。

➤ **isJobActive**

ジョブがアクティブかどうかをチェックします。

トリガ・メソッドの管理

➤ **addTriggerCI**

指定されたジョブに新しいトリガ CI を追加します。

➤ **addTriggerTQL**

指定されたジョブに新しいトリガ TQL を追加します。

➤ **disableTriggerTQL**

TQL がジョブをトリガしないようにしますが，ジョブをトリガするクエリのリストからその TQL を永久的に削除することはありません。

➤ **removeTriggerCI**

ジョブをトリガする CI のリストから，指定された CI を削除します。

➤ **removeTriggerTQL**

ジョブをトリガするクエリのリストから，指定された TQL を削除します。

➤ **setTriggerTQLProbesLimit**

指定されたリストに対して，ジョブ内で TQL がアクティブになるプローブを制限します。

ドメインおよび Probe データ・メソッド

➤ **getDomainType**

ドメイン・タイプを返します。

➤ **getDomainsNames**

現在のドメインの名前を返します。

➤ **getProbeIPs**

指定されたプローブの IP アドレスを返します。

➤ **getProbesNames**

指定されたドメイン内のプローブの名前を返します。

- **getProbeScope**

  指定されたプローブの対象範囲の定義を返します。

- **isProbeConnected**

  指定されたプローブが接続されているかどうかをチェックします。

- **updateProbeScope**

  指定されたプローブの対象範囲を設定し，既存の範囲を上書きします。

資格情報データ・メソッド

- **addCredentialsEntry**

  指定されたドメインについて，指定されたプロトコルに資格情報エントリを追加します。

- **getCredentialsEntriesIDs**

  指定されたプロトコルについて定義された資格情報の ID を返します。

- **getCredentialsEntry**

  指定されたプロトコルについて定義された資格情報を返します。暗号化された属性は空のデータとして返されます。

- **removeCredentialsEntry**

  指定された資格情報をプロトコルから削除します。

- **updateCredentialsEntry**

  指定された資格情報エントリのプロパティに新しい値を設定します。

データ更新メソッド

- **rediscoverCIs**

  指定された CI オブジェクトを検出したトリガを見つけて，それらのトリガを返します。

  rediscoverCIs は非同期的に実行されます。再検出がいつ完了するかを調べるには，**checkDiscoveryProgress** を呼び出します。

➤ **checkDiscoveryProgress**

指定された ID について，最新の **rediscoverCIs** 呼び出しの進行状況を返します。応答は，**0** から 1 までの値です。応答が 1 の場合，**rediscoverCIs** 呼び出しは完了しています。

➤ **rediscoverViewCIs**

指定されたビューに表示されるデータを作成したトリガを見つけて，それらのトリガを返します。

rediscoverViewCIs は非同期的に実行されます。再検出がいつ完了するかを調べるには，**checkViewDiscoveryProgress** を呼び出します。

➤ **checkViewDiscoveryProgress**

指定されたビューについて，最新の **rediscoverViewCIs** 呼び出しの進行状況を返します。応答は，0 から 1 までの値です。応答が 1 の場合，**rediscoverCIs** 呼び出しは完了しています。

第 IV 部

コンテンツ記述

第 10 章

コンテンツ開発と記述

本章では, 新しいディスカバリおよび依存関係マップ (DDM) コンテンツの開発の手法，方法論，および実践について説明します。

本章の内容

概念

タスク

➤ ディスカバリ・アナライザを使った作業（345 ページ）

参照先

➤ ディスカバリおよび依存関係マップのコード（357 ページ）

➤ Jython のライブラリとユーティリティ（359 ページ）

➤ ジョブとパターンの XML 形式（363 ページ）

コンテンツ開発とパターン記述について

新しいコンテンツの開発を実際に計画し始める前に，この開発とに付き物のプロセスおよびインタラクションを理解することが重要です。

以下の項は，ディスカバリ開発プロジェクトの管理と実行を成功させるために必要な知識と手順を理解するのに役立ちます。

本章の概要は次のとおりです。

➤ HP Universal CMDB に関する実用的な知識と，DDM システムの要素に関する基本的な知識があることを前提としています。本章は，ユーザの学習を支援することが目的であり，完全なガイドではありません。

➤ DDM を使った HP Universal CMDB の新しいディスカバリ・コンテンツの計画，調査，および実装段階を対象としています。また，考慮する必要があるガイドラインと注意事項も示します。

➤ Discovery and Dependency Mapping フレームワークの主な API について説明します。使用可能な API の完全なドキュメントについては，『**HP Discovery and Dependency Mapping API Reference（英語版）**』を参照してください（ほかに非公式の API も存在しますが，用意済みのパターンに使用されている場合でも，変更されることがあります）。

ビジネス価値とディスカバリ開発の関連付け

新しいディスカバリ・コンテンツ開発のユース・ケースでは，ビジネス・ケースおよびビジネス計画によって，ビジネス価値を生み出す必要があります。つまり，システム・コンポーネントを CI にマッピングし，それらを CMDB に追加することの目的は，ビジネス価値を提供することです。

開発されたコンテンツがアプリケーション・マッピングに使用されるとは限りませんが，これは多くのユース・ケースで一般的な中間段階です。コンテンツの最終的な用途に関係なく，計画ではその手法に関する以下の疑問に答える必要があります。

- 誰がコンシューマか。コンシューマは CI（および CI 間の関係）によって提供された情報に対してどのように行動すべきか。CI と関係をどのようなビジネス・コンテキストに表示すべきか。これらの CI のコンシューマは，人間か，製品か，またはその両方か。
- CI と関係の完全な組み合わせが CMDB に存在する場合，それらを使ったビジネス価値の生成をどのように計画するか。
- 完全なマッピングはどのようなものか。
  - 各 CI 間の関係を最もわかりやすく言葉で説明するとどうなるか。
  - 含める必要がある最も重要な CI のタイプは何か。
  - マップの最終的な用途とエンド・ユーザは何か。
- 完全なレポート・レイアウトはどのようなものか。

ビジネス上の判断を確立したら，次の手順として，ビジネス価値をドキュメントで具体化します。つまり，描画ツールを使って完全なマップを作成し，ユース・ケースでの必要性に応じて，CI 間の影響と依存関係，レポート，変化の追跡方法，重要な変化，監視，コンプライアンス，およびその他のビジネス価値を理解します。

この図（またはモデル）を「**青写真**」と呼びます。

たとえば，特定の構成ファイルが変更されたことを検出することがアプリケーションにとって重要な場合は，そのファイルをマップして，作成されたマップ内で適切な（そのファイルに関連する）CI にリンクする必要があります。

開発されたコンテンツのエンド・ユーザである当該分野の SME（各分野のエキスパート）とともに作業します。このエキスパートは，ビジネス価値を提供するために CMDB に存在する必要がある重要なエンティティ（属性と関係を持つ CI）を指摘する必要があります。

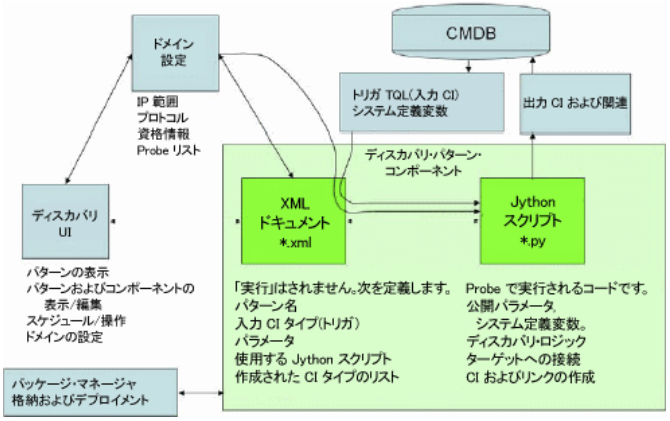
1 つの方法として，アプリケーションの所有者（この場合は SME も含む）にアンケートを行うことが考えられます。所有者は，前述の目的や青写真を明確に示すことができるはずです。所有者は，少なくともアプリケーションの現在のアーキテクチャを示す必要があります。

重要なデータのみをマップし，不要なデータをマップしないようにする必要があります。DDM は，後でいつでも拡張できます。目的は，適切に動作し，価値を提供する限定的なディスカバリを設定することであるべきです。大量のデータをマップすると，印象の強いマップができますが，混乱を招きやすく，開発に多くの時間がかかる可能性があります。

モデルとビジネス価値が明確になったら，次の段階に進みます。この段階は，その後の段階でより具体的な情報が得られるたびに実行し直すことができます。

DDM パターンと関連コンポーネント

次の図は，パターンのコンポーネントと，各コンポーネントがディスカバリの実行時にやり取りするコンポーネントを示します。緑色のコンポーネントは実際のパターンを示し，青色のコンポーネントはパターンとやり取りするコンポーネントを示します。



パターンの最小概念は，XML ドキュメントと Jython スクリプトという 2 つのファイルです。実行時には，DDM フレームワーク（入力 CI，資格情報，およびユーザ定義ライブラリを含む）がパターンに公開されます。これら 2 つのディスカバリ・パターン・コンポーネントは，DDM アプリケーションによって管理されます。また，これらは操作によって CMDB 自体に格納されます。外部パッケージは残りますが，操作では参照されません。新しいディスカバリ・コンテンツの機能は，パッケージ・マネージャを使って保存できます。

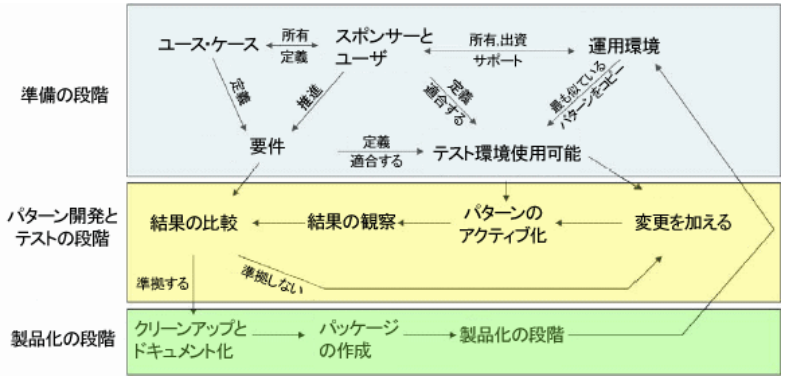
パターンへの入力 CI は TQL によって提供され，システム定義変数でパターン・スクリプトに公開されます。パターン・パラメータも宛先データとして指定されるため，パターンの特定の関数に従ってパターンの操作を設定できます。

新しいパターンは，DDM アプリケーションを使って作成およびテストします。コンテンツの記述中は，ジョブ，リソース，およびドメインの設定ページを使用します。

パターンは，パッケージとして格納および転送されます。パッケージ・マネージャ・アプリケーションと JMX コンソールを使って，新規作成されたパターンからパッケージを作成し，新しいシステムにパターンをデプロイします。

DDM の開発サイクル

次の図は，コンテンツ記述のフローチャートを示しています。ほとんどの時間は，開発とテストの反復ループである中央の部分に費やされます。



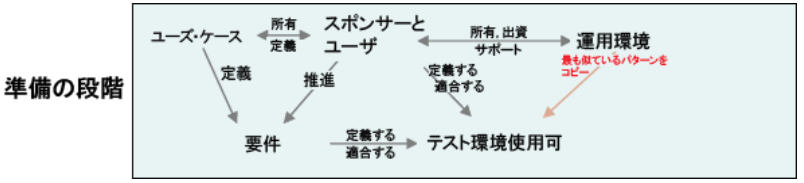
パターン開発の各段階は，直前の段階に基づいています。

パターンの記述内容と動作に満足したら，パターンをパッケージ化できます。UCMDB パッケージ・マネージャを使用するか，または手動でコンポーネントをエクスポートして，DDM パッケージの *.zip ファイルを作成します。ベスト・プラクティスとしては，実運用環境にリリースする前に別の UCMDB システム上でこのパッケージのデプロイとテストを行うことにより，すべてのコンポーネントが組み込まれ，正常にパッケージ化されたことを確認します。パッケージ化の詳細については，『**モデル管理**』の「カスタム・パッケージの作成」を参照してください。

次の各項では，最も重要な手順とベスト・プラクティスを示しながら，各段階についてさらに詳しく説明します。

- 調査と準備の段階
- パターンの開発とテスト
- パターンのパッケージ化と製品化

調査と準備の段階



調査と準備の段階には原動力となるビジネス・ニーズとユース・ケースが含まれます。また，パターンの開発とテストに必要な施設の確保もこの段階で行います。

1 既存のパターンを変更する場合は，最初の技術的な措置として，そのパターンのバックアップを作成し，元の状態に戻れるようにしておきます。新しいパターンを作成する場合は，最も似ているパターンをコピーし，それに適切な名前を付けて保存します。詳細については，249 ページ「＜パターン・ファイル＞」を参照してください。

2 パターンによってデータを収集する方法を調べます。

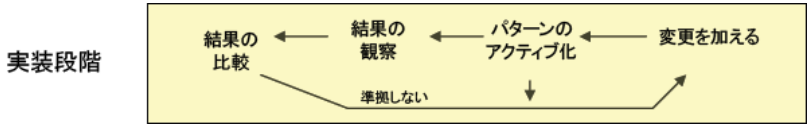
- 外部のツールやプロトコルを使ってデータを収集する
- パターンがデータに基づいて CI を作成する方法を開発する
- 同じようなパターンがどのように記述されているかをすでに知っている

3 次の要因に基づいて，最も似ているパターンを決定します。

- ➤ 同じ CI が作成されている
- ➤ 同じプロトコル (SNMP) が使用されている
- ➤ ターゲットの種類が同じである（OS のタイプやバージョンなど）

4 パッケージ全体をコピーします。

5 作業領域に展開し，パターン (XML) ファイルと Jython(.py) ファイルの名前を変更します。



パターンの開発とテスト

パターンの開発とテストの段階は，頻繁に反復されるプロセスです。パターンが具体化し始めたら，最終的なユース・ケースに対するテストを開始し，変更を行い，再度テストします。このプロセスを，パターンが要件に適合するまで繰り返します。

開始とコピーの準備

- ➤ パターンの XML 部分（1 行目の名前（ID），作成された CI タイプ，および呼び出す Jython スクリプト名）を変更します。
- ➤ 実行すると元のパターンと同じ結果となるコピーを入手します。
- ➤ コードの大部分（特に，重要な結果を生成するコード）をコメント・アウトします。

開発とテスト

- ➤ ほかのサンプル・コードを使って変更部分を開発します。
- ➤ パターンを実行してテストします。
- ➤ 専用のビューを使って複雑な結果を検証し，検索を使って簡単な結果を検証します。

パターンのパッケージ化と製品化

パターンのパッケージ化と製品化の段階は，開発の最終段階になります。ベスト・プラクティスとしては，パッケージ化に進む前に，デバッグの残存部分，ドキュメント，およびコメントを削除したり，セキュリティ上の考慮事項を確認したりするための最終パスを実施します。少なくとも Readme ドキュメントを必ず作成して，パターンの内部構造を説明する必要があります。かなり限定された内容でも，今後誰かが（おそらくユーザ自身も）このパターンを見る必要があるときに，大いに役立つはずです。

クリーンアップとドキュメント化

- デバッグ処理を削除します。
- すべての関数にコメントを付与し，メイン・セクションに開始コメントを追加します。
- ユーザがテストするためのサンプル TQL とビューを作成します。

パッケージの作成

- パッケージ・マネージャを使って，パターンや TQL などをエクスポートします。詳細については，『**モデル管理**』の「パッケージ・マネージャ」を参照してください。
- ほかのパッケージに対する依存関係（ほかのパッケージで作成された CI がパターンへの入力 CI になっているかどうかなど）を確認します。
- パッケージ・マネージャを使ってパッケージの zip ファイルを作成します。詳細については，『**モデル管理**』の「パッケージ・マネージャ」を参照してください。
- 新しいコンテンツの一部を削除して再度デプロイするか，またはほかのテスト・システムにデプロイすることによって，デプロイメントをテストします。

DDM と統合

DDM のディスカバリ・パターンは，ほかの製品と統合できます。以下の定義を考慮してください。

- DDM では，多くのターゲットから特定のコンテンツが収集されます。
- 統合では，1 つのシステムから複数のタイプのコンテンツが収集されます。

前述の定義では，収集の方法は区別されません。DDM でも同様です。新しいパターンを開発するプロセスは，新しい統合を開発するのと同じプロセスです。同じ調査を行い，新しいパターンと既存のパターンに関して同じ選択を行い，同じ方法でパターンを記述します。次のように，変更点はごくわずかです。

- 最終的なパターンのスケジュール。統合パターンは，ディスカバリよって頻繁に実行される可能性がありますが，その頻度はユース・ケースによって異なります。
- 入力 CI :
  - 統合 : CI 以外のトリガによって，入力なしで実行されます。パターン・パラメータでファイル名やソースが渡されます。
  - ディスカバリ : 通常の UCMDB CI が入力に使用されます。

統合プロジェクトでは，ほとんどの場合，既存のパターンを再利用する必要があります。統合の方向（HP Universal CMDB からほかの製品へ，またはほかの製品から HP Universal CMDB へ）が開発の方法に影響することがあります。ユーザが実績のある方法を使って特定の用途向けにコピーできるフィールド・パッケージがあります。

HP Universal CMDB からほかのプロジェクトへの統合は，次のようにして行います。

- エクスポートする CI と関係を生成する TQL を作成します。
- 汎用のラッパー・パターンを使って，TQL を実行してその結果を XML ファイルに書き込み，それを外部製品で読み取ります。

注 : フィールド・パッケージの例については，HP ソフトウェア・サポートにお問い合わせください。

ほかの製品を HP Universal CMDB に統合する場合は，ほかの製品のデータがどのように公開されるかによって，統合パターンの機能が異なります。

統合のタイプ	再利用される参考例
製品のデータベースに直接アクセスする	HP ED
エクスポートによって生成された csv または xml ファイルを読み取る	HP ServiceCenter
製品の API にアクセスする	BMC Atrium/Remedy

調査段階

この段階の前提条件は，DDM で検出する必要がある CI および関係の**青写真**があることです。この青写真には，検出する属性を含める必要があります。詳細については，302 ページ「コンテンツ開発とパターン記述について」を参照してください。

本項の内容

既存のパターンの変更

用意済みの DDM パターンやフィールド DDM パターンが存在する場合は，既存のパターンを変更します。ただし，以下の点に注意してください。

- 既存のパターンでは，必要とされる特定の属性は検出されません。
- 特定のタイプのターゲット（OS）が検出されないか，不正に検出されます。

- 特定の関係が検出されないか，作成されません。

  既存のパターンがジョブの（全部でなく）一部を行う場合は，最初の作業として，複数の既存のパターンを評価し，いずれかのパターンが必要なほとんどの処理を行うかどうかを確認します。そのような既存のパターンがある場合は，そのパターンを変更できます。

  既存のフィールド・パターンが利用可能かどうかも評価する必要があります。フィールド・パターンは，利用可能であるが用意済みではないディスカバリ・パターンです。フィールド・パターンの最新リストを受け取るには，HP ソフトウェア・サポートに連絡してください。

新しいパターンの記述

以下のような場合は，新しいパターンを開発する必要があります。

- CMDB に手動で情報を挿入するよりパターンを記述した方が速い場合（一般に，CI と関係の数が 50～100 に及ぶ場合）や，記述したパターンを再利用する場合。
- 必要性の高さがその労力に見合う場合。
- 用意済みのパターンやフィールド・パターンが利用できない場合。
- 結果を再利用できる場合。
- ターゲット環境やそのデータが利用可能な状態である場合（ユーザが確認できないものは検出できません）。

モデルの調査

- CMDB クラス・モデル（CI タイプ・マネージャ）を参照し，**青写真**から既存の CIT に対してエンティティと関係を照合します。バージョン・アップグレード時の混乱を避けるため，現在のモデルに従うことを強くお勧めします。モデルを拡張する必要がある場合は，用意済みの CIT がアップグレードによって上書きされる可能性があるため，新しい CIT を作成する必要があります。
- 一部のエンティティ，関係，または属性が現在のモデルに見つからない場合は，それらを作成する必要があります。これらの CIT を HP Universal CMDB の各インストールにデプロイできることが必要なため，これらの CIT を含むパッケージを作成することをお勧めします（このパッケージには，後で，すべてのディスカバリ，ビュー，およびこのパッケージに関連するその他の作成物も保持されます）。

テクノロジの調査

CMDB に必要な CI が保持されていることを確認したら，次の段階として，このデータを関連するシステムから取得する方法を決定します。

データを取得するには，通常，何らかのプロトコルを使って，アプリケーションの管理部分，アプリケーションの実際のデータ，またはアプリケーションに関連する構成ファイルやデータベースにアクセスする必要があります。システムに関する情報を提供できるすべてのデータ・ソースでも役に立ちます。テクノロジの調査には，問題のシステムに関する深い知識と，場合によっては創造性も必要です。

