

HP Service Manager

ソフトウェアバージョン: Service Manager 9.40、Universal CMDB 10.20 以降
サポート対象の Windows® および Unix® オペレーティングシステム向け

Universal CMDB 統合ガイド (Service Manager 拡張 汎用アダプタ使用)

ドキュメントリリース日: 2015 年 1 月 (英語版)
ソフトウェアリリース日: 2015 年 1 月 (英語版)



ご注意

保証

HP 製品、またはサービスの保証は、当該製品、およびサービスに付随する明示的な保証文によってのみ規定されるものとします。ここでの記載は、追加保証を提供するものではありません。ここに含まれる技術的、編集上の誤り、または欠如について、HP はいかなる責任も負いません。

ここに記載する情報は、予告なしに変更されることがあります。

権利の制限

機密性のあるコンピュータソフトウェアです。これらを所有、使用、または複製するには、HP からの有効な使用許諾が必要です。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に関する文書類、および商用アイテムの技術データは、FAR12.211 および 12.212 の規定に従い、ベンダの標準商用ライセンスに基づいて米国政府に使用許諾が付与されます。

著作権について

© 1994 - 2015 Hewlett-Packard Development Company, L.P.

商標について

Adobe® は、Adobe Systems Incorporated (アドビシステムズ社) の登録商標です。

Microsoft® および Windows® は、米国における Microsoft Corporation の登録商標です。

OracleとJavaは、Oracle Corporationおよびその関連会社の登録商標です。

UNIX® は、The Open Group の登録商標です。

Linux® は Linux Torvalds の米国およびその他の国における登録商標です。

オープンソースおよびサードパーティの謝辞の完全なリストについては、HP ソフトウェアサポートオンライン Web サイトで、製品マニュアル『HP Service Manager Open Source and Third Party License Agreements』を参照してください。

ドキュメントの更新情報

このマニュアルの表紙には、以下の識別情報が記載されています。

- ソフトウェアバージョンの番号は、ソフトウェアのバージョンを示します。
- ドキュメントリリース日は、ドキュメントが更新されるたびに更新されます。
- ソフトウェアリリース日は、このバージョンのソフトウェアのリリース期日を表します。

更新状況、およびご使用のドキュメントが最新版かどうかは、次のサイトで確認できます。<https://softwaresupport.hp.com>

このサイトを利用するには、HP Passport への登録とサインインが必要です。HP Passport ID の登録は、次の Web サイトから行なうことができます。

<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ます。

適切な製品サポートサービスをお申し込みいただいたお客様は、更新版または最新版をご入手いただけます。詳細は、HP の営業担当にお問い合わせください。

サポート

次のHPソフトウェアサポートオンラインのWebサイトを参照してください。<https://softwaresupport.hp.com>

このサイトでは、HP のお客様窓口のほか、HP ソフトウェアが提供する製品、サービス、およびサポートに関する詳細情報をご覧いただけます。

HPソフトウェアオンラインではセルフソルブ機能を提供しています。お客様のビジネスを管理するのに必要な対話型の技術サポートツールに、素早く効率的にアクセスできます。HP ソフトウェアサポートの Web サイトでは、次のようなことができます。

- 関心のあるナレッジドキュメントの検索
- サポートケースの登録とエンハンスメント要求のトラッキング
- ソフトウェアバッチのダウンロード
- サポート契約の管理
- HPサポート窓口の検索
- 利用可能なサービスに関する情報の閲覧
- 他のソフトウェアカスタマとの意見交換
- ソフトウェアトレーニングの検索と登録

一部のサポートを除き、サポートのご利用には、HP Passport ユーザーとしてご登録の上、サインインしていただく必要があります。また、多くのサポートのご利用には、サポート契約が必要です。HP Passport ID を登録するには、次のWebサイトにアクセスしてください。

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

アクセスレベルの詳細については、次のWebサイトをご覧ください。

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

HP Software Solutions Nowは、HPSW のソリューションと統合に関するポータルWebサイトです。このサイトでは、お客様のビジネスニーズを満たすHP製品ソリューションを検索したり、HP製品間の統合に関する詳細なリストやTILプロセスのリストを閲覧することができます。このサイトのURLは <http://h20230.www2.hp.com/sc/solutions/index.jsp> です。

目次

第1章: はじめに	9
対象読者	9
統合の目的	9
サポートされるユースケース	10
ITIL プロセスの有効化	10
予定された変更の管理	11
予定外の変更の管理	11
Service Manager チケット情報の取得	11
UCMDB CI の実際のステータスの取得	11
Service Manager から UCMDB CI へのアクセス	12
コア機能	12
プッシュ	12
連携	13
ポピュレーション	13
UCMDB と Service Manager の間で CI 情報を同期する方法	13
CI 情報の使用法	14
統合の上位レベルのコンポーネント	14
統合のコンポーネント間の関係	15
UCMDB に格納される情報について	15
Service Manager に格納される情報について	16
第2章: 統合のセットアップ	17
統合の要件	17
統合の移行方法	18
Service Manager のバージョン 9.40 へのアップグレード	19
UCMDB のバージョン 10.20 へのアップグレード	19
Service Manager でカスタム CI タイプ用の RESTful API を有効にする	19
UCMDB での Service Manager 拡張汎用アダプタを使用した統合ポイントの再構成	21
UCMDB でのカスタム CI タイプの構成の更新	22
タスク 1. マッピングスクリプトの XSLT から XML および Groovy への変換	22
タスク 2. 構成ファイルの更新	27
タスク 3. CI タイプのプッシュ、ポピュレーション、および連携を有効にする	28
統合のセットアップの概要	28
HP Service Manager のセットアップ	29

統合ユーザアカウントを作成する方法	29
UCMDB 接続情報を追加する方法	30
HP Universal CMDB のセットアップ	31
UCMDB 内に統合ポイントを作成する方法	31
CI の集中管理	34
ビジュアルマッピングツール	35
UCMDB への Service Manager CI データのポピュレーション	36
UCMDB でポピュレーションジョブを定義する方法	36
UCMDB での Service Manager CI データの表示	39
CI ポピュレーションジョブのスケジュールを設定する方法	39
UCMDB CI データの Service Manager へのプッシュ	40
UCMDB でデータプッシュジョブを定義する方法	41
データプッシュジョブのスケジュールを設定する方法	44
Service Manager で UCMDB CI データを表示する方法	45
問題レコードの主要 CI の変更履歴を表示する方法	46
Service Manager チケットデータの UCMDB への連携	46
連携クエリ	47
連携の使用例	47
例 1:すべての SM インシデントチケットを連携させる	47
例 2:UCMDB ビジネスサービスCI に影響する SM インシデントレコードを連携させる	52
例 3:UCMDB CI のインシデント、変更、および問題レコードのデータを Service Manager から連携させる	60
例 4:UCMDB CI に関連する Service Manager レコードを取得する	63
第3章: マルチテナンシー (マルチカンパニー) のセットアップ	66
マルチテナンシー (マルチカンパニー) のサポート	66
UCMDB-SM 統合でのマルチテナンシーの実装	67
データ制限 SM セキュリティレイヤ	67
UCMDB に格納されるマルチテナント情報について	67
Service Manager に格納されるマルチテナント情報について	67
一意の論理名	68
会社レコードの同期	68
UCMDB 顧客 ID	70
UCMDB ユーザID とパスワード	70
会社コード	70
CI 調整ルール	71
CI および CI 関係レコードにプッシュされる会社情報	71
インシデントレコードに複製される会社情報	71

スケジュールレコード	71
テナント固有の検出イベントマネージャ (DEM) ルール	72
マルチテナンシーのユースケース	72
マルチテナンシーの要件	73
UCMDB でのマルチテナンシー統合のセットアップ	73
各テナントに個別の Data Flow Probe をインストールする方法	74
テナント固有の Data Flow Probe を起動する方法	75
テナント固有の Data Flow Probe の IP 範囲を構成する方法	76
Service Manager でのマルチテナンシー統合のセットアップ	76
スケジュールプロセスを開始する方法	77
Service Manager システム情報レコードを構成する方法	78
テナント固有 UCMDB ユーザ ID とパスワード値の追加方法	79
UCMDB 顧客 ID 値を既存の会社に追加する方法	80
Service Manager から UCMDB に既存の会社を同期する方法	80
会社情報が UCMDB にあるかどうかを確認する方法	81
既存の会社を UCMDB と再同期する方法	82
同期された会社を非アクティブにする方法	82
非アクティブな会社を再度アクティブにする方法	83
テナント固有 DEM ルールの追加方法	83
第4章: 標準およびベストプラクティス	85
UCMDB-SM 構成のベストプラクティス	85
CI 名のマッピングに関する考慮事項	85
双方向データ同期に関する推奨事項	86
プッシュのスケジュールに関する推奨事項	88
クラスタ化された環境でのプッシュ	89
専用 Web サービス	89
クラスタ構成プロセスの手順	89
Web クライアントの構成方法	89
debugnode の構成方法	90
複数の SM プロセスの接続	90
初期ロードの構成	91
シングルスレッド環境でのプッシュのパフォーマンス	91
マルチスレッドの実装	92
マルチスレッド環境でのプッシュのパフォーマンス	93
複数 SM プロセス環境でのプッシュのパフォーマンス	93
初期ロード用の SM DEM ルールを設定する方法	94
差分ロード DEM ルールを構成する方法	95
プッシュの障害検出およびリカバリ	96

Lightweight Single Sign-On (LW-SSO) 構成を有効にする方法	97
よく寄せられる質問	97
Service Manager で新しい CI が作成されるのはいつですか。	98
SM で CI が削除された理由を分析できますか。	98
UCMDB と SM の間で関係の変更を監視するにはどうすればよいですか。	99
UCMDB から SM にどのような関係がプッシュされますか。	99
ルート CI ノードとは	99
ルート関係とは	100
UCMDB-SM 統合クエリで使用される「仮想 - 複合」関係タイプとは	100
ポピュレーション機能が必要になる状況	100
物理的に削除された CI を SM から UCMDB にポピュレートできますか。	100
SM で CI 関係の停止の依存関係の設定を維持するにはどうすればよいですか。	101
XML 構成ファイルを作成する方法を教えてください。	103
[フィールドのロード] ボタンを使用して複数の管理フィールドを追加する方法を教えてください。	104
ポピュレーション構成ファイル (smPopConf.xml) の <container> 要素の目的	104
サブアイテムの削除をポピュレートできますか。	105
ポピュレーションジョブが失敗または完了した場合の処理	105
第5章: 統合のカスタマイズ	106
統合のアーキテクチャ	106
統合のクラスモデル	106
統合クエリ	106
プッシュのクエリ	107
実際のステータスのクエリ	109
連携のクエリ	109
ポピュレーションのクエリ	110
クエリの要件	111
Service Manager Web サービス	111
管理フィールド	112
Service Manager 調整ルール	116
パフォーマンスについて	117
DEM ルールの使用	117
Service Manager 検出イベントマネージャールール	117
DEM ルールが実行される条件の変更	118
DEM ルールが実行するアクションの変更	118
変更またはインシデントのレコードをオープンするカスタム JavaScript の作成	119
新規 CI を作成するためのデフォルト値	119
新規変更を作成するためのデフォルト値	119
新規インシデントを作成するためのデフォルト値	119

統合カスタマイズオプション	120
統合アダプタ構成ファイル (sm.properties) を更新する方法	121
DEM 調整ルールの追加方法	125
検出イベントマネージャールールを追加する方法	127
DEM ルール	127
一致するレコードが存在しない場合のアクション	127
レコードは存在するが、予期しないデータが検出された場合のアクション	128
レコードが削除予定である場合のアクション	129
重複ルール	129
変更およびインシデントのレコードに表示される CI 属性	130
統合によってオープンされた変更およびインシデントのレコードの検索	131
データプッシュ用の CI 属性を統合に追加する方法	131
CI 属性を UCMDB クラスモデルに追加する方法	131
CI 属性をクエリレイアウトに追加する方法	133
Service Manager CI タイプの Web サービスフィールドを追加する方法	134
SM CI タイプへの単純な属性の追加	135
CI タイプへの構造体の配列または構造体の追加	138
CI 属性を Service Manager Web サービスフィールドにマップする方法	143
データプッシュ用の CI タイプを統合に追加する方法	145
CI タイプを UCMDB クラスモデルに追加する方法	146
CI タイプを同期するためのクエリの作成方法	148
CI タイプの属性をクエリレイアウトに追加する方法	152
Service Manager に CI タイプを追加する方法	154
CI タイプの属性を Web サービスフィールドにマップする方法	156
データプッシュ用の CI 関係タイプを統合に追加する方法	160
関係タイプをプッシュするためのクエリの作成方法	161
関係タイプクエリを Service Manager Web サービスオブジェクトにマップする方法	164
関係タイプの XML 構成ファイルを作成する方法	165
統合ジョブにカスタムクエリを追加する方法	166
ポピュレーション用の CI タイプ、属性、または関係タイプを統合に追加する方法	167
CI タイプの UCMDB ID プッシュバックを有効または無効にする方法	168
連携用のサポートされる CI タイプの属性を追加する方法	170
第 6 章: トラブルシューティング	176
データプッシュの問題のトラブルシューティング	176
失敗したプッシュジョブのエラーメッセージを確認する方法	177
プッシュジョブで失敗した CI または関係のエラーメッセージを確認する方法	178
プッシュのログファイルを確認する方法	179
プッシュの一般的なエラーメッセージと解決策	180
クエリが smPushConf.xml で構成されていない	181

マッピングファイルが well formed (整形形式) ではない	182
ポピュレーションの問題のトラブルシューティング	185
失敗したポピュレーションジョブのエラーメッセージを確認する方法	185
ポピュレーションのログファイルを確認する方法	185
ポピュレーションの一般的なエラーメッセージと解決策	186
smPopConf.xml で TQL クエリが構成されていない	186
smPopConf.xml で TQL クエリのマッピングファイル名が定義されていない	189
連携の問題のトラブルシューティング	190
失敗した連携要求のエラーメッセージを確認する方法	190
連携の一般的なエラーメッセージと解決策	191
smFedConf.xml での連携 CI タイプの構成が正しくない	191
連携 TQL クエリのマッピングファイルが well formed (整形形式) ではない	193
ドキュメントのフィードバックを送信	197

第1章: はじめに

本章では、HP Universal CMDB (UCMDB) - HP Service Manager (SM) 統合 (この文書では Universal CMDB (UCMDB) 統合または UCMDB-SM 統合とも呼ばれます) の概要について説明します。

本章の内容

- 「対象読者」(9ページ)
- 「統合の目的」(9ページ)
- 「UCMDB と Service Manager の間で CI 情報を同期する方法」(13ページ)

対象読者

本ガイドは、HP Universal CMDB (UCMDB) と Service Manager (SM) システムの間の接続を確立して保守を行うシステム実装者やシステム管理者を対象としています。本ガイドでは、読者に両システムへの管理アクセス権があることを想定しています。本ガイドにある手順は、Service Manager と UCMDB のヘルプシステムにある情報と重複する場合がありますが、便宜上掲載してあります。

注: 本書では、UCMDB 10.20 以降で利用可能になった Service Manager 拡張汎用アダプタを使用して 2 つの製品間の統合をセットアップする手順について説明します。拡張汎用アダプタを使用しない場合は、特定のアダプタに基づいている以前の UCMDB-SM 統合ドキュメントを参照してください。

統合の目的

HP Universal CMDB (UCMDB) と HP Service Manager の統合により、UCMDB システムと Service Manager システムの間で、構成アイテム (CI) の実際のステータスに関する情報を共有できるようになります。CI には一般的に IT サービス、ハードウェア、およびソフトウェアが含まれます。ベストプラクティスの構成管理および変更管理 ITIL プロセスを実装する組織は、この統合を使用して、その組織がサポートすると同意した属性値を CI が実現していることを確認できます。

この統合を用いることにより、Service Manager の変更やインシデントのレコードを自動的に作成し、予期しない属性値を取る CI を更新したり、ロールバックすることができます。Service Manager では、CI の実際のステータスが CI レコードで定義された予期されるステータスと一致しない場合に実行するアクションをプログラムにより定義できます。

この統合には、CI の実際のステータス情報を表示する方法がいくつか用意されています。

- デフォルトでは、統合により、定期的な UCMDB 同期スケジュール中に Service Manager CI レコードの管理フィールドが自動的に更新されます。記また、変更やインシデントのレコードを自動的に作成

するように統合を設定することもできます。

- Service Managerユーザは、CIレコードの[実際のステータス]セクションを確認することにより、CIの現在の実際のステータスを表示できます。[実際のステータス]セクションを開くと、Service ManagerはUCMDBに対するWebサービス要求を行い、要求が返したすべてのCI属性を表示します。Service ManagerがWebサービスコールを作成するのは、ユーザがこのセクションを開いた場合のみです。
- Service Managerユーザは[UCMDBのビュー]オプションを使用することで、UCMDBシステムにログインし、現在のCI属性をUCMDBから表示できます。Service Managerユーザは、UCMDBシステムにログインするための有効なUCMDBユーザ名とパスワードを必要とします。

サポートされるユースケース

本項では、UCMDB-SM統合によってサポートされるユースケースについて説明します。サポートされるユースケースは、UCMDB-SM統合によって有効化されるビジネスのコアプロセスを提供します。

UCMDB-SM統合によってサポートされる4つの主要なビジネスユースケースがあります。これらのユースケースを次に示します。

- 予定された変更: 公式のSM変更プロセスを介してSM内に作成される変更です。
- 予定外の変更: SMで発生した、公式のSM変更プロセスに適合していない変更またはインシデントです。
- SMチケット情報の取得: UCMDBでSMチケット情報を表示する機能です。
- 実際のステータス: SMでUCMDB CI情報を表示する機能です。

すべてのユースケースは、ユーザがITIL (ITインフラストラクチャライブラリ) プロセスを実行できるようにする重要な機能を提供します。ITILプロセスは、組織に必要なIT管理方法について定義および説明するベストプラクティスのセットを示します。

ITILプロセスの有効化

UCMDBからSMへのCIプッシュをアクティブにすることによって、ユーザは、インシデント、問題、変更管理などのITILプロセスをSMで容易に実行できるようにします。

SMは、次のモジュールで、UCMDBからプッシュされたデータを利用します。

- インシデント管理: サービスデスクオペレータ (SD エージェント) が特定のインシデントレコードの「サービス」および「影響を受けるCI」を選択します。
- 問題管理: SD エージェントが特定の問題レコードの「サービス」「影響を受けるCI」および「主要CI」を選択します。
- 変更管理: SD エージェントが特定の変更レコードの「サービス」および「影響を受けるCI」を選択します。

前述の各ITILプロセスで、SMは、UCMDBで作成されたサービス、影響を受けるCI、主要CIのCI情報を利用します。

予定された変更の管理

「予定された変更」ユースケースの目的は、完全なレビューと分析の後に IT インフラストラクチャの変更を導入するための公式プロセスを IT 組織に提供することです。これは、ITIL で定義された「変更管理」プロセスに従って実行されます。

「予定された変更」は、SM ユーザが、SM 内の公式の「変更管理」プロセスモジュールを使用して開始します。この後に、実際の変更作業が実施されます。

実際の変更は、HP DDMA などのディスカバリツールによって検出され、その後で UCMDDB で更新され、関連する変更が SM にプッシュされます。ユーザが、変更を検証した後に、SM で関連する予定された変更をクローズします。

予定外の変更の管理

「予定外の変更」ユースケースの目的は、組織の公式の承認プロセスを介して、IT インフラストラクチャで発生したすべての変更をログに記録し、かつ慣例化するための公式なプロセスを IT 組織に提供することです。

「予定外の変更」は、DDMA などのディスカバリツールによって認識される変更です。この変更は最初に更新され、UCMDDB で表示可能になってから、そのデータが SM にプッシュされます。SM が変更を認識し、結果として「インシデント」または「変更」レコードが生成されます。

これらの変更は、SM の構成アイテムフォームの[変更待ち]セクションにも表示されます。変更は承認されると、SM の「過去の変更」セクションに移動されます。

Service Manager チケット情報の取得

UCMDDB 内から SM チケット情報を取得する場合、すべての HP ソフトウェアアプリケーションユーザは、UCMDDB の連携機能およびサポート API を使用して、この情報にアクセスできます。これらのアプリケーションには、Business Service Management (BSM)、Asset Manager (AM)、Operations Orchestration (OO) などがあります。

SM チケットデータにアクセスするには、UCMDDB 内から UCMDDB 連携機能を使用します。SM チケットデータには、インシデント、問題、変更のレコード、およびそれらの属性のキーセットが含まれています。

ユーザは、UCMDDB を使用して、SM から連携されるチケットデータと UCMDDB からの CI 情報を組み合わせたレポート/ビューを作成できます。

UCMDDB CI の実際のステータスの取得

「実際のステータス」の目的は、「ディスカバリツール」によって検出され、UCMDDB に入力された CI の現在のステータスを SM ユーザが把握できるようにすることです。このステータスが提供する最新情報は、SM のコンテンツおよび範囲に表示される情報と異なる場合があります。

SM に表示される CI の「実際のステータス」により、ユーザは、UCMDDB または別のデータリポジトリ内に置かれている CI の現在のステータスを確認できます。

SM ユーザは、SM 構成アイテムフォームの CI の実際のステータスセクションを表示することによって、UCMDDB または追加のデータソースから CI の実際のステータスを取得します。

Service Manager から UCMDB CI へのアクセス

SM ユーザは、SM CI レコードの [UCMDB のビュー] ボタンをクリックして、特定の CI のコンテキストで UCMDB ユーザインタフェースを開くことができます。ユーザが [UCMDB のビュー] ボタンをクリックすると、UCMDB のログイン画面が表示されます。ユーザが UCMDB のユーザ名とパスワードを入力すると、UCMDB に、特定の CI のトポロジビューと、その CI にリンクされたすべての関連 CI が表示されます。

ヒント: 統合用の Lightweight Single Sign-On (LW-SSO) を構成できるので、Service Manager Web クライアントユーザが、[UCMDB のビュー] ボタンをクリックした後で、UCMDB ログイン画面をバイパスできます。詳細については、「[Lightweight Single Sign-On \(LW-SSO\) 構成を有効にする方法](#)」(97 ページ)を参照してください。

SM システム情報レコードで UCMDB ブラウザの URL が指定されている場合、このボタンは、[UCMDB Browser で表示] ボタンに置き換わります。[UCMDB Browser で表示] ボタンをクリックすると、UCMDB Browser のログイン画面が表示されます。UCMDB Browser のユーザ名とパスワードを入力すると、UCMDB Browser のユーザインタフェースに CI が表示されます。

コア機能

本項では、統合に関連する連携、プッシュ、ポピュレーション機能の基本的な概念について説明します。

本項の内容

- [「プッシュ」](#)(12ページ)
- [「連携」](#)(13ページ)
- [「ポピュレーション」](#)(13ページ)

プッシュ

UCMDB は、Service Manager で利用可能なほとんどのタイプの CI を自動的に検出できます。この統合を使用すると、これらのタイプの CI を UCMDB から Service Manager にプッシュすることができます。