自社開発アプリケーションの場合は，アプリケーションの所有者にアンケート用紙を提供すると，参考になることがあります。所有者は，この用紙に，青写真とビジネス価値に関して必要な情報を提供できるアプリケーション内のすべての領域を記入する必要があります。これらの情報には，管理データベース，構成ファイル，ログ・ファイル，管理インタフェース，管理プログラム，Web サービス，および送信されたメッセージやイベントなどが含まれます（ただし，これらに限定されません）。

市販製品の場合は，製品のドキュメント，フォーラム，またはサポートに焦点を合わせる必要があります。管理ガイド，プラグインおよび統合ガイド，運用ガイドなどを調べます。データがまだ管理インタフェースにない場合は，アプリケーションの構成ファイル，レジストリ・エントリ，ログ・ファイル，NT イベント・ログ，およびアプリケーションの正しい運用を制御する作成物について参照します。

データへのアクセス方法の選択に関するガイドライン

関連性 : 最も多くのデータを提供するソースまたはソースの組み合わせを選択します。ほとんどのデータが 1 つのソースから提供され，残りの情報が分散していてアクセスしにくい場合は，残りの情報の価値を取得する労力やリスクとの比較で評価します。価値やコストが投入する労力に見合わない場合は，青写真を縮小することも検討します。

再利用: 特定の接続プロトコルのサポートが HP Universal CMDB にすでに含まれている場合は，それを使用するのが良い方法です。使用する場合は，DDM フレームワークからその接続用の既製のクライアントと設定が提供されます。使用しない場合は，インフラストラクチャの開発に投資する必要が生じる可能性があります。現在サポートされている HP Universal CMDB の接続プロトコルは，［**ディスカバリ**］＞［**ディスカバリ プローブ設定**］＞［**ドメインとプローブ**］表示枠に表示されます。詳細については，199 ページ「［ドメインとプローブ］表示枠」を参照してください。

新しいプロトコルを追加するには，モデルに新しい CI を追加します。詳細については，HP ソフトウェア・サポートまでお問い合わせください。

注：Windows レジストリ・データにアクセスするには，WMI と NTCmd のいずれかを使用します。

セキュリティ: 情報へのアクセスには，通常，資格情報（ユーザ名とパスワード）が必要です。資格情報は CMDB に入力され，製品全体でセキュリティ保護されます。可能な場合，およびセキュリティの追加が設定済みのほかの原則と競合しない場合は，アクセス・ニーズに対応する最も機密性の低い資格情報またはプロトコルを選択します。たとえば，JMX（標準の管理インタフェース。限定的）と Telnet のどちらでも情報を取得できる場合は，JMX を使用することをお勧めします。これは，JMX では本質的に限定されたアクセスが提供され，基盤となるプラットフォームへのアクセスが提供されないためです。

使いやすさ：一部の管理インタフェースには，より高度な機能が含まれています。たとえば，解析のために情報ツリーをたどったり，正規表現を作成したりするより，クエリ（SQL や WMI）を発行する方が簡単な場合があります。

使用する開発者: 最終的に DDM を開発するユーザは，特定のテクノロジを好む傾向があります。2 つのテクノロジでほぼ同じ情報が提供され，ほかの要因のコストが同じである場合は，この点を考慮することもできます。

サマリ

この段階の成果物は，アクセス方法と各方法から抽出できる関連情報について記載したドキュメントです。このドキュメントには，各ソースから関連する各青写真データへのマッピングも含める必要があります。

前述の説明に従って，各アクセス方法を採点する必要があります。最終的には，どのソースを検出し，どの情報を各ソースから青写真モデルに抽出するかについての計画ができあがります（青写真モデルは，このときまでに対応する UCMDB モデルにマップされている必要があります）。

パターンの分割

技術的には，ディスカバリ全体を 1 つのパターンで定義することもできます。しかし，適切な設計では，複雑なシステムをより簡単な，管理しやすいコンポーネントに分割することが求められます。

DDM プロセスを分割するためのガイドラインとベスト・プラクティスを以下に示します。

- ディスカバリは，複数の段階に分けて行う必要があります。各段階は，システムのある領域や階層をマップするパターンによって表されます。パターンは，検出された直前の段階または階層を利用して，システムの検出を続行します。たとえば，パターン A は，アプリケーション・サーバ TQL の結果によってトリガされ，アプリケーション・サーバ層をマップします。このマッピングの中で，JDBC 接続コンポーネントがマップされます。パターン B は，JDBC 接続コンポーネントをトリガ TQL として登録し，パターン A の結果を使って（たとえば，JDBC URL 属性を介して）データベース層にアクセスし，データベース層をマップします。
- **2 段階接続の枠組み**：ほとんどのシステムでは，システムのデータにアクセスするために資格情報が必要です。つまり，これらのシステムに対してユーザ / パスワードの組み合わせを試行する必要があります。DDM 管理者は，安全な方法でシステムに資格情報を提供します。また，優先順位が設定された複数のログイン資格情報を提供できます。これは，**プロトコル辞書**と呼ばれます。システムに（何らかの理由で）アクセスできない場合は，それ以上ディスカバリを実行しても無意味です。接続に成功した場合は，その後のディスカバリ・アクセスのために，どの資格情報セットを使って成功したかを示す何らかの方法が必要です。

前述の 2 段階に合わせて，パターンも以下の 2 つに分割されます。

- **接続パターン**：これは，最初のトリガを受け入れ，そのトリガにリモート・エージェントが存在するかどうかを調べるパターンです。そのために，このエージェントのタイプと一致するプロトコル辞書内のすべてのエントリを試行します。成功すると，このパターンはその結果としてリモート・エージェント CI（SNMP や WMI など）を提供し，その後の接続のためにプロトコル辞書内の正しいエントリも示します。このエージェント CI は，コンテンツ・パターンのトリガの一部になります。
- **コンテンツ・パターン**：このパターンの前提条件は，直前のパターンで接続に成功することです（TQL によって指定される前提条件）。このタイプのパターンでは，リモート・エージェント CI から正確な資格情報を取得する方法があり，取得した資格情報を使って検出されたシステムにログインできるため，プロトコル辞書のすべてのエントリを調べる必要はありません。

- スケジュール設定に関するさまざまな考慮事項が，ディスカバリの意思決定に影響を与えることもあります。たとえば，システムのクエリが営業時間外にしか行われない場合は，パターンを別のシステムを検出する同じパターンと結合することが妥当であっても，スケジュールが異なるために 2 つのパターンを作成する必要があります。
- 異なる管理インタフェースやテクノロジを使って同じシステムを検出するディスカバリは，個別のパターンに分ける必要があります。これによって，各システムまたは組織に適したアクセス方法をアクティブにすることができます。たとえば，一部の組織では，WMI でマシンにアクセスできますが，SNMP エージェントがマシンにインストールされていません。

Jython 内での外部 Java JAR ファイルの使用

新しい Jython スクリプトを開発するときは，Java ユーティリティ・アーカイブ，接続アーカイブ（JDBC ドライバ JAR ファイルなど），または実行可能ファイル（たとえば，資格情報なしのディスカバリでは **nmap.exe** が使用されます）として，外部 Java ライブラリ （JAR ファイル）またはサードパーティの実行可能ファイルが必要になることがあります。

これらのリソースは，パッケージの**外部リソース**・フォルダ内にバンドルする必要があります。このフォルダに格納されたリソースは，HP Universal CMDB サーバに接続する Probe に自動的に送信されます。

また，ディスカバリが起動されると，JAR ファイル・リソースが Jython のクラスパスにロードされ，その中にあるすべてのクラスをインポートして使用できるようになります。

HP Discovery and Dependency Mapping API Reference

使用可能な API の完全なドキュメントについては，『**HP Discovery and Dependency Mapping API Reference（英語版）**』を参照してください。ファイルは次のフォルダにあります。

C:¥hp¥UCMDB¥UCMDBServer¥j2f¥AppServer¥webapps¥site.war¥amdocs¥eng¥doc_lib¥Discovery_and_Dependency_Mapping¥DDM_JavaDoc¥index.html

ディスカバリ・アナライザを使用したコンテンツのデバッグ

ディスカバリ・アナライザ・ツールは，パッケージ，スクリプト，またはその他のコンテンツの開発時にデバッグ目的で利用するツールです。このツールは，リモート宛先に対してジョブを実行し，情報，警告，エラーの詳細や，CI の検出結果を含むログを返します。

本項の内容

- 317 ページ「タスクとレコード」
- 317 ページ「ログ」

タスクとレコード

タスク・ファイルには，実行対象のタスクに関するデータが含まれます。タスクは，ジョブの名前や，トリガ CI を定義する必須パラメータ （リモート宛先アドレスなど）などの情報で構成されています。

レコード・ファイルには，タスク情報と特定の実行の結果が含まれます。実行とは，プローブまたはディスカバリ・アナライザ（タスクを実行したどちらかのモジュール）（とリモート宛先間の詳細な通信（応答を含む）です。

タスク・ファイルで定義したタスクはリモート宛先に対して実行できるのに対して，レコード・ファイル（特定の実行に関する特別なデータが含まれるファイル）で定義したタスクは実行と再生（レコード・ファイルに記述されたものと同じ実行の再現）が可能です。

ログ

ログは，最新の実行に関する次のような情報を提供します。

- **一般ログ**：このログには，実行中に発生したすべての情報データ，エラー，および警告が含まれます。
- **通信ログ**：このログには，ディスカバリ・アナライザとリモート宛先間の詳細な通信（応答を含む）が含まれます。実行後，ログはレコード・ファイルとして保存できます。
- **結果ログ**：検出された CI のリストが表示されます。各 CI が表示される時間は，パターンとスクリプトの設計に応じて異なります。

  すべてのログをまとめて保存することも，各ログを別々に保存することもできます。すべてのログを保存すると，ログはまとめられて 1 つの名前で保存されます。

  レコード・ファイルを再生すると，実行時間のみが異なる，同じデータが通信ログに表示されます。

パターンの実装

DDM タスクの目的は，リモート（またはローカル）システムにアクセスし，抽出されたデータを CI としてモデル化し，それらの CI を CMDB に保存することです。このタスクは次の手順で構成されます。

1 DDM パターン

パターンに含めるスクリプトを選択することにより，DDM のコンテキスト，パラメータ，および結果タイプを保持するパターン・ファイルを設定します。詳細については，次の項を参照してください。

2 DDM ジョブ

スケジュール情報とトリガ TQL を含むジョブを設定します。詳細については，327 ページ「手順 2: パターンへのジョブの割り当て」を参照してください。

3 DDM コード

パターン・ファイルに含まれ，DDM「フレームワーク」を参照する Jython コードまたは Java コードを編集できます。詳細については，329 ページ「手順 3: Jython コードの作成」を参照してください。

新しい DDM コンテンツを記述するには，前述の各コンポーネントを作成します。作成した各コンポーネントは，前の手順のコンポーネントに自動的にバインドされます。たとえば，ジョブを作成して関連するパターンを選択すると，そのパターンはジョブにバインドされます。

手順 1: ディスカバリおよび依存関係マップのパターンの作成

DDM パターンは，関数の定義とみなすことができます。この関数は，入力定義を定義し，入力に対してロジックを実行し，出力を定義して，結果を提供します。

各 DDM パターンには入力と出力が指定されます。入力も出力も，そのパターンで明示的に定義されたトリガ CI です。DDM パターンは，入力トリガ CI からデータを抽出し，そのデータを DDM コードにパラメータとして渡します （関連 CI からのデータもコードに渡されることがあります。詳細については，175 ページ「［関連 CI］ウィンドウ」を参照してください）。パターンの DDM コードは，そのコードに渡される特定の入力トリガ CI のパラメータを除いて，汎用的なものです。

入力コンポーネントの詳細については，54 ページ「トリガ CIT，トリガ CI，入力 TQL，トリガ TQL」を参照してください。

本項の内容

パターン入力 （トリガ CIT と入力 TQL）の定義

特定の CI をパターン入力として定義するには，次のようにトリガ CIT と入力「トポロジ・クエリ言語」コンポーネントを使用します。

- トリガ CIT は，パターンの入力としてどの CIT を使用するかを定義します。たとえば，IP を検出するパターンでは，入力 CIT は Network です。
- 入力 TQL は，CMDB に対するクエリを定義する通常の編集可能な TQL です。入力 TQL は，CIT に対する追加の制約を定義します（たとえば，hostID 属性と application_ip 属性がタスクに必要な場合など）。また，パターンに必要な場合は，追加の CI データを定義できます。

  トリガ CI に関連する CI からの追加情報がパターンに必要な場合は，入力 TQL にノードを追加できます。詳細については，『**モデル管理**』の 320 ページ「入力 TQL 定義の例」と「ノードと関係を TQL クエリに追加」を参照してください。
- トリガ CI のデータには，トリガ CI に関する必要なすべての情報と，（定義された場合は）入力 TQL 内のほかのノードからの情報が含まれています。DDM では，CI からデータを取得するために変数が使用されます。Probe にタスクがダウンロードされると，トリガ CI のデータ変数は実際の CI インスタンスの属性に存在する実際の値に置き換えられます。

トリガ CIT 定義の例

この例では，パターン内で IP CI を許可することがトリガ CIT によって定義されています。

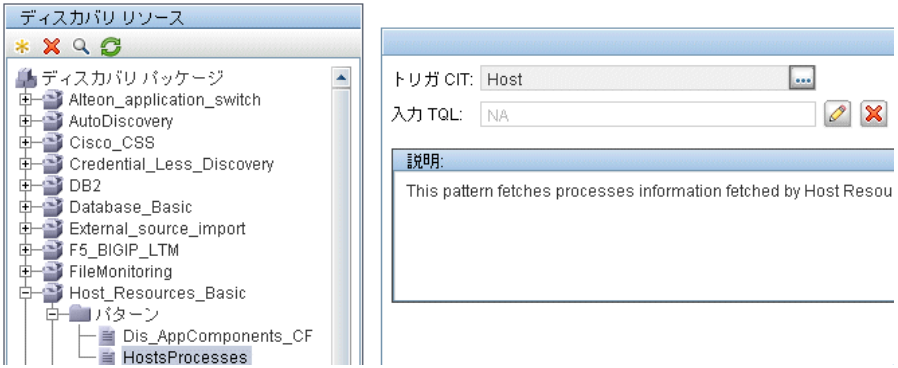
1 ［**ディスカバリ**］＞［**ディスカバリ リソースの管理**］＞［**パターン シグネチャ**］にアクセスします。HostProcesses パターンを選択します（［**ディスカバリ パッケージ**］＞［**Host_Resources_Basic**］＞［**パターン**］＞［**HostProcesses**］）。

2 ［トリガ CIT］ボックスを見つけます。詳細については，270 ページ「［トリガされた CI データ］表示枠」を参照してください。

3 ボタンをクリックすると，［検出されたクラスを選択］ダイアログ・ボックスが開きます。詳細については，241 ページ「［検出されたクラスを選択］ダイアログ・ボックス」を参照してください。

4 CIT を選択します。

この例では，パターン内で IP CI（Host）が許可されています。



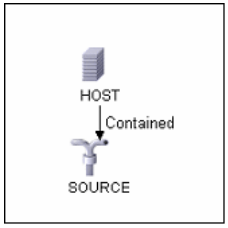
入力 TQL 定義の例

この例では，（前の例でトリガ CIT として設定された）IP CI を Host CI に接続する必要があることが入力 TQL によって定義されています。

1 ［**ディスカバリ**］＞［**ディスカバリ リソースの管理**］＞［**パターン シグネチャ**］にアクセスします。［入力 TQL］ボックスを見つけます。［**編集**］ボタンをクリックして［TQL エディタの入力］を開きます。詳細については，251 ページ「［TQL エディタの入力］ウィンドウ」を参照してください。

2 ［TQL エディタの入力］で，トリガ CI ノードに「**SOURCE**」という名前を付けます。そのためには，ノードを右クリックして，［**ノードのプロパティ**］を選択します。［**要素名**］ボックスで，名前を「**SOURCE**」に変更します。

3 IP CI に Host CI と Contains 関係を追加します。[TQL エディタの入力] を使った作業の詳細については，251 ページ「[TQL エディタの入力] ウィンドウ」を参照してください。



IP CI が **HOST** CI に接続されます。入力 TQL は，**HOST** と **IP** の 2 つのノードと，その間のリンクで構成されます。**IP** CI には「**SOURCE**」という名前が付けられています。

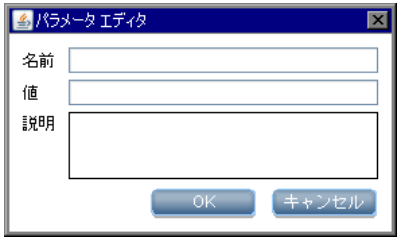
入力 TQL に変数を追加する例

この例では，前の例で作成した入力 TQL に DIRECTORY 変数と CONFIGURATION_FILE 変数を追加します。これらの変数を使って，検出する必要がある IP にリンクされたホスト上の構成ファイルを見つけるために何を検出する必要があるかを定義します。

1 前の例で作成した入力 TQL を表示します。

2 [**ディスカバリ**] > [**ディスカバリ リソースの管理**] > [**パターン シグネチャ**] にアクセスします。[トリガされた CI データ] 表示枠を見つけます。詳細については，270 ページ「[トリガされた CI データ] 表示枠」を参照してください。

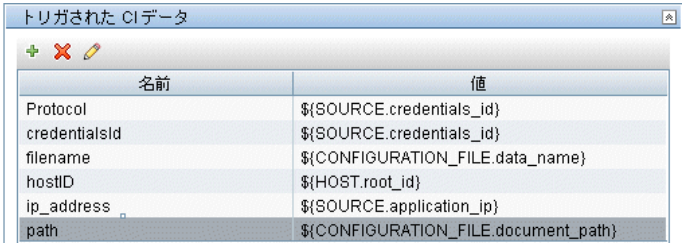
3 入力 TQL に変数を追加します。詳細については，270 ページ「[トリガされた CI データ] 表示枠」の値フィールドに関する説明を参照してください。



変数を実際のデータに置き換える例

この例では，IP CI のデータ変数を，システム内の実際の IP CI インスタンスに存在する実際の値に置き換えます。

IP CI のトリガされた CI データには，fileName 変数が含まれています。この変数を使って，入力 TQL の CONFIGURATION_FILE ノードを，ホスト上にある構成ファイルの実際の値に置き換えることができます。



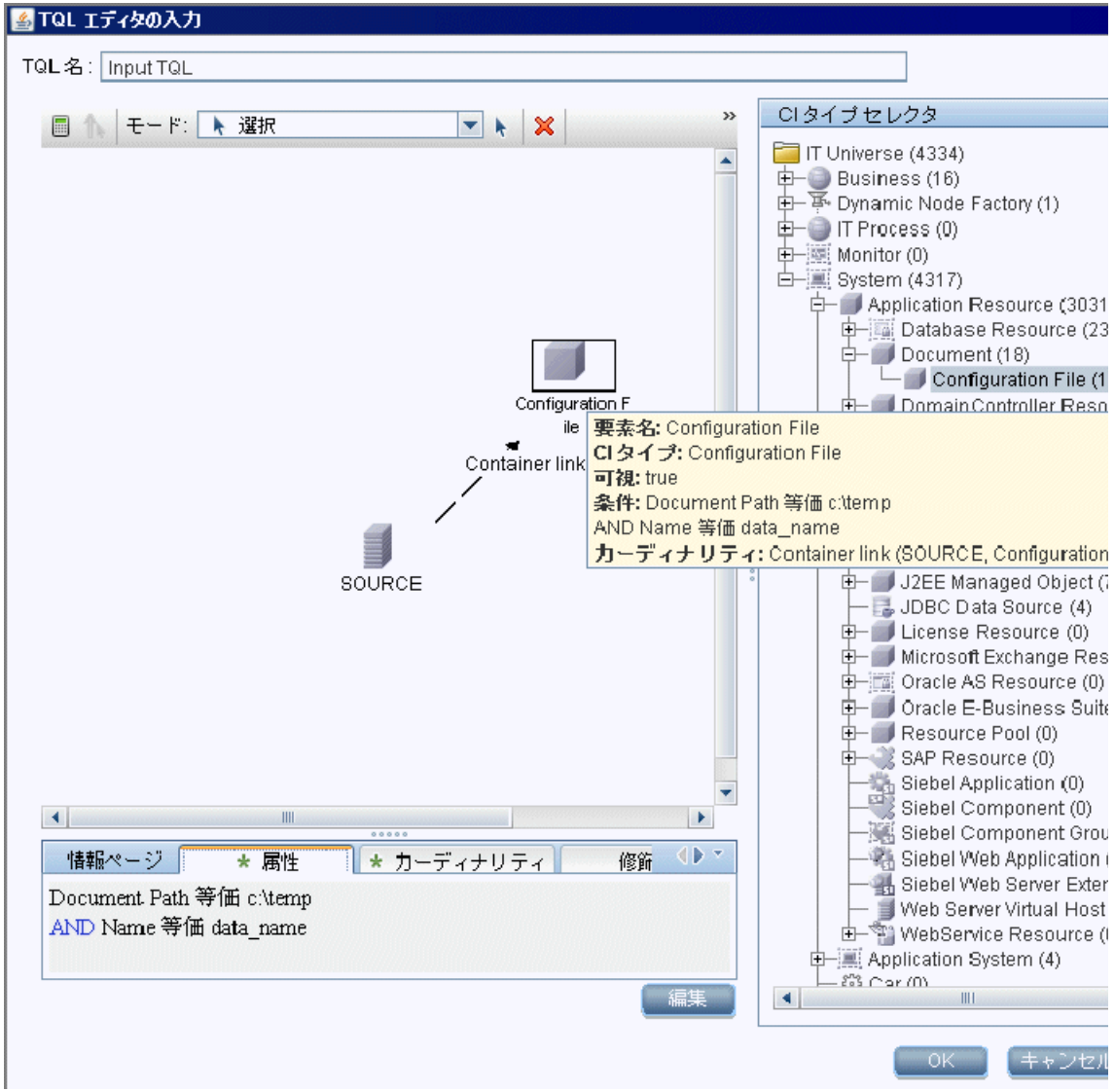
名前	値
Protocol	${SOURCE.credentials_id}
credentialsId	${SOURCE.credentials_id}
filename	${CONFIGURATION_FILE.data_name}
hostID	${HOST.root_id}
ip_address	${SOURCE.application_ip}
path	${CONFIGURATION_FILE.document_path}