次の図は、UCMDB から Service Manager (SM) にどのようにデータがプッシュされるかを示しています。データは UCMDB から SM に物理的にプッシュ(コピー)されます。データが SM 内に物理的に配置されると、このデータは、さまざまな SM プロセスでこの情報を使用する SM ユーザによって利用されます。



注: CI タイプおよび属性のプッシュ

SM にプッシュできるのは、UCMDB に物理的に存在する情報のみです。

連携

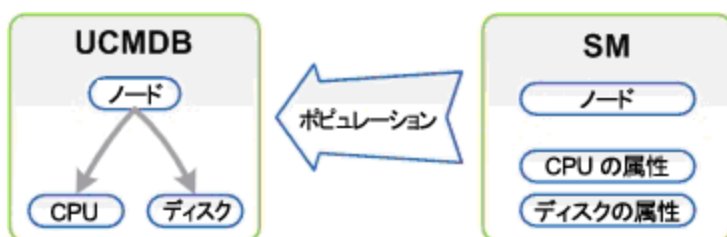
連携機能を使用して、UCMDB はさまざまなチケット情報 (インシデント、問題、変更のチケット情報など) を SM からプルします。これにより、ユーザは、UCMDB で関連するノードに接続されたチケット CI としてチケット情報を表示できます。

データが SM から UCMDB に連携 (反映または複製) される際には、データは UCMDB に物理的に存在しません。代わりに Web サービスを介して UCMDB に渡されます。

ポピュレーション

この統合を使用して、UCMDB で自動的に検出できないタイプの CI、または UCMDB システムをデプロイする前に Service Manager で作成された CI のポピュレーションを行うことができます。詳細については、「[ポピュレーション機能が必要になる状況](#)」(100ページ)を参照してください。

ポピュレーションはプッシュの逆です。次の図は、SM から UCMDB にどのようにデータがポピュレートされるかを示しています。複数の属性を持つ1つの SM CI レコードが複数の CI レコードとして UCMDB に転送されます。



UCMDB と Service Manager の間で CI 情報を同期する方法

本項では、UCMDB システムと Service Manager システムの間で CI 情報がどのように転送されるかを説明します。

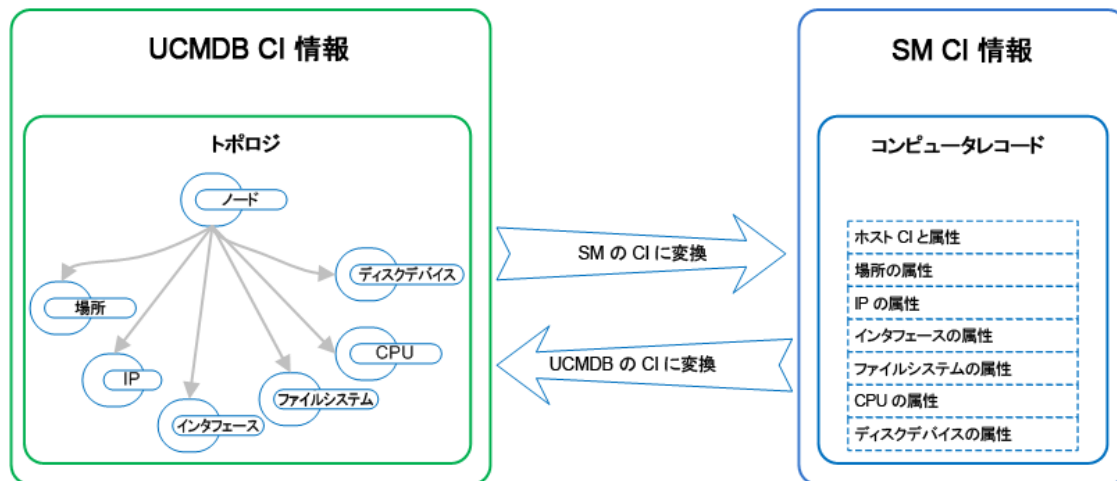
本項の内容

- 「[CI 情報の使用法](#)」(14ページ)
- 「[統合の上位レベルのコンポーネント](#)」(14ページ)
- 「[統合のコンポーネント間の関係](#)」(15ページ)
- 「[UCMDB に格納される情報について](#)」(15ページ)
- 「[Service Manager に格納される情報について](#)」(16ページ)

CI 情報の使用法

CI 情報の概念に言及するときには、UCMDB CI と Service Manager (SM) CI を明確に区別することが重要です。UCMDB モデルは、数多くの CI タイプと関係が含まれるトポロジを表します。

UCMDB のトポロジは、Service Manager では単一のエンティティとして表すことができます。UCMDB の複数の CI およびそれらの属性は、SM では単一のレコードに結合され、関連する UCMDB 属性は SM レコード内の対応する適切な属性にマップされます。



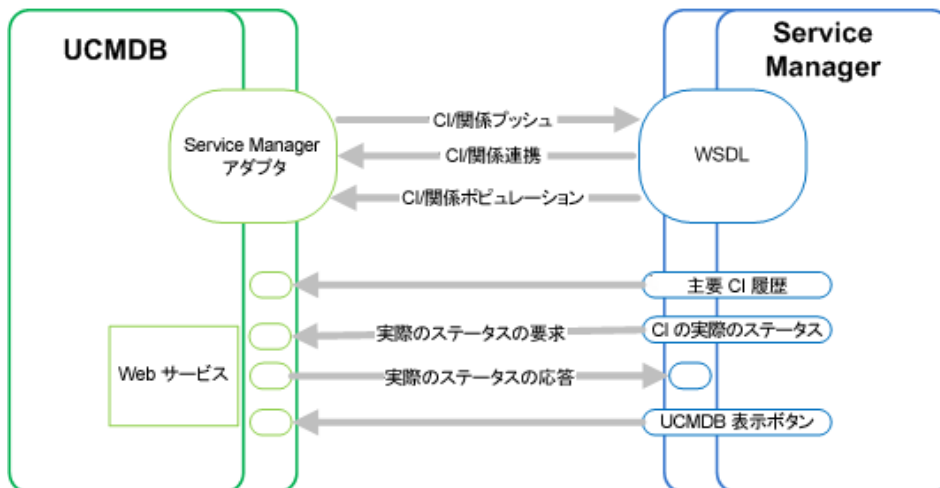
上の図は、UCMDB のトポロジモデルとコンピュータインスタンスの表現、および SM でのそれに対応する表現を示しています。SM のコンピュータ CI は、統合を介して渡されるすべての UCMDB 情報を含んでいます。

プッシュフローでは、UCMDB トポロジビュー内のノード、IP、インタフェース、場所、ファイルシステム、CPU、ディスクデバイスなどの複数の CI およびそれらの関係は、IP、MAC アドレスと場所、ファイルシステム、CPU、ディスクデバイスの属性を含む単一の SM コンピュータレコードに変換されます。

ポップレーションフローでは、逆方向に変換されます。

統合の上位レベルのコンポーネント

次の図は、統合の上位レベルのコンポーネント、および UCMDB と Service Manager の間のインタラクションを示しています。

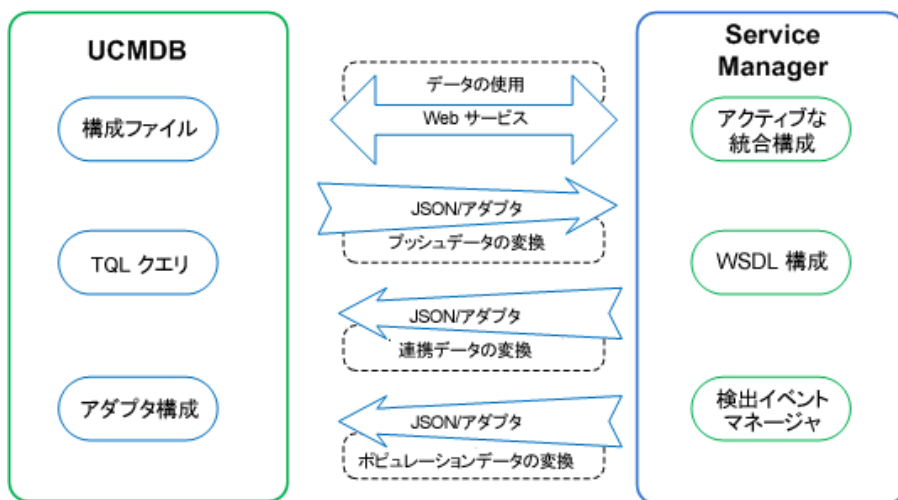


統合のコンポーネント間の関係

次の図は、UCMDB の Service Manager アダプタのコンポーネントとそれに関連付けられている Service Manager のコンポーネントの関係を示しています。

Service Manager アダプタには構成ファイルが含まれており、この構成ファイルは、データプッシュ時に UCMDB エンティティを Service Manager 内の対応するエンティティにマップするために使用され、さらにポピュレーション時に Service Manager CI を UCMDB のエンティティにマップするためにも使用されます。

この構成ファイルは、統合に関連するデータのスーパーセットを定義する UCMDB クエリを使用します。



UCMDB に格納される情報について

UCMDB システムは、CI の実際のステータスと CI 関係を CI 属性として格納します。通常、UCMDB は 1 つまたは複数の統合およびディスカバリメカニズム (フィーダ) を使用して、CI 属性値を自動的に検出します。UCMDB-SM 統合では、UCMDB システムで利用できる CI 属性のサブセットのみが使用されません。

詳細については、「[統合のカスタマイズ](#)」(106ページ)を参照してください。

Service Manager に格納される情報について

Service Manager システムは、CI の管理ステータスや予期ステータス、および CI 関係を、CI レコード内の属性値として格納します。統合に含めるには、UCMDB システムの CI 属性を Service Manager CI レコードの管理フィールドにマップする必要があります。統合を管理する Service Manager Web サービスをカスタマイズすることにより、統合に含まれる管理フィールドの追加、削除、または更新を行えます。

CI の実際のステータスが CI レコードで定義されている予期されるステータスと一致しない場合、システムはアクションを実行します。Service Manager は、このアクションを定義するルールセットに従って動作します。これらのルールは、Service Manager の検出イベントマネージャ (DEM) で定義します。次の操作を行えます。

- 実際のステータスに一覧される属性値に一致するように、CI レコードを自動的に更新する (これがデフォルトの動作です)。
- 実際のステータスと管理ステータス間の差異を確認するための変更レコードを自動的に作成する。
- 実際のステータスと管理ステータス間の差異を確認するためのインシデントレコードを自動的に作成する。

第2章: 統合のセットアップ

本番環境で統合を実装する前に、出荷時設定の統合の構成を使用してテスト環境で統合をセットアップできます。本章では、カスタマイズやマルチテナンシーの構成を含まない基本的な統合のセットアップタスクについて説明します。次のトピックについて説明します。

- 「[統合の要件](#)」(17ページ)
- 「[統合の移行方法](#)」(18ページ)
- 「[統合のセットアップの概要](#)」(28ページ)
- 「[HP Service Manager のセットアップ](#)」(29ページ)
- 「[HP Universal CMDB のセットアップ](#)」(31ページ)
- 「[UCMDB への Service Manager CI データのポピュレーション](#)」(36ページ)
- 「[UCMDB CI データの Service Manager へのプッシュ](#)」(40ページ)
- 「[Service Manager チケットデータの UCMDB への連携](#)」(46ページ)

ヒント: 本番環境での統合の実装に進む前に、以下の章で詳細情報を参照できます。

- 「[マルチテナンシー \(マルチカンパニー\) のセットアップ](#)」(66ページ)では、マルチテナンシーモードで統合をセットアップする方法について説明します。
- 「[標準およびベストプラクティス](#)」(85ページ)では、統合を実装するためのベストプラクティス、およびよく寄せられる質問について説明します。
- 「[統合のカスタマイズ](#)」(106ページ)では、ビジネスニーズに合わせて統合をカスタマイズする方法について説明します。
- 「[トラブルシューティング](#)」(176ページ)では、データのプッシュとポピュレーションの問題のトラブルシューティング方法について説明します。

統合の要件

次の表に、この統合でサポートされる製品バージョンの一覧を示します。

サポート対象の製品バージョン

Service Manager	UCMDB
9.40	10.20 以降

注: この文書では、UCMDB 10.20 以降に導入された、Service Manager 拡張汎用アダプタに基づく統合について説明します。以前のバージョンのUCMDBを使用している場合は、代わりにXSLTベースのアダプタ (Service Manager 9.x アダプタ) を使用できます。

UCMDB と Service Manager の間の統合を確立するには、次の必須コンポーネントをセットアップする必要があります。

- HP Universal CMDB インストール
ポピュレーション機能用のUCMDB Probe をまだインストールしていない場合は、これを追加します。
- HP Service Manager インストール
システム情報レコードにUCMDB URL を追加します。[「UCMDB 接続情報を追加する方法」](#)(30ページ)を参照してください。
- HP Universal CMDB システムとHP Service Manager システム間のネットワーク接続

システムのインストールと設定の方法については、UCMDB と Service Manager のマニュアルを参照してください。

統合の移行方法

UCMDB バージョン 10.20 以降、機能強化されたUCMDB統合フレームワークを基にしたService Manager 拡張汎用アダプタという名前のアダプタが導入されました。この拡張されたアダプタは、Service Manager (SM) と連携して、以前のServiceManagerAdapter9-x アダプタがサポートしていない以下の主要な機能を追加で提供します。

- ビジュアルマッピングツール: UCMDB と SM の間のフィールドマッピングを簡単に行うためのグラフィックユーザインタフェースを提供するツール。以前のアダプタを使用する場合のようにXSLTマッピングファイルを使用する必要はありません。このツールでも引き続きXMLエディタが提供されるので、マッピングファイルのコード行を直接編集することができます。
- CIタイプのカスタマイズの集中管理。詳細については、[「CIの集中管理」](#)(34ページ)を参照してください。

ビジュアルマッピングツールおよび汎用アダプタフレームワークの詳細については、Universal CMDB ヘルプセンタを参照してください。

既存のユーザがService Manager 拡張汎用アダプタを使用するには、製品システムおよび統合の構成を移行する必要があります。移行プロセスは次のとおりです。

1. [「Service Manager のバージョン 9.40 へのアップグレード」](#)(19ページ)
2. [「UCMDB のバージョン 10.20 へのアップグレード」](#)(19ページ)
3. [「Service Manager でカスタム CI タイプ用の RESTful API を有効にする」](#)(19ページ)

4. 「UCMDB での Service Manager 拡張汎用アダプタを使用した統合ポイントの再構成」(21ページ)
5. 「UCMDB でのカスタム CI タイプの構成の更新」(22ページ)

Service Manager のバージョン 9.40 へのアップグレード

UCMDB-SM 統合用の Service Manager 拡張汎用アダプタ (ServiceManagerEnhancedAdapter9-x) を使用する場合は、Service Manager をバージョン 9.40 にアップグレードする必要があります。本バージョンでは、このアダプタが機能するために必要な次の機能が導入されました。

- ブッシュ、ポピュレーション、連携のための Restful API
- デバイスタイプの取得のための Restful API
- デバイスタイプの同期のための Restful API

Service Manager をバージョン 9.40 にアップグレードする方法については、Service Manager 9.40 Installation and Upgrade Documentation Center を参照してください。

UCMDB のバージョン 10.20 へのアップグレード

Service Manager 拡張汎用アダプタ (ServiceManagerEnhancedAdapter9-x) の動作は、UCMDB 10.20 で導入された多くの新しい機能にも依存します。

UCMDB システムをバージョン 10.20 にアップグレードする方法については、『Universal CMDB 10.20 デプロイメントガイド』を参照してください。

注: 1 つまたは両方の製品システムをアップグレードした後も古いアダプタを使い続ける必要がある場合は、使用する特定のアダプタに応じて以前の UCMDB-SM 統合ドキュメントを参照してください。

Service Manager でカスタム CI タイプ用の RESTful API を有効にする

既存の UCMDB-SM 統合環境で、カスタム CI タイプを使用している場合は、Service Manager で各カスタム CI タイプの外部アクセス定義を更新して、RESTful API を有効にする必要があります。Service Manager 拡張汎用アダプタが機能するには、ucmdbIntegration RESTful API が必要です。

次の表は、各外部アクセス定義で更新する必要があるパラメータを示しています。

パラメータ	説明	値の例
RESTful 有効	RESTful API を有効にします (RESTful API を有効にするには true に設定する必要があります)	true

パラメータ	説明	値の例
リソースコレクション名	リソースコレクションの一意の名前を指定します。次のような名前の使用が推奨されます。<WS Object Name>+'s'	ucmdbATMs
リソース名	リソースの名前を指定します。次のような名前の使用が推奨されます。<WS Object Name>	ucmdbATM
固有キー	関連する device テーブルの固有キー	logical.name
クエリで返される最大レコード数	1つのクエリで返されるレコードの最大数	1000
リソースコレクションアクション - POST	リソースコレクションの POST 要求で起動されるアクション	作成
リソースアクション - POST	リソースの POST 要求で起動されるアクション	作成
リソースアクション - PUT	リソースの PUT 要求で起動されるアクション	更新
リソースアクション - DELETE	リソースの DELETE 要求で起動されるアクション	削除

これらのパラメータにアクセスするには、次の手順を実行します。

1. カスタム CI タイプの外部アクセス定義を開きます。
 - a. [カスタマイズ]>[Web サービス]>[Web サービス構成]をクリックして、外部アクセス定義フォームを開きます。
 - b. [サービス名]フィールドに、「ucmdbIntegration」と入力します。
 - c. [名前]フィールドで、カスタム CI タイプのテーブルを選択します。
 - d. [オブジェクト名]フィールドに、関連する Web サービスオブジェクト名を入力します。
 - e. [検索]をクリックします。
2. [RESTful]タブをクリックし、パラメータを更新します。

例として、次の図に、ATM Machine という名前のカスタム CI タイプの外部アクセス定義を示します。

← 戻る + 追加 🔍 検索 🔍 参照 📄 ファイル | その他 ▾

外部アクセス定義

サービス名: リリース済み:
 名前: 廃止予定:
 オブジェクト名:

許可されるアクション 式 フィールド RESTful

RESTful 有効 添付ファイル有効

リソースコレクション名:
 リソース名:
 固有キー:

クエリで返される最大レコード数:
 クエリ認証:

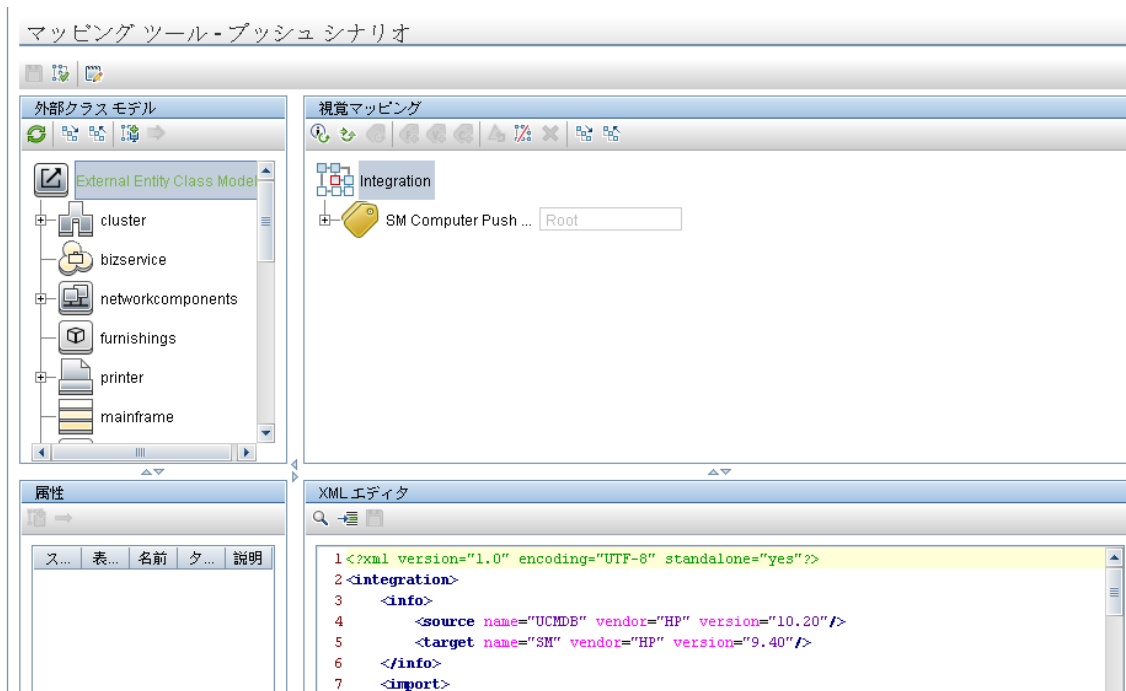
リソースコレクションアクション:
 POST:

リソースアクション:
 POST:
 PUT:
 DELETE:

UCMDB での Service Manager 拡張汎用アダプタを使用した統合ポイントの再構成

ビジュアルマッピングツールを使用するには、UCMDB 側で Service Manager 拡張汎用アダプタ (ServiceManagerEnhancedAdapater 9.x) を有効にする必要があります。これを行うには、このアダプタを使用するための統合ポイントを設定し、その統合ポイントをアクティブにします。詳細については、[「UCMDB 内に統合ポイントを作成する方法」\(31ページ\)](#)を参照してください。

このアダプタが正常に有効になったら、ビジュアルマッピングツールインタフェースを使用して、いずれかの設定済みのマッピングファイル(「SM Computer Push 2.0.xml」など)を問題なく開くことができます。次の図に、ビジュアルマッピングツールで「SM Computer Push 2.0.xml」ファイルを開いた画面を示します。



シナリオ	メモ
1対多のフィールドマッピング	<ul style="list-style-type: none"> 一方の側の1つのフィールドが他方の側の多くのフィールドにマッピングされます。 1つのフィールドの値は、多くのフィールドを基にして計算されます。
多対多のフィールドマッピング	<ul style="list-style-type: none"> 複数のフィールドが関連するCIタイプの子CIタイプとして定義されます。 1つのレコードに複数の子インスタンスがあります。
値の変換	<ul style="list-style-type: none"> 特定のアルゴリズムを基にして1つのフィールドの値が別の値に変換されます。 値の変換は、すべての種類のフィールドマッピング(1対1、1対多、多対多)で実行できます。

Service Manager 拡張汎用アダプタは、XML および Groovy マッピングスクリプトを使用します。設定済みのCIタイプ用の古いマッピングスクリプトは、デフォルトでXML および Groovy に変換されます。しかし、既存のカスタムスクリプトはユーザがXML および Groovy に変換する必要があります。

カスタムマッピングスクリプトを変換するには、最初にマッピングシナリオ(1対1、1対多、多対多、または値の変換)を識別する必要があります。その後で、使用する特定のシナリオに合わせて次のサンプルスクリプトを参照することで、スクリプトを変換できます。