トリガ CI データが Probe にアップロードされ，すべての変数が実際の値に置き換えられます。パターン・スクリプトには，DDM Framework を使って定義済み変数の実際の値を取得する次のコマンドが含まれています。

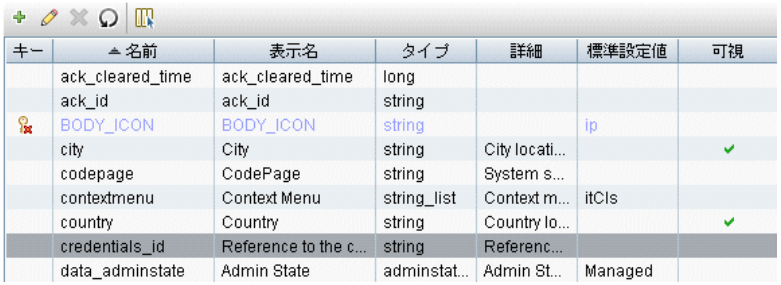
```
Framework.getTriggerCIData ('ip_address')
```

注：

➤ fileName および path 変数には，（前の例の入力 TQL で定義された）Configuration File ノードの data_name および document_path 属性が使用されます。



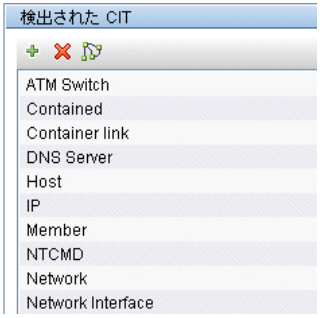
➤ Protocol，credentialsId，および ip_address 変数には，次のように root_class，credentials_id，および application_ip 属性が使用されます。



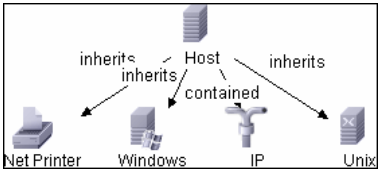
キー	名前	表示名	タイプ	詳細	標準設定値	可視
	ack_cleared_time	ack_cleared_time	long			
	ack_id	ack_id	string			
	BODY_ICON	BODY_ICON	string		ip	
	city	City	string	City locati...		✔
	codepage	CodePage	string	System s...		
	contextmenu	Context Menu	string_list	Context m...	itCIs	
	country	Country	string	Country lo...		✔
	credentials_id	Reference to the c...	string	Referenc...		
	data_adminstate	Admin State	adminstat...	Admin St...	Managed	

パターン出力の定義

パターンの出力は，検出された CI のリスト（［**ディスカバリ**］＞［**ディスカバリ リソースの管理**］＞［**パターン シグネチャ**］タブ）とその間のリンクです。



これらの CIT を，コンポーネントとそれらのリンク方法を表すトポロジ・マップで表示することもできます（［**検出した CIT をマップとして表示**］ボタンをクリックします）。



検出された CI は，DDM コード（Jython スクリプト）によって，UCMDB の ObjectStateHolderVector の形式で返されます。詳細については，335 ページ「Jython スクリプトによる結果の生成」を参照してください。

パターン出力の例

この例では，IP CI の出力にどの CIT を含めるかを定義します。

1 ［**ディスカバリ**］＞［**ディスカバリ リソースの管理**］にアクセスします。

2 ［ディスカバリ リソース］表示枠で，［**Network**］＞［**パターン**］＞［**NSLOOKUP_on_Probe**］を選択します。

3 ［パターン シグネチャ］タブで，［検出された CIT］表示枠を見つけます。

4 パターン出力に含める CIT が一覧表示されます。リストに CIT を追加するか，リストから CIT を削除します。詳細については，267 ページ「［検出された CIT］表示枠」を参照してください。

パターン・パラメータの上書き

複数のジョブに対してパターンを設定するために，パターン・パラメータを上書きできます。たとえば，パターン SQL_NET_Dis_Connection は MSSQL Connection by SQL ジョブと Oracle Connection by SQL ジョブの両方で使用されます。

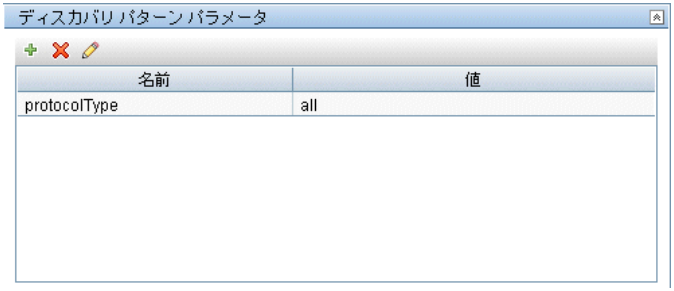
パターン・パラメータを上書きする例

この例では，1 つのパターンを使って Microsoft SQL Server と Oracle の両方のデータベースを検出できるようにパターン・パラメータを上書きする方法を示します。

1 ［**ディスカバリ**］＞［**ディスカバリ リソースの管理**］にアクセスします。

2 ［ディスカバリ リソース］表示枠で，［**Database Basic**］＞［**パターン**］＞［**SQL_NET_Dis_Connection**］を選択します。

3 ［パターン シグネチャ］タブで，［**ディスカバリ パターン パラメータ**］表示枠を見つけます。protocolType パラメータの値は **all** になっています。



4 **SQL_NET_Dis_Connection** パターンを右クリックし，［**ディスカバリ ジョブに移動**］＞［**MSSQL Connection by SQL**］を選択します。

5 ［プロパティ］タブを表示します。［パラメータ］表示枠を見つけます。

値 all を値 MicrosoftSQLServer で上書きします。

注: Oracle Connection by SQL ジョブには同じパラメータが含まれていますが，その値は値 oracle で上書きされます。

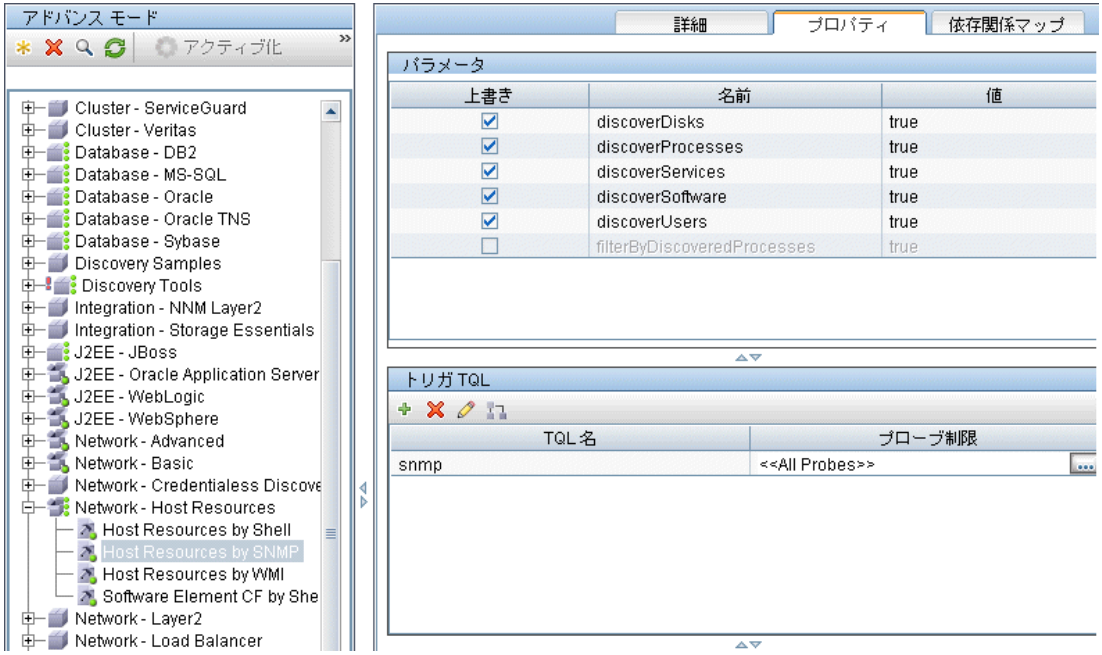
パラメータの追加，削除，編集の詳細については，267 ページ「［ディスカバリ パターン パラメータ］表示枠」を参照してください。

DDM は，このパラメータに従って Microsoft SQL Server インスタンスの検索を開始します。

手順 2: パターンへのジョブの割り当て

各パターンには，実行ポリシーを定義した 1 つ以上のジョブが関連付けられます。ジョブでは，トリガされた CI の異なるセットに対して異なる方法で同じパターンのスケジュールを設定できます。また，セットごとに異なるパラメータを設定できます。

ジョブは [ディスカバリ モジュール] ツリーに表示され，ユーザはこのエンティティをアクティブ化します。



トリガ TQL

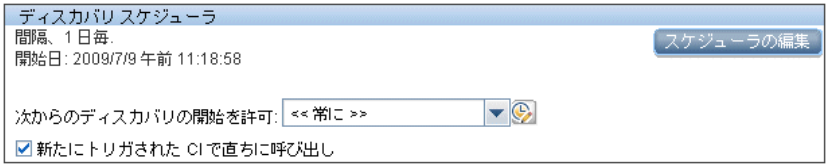
各ジョブは，トリガ TQL に関連付けられます。これらのトリガ TQL は，このジョブのパターンに対する入力トリガ CI として使用される結果を発行します。

トリガ TQL は，入力 TQL に制約を追加できます。たとえば，入力 TQL の結果が SNMP に接続された IP である場合は，トリガ TQL の結果を SNMP に接続された 195.0.0.0 ～ 195.0.0.10 の範囲内にある IP にすることができます。

注： トリガ TQL は，入力 TQL が参照する同じオブジェクトを参照する必要があります。たとえば，入力 TQL によって SNMP を実行している IP を照会する場合は，入力 TQL で必要とされる SNMP オブジェクトに接続しない IP もあるため，ホストに接続された IP を照会するトリガ TQL を（同じジョブに対して）定義できません。

スケジュール

Probe のスケジュール情報は，トリガ CI に対してコードをいつ実行するかを指定します。［**新たにトリガされた CI で直ちに呼び出し**］チェック・ボックスが選択されている場合は，以後のスケジュール設定に関係なく，トリガ CI が Probe に到達したときにも，各トリガ CI に対して 1 回ずつコードが実行されます。



各ジョブにスケジュール設定された時間ごとに，Probe はそのジョブで蓄積されたすべてのトリガ CI に対して DDM コードを実行します。詳細については，62 ページ「モジュールの実行スケジュール設定」と 146 ページ「[ディスカバリ スケジューラ］ダイアログ・ボックス」を参照してください。

パラメータ

ジョブを設定するときは，パターン・パラメータを上書きできます。詳細については，325 ページ「パターン・パラメータの上書き」を参照してください。

手順 3: Jython コードの作成

DDM コードの大部分は Jython で記述されます。「Jython は，高レベルの動的オブジェクト指向言語である Python を 100% ピュア Java で記述した実装であり，Java プラットフォームとシームレスに統合されています。したがって，Python はどの Java プラットフォームでも実行できます。」（Jython の Web サイト (http://www.jython.org/Project/index.html) から引用。）

次の項では, 実際に DDM フレームワーク内で Jython コードを記述する方法について説明します。本項では，特に，Jython スクリプトとそれを呼び出す DDM Framework との接点について取り上げ, できる限り使用する必要がある Jython のライブラリやユーティリティについても説明します。

注：

- DDM 用に記述されたスクリプトは, Jython バージョン 2.1 と互換性がある必要があります。
- 使用可能な API の完全なドキュメントについては,『**HP Discovery and Dependency Mapping API Reference（英語版）**』を参照してください

本項の内容

コードの実行

ジョブがアクティブ化されると，必要なすべての情報を含むタスクが Probe にダウンロードされます。

Probe は，タスクに指定された情報を使って DDM コードの実行を開始します。

Jython コードのフローは，スクリプトのメイン・エントリから実行を開始し，CI を検出するコードを実行し，その結果として検出された CI のベクトルを提供します。

用意済みスクリプトの変更

用意済みスクリプトを変更するときは，スクリプトの変更を最小限にとどめて，必要なメソッドを外部スクリプトに配置します。変更をより効率的に追跡できるようになり，新しいバージョンの HP Universal CMDB に移行するときにコードが上書きされません。

たとえば，次に示す用意済みスクリプト内の 1 行のコードは，アプリケーション固有の方法で Web サーバ名を計算するメソッドを呼び出します。

```
serverName = iplanet_cspecific.PlugInProcessing(serverName, transportHN,
mam_utils)
```

この名前の計算方法を決定するより複雑なロジックは，次の外部スクリプトに含まれています。

```
# implement customer specific processing for 'servername' attribute of httpplugin

#

def PlugInProcessing(servername, transportHN, mam_utils_handle):

        # support application-specific HTTP plug-in naming

        if servername == "appsrv_instance":

                # servername is supposed to match up with the j2ee server name,
however some groups do strange things with their

                # iPlanet plug-in files. this is the best work-around we could find. this join
can't be done with IP address:port

                # because multiple apps on a web server share the same IP:port for
multiple websphere applications

                logger.debug('httpcontext_webapplicationserver attribute has been
changed from [' + servername + '] to [' + transportHN[:5] + '] to facilitate websphere
enrichment')

                servername = transportHN[:5]

        return servername
```

この外部スクリプトを外部リソース・フォルダに保存します。詳細については，246 ページ「[ディスカバリ リソース] 表示枠」を参照してください。このスクリプトをパッケージに追加すると，ほかのジョブでもこのスクリプトを使用できるようになります。パッケージ・マネージャを使った作業の詳細については，**『モデル管理』**の「パッケージ・マネージャ」を参照してください。

アップグレードを行ったときは，この 1 行のコードに対して行った変更が用意済みスクリプトの新しいバージョンによって上書きされるため，行を置き換える必要があります。しかし，外部スクリプトは上書きされません。

Jython ファイルの構造

Jython ファイルは，一定の順序で並んだ以下の 3 つの部分で構成されます。

1 インポート

2 関数定義（任意）

3 メイン関数 - DiscoveryMain

以下に，Jython スクリプトの例を示します。

```
#   imports section
from appilog.common.system.types import ObjectStateHolder
from appilog.common.system.types.vectors import ObjectStateHolderVector

# Function definition
def foo:
 # do something

#   Main Function
def DiscoveryMain(Framework):
 OSHVResult = ObjectStateHolderVector()

 ## Write implementation to return new result CIs here...

 return OSHVResult
```

インポート

Jython のクラスは，階層構造の名前空間に存在します。バージョン 7.0 以降では，以前のバージョンと異なり，暗黙的なインポートがないため，使用するすべてのクラスを明示的にインポートする必要があります（この変更は，パフォーマンス上の理由と，必要な詳細を隠さないようにすることで Jython スクリプトをわかりやすくする目的で行われました）。

➤ Jython スクリプトをインポートするには，次のようにします。

```
import logger
```

➤ Java クラスをインポートするには，次のようにします。

```
from appilog.collectors.clients import ClientsConsts
```

メイン関数 ー DiscoveryMain

実行可能な各 Jython スクリプト・ファイルには，メイン関数である DiscoveryMain が含まれています。

DiscoveryMain 関数は，スクリプトのメイン・エントリ（最初に実行される関数）です。メイン関数は，スクリプト内で定義されたほかの関数を呼び出すことができます。

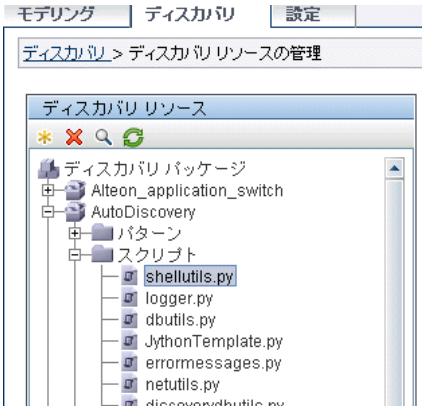
```
def DiscoveryMain(Framework):
```

メイン関数の定義では，Framework 引数を指定する必要があります。この引数は，メイン関数がスクリプトの実行に必要な情報（トリガ CI の情報やパラメータなど）を取得するために使用します。また，スクリプトの実行中に発生したエラーについて報告するためにも使用できます。

メイン・メソッドなしで Jython スクリプトを作成することもできます。このようなスクリプトは，ほかのスクリプトから呼び出されるライブラリ・スクリプトとして使用されます。

関数定義

各スクリプトには，メイン・コードから呼び出される追加の関数を含めることができます。このような関数から，現在のスクリプトまたは（import ステートメントを使って）別のスクリプトに存在する別の関数を呼び出すこともできます。ほかのスクリプトを使用するには，それをパッケージの［スクリプト］セクションに追加する必要があります。



関数から別の関数を呼び出す例

次の例では，メイン・コードから doQueryOSUsers(..) メソッドを呼び出し，そこからさらに内部メソッドの doOSUserOSH(..) を呼び出しています。

```
def doOSUserOSH(name):
 sw_obj = ObjectStateHolder('winosuser')

 sw_obj.setAttribute('data_name', name)
 # return the object
 return sw_obj

def doQueryOSUsers(client, OSHVResult):
 _hostObj = modeling.createHostOSH(client.getIpAddress())
 data_name_mib = '1.3.6.1.4.1.77.1.2.25.1.1,1.3.6.1.4.1.77.1.2.25.1.2,string'
 resultSet = client.executeQuery(data_name_mib)
 while resultSet.next():
       UserName = resultSet.getString(2)
       ########## send object ###############
       OSUserOSH = doOSUserOSH(UserName)
       OSUserOSH.setContainer(_hostObj)
       OSHVResult.add(OSUserOSH)

def DiscoveryMain(Framework):
 OSHVResult = ObjectStateHolderVector()
 try:
       client =
Framework.getClientFactory(ClientsConsts.SNMP_PROTOCOL_NAME).createClient()
 except:
       Framework.reportError('Connection failed')
 else:
       doQueryOSUsers(client, OSHVResult)
       client.close()
 return OSHVResult
```

このスクリプトが多くのパターンに関係するグローバルなライブラリである場合は，個々のパターンに追加する代わりに，このスクリプトを jythonGlobalLibs.xml 構成ファイル内のスクリプトのリストに追加できます（[**ディスカバリ**] > [**ディスカバリ リソースの管理**] > [**ディスカバリ パッケージ**] > [**AutoDiscovery**] > [**構成ファイル**]）。

Jython スクリプトによる結果の生成

各 Jython スクリプトは，特定のトリガ CI に対して実行され，DiscoveryMain 関数の戻り値によって返される結果とともに終了します。

スクリプトの結果は，実際には CMDB で挿入または更新される CI とリンクのグループです。スクリプトは，この CI とリンクのグループを ObjectStateHolderVector の形式で返します。

ObjectStateHolder クラスは，CMDB で定義されたオブジェクトまたはリンクを表す手段です。ObjectStateHolder オブジェクトには，CIT の名前と，属性とその値のリストが含まれています。ObjectStateHolderVector は，ObjectStateHolder インスタンスのベクトルです。

ObjectStateHolder の構文

本項では，DDM の結果を CMDB モデルに組み込む方法について説明します。

CI の属性設定の例

ObjectStateHolder クラスは，DDM の結果グラフを記述します。個々の CI とリンク（関係）は，次の Jython コード例のように，ObjectStateHolder クラスのインスタンスの内部に置かれます。

```
# siebel application server
1 appServerOSH = ObjectStateHolder('siebelappserver' )
2 appServerOSH.setStringAttribute('data_name', sblsvrName)
3 appServerOSH.setStringAttribute ('application_ip', ip)
4 appServerOSH.setContainer(appServerHostOSH)
```

- 第 1 行では，**siebelappserver** タイプの CI を作成します。
- 第 2 行では，サーバ名として検索された値が設定された Jython 変数である **sblsvrName** の値を持つ **data_name** という名前の属性を作成します。
- 第 3 行では，CMDB で更新される非キー属性を設定します。
- 第 4 行では，包含関係を構築します（結果はグラフになります）。このアプリケーション・サーバがホスト（範囲内の別の ObjectStateHolder クラス）に含まれることを指定します。