ヒント: すべてのマッピングシナリオで、ビジュアルマッピングツールを使用してXMLスクリプトのスケルトンを生成できます。元のXSLTアダプタを使用する場合と比較すると、すべてのコード行を新規に記述する必要がなく、代わりにマップされるフィールドをドラッグアンドドロップしてスクリプトを生成できるので操作が簡単になっています。UCMDB統合フレームワークでのXMLおよびGroovyスクリプトの使用の詳細については、『HP Universal CMDB 10.20 開発者向け参照情報ガイド』を参照してください。

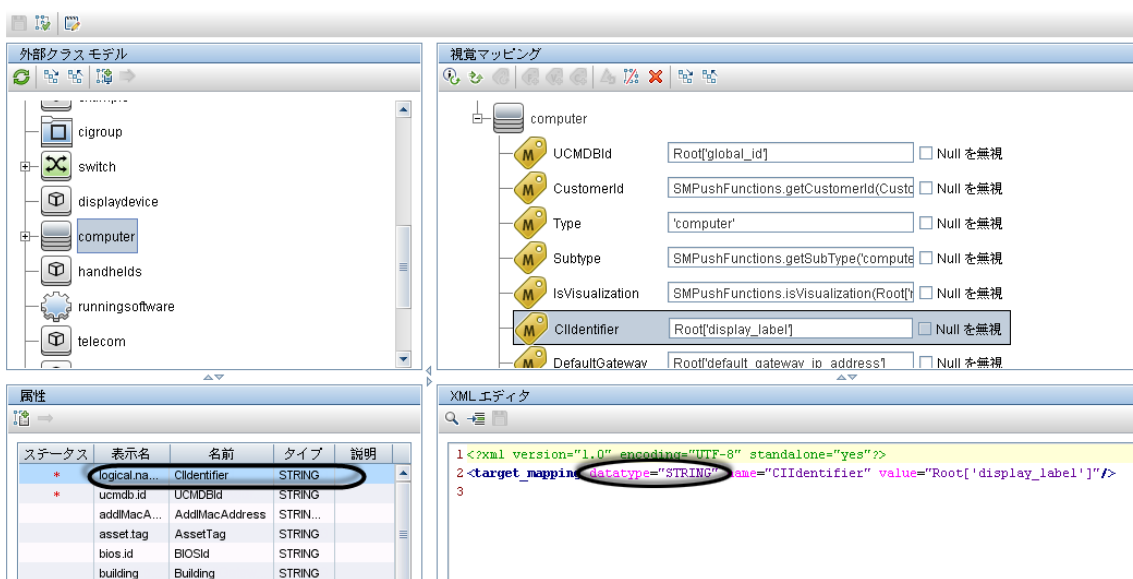
ヒント: Service Manager 拡張汎用アダプタには、参照用に使用できる4の初期設定のGroovyスクリプトが含まれています。

- SMUtils.groovy: プッシュ、ポピュレーション、連携に使用される一般的なメソッドを定義します。
- SMPushFunctions.groovy: 設定済みのプッシュマッピングスクリプトで使用されるメソッドを定義します。
- SMPopulateFunctions.groovy: 設定済みのポピュレーションマッピングスクリプトで使用されるメソッドを定義します。
- SMFederationFunctions.groovy: 設定済みの連携マッピングスクリプトで使用されるメソッドを定義します。

さらに、SMFederationConverter.groovy スクリプトが連携用の値の変換を定義します。

フィールドマッピングでは、各属性の正しいデータ型を使用する必要があります。ビジュアルマッピングツールを使用すると、属性のデータ型を簡単に判断できます。次の図は、CIIdentifier 属性のデータ型を示しています。外部クラスモデルの[属性]表示枠でデータ型を確認するか、表示枠から属性をビジュアルマッピング領域にドラッグアンドドロップすることで自動的に正しいデータ型を取得することもできます。

マッピング ツール - ブッシュシナリオ



1 対 1 のフィールド マッピング

XSLT でのサンプルスクリプト:

```
<Type>computer</Type>
```

または

```
<UCMDBid><xsl:value-of select="@id"/></UCMDBid>
```

または

```
<xsl:for-each select="@display_label">  
  <CIIdentifier><xsl:value-of select="."/></CIIdentifier>  
</xsl:for-each>
```

XML でのサンプルスクリプト (このシナリオでは Groovy は必要ありません):

```
<target_mapping datatype="STRING" name="CIIdentifier" value="Root['display_label']"  
  "/>
```

ここで、**datatype** は、Service Manager で定義された属性の正しいデータ型にする必要があります。ビジュアルマッピングツールインタフェースでデータ型を確認できます。

1 対多のフィールドマッピング

XSLTでのサンプルスクリプト:

```
<xsl:variable name="prefix" select="'Value&gt;'" />
  <xsl:variable name="suffix" select="'&lt;/Value'" />
  <Subtype>
    <xsl:choose>
      <xsl:when test="contains(@node_role,concat($prefix,'desktop',$suffix))"
      >Desktop</xsl:when>
      <xsl:when test="@os_family">
        <xsl:value-of select="@os_family" />
      </xsl:when>
      <xsl:otherwise>Server</xsl:otherwise>
    </xsl:choose>
  </Subtype>
```

XMLでのサンプルスクリプト:

```
<target_mapping datatype="STRING" name="Subtype" value="SMPushFunctions.getSubType
('computer',Root['node_role'],Root['os_family'])" />
```

Groovyでのサンプルスクリプト:

マッピングロジックを実装するためのGroovy関数を開発する必要があります。通常、対応するシステムの関連フィールドがGroovy関数のパラメータとして使用されます。

多対多のフィールドマッピング

XSLTでのサンプルスクリプト:

```
<xsl:for-each select="cpus">
  <cpu>
    <xsl:for-each select="cpu">
      <cpu>
        <CpuID><xsl:value-of select="@cpu_id" /></CpuID>
        <CpuName><xsl:value-of select="@name" /></CpuName>
        <CpuClockSpeed>
          <xsl:value-of select="@cpu_clock_speed" />
        </CpuClockSpeed>
      </cpu>
    </xsl:for-each>
  </cpu>
</xsl:for-each>
```

XMLでのサンプルスクリプト:

```
<for-each-source-entity count-index="i" source-entities="Root.Cpu">
  <target_entity name="cpu">
    <target_mapping datatype="STRING" name="CpuID" value="Root.Cpu[i]
['cpu_id']" />
    <target_mapping datatype="STRING" name="CpuName" value="Root.Cpu[i]
['name']" />
```

```
        <target_mapping datatype="STRING" name="CpuClockSpeed" value="Root.Cpu  
[i]['cpu_clock_speed']"/>  
    </target_entity>  
</for-each-source-entity>
```

ここで、<source-entities> (**Root Cpu**) は、ローカル TQL クエリ構造から取得され、<target_entity name> (**cpu**) は、XSLT ファイルの <cpu> タグから取得されます。

注: Groovy スクリプトはこのシナリオでは必要ありません。

値の変換

XSLT でのサンプルスクリプト:

```
<xsl:if test="@customer_id">  
    <xsl:variable name="ucmdbCustomerId" select="@customer_id"/>  
    <xsl:variable name="tenantMappingEntry" select="document('SM_MT_  
mapping.xml')/list['TenantMapping']/entry[@ucmdb=$ucmdbCustomerId]/>  
    <xsl:choose>  
        <xsl:when test="$tenantMappingEntry">  
            <CustomerId><xsl:value-of  
select="$tenantMappingEntry/@sm"/></CustomerId>  
        </xsl:when>  
        <xsl:otherwise>  
            <CustomerId><xsl:value-of select="@customer_id"/></CustomerId>  
        </xsl:otherwise>  
    </xsl:choose>  
</xsl:if>
```

または

```
<xsl:if test="contains(@node_role,concat($prefix,'virtualized_system',$suffix))">  
    <IsVisualization>true</IsVisualization>  
</xsl:if>
```

XML でのサンプルスクリプト:

```
<target_mapping datatype="STRING" name="CustomerId"  
value="SMPushFunctions.getCustomerId(CustomerInformation)"/>
```

または

```
<target_mapping datatype="BOOLEAN" name="IsVisualization"  
value="SMPushFunctions.isVisualization(Root['node_role'])"/>
```

注: 通常、対応するシステムの関連フィールドが Groovy 関数の 1 つのパラメータとして使用されま
す。

Groovy でのサンプルスクリプト:

```
/**  
    * Priority Mapping
```

```
    * [uCMDB value :SM value ]
    */
private static final def PriorityMapping = [
    "1_critical":"1",
    "2_high":"2",
    "3_average":"3",
    "4_low":"4"];

/**

    * Convert the Priority value from uCMDB to SM
    *
    * @param priority uCMDB Priority value
    * @return SM Priority value
    */
public static String convertPriority(String priority, DataAdapterLogger log)
{
    return convertEnumValue(priority,"Priority");
}
```

タスク2. 構成ファイルの更新

次の表のように、ファイルを変更します。

ファイル	説明
icon.properties	<p>この構成ファイルは、Service Manager デバイスタイプのマッピングツールに表示される UCMDB アイコンを定義します。SM でカスタムデバイスタイプを定義した場合は、この構成ファイルを更新して、適切なアイコンをそれらに割り当てることができます。</p> <p>詳細については、『Universal CMDB データ・フロー管理ガイド』を参照してください。</p>
sm.properties	<p>これは、統合アダプタ構成ファイルです。古い sm.properties で出荷時設定 (同時スレッドの数など)を変更した場合は、Service Manager 拡張汎用アダプタでまだ有効なそれらのオプションに合わせて、この構成ファイルを更新する必要があります。</p> <p>詳細については、「統合アダプタ構成ファイル (sm.properties) を更新する方法 (121ページ)」を参照してください。</p>
smFedConf.xml	<p>この構成ファイルは、連携に使用されるファイルで、Service Manager 拡張汎用アダプタによって導入されます。フェデレート CI のカスタマイズされたフィールドなどの連携機能用のカスタムロジックがある場合は、この構成ファイルを更新する必要があります。</p> <p>詳細については、「連携の問題のトラブルシューティング (190ページ)」を参照してください。</p>

ファイル	説明
smPopConf.xml	<p>この構成ファイルは、ポピュレーションに使用されるファイルで、古い smPopConfFile.xml 構成ファイルを置き換えます。カスタムクエリ条件などのポピュレーション機能用のカスタムロジックがある場合は、この構成ファイルを更新する必要があります。</p> <p>詳細については、以下のトピックを参照してください。</p> <p>「ポピュレーション構成ファイル (smPopConf.xml) の <container> 要素の目的」(104ページ)</p> <p>「smPopConf.xml で TQL クエリが構成されていない」(186ページ)</p>
smPushConf.xml	<p>この構成ファイルは、関係データのプッシュに使用されるファイルで、古い smSyncConfFile.xml 構成ファイルを置き換えます。</p> <p>このファイルの詳細については、以下の項目を参照してください。</p> <p>「関係タイプクエリを Service Manager Web サービスオブジェクトにマップする方法」(164ページ)</p> <p>「クエリが smPushConf.xml で構成されていない」(181ページ)</p>

タスク3. CI タイプのプッシュ、ポピュレーション、および連携を有効にする

上で説明したタスクが完了したら、統合ポイントを編集して、CI タイプのプッシュおよびポピュレーションジョブを追加し、さらにサポートされる CI タイプの連携を有効にする必要があります。詳細については、以下のトピックを参照してください。

[「UCMDB 内に統合ポイントを作成する方法」\(31ページ\)](#)

[「UCMDB でデータプッシュジョブを定義する方法」\(41ページ\)](#)

[「UCMDB でポピュレーションジョブを定義する方法」\(36ページ\)](#)

[「連携用のサポートされる CI タイプの属性を追加する方法」\(170ページ\)](#)

統合のセットアップの概要

統合には、UCMDB と Service Manager の両システムでセットアップが必要になります。

本タスクの手順は次のとおりです。

1. Service Manager システムをセットアップします。
[「HP Service Manager のセットアップ」\(29ページ\)](#) を参照してください。
2. UCMDB システムをセットアップします。
[「HP Universal CMDB のセットアップ」\(31ページ\)](#) を参照してください。

3. UCMDB ポピュレーションジョブを実行して、UCMDBにCIを同期します。
「UCMDB への Service Manager CI データのポピュレーション」(36ページ)を参照してください。
4. UCMDB データプッシュジョブを実行して、Service ManagerにCIを転送します。
「UCMDB への Service Manager CI データのポピュレーション」(36ページ)を参照してください。

HP Service Manager のセットアップ

統合をサポートするには、Service Manager システムで次のタスクを実行する必要があります。

1. Service Manager で専用の統合ユーザアカウントを作成します。
「統合ユーザアカウントを作成する方法」(29ページ)を参照してください。
2. システム情報レコードにUCMDB接続情報を追加します。
「UCMDB接続情報を追加する方法」(30ページ)を参照してください。

統合ユーザアカウントを作成する方法

この統合でUCMDBをService Managerに接続するには、管理者ユーザアカウントが必要です。このユーザアカウントは、Service ManagerとUCMDBの両方ですでに存在する必要があります。

注: 統合でService Manager拡張汎用アダプタを使用するには、統合ユーザアカウントは、Service Managerで**RESTful API** 権限を持っている必要があります。これは、Service Manager ucmbdIntegration RESTful APIを使用するために必要です。

Service Manager で専用の統合ユーザを作成するには:

1. システム管理者としてService Managerにログインします。
2. Service Manager コマンドラインで「contacts」と入力して、[Enter]キーを押します。
3. 統合ユーザアカウントの新しい連絡先レコードを作成します。
 - a. [連絡先名]フィールドに名前を入力します。「UCMDB」などです。
 - b. [追加]をクリックし、[OK]をクリックします。
4. Service Manager コマンドラインで「operator」と入力して、[Enter]キーを押します。
5. [ログイン名]フィールドに、既存のシステム管理者アカウントのユーザ名を入力し、[検索]をクリックします。
システム管理者アカウントが表示されます。
6. 既存のアカウントを基にして新しいユーザアカウントを作成します。
 - a. ログイン名を必要な統合アカウント名 (ucmdb など)に変更します。
 - b. 名前を入力します。「UCMDB」などです。

- c. [連絡先 ID]フィールドで、[フィル]アイコンをクリックし、作成した連絡先レコードを選択します。
- d. [追加]をクリックします。
- e. [セキュリティ]タブを選択し、パスワードを変更します。
- f. [起動]タブを選択し、オペレータに **RESTful API** ケイパビリティワードを追加します。
- g. [OK]をクリックします。

統合ユーザアカウントが作成されます。後で、UCMDB でこのユーザアカウント (ユーザ名/パスワード) を追加し、UCMDB で統合ポイントを作成するときに、[資格情報 ID]フィールドでこのユーザアカウントを指定する必要があります。「[UCMDB 内に統合ポイントを作成する方法](#)」(31ページ)を参照してください。

UCMDB 接続情報を追加する方法

Service Manager は、UCMDB システムから CI 属性情報を取得して、Service Manager 構成アイテムフォームの[実際のステータス]セクションに表示するために UCMDB 接続情報を必要とします。

注意: 正しい接続情報を指定しない場合は、[実際のステータス]セクションに UCMDB CI 情報ではなくエラーが表示されます。

Service Manager で UCMDB 接続情報を追加するには:

1. システム管理者として Service Manager にログインします。
2. [システム管理]>[ベースシステム構成]>[その他]>[システム情報レコード]をクリックします。
3. [アクティブ統合]タブをクリックします。
4. [HP Universal CMDB]オプションを選択します。
5. [UCMDB Web サービス URL]フィールドに、HP Universal CMDB Web サービス API への URL を入力します。URL、は次の形式です:
`http://<UCMDBサーバ名>:<ポート>/axis2/services/ucmdbSMService`

<UCMDBサーバ名>にはUCMDBサーバのホスト名、<ポート>には UCMDB サーバが使用する通信ポートを入力します。
6. [ユーザ ID]フィールドと[パスワード]フィールドに、UCMDB システム上で CI を管理するのに必要なユーザ資格情報を入力します。たとえば、出荷時設定の資格情報は「**admin/admin**」です。
7. オプションで、UCMDB Browser との統合を有効にする場合は、[UCMDB Browser URL]フィールドに、次の形式で UCMDB Browser の URL を入力します。
`http://<UCMDB Browser サーバ名>:<ポート>/ucmdb-browser`

例 : `http://myucmdbbrowserserver:8081/ucmdb-browser`

注: UCMDB Browser URL をここで指定した場合、UCMDB から同期される CI レコードで [UCMDB Browser で表示] ボタンが [UCMDB のビュー] ボタンに置き換わります。このフィールドを空白のままにした場合のみ、[UCMDB のビュー] ボタンが表示されます。

8. [保存] をクリックします。Service Manager のメッセージが表示されます。「情報レコードは更新されました。」と表示されます。
9. Service Manager システムからログアウトします。
10. Service Manager システムに管理者アカウントでログインします。

[実際のステータス] セクションと [UCMDB Browser で表示] または [UCMDB のビュー] ボタンが、UCMDB からプッシュされる CI レコードで利用可能になります。

HP Universal CMDB のセットアップ

統合をサポートするには、UCMDB システムで次のタスクを実行する必要があります。


UCMDB と Service Manager 間に統合ポイントを作成します。[「UCMDB 内に統合ポイントを作成する方法」\(31ページ\)](#)を参照してください。


UCMDB 内に統合ポイントを作成する方法

デフォルトの UCMDB のインストールには、ServiceManagerEnhancedAdapter9-x パッケージがすでに含まれています。統合パッケージを使用するには、統合の接続プロパティを一覧する統合ポイントを作成する必要があります。

注意: データポプレーションの場合、この統合は、Service Manager システム用のプローブの使用を1つのみサポートします。言い換えると、Service Manager システム用の異なる複数のプローブを使用して複数の統合ポイントを設定アップすることで異なる複数のプローブでポプレーションジョブを実行することはできません。1つの Service Manager システムに対して1つのプローブのみが許可されません。

統合ポイントを作成するには、次の手順に従います。

1. 管理者として UCMDB にログインします。
2. Service Manager で作成した統合ユーザアカウントを追加します。
 - a. [管理]>[ユーザとロール] をクリックします。
 - b. [新規ユーザの追加] アイコン  をクリックします。
 - c. [ユーザ名 およびパスワード] に、Service Manager で作成したユーザ名とパスワードを入力します。「[統合ユーザアカウントを作成する方法」\(29ページ\)](#)を参照してください。

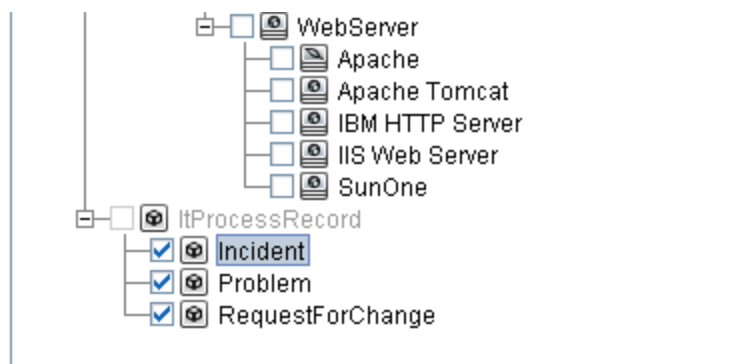
- d. [次へ]をクリックし、ロールのリストから[管理]を選択します。
 - e. [完了]をクリックします。統合ユーザアカウントが追加されます。
3. [データフロー管理]>[Integration Studio]に移動します。UCMDB に、既存の統合ポイントのリストが表示されます。
 4. [新規統合ポイント]アイコン  をクリックします。
UCMDB に、新規統合ポイントのプロパティウィンドウが表示されます。
 5. 次の表の説明に従って、統合とアダプタのプロパティフィールドに入力します。

フィールド名	必須か?	説明
統合名	はい	統合ポイントの名前 (固有キー)を入力します。「sm_integration」などです。
統合の説明	いいえ	統合ポイントの説明を入力します。
アダプタ	はい	[HP Software Products]>[Service Manager]>[ServiceManagerEnhancedAdapter9.x]を選択します。
統合はアクティブ化されています	はい	このオプションが有効な場合は、統合ポイントがアクティブなことを示します。
ホスト名/IP	はい	Service Manager サーバのホスト名または IP アドレスを入力します。 「localhost」などです。
ポート	はい	Service Manager サーバの通信ポートを入力します。 「13080」などです。

(続き)


フィールド名	必須か?	説明
URL 上書き	いいえ	<p>このフィールドの値 (存在する場合) は、上で説明した [ホスト名/IP] および [ポート] の設定を上書きします。</p> <p>次の方法の任意の組み合わせで、UCMDB を Service Manager に接続する場合に、このフィールドを使用します。</p> <ul style="list-style-type: none"> ■ HTTPS または HTTP と HTTPS の両方を介して Service Manager に接続する。 ■ 複数の Service Manager サーバードに接続する (水平スケール環境) ■ 複数のポートを介して単一の Service Manager サーバードに接続する (垂直スケール環境) <p>詳細については、「クラスタ化された環境でのプッシュ」(89ページ)を参照してください。</p> <p>このフィールドに1つまたは複数の Service Manager Web サービス URL (セミコロンで区切ります) を入力します。</p> <p>次にこのフィールドの値の例を2つ示します (各 URL は http(s)://<ホスト名>:<ポート>/SM/9/rest の形式を使用する必要があります)。</p> <ul style="list-style-type: none"> ■ https://localhost:13443/SM/9/rest ■ http://localhost:13080/SM/9/rest; https://localhost:13443/SM/9/rest; http://smfpe04:13080/SM/9/rest
資格情報 ID	はい	<p>[Generic Protocol] をクリックして、[追加] アイコンをクリックし、作成した統合ユーザアカウントを追加して、そのアカウントを選択します。このアカウントは、Service Manager と UCMDB の両方に存在している必要があります。「統合ユーザアカウントを作成する方法」(29ページ)を参照してください。</p>
Data Flow Probe	はい	<p>ポピュレーションジョブの実行に使用する Data Flow Probe の名前を選択します。UCMDB をインストールした後に統合用の Data Flow Probe をすでに追加している必要があります。「統合の要件」(17ページ)を参照してください。</p>

6. [テスト接続]をクリックして、正常な接続が確立されたことを確認します。
7. [OK]をクリックします。
統合ポイントが作成され、その詳細が表示されます。
8. [連携]タブをクリックし、次の構成を指定します。
 - a. [サポートおよび選択された CI タイプ]で、[ItProcessRecord]から必要に応じて次の CI タイプを選択します:
 - i. Incident
 - ii. Problem
 - iii. RequestForChange



- b. 選択した各 CI タイプ (Incident、Problem、RequestForChange) に対して、[CI タイプ取得モード]で[選択した CI タイプの CI を取得]を選択します。
9. [ポピュレーション]タブと[データ プッシュ]タブをクリックして、デフォルトの統合ジョブの詳細を表示します。

注: 統合ポイントを作成するときには、UCMDB によって、複数のデフォルトのポピュレーションジョブおよびデータプッシュジョブが作成されます。必要な場合は、統合ポイント用の新しいジョブを作成できます。統合ジョブの作成の詳細については、「[UCMDB でデータプッシュジョブを定義する方法](#)」(41ページ)および「[UCMDB でポピュレーションジョブを定義する方法](#)」(36ページ)を参照してください。