注：Jython スクリプトによって報告される各 CI には，その CI の CI タイプのすべてのキー属性の値を含める必要があります。

関係（リンク）の例

次のリンクの例を使って，グラフがどのように表されるかを説明します。

```
1 linkOSH = ObjectStateHolder('route')
2 linkOSH.setAttribute('link_end1', gatewayOSH)
3 linkOSH.setAttribute('link_end2', appServerOSH)
```

- 第 1 行では，リンクを作成します（このリンクは ObjectStateHolder クラスでもあります。唯一の違いは，route がリンク CI タイプであることです）。
- 第 2 行と第 3 行では，各リンクの端にあるノードを指定します。そのためには，リンクの **end1** および **end2** 属性を使用します。これらの属性は，（各リンクの最低限のキー属性であるため）必ず指定します。属性の値は，ObjectStateHolder インスタンスです。エンド 1 とエンド 2 の詳細については，242 ページ「リンク」を参照してください。

重要：リンクには方向があります。エンド 1 ノードとエンド 2 ノードが各端の有効な CIT に対応していることを確認する必要があります。ノードが有効でない場合は，結果オブジェクトが DDM 検証に失敗し，正しく報告されません。詳細については，『**モデル管理**』の「CI タイプ・マネージャ」を参照してください。

ベクトル（CI 収集）の例

属性とともにオブジェクトを作成し，端にあるオブジェクトとともにリンクを作成したら，それらをグループにまとめる必要があります。そのためには，次のように ObjectStateHolderVector インスタンスにそれらを追加します。

```
oshvMyResult = ObjectStateHolderVector()
oshvMyResult.add(appServerOSH)
oshvMyResult.add(linkOSH)
```

この複合された結果を「フレームワーク」に報告して CMDB サーバに送信できるようにする方法の詳細については，sendObjects メソッドを参照してください。

DDM の結果グラフを ObjectStateHolderVector インスタンスにまとめたら，それを DDM Framework に返して CMDB に挿入する必要があります。そのためには，DiscoveryMain() 関数の結果として ObjectStateHolderVector インスタンスを返します。

注: 一般的な CIT の **OSH** を作成する方法の詳細については, 359 ページ「Jython のライブラリとユーティリティ」の **modeling.py** を参照してください。

Framework インスタンス

「フレームワーク」インスタンスは，Jython スクリプトのメイン関数に渡される唯一の引数です。これは，スクリプトの実行に必要な情報（トリガ CI の情報やパターン・パラメータなど）を取得するために使用できます。また，スクリプトの実行中に発生したエラーについて報告するためにも使用できます。詳細については, 316 ページ「HP Discovery and Dependency Mapping API Reference」を参照してください。

本項では，Framework の最も重要な使用法について説明します。

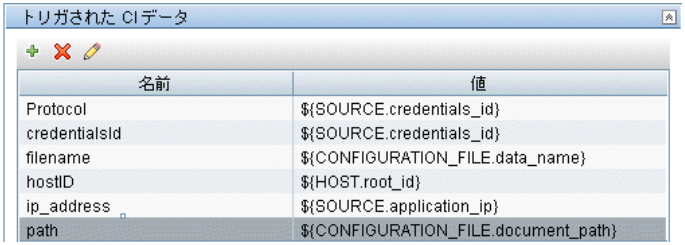
- 337 ページ「Framework.getTriggerCIData(String attributeName)」
- 338 ページ「Framework.createClient(credentialsId, props)」
- 340 ページ「Framework.getParameter (String parameterName)」
- 340 ページ「Framework.reportError(String message) および Framework.reportWarning(String message)」

Framework.getTriggerCIData(String attributeName)

この API は, パターンで定義されたトリガ CI データとスクリプトの間の中間段階を提供します。

資格情報を取得する例

次のトリガ CI データの情報を要求します。



タスクから資格情報を取得するには，次の API を使用します。

```
credId = Framework.getTriggerCIData('credentialsId')
```

Framework.createClient(credentialsId, props)

リモート・マシンと接続するには，クライアント・オブジェクトを作成し，そのクライアントに対してコマンドを実行します。クライアントを作成するには，ClientFactory クラスを取得します。getClientFactory() メソッドによって，要求されるクライアント・プロトコルのタイプを取得します。プロトコルの定数は，ClientsConsts クラスに定義されています。資格情報とサポートされているプロトコルの詳細については，203 ページ「ドメイン資格情報リファレンス」を参照してください。

資格情報 ID に対応する Client インスタンスを作成する例

資格情報 ID に対応する Client インスタンスを作成するには, 次のようにします。

```
properties = Properties()
codePage = Framework.getCodePage()
properties.put( BaseAgent.ENCODING, codePage)
client = Framework.createClient(credentailsID ,properties)
```

これで, Client インスタンスを使って該当するマシンまたはアプリケーションに接続できます。

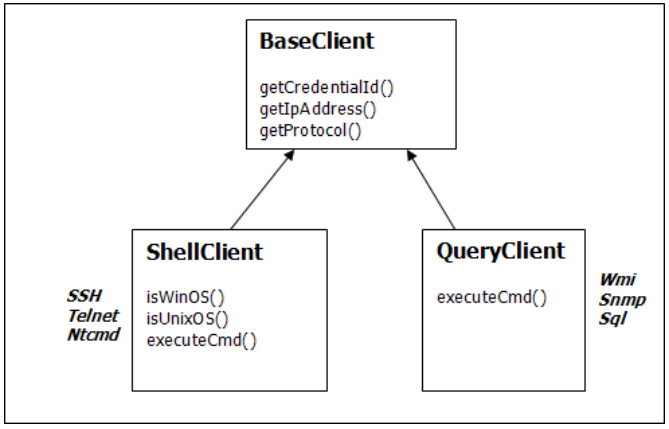
WMI クライアントを作成して WMI クエリを実行する例

WMI クライアントを作成し，そのクライアントを使って WMI クエリを実行するには，次のようにします。

```
wmiClient = Framework.createClient(credential)
resultSet = wmiClient. executeQuery("SELECT TotalPhysicalMemory
                                     FROMWin32_LogicalMemoryConfiguration")
```

注：createClient() API を機能させるには，［トリガされた CI データ］表示枠でトリガ CI データのパラメータに **credentialsId = ${SOURCE.credentials_id}** を追加します。または，関数 **wmiClient = clientFactory().createClient(credentials_id)** を呼び出したときに資格情報 ID を手動で追加できます。

次の図は，クライアントで一般的にサポートされる API とともにクライアントの階層を示します。



クライアントおよびクライアントでサポートされる API の詳細については，『**HP Discovery and Dependency Mapping API Reference**』（英語版）の「BaseClient」，「ShellClient」および「QueryClient」を参照してください。

Framework.getParameter (String parameterName)

トリガ CI に関する情報を取得するのに加えて，多くの場合，パターン・パラメータの値を取得する必要があります。次に例を示します。

protocolType パラメータの値を取得する例

Jython スクリプトから protocolType パラメータの値を取得するには，次の API を使用します。

```
protocolType = Framework.getParameterValue('protocolType')
```

Framework.reportError(String message) および Framework.reportWarning(String message)

スクリプトの実行中に，何らかのエラー（接続の障害，ハードウェアの問題，タイムアウトなど）が発生することがあります。このようなエラーが検出されたときは，Framework によってその問題を報告できます。報告されたメッセージはサーバに到達し，ユーザに対して表示されます。

レポート・エラーとメッセージの例

次の例は，reportError（<エラー・メッセージ>）API の使用法を示します。

```
try:
    client = Framework.getClientFactory(ClientsConsts.SNMP_PROTOCOL_NAME)
    createClient()
except:
    strException = str(sys.exc_info()[1]).strip()
    Framework. reportError ('Connection failed: %s' % strException)
```

問題を報告するには，Framework.reportError(String message) と Framework.reportWarning(String message) のいずれかの API を使用します。2 つの API の違いは，エラーを報告する場合はプローブがセッション全体のパラメータを含む通信ログ ファイルをファイル・システムに保存することです。これによって，セッションを追跡し，エラーをより深く理解することができます。

（接続パターン用の）正しい資格情報の検索

リモート・システムに接続するパターンでは，可能なすべての資格情報を試行する必要があります。（ClientFactory を使って）クライアントを作成するときに必要なパラメータの 1 つは，資格情報 ID です。接続スクリプトは，可能な資格情報セットにアクセスし，clientFactory.getAvailableProtocols() メソッドを使って資格情報を 1 つずつ試行します。ある資格情報セットが成功すると，パターンはそのトリガ CI のホスト上にある CI 接続オブジェクトを（その IP と一致する資格情報 ID とともに）CMDB に報告します。その後のパターンでは，この接続オブジェクト CI を使って資格情報セットに直接接続できます（つまり，各パターンで再び可能なすべての資格情報を試行する必要はありません）。

次の例は，SNMP プロトコルのすべてのエントリを取得する方法を示します。この例では，IP をトリガ CI データから取得しています（# Get the Trigger CI data values）。

接続スクリプトは，可能なすべてのプロトコル資格情報を要求し (# Go over all the protocol credentials)，いずれかが成功するまでループでそれらを試行します（resultVector）。詳細については，314 ページ「パターンの分割」の「**2 段階接続の枠組み**」を参照してください。

```
import logger
from appilog.collectors.clients import ClientsConsts
from appilog.common.system.types.vectors import ObjectStateHolderVector

      def mainFunction(Framework):
resultVector = ObjectStateHolderVector()

      # Get the Trigger CI data values
      ip_address = Framework.getDestinationAttribute('ip_address')
      ip_domain  = Framework.getDestinationAttribute('ip_domain')

      # Create the client factory for SNMP
      clientFactory = framework.getClientFactory(ClientsConsts.SNMP_PROTOCOL_NAME)
      protocols = clientFactory.getAvailableProtocols(ip_address, ip_domain)

      connected = 0
      # Go over all the protocol credentials
      for credentials_id in protocols:
            client = None
            try:
                  # try to connect to the snmp agent
                  client = clientFactory.createClient(credentials_id)

                  // Query the agent
                  ….

                  # connection succeed
                  connected = 1
            except:
                  if client != None:
                        client.close()
      if (not connected):
            logger.debug('Failed to connect using all credentials')
      else:
            // return the results as OSHV
            return resultVector
```

Java の例外の処理

一部の Java クラスは障害発生時に例外をスローします。この例外をキャッチして処理することをお勧めします。そうしないと，パターンが予期せずに終了する原因になります。

既知の例外をキャッチするときは，ほとんどの場合，例外のスタック・トレースをログに出力し，適切なメッセージを UI に発行する必要があります。次に例を示します。

```
try:
    client = Framework.getClientFactory().createClient()
except Exception, msg:
    Framework.reportError('Connection failed')
    logger.debugException('Exception while connecting: %s' % (msg))
    return
```

例外が致命的なものでなく，スクリプトを続行できる場合は，reportError() メソッドの呼び出しを省略して，スクリプトを続行できるようにする必要があります。

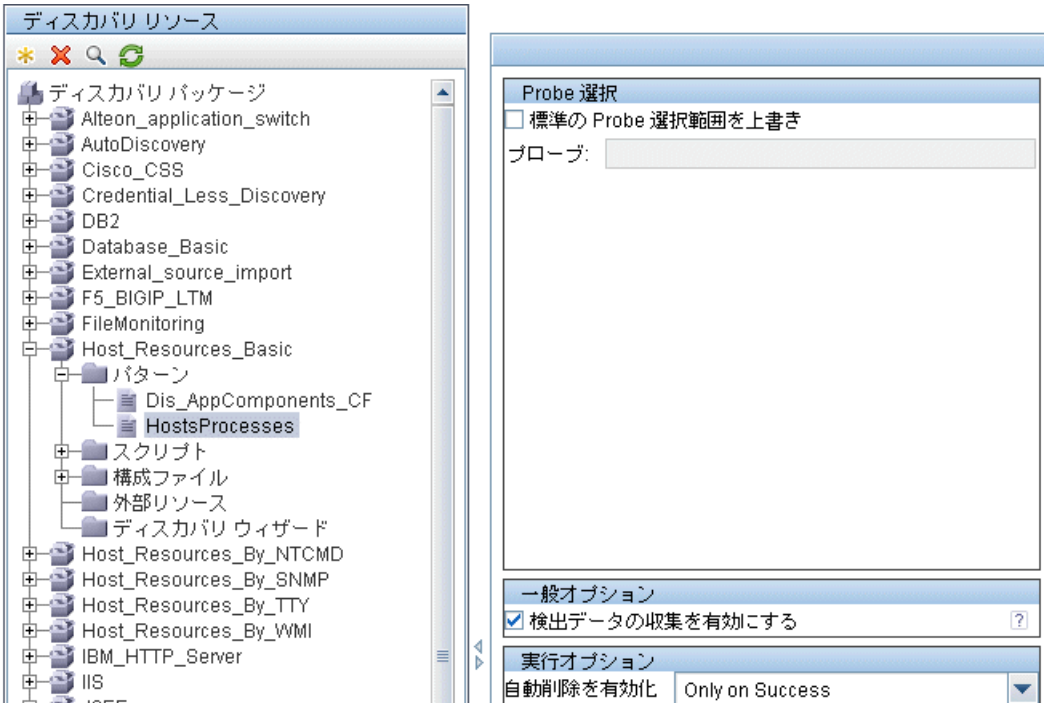
DDM コードの記録

デバッグやコードのテストなどを行うときは，すべてのパラメータを含む実行全体を記録すると非常に便利です。このタスクでは，関連するすべての変数を使って実行全体を記録する方法について説明します。また，通常はデバッグ・レベルでもログ・ファイルに出力されない追加のデバッグ情報を参照することもできます。

DDM コードを記録するには，次の手順を実行します。

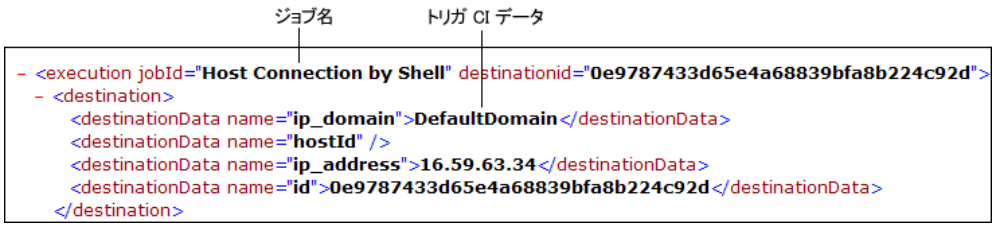
1 ［**ディスカバリ**］＞［**ディスカバリ実行**］にアクセスします。実行をログに記録する必要があるジョブを右クリックし，［**パターンの編集**］を選択して［ディスカバリ リソースの管理］アプリケーションを開きます。

2［パターン管理］タブの［**実行オプション**］表示枠を見つけます。

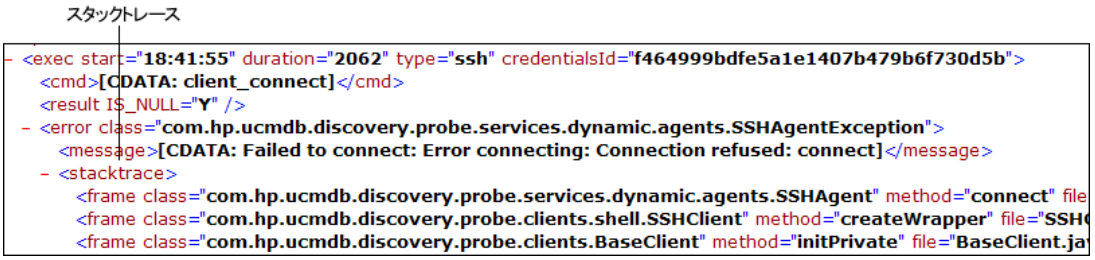


3［**通信ログの作成**］ボックスを［**常時**］に変更します。ログ記録オプションの詳細については，260 ページ「[実行オプション］表示枠」を参照してください。

次の例は，Host Connection by Shell ジョブを実行し，［**通信ログの作成**］ボックスを［**常時**］または［**失敗時**］に設定したときの XML ログ・ファイルです。



次の例は，メッセージとスタックトレース・パラメータを示します。



ディスカバリ・アナライザを使った作業

次に，ディスカバリ・アナライザでの作業方法について説明します。

本項の内容

1 前提条件

➤ プローブがインストールされている必要があります（ディスカバリ・アナライザは，プローブのインストール・プロセスの一部としてインストールされ，リソースをプローブと共有します)。

➤ ディスカバリ・アナライザでの作業中にプローブを実行する必要はありません。

ただし，プローブがすでに UCMDB サーバに対して実行されている場合，必要なリソースはすべて，すでにファイル・システムにダウンロードされています。プローブがまだ実行されていない場合は，ディスカバリ・アナライザで必要なリソースを［Settings］メニューからアップロードできます。詳細については，351 ページ「設定のインポート」を参照してください。

➤ UCMDB サーバがインストールされている必要はありません。

2 ディスカバリ・アナライザへのアクセス

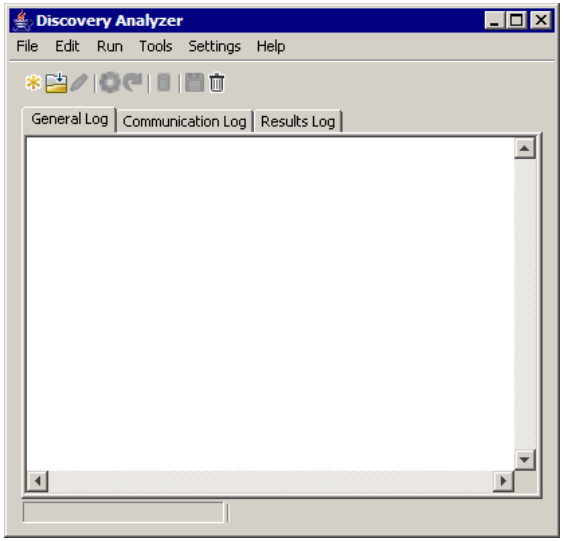
ディスカバリ・アナライザには次のいずれかの方法でアクセスできます。

➤ Eclipse で作業している場合 :**pydev** プラグインをお勧めします。PyDev および PyDev Extensions をインストールします。これらは，**http://pydev.sourceforge.net/**(PyDev) および **http://www.fabioz.com/pydev/**(PyDev Extensions) で入手できます。

プローブのインストールは，**C:¥hp¥DDM¥DiscoveryProbe¥discoveryAnalyzerWorkspace** にある標準設定の Eclipse ワーク・スペースに付属しています。このワーク・スペースには，ディスカバリ・アナライザを起動するための Jython スクリプト (**startDiscoveryAnalyzerScript.py**)，およびすべての DDM スクリプトへのリンクが含まれます。この方法でツールを起動すると，デバッグ目的で Jython スクリプト内にブレークポイントを配置できます。

➤ 直接アクセスする場合：次のフォルダにあるファイルをダブルクリックします。**C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥discoveryAnalyzer.cmd** 詳細については，次の項を参照してください。

［Discovery Analyzer］ウィンドウが開きます。



3 タスクの定義

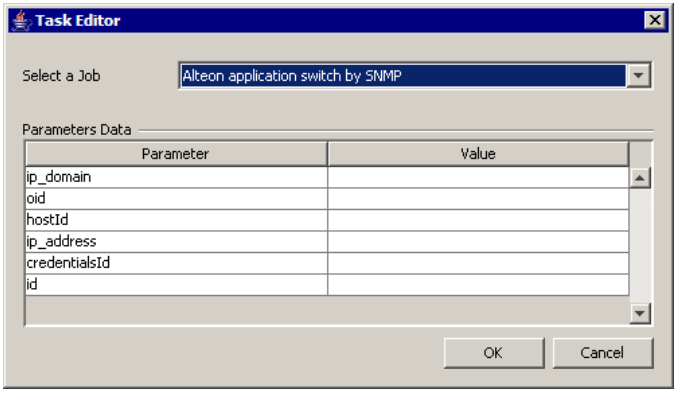
タスクは，次のいずれかの方法で定義します。

- 新しいタスクを定義する。詳細については，348 ページ「新しいタスクの定義」を参照してください。
- レコード・ファイルからタスクをインポートする。詳細については，349 ページ「レコードの取得」を参照してください。
- 保存したタスクをタスク・ファイルからインポートする。詳細については，349 ページ「タスク・ファイルを開く」を参照してください。
- プローブの内部データベースからジョブを取得する。詳細については，349 ページ「データベースからのジョブのインポート」を参照してください。

4 新しいタスクの定義

a ［**New Task**］ボタンをクリックして，タスク・エディタを表示します。

タスク エディタが開き，ファイル・システムに現在存在するジョブのリストが表示されます。このリストは，プローブがサーバからタスクを受信するたびに，または［Settings］メニューからパッケージを手動で展開するたびに更新されます。



b ジョブを選択します。

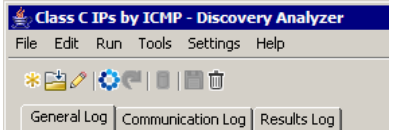
c すべてのパラメータの値を入力します。

ここに表示されるパラメータは，DDM パターン・パラメータです。このパラメータは，［パターン シグネチャ］タブの［ディスカバリ パターン パラメータ］表示枠で表示できます。詳細については，267 ページ「［ディスカバリ パターン パラメータ］表示枠」を参照してください。