10. [統合ポイントの保存]アイコン  をクリックします。

CI の集中管理

Service Manager 拡張汎用アダプタを使用して、CI タイプを集中管理することができます。CI の集中管理によって、UCMDB から CI を管理できるようになり、Service Manager (SM) 側で多くのカスタマイズを行う必要がなくなります。その後で、データプッシュを使用して、UCMDB から SM に CI を同期できます。

設定済みの CI タイプの管理

Service Manager および UCMDDB には、設定済みの CI タイプおよびそれらのマッピングファイルが用意されています。設定済みの CI タイプを変更する場合は (例えば、ビジネスニーズに応じて新しい属性を追加する場合など)、ビジュアルマッピングツールを開いて、この CI タイプに新しい属性を手動で追加し、更新を適用するだけで済みます。その際に UCMDDB を閉じる必要はありません。新しい属性は、SM 側の関連する Web サービスオブジェクトに自動的に追加されます。

新しいカスタム CI タイプの管理

UCMDDB で新しい CI タイプを作成すると、フォーマットを除いて (新しい CI タイプはデフォルトのフォーマットとして **configurationItem** を使用します)、新しい CI タイプに関連するすべての SM オブジェクト (DBDICT、結合定義、Web サービス API、DEM ルールなど) が、SM 側で自動的に作成されます。

ビジネスニーズに応じて新しい CI タイプを追加する場合は、新しい CI タイプを追加し、UCMDDB で TQL クエリを作成してから、ビジュアルマッピングツールを使用して一致する SM CI タイプを作成するだけで済みます。その際に UCMDDB を閉じる必要はありません。

注: ポピュレーション機能を使用すると、SM から UCMDDB に CI を同期することができます。ポピュレーション機能を使用する必要がある状況の詳細については、「[ポピュレーション機能が必要になる状況](#)」(100ページ)を参照してください。

ビジュアルマッピングツール

Service Manager 拡張汎用アダプタに付属しているビジュアルマッピングツールには、複雑なデータプッシュやポピュレーションの構成のために値とフィールドのマッピングを構成できるグラフィックユーザインターフェースが用意されています。

次の表では、このグラフィックユーザインターフェースの表示枠について説明しています。

表示枠	説明
ローカルクエリ	CMDB に含まれるローカル TQL クエリの階層ツリー構造が表示されます。
ローカルクエリの属性	ローカルクエリの[属性]表示枠です。ローカル統合 TQL クエリの属性が表示されます。
視覚マッピング	[ローカルクエリ]表示枠および[外部クラスモデル]表示枠から選択し、ドラッグアンドドロップでアイテムのマッピングを確立できます。
XML エディタ	テキストエディタで XML マッピングファイルを編集できます。
外部クラスモデル	外部クラスモデルの階層ツリー構造が表示されます。UCMDDB-SM 統合では、この表示枠に、Service Manager (SM) 側からサポートされる CI タイプが表示されます。
外部クラスモデルの属性	外部クラスモデルの[属性]表示枠には、外部クラスモデルの属性が表示されます。UCMDDB-SM 統合では、この表示枠に、SM 側から属性が表示されます。

UCMDB への Service Manager CI データのポピュレーション

CI データを UCMDB から Service Manager にプッシュすることに加えて、この統合では、Service Manager から UCMDB への CI データ (CI および CI 関係を含む) のポピュレーションもサポートされます。Service Manager で新規 CI や新規属性値が検出されると、統合によって UCMDB 内の CI の一覧が更新されます。Service Manager から UCMDB へのデータのポピュレーションは、UCMDB の Integration Studio で定義されます。ポピュレーションジョブは手動で実行できますが、HP では、CI と CI 属性を最新の状態に保つため、これらのジョブのスケジュールを設定することを推奨します。

本タスクの手順は次のとおりです。

1. UCMDB で CI/CI 関係のポピュレーションを定義します。
「UCMDB でポピュレーションジョブを定義する方法」(36ページ)を参照してください。
2. UCMDB で転送された CI/CI 関係データを表示します。
「UCMDB での Service Manager CI データの表示」(39ページ)を参照してください。
3. CI と CI 属性を最新の状態に保つため、CI ポピュレーションジョブのスケジュールを設定します。
「CI ポピュレーションジョブのスケジュールを設定する方法」(39ページ)を参照してください。

UCMDB でポピュレーションジョブを定義する方法

CI または CI 関係ポピュレーションジョブは、特定のタイプの CI または CI 関係を Service Manager から UCMDB にコピーします。

CI または CI 関係ポピュレーションジョブを定義するには、次の手順に従います。

1. 管理者として UCMDB にログインします。
2. [データフロー管理]>[Integration Studio]に移動します。UCMDB に、既存の統合ポイントのリストが表示されます。
3. Service Manager 向けに作成した統合ポイントを開きます。
4. [ポピュレーション]タブをクリックし、次のように新しいジョブを追加します。



注: 統合ポイントを作成するときには、UCMDB によって、複数のデフォルトのポピュレーションおよびデータプッシュジョブが作成されます。次の表に、デフォルトのポピュレーションジョブとそのクエリを示します。必要な場合は、各ジョブのクエリを作成、更新、または削除できます。



CI/CI 関係ポピュレーション用のクエリ

統合ジョブ	CI/CI 関係ポピュレーション用のクエリ
SM 構成アイテムポピュレーションジョブ	<p data-bbox="607 321 1372 422">出荷時設定で、このジョブでは以下のクエリが利用可能です。このジョブは、Service Manager からの CI レコードを UCMDB に入力します。</p> <ul data-bbox="607 453 1385 810" style="list-style-type: none"><li data-bbox="607 453 1385 520">■ SM Business Service Population 2.0: bizservice タイプの CI を UCMDB に入力します。<li data-bbox="607 552 1385 619">■ SM RunningSoftware Population 2.0: RunningSoftware タイプの CI を UCMDB に入力します。<li data-bbox="607 651 1385 718">■ SM Computer Population: computer タイプの CI を UCMDB に入力します。<li data-bbox="607 749 1385 816">■ SM CLIP Down Time Population 2.0: planned outage (予定された停止) タイプの CI を UCMDB に入力します。

CI/CI 関係ポピュレーション用のクエリ (続き)

統合ジョブ	CI/CI 関係ポピュレーション用のクエリ
SM 関係ポピュレーションジョブ	<p>出荷時設定で、このジョブでは以下のクエリが定義されています。このジョブは、Service Manager からの CI 関係レコードを UCMDDB に入力します。</p> <ul style="list-style-type: none"> ■ SM Biz To Biz With Containment 2.0: 関係内で bizservice CI が別の CI を含んでいる CI 関係を UCMDDB に入力します。 ■ SM Biz To Biz With Usage 2.0: 関係内で bizservice CI が別の CI を使用する CI 関係を UCMDDB に入力します。 ■ SM Biz To Computer With Containment 2.0: 関係内で bizservice CI が computer CI を含んでいる CI 関係を UCMDDB に入力します。 ■ SM Biz To Computer With Usage 2.0: 関係内で bizservice CI が computer CI を使用する CI 関係を UCMDDB に入力します。 ■ SM Computer To Computer With Connects 2.0: 関係内で computer CI が別の CI に接続する CI 関係を UCMDDB に入力します。 ■ SM Biz To Software With Containment 2.0: 関係内で bizservice CI が software CI を含んでいる CI 関係を UCMDDB に入力します。 ■ SM Biz To Software With Usage 2.0: 関係内で bizservice CI が software CI を使用する CI 関係を UCMDDB に入力します。 ■ SM Computer Composition Software 2.0: 関係内で computer CI が software CI に接続する CI 関係を UCMDDB に入力します。 ■ SM CI Connection Down Time CI 2.0: 関係内で CI が down time CI に接続する CI 関係を UCMDDB に入力します。

- a. [新規統合ジョブ]アイコン  をクリックします。
- b. 統合ジョブの名前を入力します。「CI_Population_Job1」などです。
- c. クエリの[追加]アイコン  をクリックして既存のクエリをジョブに追加します (上の表を参照してください)。

- d. クエリの[統合ジョブで、削除されたデータを消去することを許可します]チェックボックスをオンにします。
 - e. [OK]をクリックして、ジョブを保存します。
5. ジョブを手動で実行し、統合ジョブが正常に機能するかどうかを確認します。
 - a. ジョブのすべての関連するデータを UCMDB に入力するには、 アイコンをクリックします。
 - b. ジョブの最後の実行以降の CI データの変更のみを UCMDB に入力するには、 アイコンをクリックします。
 6. ジョブの完了を待機し、ジョブが完了するまで[更新]アイコンを必要な回数だけクリックします。

注: ジョブが完了すると、ジョブのステータスが次のいずれかになります。[成功]、[エラーで終了]、[失敗]。
 7. [統計情報]タブをクリックして、結果を表示します。ジョブが失敗した場合、[クエリのステータス]タブと[ジョブ エラー]タブをクリックして、詳細情報を表示します。詳細については、「[ポピュレーションの問題のトラブルシューティング](#)」(185ページ)を参照してください。
 8. [OK]をクリックします。

ジョブが正常に完了した場合は、転送された CI データを UCMDB で表示して、ジョブを自動的に実行できるようにスケジュールを設定できます。

UCMDB での Service Manager CI データの表示

ポピュレーションジョブが正常に完了したら、UCMDB で Service Manager CI レコードを検索し、属性が正しく入力されていることを確認できます。

[Service Manager CI 識別子]フィールドは、UCMDB の[構成アイテムのプロパティ]表示枠の[名前]フィールドに入力されます。

注: CI タイプの属性マッピングの全体を表示するには、ポピュレーション XML ファイル (**SM Business Service Population.xml** など)を開きます。このファイルで UCMDB 属性のフィールド名およびマップされた Service Manager Web サービスフィールドキャプション名が定義されます。詳細については、「[統合のカスタマイズ](#)」(106ページ)を参照してください。

CI ポピュレーションジョブのスケジュールを設定する方法

Service Manager フィーダのディスクバリメンテナンスケジュールと一致するように、CI ポピュレーションジョブのスケジュールを設定することができます。たとえば、Service Manager フィーダが日次スケジュールで CI データの更新を送信する場合は、ポピュレーションジョブも日次スケジュールで実行する必要があります。一致するスケジュールを用いることにより、UCMDB システムが常に最新の CI データを保持できるようになります。

本タスクの手順は次のとおりです。

1. 管理者として UCMDB にログインします。
2. **[データフロー管理]**>**[Integration Studio]**に移動します。UCMDB に、統合ポイントのリストが表示されます。
3. この Service Manager 用の統合ポイントを開きます。
4. **[ポピュレーション]**タブをクリックし、リストからポピュレーションジョブを選択します。
5. **[統合ジョブの編集]**アイコンをクリックします。
6. **[スケジューラを有効にする]**オプションを選択します。
7. 使用するスケジュールオプションを選択します。たとえば、**[繰り返し間隔: 日]**および**[終了: 実行しない]**を選択します。
8. タイムゾーンを選択します。
9. **[OK]**をクリックします。

UCMDB CI データの Service Manager へのプッシュ

統合は、Service Manager システムに CI を入力するため、UCMDB から Service Manager に CI の1回限りの転送を必要とします。UCMDB が新規 CI や新規属性値を検出すると、統合によって Service Manager 内の CI の一覧が更新されます。統合では、UCMDB システムのデータプッシュジョブを使用して、CI データのプッシュが行われます。HP では、CI と CI 属性を最新に保つため、これらのジョブのスケジュールを設定することを推奨します。