フィールドは，すべて必須フィールドです（ジョブのスクリプトでフィールドを空白にする必要がある場合を除く）。

ID または資格情報 ID の入力値を必要とするパラメータの場合，ランダムに作成した ID を使用できます。Value ボックスを右クリックし，［**Generate random CMDB ID**］または［**Credential Chooser**］を選択します。

タスクがアクティブになり，開かれたタスクの名前がタイトル・バーに表示されます。



d タスクを定義する手順を続行します。詳細については，350 ページ「タスクとログの保存」を参照してください。

5 レコードの取得

特定の実行に関するデータが含まれるレコード・ファイルを開いて，タスクを定義できます。タスクをこの方法で定義すると，再生オプションを選択して特定の実行を再現できます（タスクを再生すると，応答は，リモート宛先ではなく，レコード・ファイルに保存されたデータから受信されます)。

［**File**］＞［**Open Record**］を選択します。レコードを保存したフォルダを参照します。レコードがアクティブになり，タスクの名前がタイトル・バーに表示されます。

6 タスク・ファイルを開く

タスク・ファイルからタスクを定義できます。［**File**］＞［**Open Task**］を選択します。

7 データベースからのジョブのインポート

プローブがすでに実行されており，プローブの内部データベースにアクティブなタスクがある場合，プローブ・データベースからジョブを取得できます。パラメータ値を使用して，タスクを定義できます。

a ［**File**］＞［**Import Task from Probe DB**］を選択します。

b 開いたダイアログ・ボックスで，実行するジョブを選択し，［**OK**］をクリックします。

c タスクを定義する手順を続行します。詳細については，350 ページ「タスクとログの保存」を参照してください。

8 タスクの編集

タスクを定義すると，タスク（またはファイル）の名前がタイトル・バーに表示されます。これで，ファイルを編集できます。

a ［**Edit**］＞［**Edit Task**］を選択します。

b タスクに変更を加え，［**OK**］をクリックします。

9 タスクとログの保存

タスク・パラメータを保存できます。［**File**］＞［**Save Task**］を選択します。

タスクの実行後に限り，次のオプションを選択できます。

- タスクのレコードを保存します。タスク・パラメータとタスク実行の結果を保存できます。［**File**］＞［**Save Record**］を選択します。
- タスクのログを保存できます。［**File**］＞［**Save General Log**］を選択します。
- 結果を保存します。［**File**］＞［**Save Result**］を選択します。

10 タスクの実行

手順の次の手順では，作成したタスクを実行します。

a リモート宛先に対してのみタスクを実行するには，［**Run Task**］ボタンをクリックします。

ディスカバリ・アナライザによってジョブが実行され，3 つのログ・ファイル（**一般**，**通信**，**結果**）に情報が表示されます。

b ログ・ファイルはまとめて，または別々に保存できます。［**File**］の後に，［**Save General Log**］，［**Save Record**］，［**Save Results**］，［**Save All logs**］のいずれかを選択します。ログ・ファイルの詳細については，317 ページ「ログ」を参照してください。

c タスクをレコード・ファイルから取得する場合，［**Playback**］ボタンをクリックすると，レコード・ファイルに記述された実行を再現できます。表示される通信ログは同じですが，実行時間は更新されます。

11 タスク結果のサーバへの送信

タスクの実行が終了し，結果が生成された場合（つまり，［結果ログ］タブに検出された CI のリストが表示された場合），結果を UCMDB サーバに送信できます。これは，サーバの停止中にスクリプトのテストを行い，後から結果を送信する場合などに便利です。

注：結果の送信先にできるのは，ディスカバリ・アナライザと同じマシンにインストールされているプローブからタスクを受信する UCMDB サーバのみです。

---

12 設定のインポート

プローブがまだ実行されていない場合，ディスカバリ・アナライザで必要なファイルをインポートできます。［Settings］メニューにアクセスし，**domainScopeDocument.xml** または **domainScopeDocument.bin** をインポートします。**.bin** ファイルをインポートする場合，**key.bin** もインポートする必要があります。［**Settings**］＞［**Import Packages**］を選択すると，必要な DDM パッケージ（スクリプト，パターンなど）を展開できます。

13 ブレークポイント

Python スクリプトからディスカバリ・アナライザを実行する場合，スクリプトにブレークポイントを追加できます。このタスクでは

Eclipse ワークスペースの設定

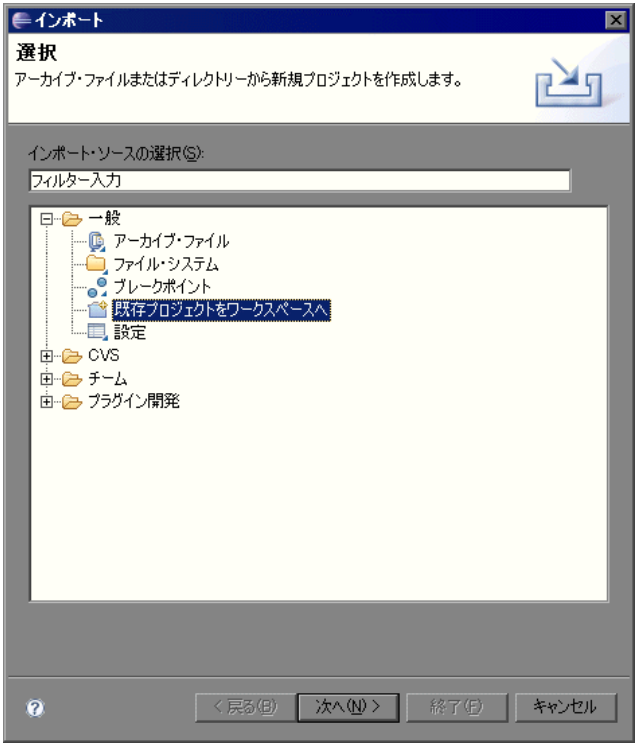
このタスクでは Eclipse ワークスペースを設定する方法について説明します。

Eclipse を設定するには，次の手順を実行します :

1 前提条件 :

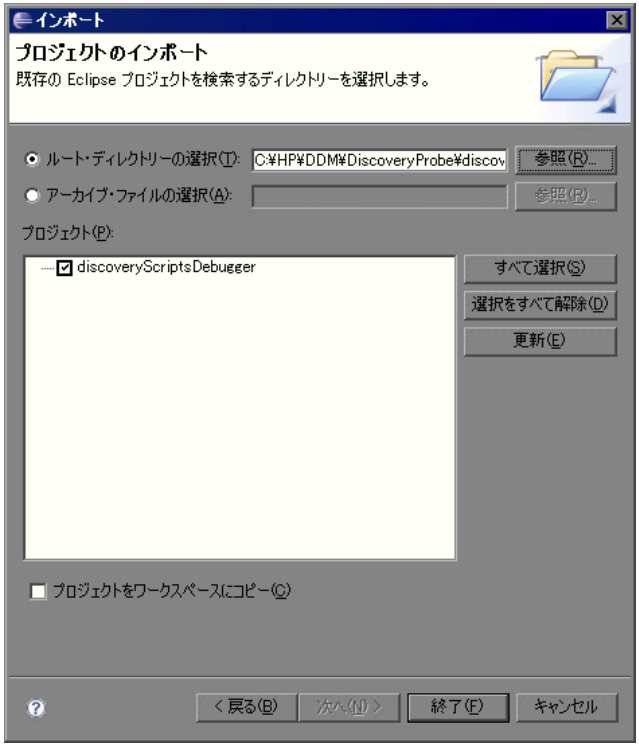
- お使いのコンピュータに Java 5.0 SE をインストールします(まだインストールされていない場合）。必要なファイルは，Java Developers Web サイト (http://www.java.sun.com) で入手できます
- お使いのコンピュータに Eclipse の最新バージョンをインストールします。このアプリケーションは，www.eclipse.org で入手できます。
- PyDev および PyDev Extensions をインストールします。これらは，http://pydev.sourceforge.net/(PyDev) および http://www.fabioz.com/pydev/(PyDev Extensions) で入手できます。
- このコンピュータに，ディスカバリ Probe がインストールされている必要があります。

2［**ファイル**］>［**インポート**］>［**一般**］>［**既存プロジェクトをワークスペースへ**］を選択します。



3［**次へ**］をクリックします。［**参照**］をクリックして次のディレクトリを選択します。**C:¥hp¥DDM¥DiscoveryProbe¥discoveryAnalyzerWorkspace**:

4 ［**プロジェクト**］表示枠で［**discoveryAnalyzerWorkspace**］が選択されていることを確認し，［終了］をクリックします。

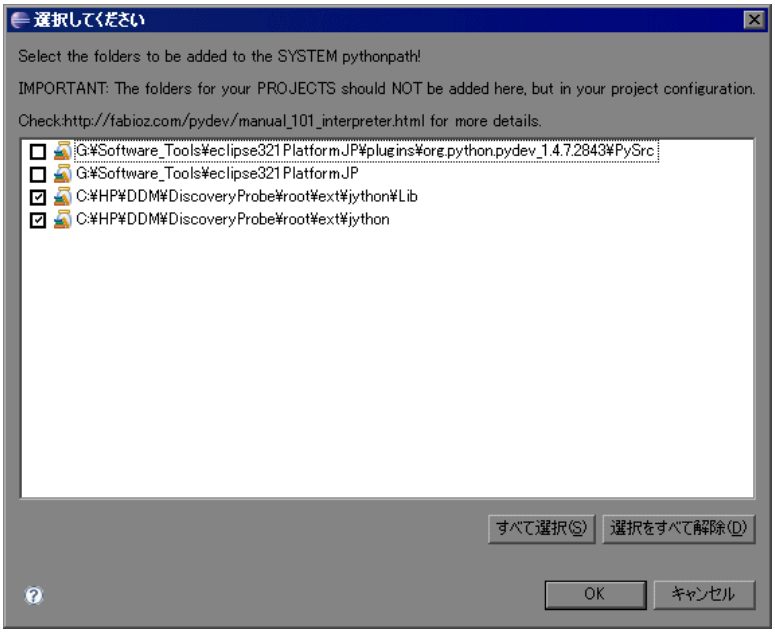


discoveryAnalyzerWorkspace プロジェクトが Eclipse のワークスペースに追加されます。

5 ［**ウィンドウ**］＞［**設定**］を選択して，［**設定**］ダイアログ・ボックスを開きます。［**Pydev**］＞［**Interpreter - Jython**］＞［**新規**］を選択します。

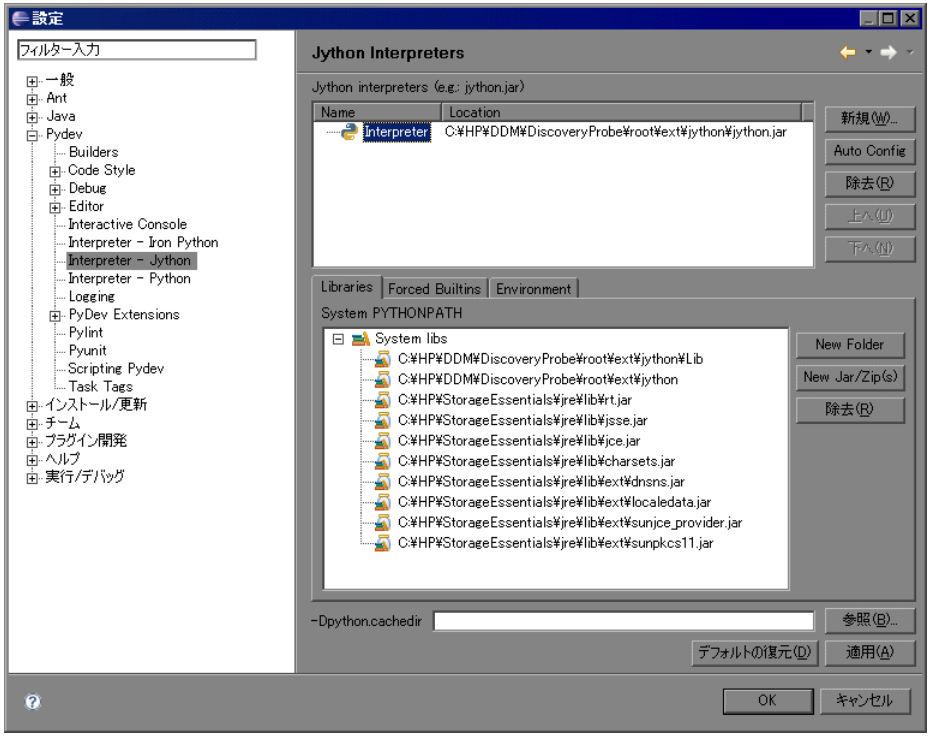
6 以下のディレクトリを選択します。

- **C:¥hp¥DDM¥DiscoveryProbe¥root¥ext¥jython¥**
- **C:¥hp¥DDM¥DiscoveryProbe¥root¥ext¥jython¥lib**



［**OK**］をクリックします。

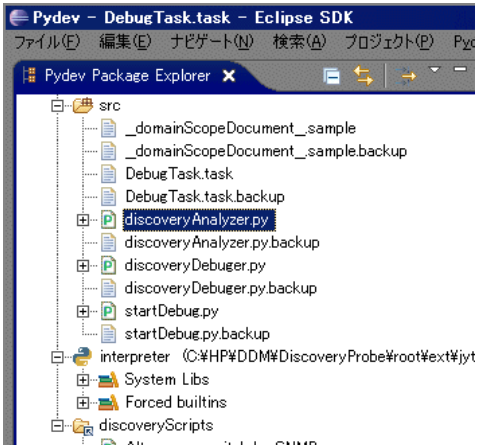
7 ［**設定**］ダイアログ・ボックスに戻ります。［**Jython interpreters**］リストに jython.jar ファイルが表示されていることを確認します。



［**OK**］をクリックします。

8 ［**ウィンドウ**］＞［**パースペクティブを開く**］＞［**その他**］を選択して，［**パースペクティブを開く**］ダイアログ・ボックスを開きます。［**PyDev**］を選択します。

9 Discovery Analyzer を使用するには, **discoveryAnalyzerWorkspacerc** プロジェクトの **startDiscoveryAnalyzerScript.py** ファイルを選択します。このファイルを右クリックして，［**実行**］＞［**Jython run**］を選択します。



トラブルシューティングおよび制限事項

Jython インタープリタを含む pydev プラグインは, 適切に設定する必要があります。DDM プローブのインストールの一部である Jython インタープリタを選択する方法をお勧めします（[**Eclipse] ウィンドウ> [プリファレンス]**）。pydev のクラス パスを定義する **.pydevproject** が含まれるワーク・スペースを使用する必要があります。**pydev** が正しくインストールされていれば，Jython インタープリタは自動的に選択されます。

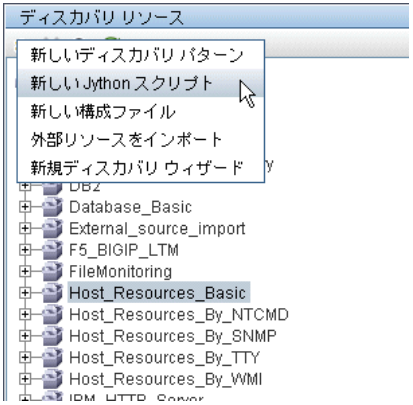
ディスカバリおよび依存関係マップのコード

リモート・システムに接続し，そのデータを照会し，それを CMDB データとしてマップする処理の実際の実装は，DDM コードによって実行されます。コードには，たとえば，データベースに接続してそこからデータを抽出するためのロジックが含まれています。この場合，コードは JDBC URL，ユーザ名，パスワード，ポートなどを受け取ることを期待しています。これらのパラメータは，TQL クエリに応答するデータベースの各インスタンスに固有のものです。これらの変数はパターン（トリガ CI データ）で定義され，ジョブを実行すると，これらの詳細がコードに渡されて実行に使用されます。

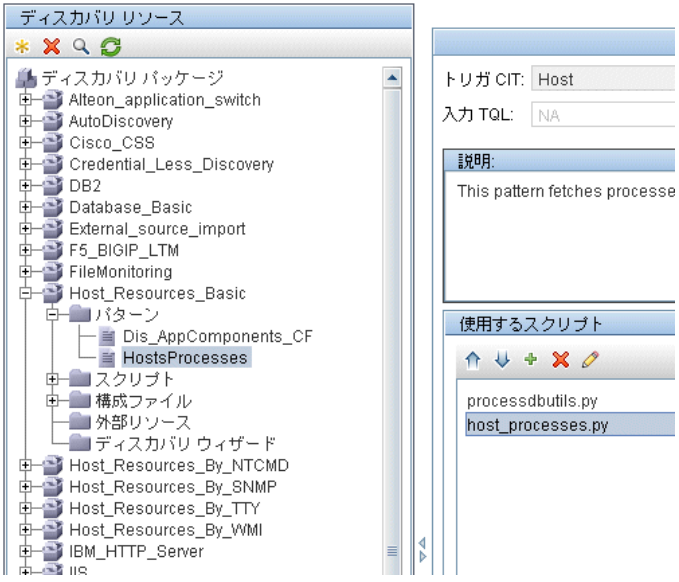
パターンは，Java クラス名または Jython スクリプト名によってこのコードを参照できます。本項では，Jython スクリプトとして DDM コードを記述する方法について説明します。

パターンには, DDM の実行時に使用されるスクリプトのリストが含まれています。新しいパターンを作成するときは，通常，新しいスクリプトを作成し，それをパターンに割り当てます。新しいスクリプトには基本的なテンプレートが含まれていますが，ほかのいずれかのスクリプトを右クリックして［**名前を付けて保存**］を選択すれば，それをテンプレートとして使用できます。

新しい Jython スクリプトを記述する方法の詳細については，329 ページ「手順 3: Jython コードの作成」を参照してください。スクリプトを追加するには，［ディスカバリ リソースの管理］ウィンドウを使用します。



スクリプトのリストは，パターンに定義された順序で 1 つずつ実行されます。



注： スクリプトは，別のスクリプトでライブラリとしてのみ使用される場合でも指定する必要があります。その場合は，ライブラリ・スクリプトを使用するスクリプトより先に，ライブラリ・スクリプトを定義します。 この例では，processdbutils.py スクリプトは最後の host_processes.py スクリプトが使用するライブラリです。ライブラリは，DiscoveryMain() 関数がないことによって通常の実行可能なスクリプトと区別されます。

Jython のライブラリとユーティリティ

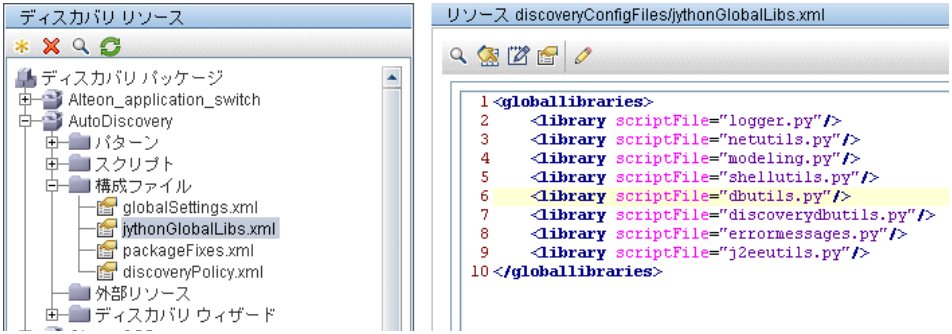
DDM のパターンでは，いくつかのユーティリティ・スクリプトが広く使用されます。これらのスクリプトは，Probe にダウンロードされたほかのスクリプトとともに，AutoDiscovery パッケージに含まれており，**C:¥hp¥DDM ¥DiscoveryProbe¥root¥lib¥collectors¥probeManager¥discoveryScripts** にあります。

注： discoveryScript フォルダは，Probe が動作を開始したときに動的に作成されます。

ユーティリティ・スクリプトを使用するには，スクリプトのインポート・セクションに以下の import 行を追加します。

```
import <スクリプト名>
```

AutoDiscovery Python ライブラリには，Jython ユーティリティ・スクリプトが含まれています。これらのライブラリ・スクリプトは，DDM の**ライブラリとみなされ，（構成ファイル・フォルダにある）**jythonGlobalLibs.xml ファイルで定義されています。



jythonGlobalLibs.xml ファイルに表示される各スクリプトは，標準設定では Probe の起動時にロードされるため，それらをパターン定義で明示的に使用する必要はありません。

本項の内容

logger.py

logger.py スクリプトには，エラー報告用のログ・ユーティリティとヘルパー関数が含まれています。そのデバッグ API，情報 API，およびエラー API を呼び出して，ログ・ファイルに書き込むことができます。ログ・メッセージは，**C:¥hp¥DDM¥DiscoveryProbe¥root¥logs¥probeMgr-patternsDebug.log** に記録されます。

メッセージは，**C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥probeManager¥probeMgrLog4j.properties** ファイルの PATTERNS_DEBUG アペンダに定義されたデバッグ・レベルに従ってログ・ファイルに記入されます。（標準設定のレベルは DEBUG です）。詳細については，63 ページ「重大度レベル」を参照してください。

```
#########################################################################
################       PATTERNS_DEBUG log              ###################
#########################################################################
log4j.category.PATTERNS_DEBUG=DEBUG, PATTERNS_DEBUG
log4j.appender.PATTERNS_DEBUG=org.apache.log4j.RollingFileAppender
log4j.appender.PATTERNS_DEBUG.File=C:/hp/DDM/DiscoveryProbe/root/logs/probe
Mgr-patternsDebug.log
log4j.appender.PATTERNS_DEBUG.Append=true
log4j.appender.PATTERNS_DEBUG.MaxFileSize=15MB
log4j.appender.PATTERNS_DEBUG.Threshold=DEBUG
log4j.appender.PATTERNS_DEBUG.MaxBackupIndex=10
log4j.appender.PATTERNS_DEBUG.layout=org.apache.log4j.PatternLayout
log4j.appender.PATTERNS_DEBUG.layout.ConversionPattern=<%d> [%-5p][%t] -
%m%n
log4j.appender.PATTERNS_DEBUG.encoding=UTF-8
```

情報メッセージとエラー・メッセージは，コマンド・プロンプト・コンソールに表示されます。

次の 2 つの API セットがあります。

- logger.<debug/info/warn/error>
- logger.<debugException/infoException/warnException/errorException>

1 つ目のセットは，該当するログ・レベルでそのすべての文字列引数を連結したものを発行します。2 つ目のセットは，連結した文字列とともに，最後にスローされた例外のスタック・トレースを発行して，詳細な情報を提供します。次に例を示します。

```
logger.debug('found the result')
logger.errorException('Error in discovery')
```

modeling.py

modeling.py スクリプトには，ホスト，IP，プロセス CI などを作成するための API が含まれています。これらの API を使って，共通のオブジェクトを作成し，コードを読みやすくできます。次に例を示します。

```
ipOSH= modeling.createIpOSH(ip)
host = modeling.createHostOSH(ip_address)
member1 = modeling.createLinkOSH('member', ipOSH, networkOSH)
```

netutils.py

netutils.py ライブラリは，オペレーティング・システム名の取得，MAC アドレスの有効性の確認，IP アドレスの有効性の確認など，ネットワークや TCP の情報を取得するために使用されます。次に例を示します。

```
dnsName = netutils.getHostName(ip, ip)
isValidIp = netutils.isValidIp(ip_address)
address = netutils.getHostAddress(hostName)
```

shellutils.py

shellutils.py ライブラリは，シェル・コマンドを実行して，実行されたコマンドの終了ステータスを取得するための API を提供します。また，その終了ステータスに基づいて複数のコマンドを実行できます。このライブラリは，シェル・クライアントによって初期化され，そのクライアントを使ってコマンドを実行し，結果を取得します。次に例を示します。

```
ttyClient = clientFactory.createClient(Props)
clientShUtils = shellutils.ShellUtils(ttyClient)
if (clientShUtils.isWinOs()):
        logger.debug ('discovering Windows..')
```

ジョブとパターンの XML 形式

ジョブとパターンは，XML 形式で CMDB に保存されます。ジョブ名は，そのジョブの XML ファイルに表示され，**id** 属性によって参照されます。各ジョブには対応するパターンがあり，**patternId** 属性によって参照されます。

ジョブ XML の例

ジョブ名は **CPUs by TTY** です。対応するパターンは **TTY_HR_CPU** です。

```
<job id="CPUs by TTY" displayName="CPUs by TTY">
      <patternId>TTY_HR_CPU</patternId>
      <triggers>
           <trigger>shell_on_unix</trigger>
      </triggers>
      <schedule>
           <offsetElem>0</offsetElem>
           <generatedFromElem>1</generatedFromElem>
           <schedularType>scheduler_simple_Schedule</schedularType>
           <timeExpElem>Days_1</timeExpElem>
      </schedule>
</job>
```

パターン XML の例

パターン名は **TTY_HR_CPU** です。パターン入力は **<inputClass>** タグによって定義されています。パターン出力は**<discoveredClasses>**タグによって定義されています。

```
<pattern id="TTY_HR_CPU" description="Discover CPU on Unix boxes."
schemaVersion="7.0">
 <discoveredClasses>
      <discoveredClass>container_f</discoveredClass>
      <discoveredClass>cpu</discoveredClass>
 </discoveredClasses>
 <parameters />
 <taskInfo className="appilog.collectors.services.dynamic.core.DynamicService">
      <destinationInfo className="appilog.collectors.tasks.BaseDestinationData">
           <destinationData
name="ip_address">${SOURCE.application_ip}</destinationData>
           <destinationData
name="Protocol">${SOURCE.root_class}</destinationData>
           <destinationData
name="credentialsId">${SOURCE.credentials_id}</destinationData>
           <destinationData
name="language">${SOURCE.language:NA}</destinationData>
           <destinationData
name="codepage">${SOURCE.codepage:NA}</destinationData>
      </destinationInfo>
      <params
className="appilog.collectors.services.dynamic.core.DynamicServiceParams">
           <script>TTY_HR_CPU_Lib.py</script>
           <script>TTY_HR_CPU.py</script>
      </params>
 </taskInfo>
 <inputClass>shell</inputClass>
</pattern>
```

第 11 章

多言語ロケールのサポート

注： この機能は，Content Pack 3.00 以降で使用できます。

本章の内容

概念

タスク

参照先

多言語ロケールのサポートの概要

多言語ロケール機能によって，DDM をさまざまなオペレーティング・システム (OS) 言語で使用し，実行時に適切なカスタマイズを有効にすることができます。

Content Pack 3.00 より前では，DDM では静的に指定したエンコーディングですべてのネットワーク・ターゲットの出力を処理していました。しかし，この方法は多言語の IT ネットワークには適していません。Probe の管理者は，さまざまな OS 言語のホストを検出するため，ジョブ・パラメータをその都度変更して DDM ジョブを手動で複数回再実行する必要がありました。この手順では，ネットワークに多大な負荷がかかるうえに，トリガ CI での即時のジョブ呼び出しや，スケジュール・マネージャによる UCMDB のデータの自動更新など，DDM の重要な機能を使用できませんでした。

現在，標準設定でサポートされているロケール言語は，日本語，ロシア語，ドイツ語です。標準設定のロケールは英語です。

エンコーディングの文字セットの決定

コマンド出力のデコーディングに適切な文字セットは，実行時に判別されます。多言語の解決は，次の事実と前提条件に基づいています。

1. OS 言語は，ロケールに依存しない方法で判別できます。たとえば，Windows の場合は **chcp** コマンド，Linux の場合は **locale** コマンドを実行して判別できます。
2. 言語とエンコーディングの関係はよく知られており，静的に定義できます。たとえば，ロシア語には，Cp866 と Windows-1251 という 2 つの一般的なエンコーディングがあります。
3. 各言語に 1 つの文字セットを使用することをお勧めします。たとえば，ロシア語の推奨される文字セットは Cp866 です。これによって，ほとんどのコマンドの出力がこのエンコーディングで行われます。
4. 次のコマンド出力のエンコーディングは予測できませんが，所定の言語で使用可能なエンコーディングのいずれかです。たとえば，Windows マシンでロシア語ロケールを使用している場合，**ver** コマンド出力は Cp866 で行われますが，**ipconfig** コマンド出力は Windows-1251 で行われます。
5. 既知のコマンドの出力には，既知のキーワードが含まれています。　たとえば，**ipconfig** コマンドには，**IP-Address** 文字列が変換されて含まれています。したがって，**ipconfig** コマンドの出力では，英語の OS の場合は **IP-Address** が，ロシア語の OS の場合は IP-Адрес が，ドイツ語の OS の場合は **IP-Adresse** が含まれます。

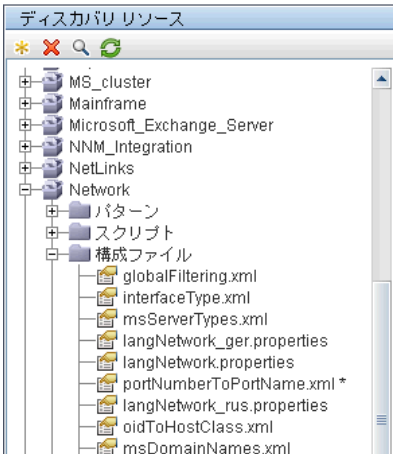
   コマンド出力の言語が検出されると (# 1)，使用可能な文字セットは 1 つか 2 つに限定されます (# 2)。さらに，この出力には既知のキーワードが含まれています (# 5)。

   したがって，結果でキーワードを検索し，使用可能なエンコーディングのいずれかを使用して，コマンド出力をデコードすることで解決できます。キーワードが見つかった場合，現在の文字セットが適切な文字セットとみなされます。

リソース・バンドル

リソース・バンドルは，プロパティの拡張子 (***.properties**) を持つファイルです。プロパティ・ファイルは，データが「**キー = 値**」という形式で保存されている辞書とみなすことができます。プロパティ・ファイルの各行には，1 組の「**キー = 値**」の関連付けが収められています。リソース・バンドルの主な機能は，キーによって値を返すことです。

リソース・バンドルは，Probe マシンの **C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥probeManager¥discoveryConfigFiles** にあります。これらは他の構成ファイルと同様に UCMDB サーバからダウンロードします。［ディスカバリ リソースの管理］ウィンドウで，編集，追加，削除が可能です。詳細については，243 ページ「［構成ファイル］表示枠」を参照してください。



DDM で宛先を検出するときは，通常，コマンド出力またはファイル・コンテンツからテキストを解析する必要があります。多くの場合，この解析は正規表現に基づいています。解析に使用する正規表現は，言語によって異なります。すべての言語に対応するコードを記述するには，言語固有のすべてのデータをリソース・バンドルに抽出する必要があります。言語ごとにリソース・バンドルがあります（リソース・バンドルに複数の異なる言語のデータが含まれていることもありますが，DDM では，1 つのリソース・バンドルに常に 1 つの言語のデータが含まれています）。

Jython スクリプト自体には，ハード・コーディングされた言語固有のデータ（言語固有の正規表現など）はありません。このスクリプトによって，リモート・システムの言語の判別，適切なリソース・バンドルのロード，特定のキーによる言語固有のすべてのデータの取得が行われます。

DDM では，リソース・バンドルは **< ベース名 > _< 言語識別子 >** .properties という特定の名前形式になります。たとえば，langNetwork_spa.properties です（標準設定のリソース・バンドルの形式は **< ベース名 >** .properties です。例 :langNetwork.properties）。

形式の**ベース名**は，このバンドルの目的を表します。たとえば，**langMsCluster** は，そのリソース・バンドルに MS Cluster DDM ジョブで使用する言語固有リソースが含まれていることを示しています。

形式の**言語識別子**は，言語を識別する 3 文字の略語です。たとえば，rus はロシア語を，ger はドイツ語を意味します。言語識別子は，Language オブジェクトの宣言に含まれます。

新しい言語サポートの追加

このタスクでは，新しい言語のサポートを追加する方法を説明します。

このタスクには次の手順が含まれます。

- 370 ページ「リソース・バンドル（*.properties ファイル）の追加」
- 370 ページ「Language オブジェクトの宣言と登録」

1 リソース・バンドル（*.properties ファイル）の追加

実行するジョブに合わせて，リソース・バンドルを追加します。次の表に，DDM のジョブと各ジョブで使用するリソース・バンドルを示します。

ジョブ	リソース・バンドルのベース名
File Monitor by Shell	langFileMonitoring
Host Resources and Applications by Shell	langHost_Resources_By_TTY, langTCP
Hosts by Shell using NSLOOKUP in DNS Server	langNetwork
Host Connection by Shell	langNetwork
Collect Network Data by Shell or SNMP	langTCP
Host Resources and Applications by SNMP	langTCP
Microsoft Exchange Connection by NTCMD, Microsoft Exchange Topology by NTCMD	msExchange
MS Cluster by NTCMD	langMsCluster

バンドルの詳細については，368 ページ「リソース・バンドル」を参照してください。

2 Language オブジェクトの宣言と登録

新しい言語を定義するには，次の 2 行のコードを **shellutils.py** スクリプトに追加します。これには，現在，サポートされているすべての言語のリストが含まれています。このスクリプトは AutoDiscoveryContent パッケージに格納されています。スクリプトを表示するには，［ディスカバリ リソースの管理］ウィンドウにアクセスします。詳細については，256 ページ「［ディスカバリ リソースの管理］ウィンドウ」を参照してください。

a 次のように，言語を宣言します。

```
LANG_RUSSIAN = Language(LOCALE_RUSSIAN, 'rus', ('Cp866', 'Cp1251'),
(1049,), 866)
```

クラス言語の詳細については，373 ページ「API リファレンス」を参照してください。Class Locale オブジェクトの詳細については，http://java.sun.com/j2se/1.5.0/docs/api/java/util/Locale.html を参照してください。既存のロケールを使用するか，新しいロケールを定義できます。

b 言語を次のコレクションに追加して登録します。

```
LANGUAGES = (LANG_ENGLISH, LANG_GERMAN, LANG_SPANISH,
LANG_RUSSIAN, LANG_JAPANESE)
```

ローカライズしたデータを使用する新しいジョブの定義

このタスクでは，ローカライズしたデータを使用する新しいジョブを作成する方法について説明します。

Jython スクリプトでは，通常，コマンドを実行して出力を解析します。このコマンド出力を適切にデコーディングして受け取るには，**ShellUtils** クラス用の API を使用します。詳細については，291 ページ「DDM Web サービス API」を参照してください。

このコードは，通常，次の形式になっています。

```
client = Framework.createClient(protocol, properties)
shellUtils = shellutils.ShellUtils(client)
languageBundle = shellutils.getLanguageBundle (’angNetwork’, shellUtils.osLanguage,
Framework)
strWindowsIPAddress = languageBundle.getString(’windows_ipconfig_str_ip_address’)
ipconfigOutput = shellUtils.executeCommandAndDecode(’ipconfig /all’,
strWindowsIPAddress)
#Do work with output here
```

1 クライアントを作成します。

```
client = Framework.createClient(protocol, properties)
```

2 **ShellUtils** クラスのインスタンスを作成し，オペレーティング・システムの言語を追加します。この言語を追加しないと，標準設定の言語 （通常は英語）が使用されます。

```
shellUtils = shellutils.ShellUtils(client)
```

オブジェクトの初期化中，DDM によってマシンの言語が自動的に検出され，定義済みの Language オブジェクトから推奨されるエンコーディングが設定されます。推奨されるエンコーディングは，エンコーディング・リストで最初に表示されているインスタンスです。

3 **getLanguageBundle** メソッドを使用して，**shellclient** から適切なリソース・バンドルを取得します。

```
languageBundle = shellutils.getLanguageBundle ('langNetwork', shellUtils.osLanguage, Framework)
```

4 リソース・バンドルから，特定のコマンドに対して適切なキーワードを取得します。

```
strWindowsIPAddress = languageBundle.getString('windows_ipconfig_str_ip_address')
```

5 **executeCommandAndDecode** メソッドを呼び出し，**ShellUtils** オブジェクトでキーワードを渡します。

```
ipconfigOutput = shellUtils.executeCommandAndDecode('ipconfig /all', strWindowsIPAddress)
```

ShellUtils オブジェクトは，（このメソッドの詳細が説明されている）API リファレンスをユーザに示すためにも必要です。

6 出力を通常どおり解析します。

キーワードを使用しないコマンドのデコーディング

現在のローカライズ方法では，すべてのコマンド出力のデコードにキーワードを使用しています。詳細については，371 ページ「ローカライズしたデータを使用する新しいジョブの定義」の手順 372 ページ「4」を参照してください。

ただし，最初のコマンド出力にのみキーワードを使用し，それ以降のコマンドには，最初のコマンドのデコードに使用した文字セットを使用する方法もあります。この方法を使うには，**ShellUtils** オブジェクトの **getCharsetName** メソッドと **useCharset** メソッドを使用します。

次に，一般的な使用例を示します。

1 **executeCommandAndDecode** メソッドを 1 回呼び出します。

2 **getCharsetName** メソッドにより，直近に使用した文字セットの名前を取得します。

3 **shellUtils** の標準設定でこの文字セットを使用するため，**ShellUtils** オブジェクトで **useCharset** メソッドを呼び出します。

4 **ShellUtils** の **execCmd** メソッドを 1 回以上呼び出します。手順 372 ページ「3」で指定した文字セットで出力が返されます。これ以外のデコーディング操作は発生しません。

標準設定の言語の変更

OS の言語を判別できない場合，標準設定の言語が使用されます。標準設定の言語は，**shellutils.py** ファイルで指定されています。

```
#default language for fallback
DEFAULT_LANGUAGE = LANG_ENGLISH
```

標準設定の言語を変更するには, DEFAULT_LANGUAGE 変数を別の言語で初期化します。詳細については，369 ページ「新しい言語サポートの追加」を参照してください。

API リファレンス

本項の内容

Language クラス

このクラスは，リソース・バンドルのポストフィックスや使用可能なエンコーディングなど，言語に関する情報をカプセル化します。

フィールド

名前	説明
locale	ロケールを表す Java オブジェクト。
bundlePostfix	リソース・バンドルのポストフィックス。このポストフィックスは，リソース・バンドル名で言語を示すために使用されます。たとえば，**langNetwork_ger.properties** というバンドルには，**ger** がバンドルのポストフィックスとして含まれています。
charsets	この言語のエンコードに使用する文字セット。1 つの言語に対し，複数の文字セットが存在できます。たとえば，ロシア語には，一般的なエンコーディングとして Cp866 と Windows-1251 があります。
wmiCodes	Microsoft Windows OS が言語を識別するために使用する WMI コードのリスト。使用可能なコードのリストは http://msdn.microsoft.com/en-us/library/aa394239(VS.85).aspx（OSLanguage の項）にあります。OS の言語を識別する方法には，WMI クラス OS に対して OSLanguage プロパティを問い合わせる方法があります。
codepage	特定の言語で使用するコード・ページ。たとえば，ロシア語のマシンでは 866 を，英語のマシンでは 437 を使用します。OS の言語を識別するには，標準設定のコード・ページを取得する方法があります（例 : chcp コマンドを使用）。

executeCommandAndDecode メソッド

このメソッドは，Jython スクリプトのビジネス・ロジックで使用されます。デコーディング操作をカプセル化し，デコーディングしたコマンド出力を返します。

引数

名前	説明
cmd	実行する実際のコマンド。
keyword	デコーディング操作で使用するキーワード。
framework	DDM で実行可能なすべての Jython スクリプトに渡す Framework オブジェクト。
timeout	コマンドのタイムアウト。
waitForTimeout	タイムアウトを超えた場合にクライアントが待機するかどうかを指定します。
useSudo	sudo を使用するかどうかを指定します（UNIX マシン・クライアントのみ）。
language	言語を自動検出せず，直接指定できるようにします。

getCharsetName メソッド

このメソッドは，直近に使用した文字セットの名前を返します。

useCharset メソッド

このメソッドは，ShellUtils インスタンスに文字セットを設定します。ShellUtils インスタンスはこの文字セットを最初のデータ・デコーディングに使用します。

引数

名前	説明
charsetName	文字セットの名前（windows-1251，UTF-8 など）。

375 ページ「getCharsetName メソッド」も参照してください。

getLanguageBundle メソッド

適切なリソース・バンドルを取得するために使用するメソッドです。次の API の代わりになるものです。

```
Framework.getEnvironmentInformation().getBundle(…)
```

引数

名前	説明
baseName	言語のサフィックスのないバンドルの名前（langNetwork など）。
language	Language オブジェクト。ここで，ShellUtils.osLanguage を渡します。
framework	DDM で実行可能なすべての Jython スクリプトに渡す，一般的な Framework オブジェクト。

osLanguage フィールド

このフィールドには，言語を表すオブジェクトが格納されます。

第 V 部

ディスカバリおよび依存関係マップ・セキュリティ

第 12 章

DDM のセキュリティ強化

本章では, ディスカバリおよび依存関係マップ (DDM) のセキュリティ強化について説明します。

本章の内容

概念

タスク

ディスカバリおよび依存関係マップのセキュリティ強化 ー概要

[ディスカバリ プローブの設定] ウィンドウに入力した資格情報は，ドメイン・スコープ・ドキュメント (DSD: domain scope document) という暗号化されたファイルに保存されます。この DSD には，ディスカバリ・ドメイン・データが含まれます。ドキュメントに含まれる各ディスカバリ・ドメイン・エントリには，ドメインの Probe と，Probe がリモート・マシンとの通信に使用する可能性のある資格情報のための，ネットワーク・スコープが含まれます。

注: HP Universal CMDB のユーザ管理に関連するセキュリティ機能（認証と承認など）についてはここでは説明しません。

本項の内容

セキュリティ上の基本的な前提条件

セキュリティ上の前提条件は次のとおりです。

- HP Universal CMDB サーバおよび Probe ファイル・システムへのアクセスを，許可のあるものに限定することでセキュリティが確保されている。
- UCMDB サーバ JMX コンソールのセキュリティが確保されていて，UCMDB システム管理者だけがアクセスを許可されるようになっている。localhost からのアクセスに限定されているのが望ましい。

資格情報暗号管理

次の資格情報暗号の管理のためのガイドラインに注意してください。

- **domainScopeDocument**(DSD) ファイルは，標準の対称暗号化方式により暗号化されます。DSD ファイルは転送中にその場所で暗号化されます（鍵はサーバ・データベースと Probe ファイル・システムに保存されます）。暗号化には，192 バイトの標準設定の鍵で AES アルゴリズムが使用されます（ただし，暗号鍵 / 復号鍵とほかのセキュリティ・パラメータの長さを指定できます）。鍵のサイズの変更方法の詳細については，387 ページ「暗号鍵の更新」を参照してください。
- 標準の対称鍵は，UCMDB のインストールにより配布されます。標準設定の鍵はすべて同一なので，この鍵をローカルで生成された鍵で置き換える必要があります。
- DSD ファイルは，暗号化ファイル形式でのエクスポートおよびインポートが可能です。ファイルをインポートするには，ファイルの暗号化に使用する鍵を指定する必要があります。サーバの JMX コンソール（ディスカバリ・マネージャ・サービス）を通してインポートおよびエクスポート操作を実行します。詳細については，391 ページ「domainScopeDocument (DSD) ファイルの暗号化形式でのエクスポートとインポート」を参照してください。

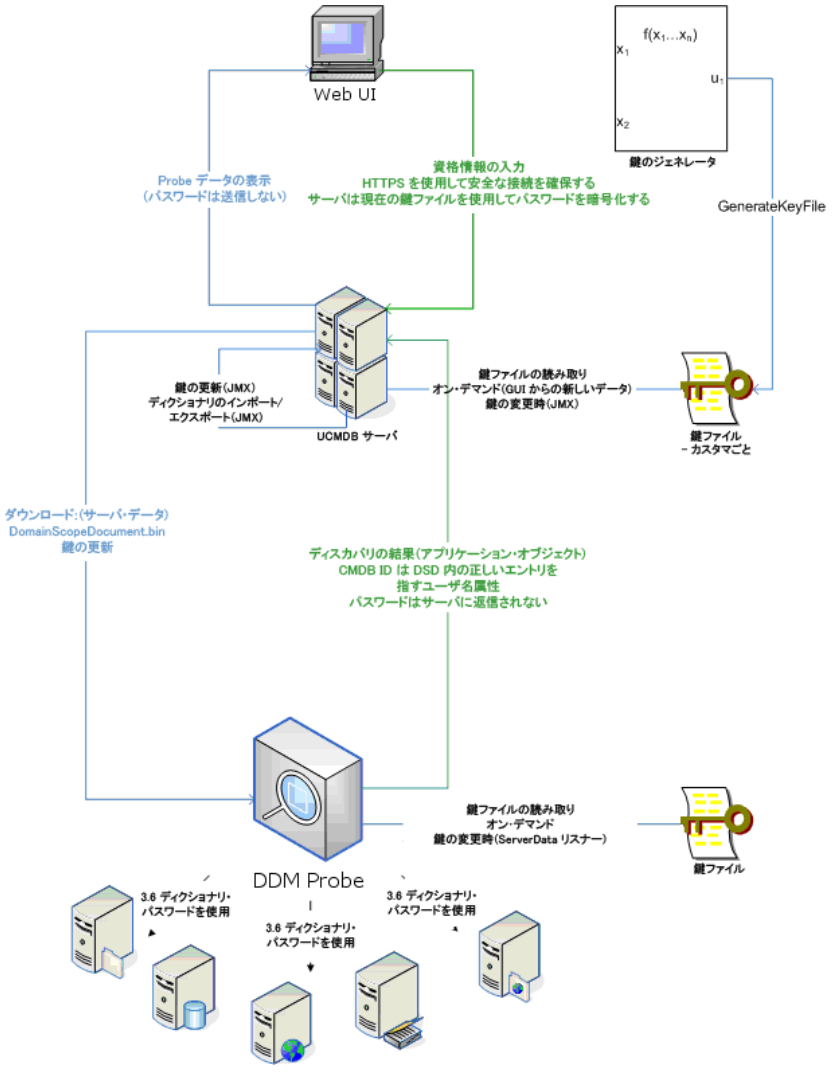
- システムが起動している間は，データを損失することなくシステムの一貫性が保たれるよう鍵を交換できます。これはサーバの JMX コンソールを通して管理されます。詳細については，384 ページ「暗号鍵の生成または更新」を参照してください。
- 鍵が更新されると，自動的に新しい鍵を Probe に配布できます。このオプションは，デプロイがより簡単ではありますが，安全性は低下すると考えられます。新しい鍵は，古い鍵を使って暗号化されます。セキュリティを強化するため，手動で鍵を変更できます。詳細については，386 ページ「新規暗号鍵の生成」を参照してください。
- DSD 暗号化の鍵はそれ自体が DPAPI を使用して暗号化され，Probe と UCMDB サーバ・ファイル・システムにクリア・テキストではなく暗号化された形式で保存されます。DPAPI は，暗号化処理で Windows のユーザ・パスワードに依存します。このため，Probe が確実に鍵を読み取れるように，常に同じ Windows ユーザ（ユーザ・パスワードの変更は可能）で Probe を実行する必要があります（DPAPI は，Windows 上で証明書と秘密鍵などの機密データを保護する標準のメソッドです）。

HTTPS/SSL の設定

HP Universal CMDB サーバと DDM Probe 間の通信に，HTTPS/SSL を使用するように設定できます。これにより，移行中により高度な DSD セキュリティが実現されます。

注：SSL を使用すると，HP Universal CMDB 製品のほかの面（ディスカバリ・タスクや収集された結果など）の安全性がより高くなります。

次の図は，セキュリティが強化されたシステムを示しています。



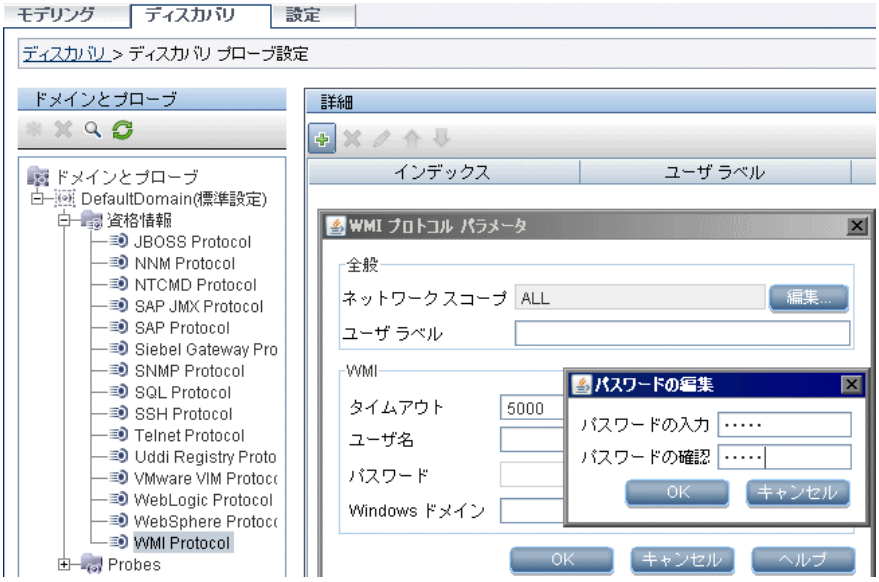
資格情報のストレージの管理

本項では DSD ファイルの管理方法について説明します。

本項の内容

1 資格情報の表示（データの方向 : サーバから HP Universal CMDB へ）

パスワードは，サーバからアプリケーションへは送信されません。つまり，HP Universal CMDB は，パスワード・フィールドの内容に関係なくアスタリスク（*）を表示します。



2 資格情報の更新（データの方向 : HP Universal CMDB からサーバへ）

- この方向の通信は暗号化されないため，UCMDB サーバへは HTTPS/SSL を使用して接続するか，信頼されたネットワークを通して接続する必要があります。

  通信は暗号化されませんが，パスワードはネットワークでクリア・テキストとしては送信されません。パスワードは標準設定の鍵で暗号化されるため，送信中の機密性を高めるために SSL を使用することを強くお勧めします。

- password フィールドは，40 文字に制限されています。パスワードはファイルにのみ保存されるため，パスワードの長さに関してほかの制限はありません。
- パスワードには，特殊文字や英字以外の文字も使用できます。

暗号鍵の生成または更新

暗号鍵を生成または更新できます。いずれの場合にも，DDM は，指定したパラメータ（鍵の長さ，追加 PBE サイクル，JCE プロバイダ) に基づいて新しい暗号鍵を作成します。 DPAPI 暗号化を無効にすることもできます。

注 :

- 鍵の作成に使用するメソッド (generateEncryptionKey) と鍵の更新に使用するメソッド (changeEncryptionKey) の違いは，generateEncryptionKey では新しいランダム暗号鍵を作成し，changeEncryptionKey では指定した名前の暗号鍵をインポートするという点です。
- インストールされた Probe の数に関わらず，システムで使用できる暗号鍵は 1 つのみです。

本項の内容

新規暗号鍵の生成

UCMDB サーバと暗号化 / 複合化用の Probe で使用される新しい鍵を生成できます。DDM は，以前の鍵を新しく生成した鍵と置き換え，この鍵を Probe に配布します。

JMX コンソールを通して新しい暗号鍵を生成するには，次の手順を実行します。

1 Web ブラウザを起動し，サーバ・アドレスに http://localhost:8080/jmx-console と入力します。

ユーザ名とパスワードでのログインが必要な場合もあります。

2 MAM の下の **service=Discovery manager** をクリックして［JMX MBEAN View］ページを開きます。

3 **generateEncryptionKey()** 演算を見つけます。

- ［**customerId**］パラメータ・ボックスに **1**（標準設定）を入力します。
- **keySizeInBits**: 暗号鍵の長さです（128，192，256 の長さが指定可能）。
- **usePBE: true**: 追加 PBE ハッシュ・サイクルを使用します。**false**: 追加 PBE ハッシュ・サイクルを使用しません。
- **jceVendor**: 標準設定でない JCE プロバイダの使用を選択できます。このボックスが空のときは，標準設定のプロバイダが使用されます。
- **autoUpdateProbe**: True - サーバは新しい鍵を自動的に Probe に配布します。False - 新しい鍵を手動で Probe に配置してください。
- **exportEncryptionKey**: **True** - 新しいパスワードを作成して安全なストレージに保存することの他に，サーバは新しいパスワードをファイル・システム（**C:¥hp¥UCMDB¥UCMDBServer¥root¥lib¥server¥discovery¥key.bin**）にエクスポートします。このオプションにより，新しいパスワードを使用して Probe を手動で更新できます。**False** - 新しいパスワードはファイル・システムにエクスポートされません。

  Probe を手動で更新するには，autoUpdateProbe を False に設定し，exportEncryptionKey を True に設定します。

重要：

Probe が稼動していて，サーバに接続されていることを確認します。Probe が停止している場合，鍵は Probe に接続できません。

Probe が停止する前に鍵を変更した場合は, Probe が再起動するとその Probe に鍵が再送信されます。ただし，Probe が停止する前に鍵を複数回変更した場合は，JMX コンソールを通して手動でその鍵を変更する必要があります（**False** を選択）。

4 ［**Invoke**］をクリックして，暗号鍵を生成します。

暗号鍵の更新

changeEncryptionKey メソッドを使用して，暗号鍵をインポートします。

JMX コンソールを通して暗号鍵を更新するには，次の手順を実行します。

1 Web ブラウザを起動し，サーバ・アドレスに http://localhost:8080/jmx-console と入力します。

2 ユーザ名とパスワードでのログインが必要な場合もあります。

3 MAM の下の **service=Discovery manager** をクリックして［JMX MBEAN View］ページを開きます。

4 **changeEncryptionKey()** 演算を見つけます。

- ［**customerId**］パラメータ・ボックスに **1**（標準設定）を入力します。
- **newKeyFileName**: 新しい鍵の名前を入力します。
- **keySizeInBits**: 暗号鍵の長さです（128，192，256 の長さが指定可能）。
- **usePBE: true**: 追加 PBE ハッシュ・サイクルを使用します。**false**: 追加 PBE ハッシュ・サイクルを使用しません。
- **jceVendor**: 標準設定でない JCE プロバイダの使用を選択できます。このボックスが空のときは，標準設定のプロバイダが使用されます。

➤ **autoUpdateProbe**: 自動的に変更した鍵を Probe に配布するサーバの場合は，**True** のままにします。

Probe JMX コンソールを使用して手動で変更した鍵を配布する場合は，**False** を選択します。

重要：

Probe が稼動していて，サーバに接続されていることを確認します。Probe が停止している場合，鍵は Probe に接続できません。

Probe が停止する前に鍵を変更した場合は，Probe が再起動するとその Probe に鍵が再送信されます。ただし，Probe が停止する前に鍵を複数回変更した場合は，JMX コンソールを通して手動でその鍵を変更する必要があります（**False** を選択）。

---

5 ［**Invoke**］をクリックして，暗号鍵を生成および更新します。

暗号鍵ファイル名の取得

サーバで暗号鍵を変更する場合（**changeEncryptionKey** メソッドを使用），セキュリティを考慮して新しい鍵の Probe への自動ダウンロードを回避できます。手動で Probe に暗号鍵ファイルをダウンロードできます。

自動ダウロードを防ぐには，importEncryptionKey メソッドを実行します。

1 暗号鍵ファイルを **C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥probeManager¥binaryData** ディレクトリに置きます。

2 Web ブラウザを起動し，Probe アドレスに http://localhost:1977/ と入力します。

ユーザ名とパスワードでのログインが必要な場合もあります。

3 Probe ドメインの下の **type=MainProbe** をクリックして，［MBean View］ページを開きます。

4 **importEncryptionKey** メソッドを見つけます。

5 次のディレクトリに置かれる暗号鍵ファイルの名前を入力します：**C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥probeManager¥binaryData**。

このファイルには，インポートされる鍵が含まれています。

6 ［**importEncryptionKey**］ボタンをクリックします。

DPAPI 暗号化の無効化

鍵はファイル・システムの DPAPI で暗号化されます。この暗号化を無効にできます。

DPAPI の暗号化を無効にするには，次の手順を実行します。

1 Web ブラウザを起動し，サーバ・アドレスに http://localhost:8080/jmx-console と入力します。

ユーザ名とパスワードでのログインが必要な場合もあります。

2 **MAM**の下の **service=Discovery manager** をクリックして［JMX MBEAN View］ページを開きます。

3 **setDPApiUsage()** 演算を見つけて，次のプロパティを入力します。

- ［**customerId**］パラメータ・ボックスに **1**（標準設定）を入力します。
- **useDPAPI: true**: DPAPI を使用します。**false**: DPAPI を使用しません。

4 ［**Invoke**］をクリックして，DPAPI 暗号鍵を無効にします。

複数の JCE プロバイダの定義

JMX コンソールを通して暗号化鍵を生成する場合，**changeEncryptionKey** と **generateEncryptionKey** メソッドで複数の JCE プロバイダを定義できます。標準設定の JCE プロバイダを変更するには，次の手順を実行します。

1. **$JRE_HOME/lib/ext** ディレクトリの JCE プロバイダ jar ファイルを登録します。
2. jar ファイルを次のディレクトリにコピーします。
   - **UCMDB サーバ**の場合：次のディレクトリにある $JRE_HOME へコピーします。
     - **C:¥hp¥UCMDB¥UCMDBServer¥jre**
     - **C:¥hp¥UCMDB¥UCMDBServer¥j2f¥JRE**
   - **DDM Probe** の場合 : 次のディレクトリにある **$JRE_HOME** へコピーします。
     - **C:¥hp¥DDM¥DiscoveryProbe¥jre**
3. **$JRE_HOME¥lib¥security¥java.security** ファイルのプロバイダ・リストの最後にプロバイダ・クラスを追加します。
4. 無制限 JCE ポリシーを含めるように，**local_policy.jar** と **US_export_policy.jar** ファイルを更新します。これらの jar ファイルは Sun の Web サイトからダウンロードできます。
5. UCMDB サーバと DDM Probe を再起動します。
6. **changeEncryptionKey** または **generateEncryptionKey** メソッド用の JCE ベンダ・フィールドを見つけて，JCE プロバイダの名前を追加します。

domainScopeDocument (DSD) ファイルの暗号化形式でのエクスポートとインポート

DSD ファイルを暗号化形式でエクスポートおよびインポートできます（DSD ファイルのインポートは，おそらくシステム・クラッシュ後の回復中またはアップグレード中に行うことになります）。

- **DSD ファイルをエクスポートする場合**，（任意の）パスワードを入力する必要があります。ファイルはこのパスワードで暗号化されます。UCMDB データベースに DSD ファイルを保存するために使用する暗号化鍵は，使用されなくなりました。
- **DSD ファイルをインポートする場合**，DSD ファイルをエクスポートするときに設定したパスワードを使用する必要があります。

重要：UCMDB バージョン 8.02 から domainScopeDocument ファイルをエクスポートした場合，現在のバージョンにファイルをインポートするには，バージョン 8.02 のシステムにある key.bin ファイル内からパスワードをコピーします。

DSD ファイルをエクスポートまたはインポートするには，次の手順を実行します。

1. Web ブラウザを起動し，アドレスに http://localhost:8080/jmx-console と入力します。

   ユーザ名とパスワードでのログインが必要な場合もあります。

2. MAM の下の **service=Discovery manager** をクリックして［JMX MBEAN View］ページを開きます。

3. **ExportDomainScopeDocument** または **importDomainScopeDocument** 操作を見つけて既存のファイル名とパスワードを入力します。

4. ［**Invoke**］をクリックして，domainScopeDocument ファイルをエクスポートまたはインポートします。

   保存された domainScopeDocument ファイルは **C:¥hp¥UCMDB¥UCMDBServer¥root¥lib¥server¥discovery¥<カスタマ・ディレクトリ>**にあります。

第 13 章

DDM Probe のセキュリティ強化

本章では，ディスカバリおよび依存関係マップ（DDM）Probe のセキュリティ強化について説明します。

本章の内容

タスク

DDM Probe 用 MySQL データベースのセキュリティ強化

スクリプトを実行することにより，DDM Probe の MySQL データベースのユーザ・パスワードを設定できます。パスワードを設定すると，**delCollectors.bat** スクリプトは，実行されるたびにデータベースのパスワードを引数として取得することが必要になります。

スクリプト名 : **set_dbuser_password.cmd**

スクリプトの場所 : **C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors**

このスクリプトは，次の方法で使用できます。

- パスワードを設定または変更する必要があるときにはいつでも手動で使用できます。新しいパスワードを引数として指定して，コマンド・プロンプトでファイルを実行します。
- **delCollectors** スクリプトを実行して Probe の MySQL データベースをクリーンアップすると，自動的に，**delCollectors.bat** スクリプトの引数として指定されたパスワードを使用して **mamprobe** ユーザが再作成されます。

注：delCollectors.bat とはすべてのテーブル，リポジトリ，および保留中のタスクを含むタスク・データを Probe から削除し，最初からデータを構築するコマンド・ファイルです。

ユーザ・パスワードを設定するには，次の手順を実行します。

1 DDM Probe を停止します。

2 新しいパスワードを引数として使用して set_dbuser_password.cmd スクリプトを実行します。

3 DDM Probe の構成ファイルにあるパスワードを更新します。

a 構成ファイルに書き込むパスワードは暗号化する必要があります。

b 暗号化された形式のパスワード（AES，192 ビット鍵）を取得するには，**getEncryptedDBPassword** JMX メソッドを使用します。このメソッドは，DDM Probe JMX コンソールの **MainProbe MBean** の下にあります。

c 暗号化されたパスワードを **C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥DiscoveryProbe.properties** ファイルに追加します。

➤ 暗号化されたパスワードを次のプロパティに追加します。

appilog.agent.probe.jdbc.pwd

appilog.agent.local.jdbc.pwd

d 暗号化されたパスワードを **C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥probeManager¥probeMgr-quartz.properties** ファイルに追加します。

➤ 暗号化されたパスワードを次のプロパティに追加します。

org.quartz.dataSource.QUARTZ_DB.password

e 暗号化されたパスワードを **C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥probeManager¥netlinks¥SQL-MySQL.properties** ファイルに追加します。

➤ 暗号化されたパスワードを次のプロパティに追加します。

JDBC.Password=

4 C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥probe_setup.log ファイルにエラーがないかどうかを調べます。

5 データベースのパスワードが記録されているため，C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥probe_setup.log ファイルを削除します。

6 DDM Probe を起動します。

ローカル・マシンにのみ DDM Probe の MySQL データベースへのアクセスを許可するには，次の手順を実行します。

remove_remote_user_access.cmd スクリプトをコマンド・プロンプトで，またはダブルクリックして実行します。リモート・コンピュータからアクセスしようとするユーザは，すべてアクセスを拒否されます。

MySQL データベースの暗号化されたパスワードの設定

本項では，MySQL データベース・ユーザのパスワードの暗号化方法について説明します。

1 パスワードの暗号化形式（AES，192 ビット鍵）を作成します。

a DDM Probe JMX コンソールにアクセスします。Web ブラウザを起動し，アドレスとして「**http://<DDM Probe マシン名または IP アドレス>:1977**」と入力します。DDM Probe をローカルで実行している場合は，「**http://localhost:1977**」と入力します。

ユーザ名とパスワードでのログインが必要な場合もあります。

b **Type=MainProbe** サービスを見つけ，リンクをクリックして［JMX MBEAN View］ページを開きます。

c **getEncryptedDBPassword** 操作を見つけます。

d ［**DB Password**］フィールドに，暗号化するパスワードを入力します。

e ［**getEncryptedDBPassword**］ボタンをクリックして操作を呼び出します。

この呼び出しの結果は，次のような暗号化されたパスワード文字列となります。

```
66,85,54,78,69,117,56,65,99,90,86,117,97,75,50,112,65,53,67,114,112,65,61,61
```

2 DDM Probe を停止します。

3 set_dbuser_password.cmd スクリプトを実行します。

a 新しいパスワードを引数として指定して **set_dbuser_password.cmd** スクリプトを実行します。たとえば，**set_dbuser_password ＜自分のパスワード＞** のように指定します。

パスワードは，暗号化していない形式（平文）で入力する必要があります。

このスクリプトは，次のフォルダにあります。
C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors

4 DDM Probe の構成ファイルにあるパスワードを更新します。

a 構成ファイルに書き込むパスワードは暗号化する必要があります。暗号化された形式のパスワードを取得するには，396 ページ の説明に従って **getEncryptedDBPassword** JMX メソッドを使用します。

b 暗号化されたパスワードを，**C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥DiscoveryProbe.properties** ファイルの次のプロパティに追加します。

- **appilog.agent.probe.jdbc.pwd**

  例：

```
appilog.agent.probe.jdbc.user = mamprobe
appilog.agent.probe.jdbc.pwd =
66,85,54,78,69,117,56,65,99,90,86,117,97,75,50,112,65,53,67,114,112,65,61,61
```

- **appilog.agent.local.jdbc.pwd**

c 暗号化されたパスワードを **C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥probeManager¥probeMgr-quartz.properties** ファイルの次のプロパティに追加します。

- **org.quartz.dataSource.QUARTZ_DB.password**

5 DDM Probe を起動します。

delCollectors.bat スクリプト：使用法

delCollectors.bat スクリプトは，スクリプトに引数として指定されたパスワードを使用してデータベース・ユーザを再作成します。

パスワードを設定すると，**delCollectors.bat** スクリプトは，実行されるたびにデータベースのパスワードを引数として取得します。

スクリプトの実行後：

- **C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥probe_setup.log** ファイルを調べ，エラーがないかどうか確認します。
- データベースのパスワードが記録されているため，**C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥probe_setup.log** ファイルを削除します。

JMX コンソールの暗号化されたパスワードの設定

本項では，JMX ユーザのパスワードの暗号化方法について説明します。暗号化されたパスワードは，DiscoveryProbe.properties ファイルに保存されます。

1 パスワードの暗号化形式（AES，192 ビット鍵）を作成します。

a DDM Probe JMX コンソールにアクセスします。Web ブラウザを起動し，アドレスとして「**http://<DDM Probe マシン名または IP アドレス >:1977**」と入力します。DDM Probe をローカルで実行している場合は，「**http://localhost:1977**」と入力します。

ユーザ名とパスワードでのログインが必要な場合もあります。

b **Type=MainProbe** サービスを見つけ，リンクをクリックして［JMX MBEAN View］ページを開きます。

c **getEncryptedKeyPassword** 操作を見つけます。

d ［**Key Password**］フィールドに，暗号化するパスワードを入力します。

e ［**getEncryptedDBPassword**］ボタンをクリックして操作を呼び出します。

この呼び出しの結果は，次のような暗号化されたパスワード文字列となります。

```
85,-9,-61,11,105,-93,-81,118
```

2 DDM Probe を停止します。

3 暗号化されたパスワードを，C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥DiscoveryProbe.properties ファイルの次のプロパティに追加します。

appilog.agent.Probe.JMX.BasicAuth.Pwd

例：

```
appilog.agent.Probe.JMX.BasicAuth.User=admin
appilog.agent.Probe.JMX.BasicAuth.Pwd=-85,-9,-61,11,105,-93,-81,118
```

4 DDM Probe を起動します。

UCMDB サーバと DDM Probe の間の相互認証による SSL の有効化

DDM Probe と UCMDB サーバの両方で, 証明書による認証を設定できます。設定すると，接続を確立する前に両者の証明書が送信されて認証されます。

重要：DDM Probe で SSL を有効にする次の方法により，使用されなくなった基本認証の手順が置き換えられています。基本認証の詳細については，408 ページ「基本認証による DDM Probe での SSL の有効化」を参照してください。

本項の内容

概要

UCMDB は，UCMDB サーバと DDM Probe の間の通信で次のモードをサポートしています。

- **相互認証**：このモードでは，SSL を使用し，Probe によるサーバ認証とサーバによるクライアント認証の両方を実行できます。詳細については，400 ページ「相互証明書認証の有効化」を参照してください。

- **サーバ認証**：このモードでは，SSL を使用し，Probe は UCMDB サーバの証明書を認証します。詳細については，407 ページ「サーバ証明書認証のみの有効化」を参照してください。
- **標準 HTTP**: SSL 通信は行われません。これは標準設定モードで，UCMDB の SSL ポートは無効になっており，サーバは標準 HTTP プロトコルに従って DDM Probe と通信します。

キー・ストアとトラスト・ストア

UCMDB サーバと DDM Probe は，キー・ストアとトラスト・ストアを使用して動作します。

- **キー・ストア**：キー・エントリ（証明書および一致する秘密鍵）を保持するファイル。
- **トラスト・ストア**：リモート・ホストを検証するために使用する証明書を保持するファイル（たとえば，サーバ認証で使用する場合，DDM Probe のトラスト・ストアには UCMDB サーバの証明書が含まれている必要があります）。

相互証明書認証の有効化

HP Universal CMDB Web サーバによって使用されている証明書が良く知られている認証局（CA）で発行されたものである場合は，おそらく次の手順を実行する必要はありません。トラストを検証する場合は，SSL を使用して Web サーバに接続し，証明書が信頼できるものであるかどうかを調べます。

認証時，UCMDB はその UCMDB の証明書を DDM Probe クライアント・マシンに送信し，DDM Probe はその DDM Probe の証明書を UCMDB サーバに送信します。

1 UCMDB の SSL ポートを有効にします。

a 次のフォルダにある **server.xml** ファイルで, 次のセクションのコメントを解除します。
C:¥hp¥UCMDB¥UCMDBServer¥j2f¥EJBContainer¥server¥mercury¥deploy¥jbossweb-tomcat55.sar¥:

```
<Connector port="8443" address="${jboss.bind.address}"
        maxThreads="100" strategy="ms" maxHttpHeaderSize="8192"
        emptySessionPath="true"
        scheme="https" secure="true" clientAuth="false"
        keystoreFile="${jboss.server.home.dir}/conf/chap8.keystore"
        keystorePass="rmi+ssl" sslProtocol = "TLS"
        truststoreFile="${jboss.server.home.dir}/conf/server.truststore"
        truststorePass="truststore_password"/>
```

注 : SSL によって接続するよう以前にセットアップした UCMDB サーバ（たとえば，前バージョン）を相互認証を使用するように設定する場合は, **server.xml** ファイルのコードを使用できますが, **truststoreFile** パラメータと **truststorePass** パラメータを追加する必要があります。詳細については,『**HP Universal CMDB デプロイメント・ガイド**』（PDF）の「Tomcat Web サーバでの SSL の有効化」を参照してください。

---

b **keystoreFile** 属性と **keystorePass** 属性を，キー・ストア・ファイルとパスワードで置き換えます。

c **truststoreFile** 属性と **truststorePass** 属性を，トラストストア・ファイルとパスワードで置き換えます。

注 : バージョン 8.04 以降，DDM Probe では，標準設定の Java トラストストア（**C:¥hp¥DDM¥DiscoveryProbe¥jre¥lib¥security¥cacerts**）と ssl.properties ファイルのトラストストア（**javax.net.ssl.trustStore** で定義）から読み取ることによって証明書を検証するようになっています。これらのトラストストアのどちらによっても証明書が有効と判断されない場合は，接続が拒否されます。

---

d キー・ストアに複数の証明書がある場合は，次の属性を追加してください。

```
keyAlias="certificate_alias"
```

certificate_alias を，UCMDB キー・ストアにある別名で置き換えます。

注：キー・ストアの作成およびインポートの詳細については，405 ページ「キー・ストアの作成とインポート」を参照してください。

---

2 UCMDB サーバでのクライアント証明書認証を有効にします。

a **web.xml** ファイルにアクセスします。

web.xml ファイルは，次の場所の **mam-collectors.war** アーカイブ・ファイルの **WEB-INF** フォルダにあります。**C:¥hp¥UCMDB¥UCMDBServer¥j2f¥AppServer¥deploy¥mam-jars¥** フォルダ。

b **web.xml** ファイルで，次のセクションのコメントを解除します。

```
<security-constraint>
      <web-resource-collection>
         <web-resource-name>AutoDiscovery_Servlets</web-resource-name>
           <description>Require users to authenticate</description>
           <url-pattern>/collectors/*</url-pattern>
           <url-pattern>/collectorsResults/*</url-pattern>
           <url-pattern>/collectorsUnSerializedResults/*</url-pattern>
           <url-pattern>/downloader/*</url-pattern>
      </web-resource-collection>
      <auth-constraint>
         <role-name>*</role-name>
      </auth-constraint>
  </security-constraint>
  <login-config>
      <auth-method>CLIENT-CERT</auth-method>
      <realm-name>AutoDiscovery</realm-name>
  </login-config>
 <security-role>
    <role-name>*</role-name>
 </security-role>
```

3 DDM Probe での SSL 通信を有効にします。

a **C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥** フォルダにある **DiscoveryProbe.properties** ファイルを編集します。

b **appilog.agent.probe.protocol** プロパティを **HTTPS** に変更します。

c **serverPortHttps** 値が，手順 401 ページ「1」での接続の **port** 属性と同じになっていることを確認します。

➤ **DiscoveryProbe.properties:**

```
# Ports used for HTTP/s traffic
serverPort = 8080
serverPortHttps = 8443
```

➤ **server.xml:**

```
<Connector port="8443" address="${jboss.bind.address}"
```

標準設定ポートは **8443** です。

4 DDM Probe のキー・ストア・ファイルとトラスト・ストア・ファイルを定義します。

a **C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥security¥** フォルダの **ssl.properties** ファイルを編集します。

b キー・ストア・ファイルの名前を **javax.net.ssl.keyStore** プロパティに追加します。

重要：C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥security¥ssl.properties で定義する DDM Probe キー・ストアには，キー・エントリが 1 つだけ含まれている必要があります。

---

c トラスト・ストア・ファイルの名前を **javax.net.ssl.trustStore** プロパティに追加します。

d キー・ストアの暗号化パスワードを **javax.net.ssl.keyStorePassword** プロパティで定義します。

e トラスト・ストアの暗号化パスワードを **javax.net.ssl.trustStorePassword** プロパティで定義します。

注：キー・ストアとトラスト・ストアのパスワードが暗号化されます。詳細については，404 ページ「パスワードの暗号化」を参照してください。

パスワードの暗号化

DDM のキー・ストアおよびトラスト・ストアの暗号化パスワードは，**C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥security¥** フォルダの **ssl.properties** ファイルに保存されます。

1 DDM Probe を実行します（[**スタート**] > [**プログラム**] > [**HP DDM**] > [**DDM Probe**]）。

2 DDM Probe JMX コンソールにアクセスします。Web ブラウザを起動し，アドレスとして「**http://<DDM Probe マシン名または IP アドレス >:1977**」と入力します。DDM Probe をローカルで実行している場合は，「**http://localhost:1977**」と入力します。

ユーザ名とパスワードでのログインが必要な場合もあります。

3 **Type=MainProbe** サービスを見つけ，リンクをクリックして［JMX MBEAN View］ページを開きます。

4 **getEncryptedKeyPassword** 操作を見つけます。

5 ［**キー パスワード**］フィールドにキー・ストアまたはトラスト・ストアのパスワードを入力し，［**getEncryptedKeyPassword**］をクリックします。

6 **C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥security¥** フォルダで，**ssl.properties** ファイルを開きます。

7 暗号化されたパスワード（たとえば 1,2,3,4,5 のように，数字をカンマで区切ります）をコピーし，**ssl.properties** ファイルの関連するキー・ストアまたはトラスト・ストアの行に貼り付けます。

8 ファイルを保存します。

キー・ストアの作成とインポート

以降の各項では，次の操作の方法について説明します。

- DDM Probe の新規キー・ストアを作成する
- その証明書をエクスポートする
- UCMDB サーバのトラスト・ストアに証明書をインポートする

DDM Probe のキー・ストアを作成するには，次の手順を実行します。

1 次のコマンドを実行します。

```
C:¥hp¥DDM¥DiscoveryProbe¥jre¥bin¥keytool -genkey -alias ddmkey -keyalg RSA -
keystore C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥security¥client.keystore
```

2 パスワード，キー・ストアのパスワード，名前，組織など，必要に応じて情報を入力します。

3 ［**Is CN=... C=... Correct?**］と表示された場合は，「**yes**」と入力して ENTER キーを押します。

4 もう一度 ENTER キーを押して，そのキー・ストア・パスワードをキー・パスワードとして受け入れます。

5 ファイル **C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥security¥client.keystore** が作成されたことを確認します。

作成されたキー・ストアから DDM Probe の証明書をエクスポートするには，次の手順を実行します。

1 次のコマンドを実行します。

```
C:¥hp¥DDM¥DiscoveryProbe¥jre¥bin¥keytool.exe -export -alias ddmkey -keystore
C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥security¥client.keystore -file
C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥security¥client.cer
```

2 前に作成したキー・ストアのパスワードを入力します。

3 ファイル **C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥security¥client.cer** が作成されたことを確認します。

DDM Probe の証明書を UCMDB サーバのトラスト・ストアにインポートするには，クライアント認証を有効にします。

1 次のコマンドを実行します。

```
C:¥hp¥DDM¥DiscoveryProbe¥jre¥bin¥keytool.exe -import -v -keystore
<SERVER_TRUSTSTORE_PATH> -file
C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥security¥client.cer -alias ddmkey
```

2 UCMDB サーバのトラスト・ストア・パスワードを入力します。

3 ［**Trust this certificate?**］と表示された場合は，「**yes**」と入力して ENTER キーを押します。

4 出力が［**証明書がキー・ストアに追加されました**］であることを確認します。

トラブルシューティング

➤ **C:¥hp¥UCMDB¥UCMDBServer¥j2f¥EJBContainer¥server¥mercury¥conf¥login-config.xml** ファイルに次のセクションが存在し，コメントアウトされていないことを確認します。

```
<application-policy name="auto_discovery">
<authentication>
      <login-module code="org.jboss.security.auth.spi.BaseCertLoginModule" flag =
"required">
      <module-option name="securityDomain">java:/jaas/auto_discovery</module-
option>
      <module-option
name="verifier">org.jboss.security.auth.certs.AnyCertVerifier</module-option>
      </login-module>
 </authentication>
</application-policy>
```

➤ **C:¥hp¥UCMDB¥UCMDBServer¥j2f¥AppServer¥deploy¥mam-jars¥mam-collectors.war** アーカイブ・ファイルの **WEB-INF** フォルダに **jboss-web.xml** ファイルが存在し，次のセクションがあることを確認します。

```
<jboss-web>
      <security-domain>java:/jaas/auto_discovery</security-domain>
</jboss-web>
```

サーバ証明書認証のみの有効化

UCMDB サーバ認証のみを有効にすることができます。つまり UCMDB サーバは，その UCMDB サーバの証明書を認証のために DDM Probe に送信します。

手順 401 ページ「1」，手順 403 ページ「3」，および手順 404 ページ「4」に従ってください。

基本認証による DDM Probe での SSL の有効化

重要： DDM Probe で SSL を有効にするために行っていた次の方法（基本認証セキュリティ）は使用されなくなっています。セキュリティの有効性がはるかに高いので，手動による認証セキュリティを使用することをお勧めします。詳細については，399 ページ「UCMDB サーバと DDM Probe の間の相互認証による SSL の有効化」を参照してください。

基本認証を設定するには，次の手順を実行します。

1 次のファイルを見つけます。
C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥DiscoveryProbe.properties.

2 次のプロパティからコメント記号（**#**）を削除し，関連する資格情報を入力します。

```
appilog.agent.Probe.BasicAuth.Realm=
appilog.agent.Probe.BasicAuth.User=
appilog.agent.Probe.BasicAuth.Pwd=
```

資格情報は，UCMDB サーバで定義されている情報と一致する必要があります。

リバース・プロキシを使用した DDM Probe の接続

リバース・プロキシを使用して DDM プローブに接続するには，次の手順を実行します。

注： リバース・プロキシを使用して接続する場合は，UCMDB サーバと DDM Probe の間で SSL を使用するときに相互認証を有効にすることができません。

DDM プローブがリバース・プロキシに反して動作するよう設定するには，次の手順を実行します。

1 **discoveryProbe.properties** ファイルを編集します（場所は，**<DDM Probe root directory>¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors**）。

2 **serverIP** プロパティをリバース・プロキシ・サーバの IP または DNS 名に設定します。

3 ファイルを保存します。

リバース・プロキシを経由して DDM Probe のみを HP Universal CMDB に接続する場合は，次のプロキシ・サーバ設定が必須です。

リバース・プロキシ・サーバでの要求	プロキシ要求の処理先
/mam-collectors/*	http://[HP Universal CMDB サーバ]/mam-collectors/*

次の設定は，SOAP アダプタがリバース・プロキシ経由でセキュアな（セキュリティ強化された）HP Universal CMDB へのレプリケーションに使用される場合に必要です。

リバース・プロキシ・サーバでの要求	プロキシ要求の処理先
/axis2/*	http://[HP Universal CMDB server]/axis2/*

DDM プローブと Web クライアントのリバース・プロキシによる接続

DDM プローブとアプリケーション・ユーザの両方がリバース・プロキシ経由で HP Universal CMDB に接続されている場合，次の設定が必須です。

リバース・プロキシ・サーバでの要求	プロキシ要求の処理先
/mam/*	[HP Universal CMDB サーバ]/mam/*
/mam_images/*	[HP Universal CMDB サーバ]/mam_images/*
/mam-collectors/*	[HP Universal CMDB サーバ]/mam-collectors/*

リバース・プロキシ・サーバでの要求	プロキシ要求の処理先
/ucmdb/*	[HP Universal CMDB サーバ]/ucmdb/*
/site	[HP Universal CMDB サーバ]/site/*

DDM Probe の JMX コンソールへのログインの実装

DDM Probe の JMX コンソールへの不正アクセスを防止できます。Probe の JMX コンソールにアクセスするためには，ユーザ名とパスワードを入力する必要があります。次に，ユーザ名とパスワードを実装する手順を示します。

1 DDM Probe サービスを停止します。

2 テキスト・エディタで（**root¥lib¥collector¥probemanager** フォルダの）**DiscoveryProbe.properties** ファイルを開き，次の値を更新します。

```
# Authentication data for the probe JMX
# Values can be empty to represent non-values.
appilog.agent.Probe.JMX.BasicAuth.User=
appilog.agent.Probe.JMX.BasicAuth.Pwd=
```

次のように変更します。

```
# Authentication data for the probe JMX
# Values can be empty to represent non-values.
appilog.agent.Probe.JMX.BasicAuth.User=admin
appilog.agent.Probe.JMX.BasicAuth.Pwd=admin
```

3 結果をテストします。Web ブラウザに，「**http://<DDM Probe machine name or IP address>:1977**」と入力します。[ユーザ名およびパスワード] ダイアログ・ボックスが表示されます。

domainScopeDocument ファイルの場所の制御

Probe のファイル・システムは，暗号鍵と domainScopeDocument ファイルの両方を（標準設定で）保持しています。Probe は，起動するたびに，サーバから domainScopeDocument ファイルを取得してファイル・システムに格納します。承認されていないユーザがそれらの資格情報を取得するのを防ぐために，domainScopeDocument ファイルが Probe のメモリに保持され，Probe のファイル・システムには格納されないように Probe を設定できます。

domainScopeDocument ファイルの場所を制御するには, 次の手順を実行します。

1 **C:¥hp¥DDM¥DiscoveryProbe¥root¥lib¥collectors¥DiscoveryProbe.properties** ファイルを開き，次のように変更します。

```
appilog.collectors.storeDomainScopeDocument=true
```

次のように変更します。

```
appilog.collectors.storeDomainScopeDocument=false
```

これで，Probe Gateway と Probe Manager の serverData フォルダに domainScopeDocument ファイルが存在しなくなります。

domainScopeDocument ファイルを使用して DDM のセキュリティを強化する方法の詳細については，383 ページ「資格情報のストレージの管理」を参照してください。

2 Probe を再起動します。

索引

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

あ

い

う

え

か

き

く

け

こ

さ

し

す

せ

そ

た

つ

て

と

ね

は

ふ

へ

ほ

め

も

よ

り

ろ

わ