CI の集中管理

Service Manager 拡張汎用アダプタを使用して、CI タイプを集中管理することができます。UCMDB で新しい CI タイプを作成すると、フォーマット (新しい CI タイプはデフォルトのフォーマットとして **configurationItemNode** を使用します)を除いて、新しい CI タイプに関連するすべての SM オブジェクト (DBDICT、結合定義、Web サービス API、DEM ルールなど) が、SM 側で自動的に作成されます。CI を集中管理すると、手動で操作しなくても UCMDB で CI を管理できます。


本タスクの手順は次のとおりです。

1. CI/CI 関係のデータプッシュジョブを定義します。
「[UCMDB でデータプッシュジョブを定義する方法](#)」(41ページ)を参照してください。
2. UCMDB からプッシュされた CI/CI 関係データを表示します。
「[Service Manager で UCMDB CI データを表示する方法](#)」(45ページ)を参照してください。
3. CI/CI 関係データを最新の状態に保つため、データプッシュジョブのスケジュールを設定します。
「[データプッシュジョブのスケジュールを設定する方法](#)」(44ページ)を参照してください。

UCMDB でデータプッシュジョブを定義する方法

データプッシュジョブは、UCMDB システムから Service Manager システムに CI または CI 関係レコードをコピーします。

CI または CI 関係プッシュジョブを定義するには、次の手順を実行します。

1. 管理者として UCMDB にログインします。
2. [データフロー管理]>[Integration Studio]に移動します。UCMDB に、既存の統合ポイントのリストが表示されます。
3. Service Manager 向けに作成した統合ポイントを選択します。「sm_integration」などです。
4. [データプッシュ]タブを選択します。
5. 新しいデータプッシュジョブを追加するには、次の手順を実行します。
 - a. [新規統合ジョブ]アイコン  をクリックします。

注: 統合ポイントを作成するときには、UCMDB によって、デフォルトのデータプッシュジョブが作成されます。次の表に、デフォルトのデータプッシュジョブとそのクエリのリストを示します。必要な場合は、プッシュジョブのクエリを作成、更新、または削除できます。これらの出荷時設定のジョブのクエリにアクセスするには、[モデリング]>[モデリングスタジオ]>[リソース]に移動し、リソースタイプの[クエリ]を選択して、[Root]>[Integration]>[Push]に移動します。データプッシュクエリのカスタマイズの詳細については、「[CI タイプを同期するためのクエリの作成方法](#)」(148ページ)を参照してください。

CI/CI 関係プッシュ用のクエリ


統合ジョブ	クエリ
統合ジョブ	クエリ

CI/CI 関係プッシュ用のクエリ (続き)



統合ジョブ	クエリ
SM プッシュジョブ	<p>出荷時設定で、このジョブでは以下のクエリが利用可能です。このジョブは、CI レコードを UCMDB から Service Manager にプッシュします。</p> <ul style="list-style-type: none"> ■ SM Mainframe Push 2.0: メインフレームタイプの CI をプッシュします。 ■ SM Service Element Push 2.0: ビジネスアプリケーションとインフラストラクチャサービスタイプの CI をプッシュします。 ■ SM Network Component Push 2.0: ネットワークコンポーネントタイプの CI をプッシュします。 ■ SM Running Software Push 2.0: 実行中のソフトウェアタイプの CI をプッシュします。 ■ SM Business Service Push 2.0: ビジネスサービスタイプの CI をプッシュします。 ■ SM Computer Push 2.0: コンピュータタイプの CI をプッシュします。 ■ SM Storage Push 2.0: ストレージタイプの CI をプッシュします。 ■ SM Switch Push 2.0: スイッチタイプの CI をプッシュします。 ■ SM Net Printer Push 2.0: ネットプリンタタイプの CI をプッシュします。 ■ SM Cluster Push 2.0: クラスタタイプの CI をプッシュします。 ■ SM Mobile Device Push 2.0: モバイルデバイスタイプの CI をプッシュします。 ■ SM Local Printer Push 2.0: ローカルプリンタタイプの CI をプッシュします。 <p>このジョブでは出荷時設定の以下のクエリが利用可能です。このジョブは、CI 関係レコードを UCMDB から Service Manager にプッシュします。</p> <ul style="list-style-type: none"> ■ SM Layer2 Topology Relations Push 2.0: ノード間の複合 CI 関係をプッシュします。 ■ SM Business Service Relations Push 2.0: アップストリーム CI タイプがビジネスサービスである CI 関係をプッシュします。 ■ SM CRG Relations Push 2.0: アップストリーム CI タイプがクラスタである CI 関係をプッシュします。


CI/CI 関係プッシュ用のクエリ (続き)

統合ジョブ	クエリ
	<ul style="list-style-type: none"> ■ SM Node Relations Push 2.0: アップストリーム CI タイプがノードである直接 CI 関係をプッシュします。

- b. [名前]にジョブの一意な名前を入力します。「CI_Push_Job1」などです。
 - c. クエリの[追加]アイコン  をクリックして既存のクエリをジョブに追加します。
 - d. 各クエリの[統合ジョブで、削除されたデータを消去することを許可します]オプションを選択します。
 - e. [OK]をクリックして、ジョブを保存します。
6. ジョブを手動で実行し、統合ジョブが正常に機能するかどうかを確認します。

注意: UCMDDB システムに大量の CI データがあり、CI/CI 関係データを初めて Service Manager にプッシュする場合は、各検出イベントマネージャールール定義の[一致するレコードが存在しない場合のアクション]で[変更のオープン]または[インシデントのオープン]の代わりに[レコードの追加]オプションを選択することをお勧めします。そのようにしないと、不要なパフォーマンスの問題の原因になることがあります。詳細については、「[検出イベントマネージャールールを追加する方法](#)」(127ページ)を参照してください。

- a. ジョブのすべての関連するデータをプッシュするには、 アイコンをクリックします。
- b. ジョブの最後の実行以降に変更されたデータのみをプッシュするには、 アイコンをクリックします。

ヒント: [選択したジョブを停止]アイコン  をクリックすると、実行中のプッシュジョブを停止することができます。

7. ジョブの完了を待機し、ジョブが完了するまで[更新]アイコンを必要な回数だけクリックします。

注: ジョブが完了すると、結果に応じてジョブのステータスが次のいずれかになります。[正常に完了]、[完了]、[失敗]。


8. [統計情報]タブをクリックして結果を表示します。エラーが発生した場合は、[クエリのステータス]タブおよび[ジョブ エラー]タブをクリックして詳細情報を表示します。詳細については、「[データプッシュの問題のトラブルシューティング](#)」(176ページ)を参照してください。
9. [OK]をクリックします。

ジョブが正常に完了した場合は、Service Manager で UCMDDB CI データを表示して、ジョブを自動的に実行できるようにスケジュールを設定できます。

データプッシュジョブのスケジュールを設定する方法

Service Manager フィーダのディスクバリエーションと一致するように、データプッシュジョブのスケジュールを設定することが重要です。たとえば、Service Manager フィーダが日次スケジュールで CI データの更新を送信する場合は、データプッシュジョブも日次スケジュールで実行する必要があります。一致するスケジュールを用いることにより、Service Manager システムが常に最新の CI データを保持できるようになります。

UCMDB では、データプッシュジョブから直接更新のスケジュールを設定できます。本タスクの手順は次のとおりです。

1. 管理者として UCMDB にログインします。
2. [データフロー管理]>[Integration Studio]に移動します。UCMDB に、統合ポイントのリストが表示されます。
3. Service Manager 向けに作成した統合ポイントを選択します。「SM Integration」などです。
4. [データプッシュ]タブを選択します。
5. プッシュジョブを選択します。[SM Configuration Item Push Job 2.0]などです。
6. [統合ジョブの編集]アイコン  をクリックします。

ヒント: UCMDB では、2つのタイプのデータプッシュ用に2つの異なるスケジュールを定義できます。デルタ同期と完全同期です。プッシュのスケジュール設定に関する推奨事項については、「[プッシュのスケジュールに関する推奨事項](#)」(88ページ)を参照してください。

7. デルタ同期のスケジュールを定義します。
 - a. [デルタ同期]タブをクリックします。
 - b. [スケジュールを有効にする]オプションを選択します。
 - c. 使用するスケジュールオプションを選択します。

スケジュールの定義


デルタ同期 | 完全同期

スケジュールを有効にする

繰り返す: 1回
間隔
日付
週ごと
月ごと
年ごと
Cron

開始: 終了: 実行... 次まで

繰り返す: 日 月 火 水 木 金 土

タイムゾーン:  サーバ時間:

8. [完全同期]タブをクリックし、使用するスケジュールオプションを選択します。

9. **[OK]**をクリックして、データプッシュジョブを保存します。
10. 統合ポイントの残りのデータプッシュジョブについて、**手順 6** から**手順 9**を繰り返します。
11. 統合ポイントを保存します。

Service Manager で UCMDB CI データを表示する方法

プッシュジョブが正常に完了したら、Service Manager でプッシュされた CI/CI 関係データを検索して確認することができます。

UCMDB からプッシュされる CI レコードに含まれる**[UCMDB のビュー]**または**[UCMDB Browser で表示]**ボタンを使用して、UCMDB または UCMDB Browser にアクセスして CI 情報を表示できます。

注:

- SM のシステム情報レコードで UCMDB Browser の URL を指定した場合は**[UCMDB Browser で表示]**ボタンが表示され、それ以外の場合は**[UCMDB のビュー]**ボタンが表示されます。
- UCMDB Browser は、UCMDB の構成情報に簡単にアクセスできるように設計されたシンプルな UI です。このツールで、構成関連データを検索して利用できます。これは、UCMDB のオプションのアドオンです。詳細については、UCMDB Browser のマニュアルを参照してください。

Service Manager で UCMDB CI データを表示するには:

1. システム管理者として Service Manager にログインします。
2. **[構成管理]**>**[CI の検索]**をクリックします。
3. UCMDB からプッシュされる CI レコードを開きます。
4. **[UCMDB のビュー]**ボタンが利用可能な場合は、UCMDB で CI レコードを表示します。
 - a. **[UCMDB のビュー]**ボタンをクリックします。UCMDB のログイン画面が表示されます。
 - b. UCMDB のユーザ名とパスワードを入力してログインします。

UCMDB で CI レコードが開きます。レコードのプロパティを表示できます。

注: 統合用の Lightweight Single Sign-On (LW-SSO) を有効にして、Service Manager Web クライアントユーザが UCMDB ログイン画面をバイパスできるようにすることができます。詳細については、「[Lightweight Single Sign-On \(LW-SSO\) 構成を有効にする方法](#)」(97ページ)を参照してください。

5. **[UCMDB Browser で表示]**ボタンが利用可能な場合は、UCMDB Browser で CI レコードを表示します。
 - a. **[UCMDB Browser で表示]**ボタンをクリックします。UCMDB Browser のログイン画面が表示されます。

- b. UCMDB Browser のユーザ名とパスワードを入力してログインします。UCMDB Browser で CI レコードが開きます。レコードのプロパティやその他の情報を表示できます。
6. **[実際のステータス]** セクションを開きます。Service Manager が Web サービス要求を UCMDB に送信し、要求で返されたすべての CI 属性を表示します。

注: Web サービス要求では、Service Manager のシステム情報レコードで定義された UCMDB Web サービスの URL とアカウント (admin/admin など) を使用します。[「UCMDB 接続情報を追加する方法」\(30ページ\)](#)を参照してください。

問題レコードの主要 CI の変更履歴を表示する方法

UCMDB Browser と統合されると、Service Manager は、主要 CI が UCMDB から同期されている問題レコードに **[UCMDB の主要 CI 履歴]** セクションを表示します。その主要 CI に関する CI の変更を表示して原因を調べることができます。

UCMDB Browser で統合を有効にする方法の詳細については、[「UCMDB 接続情報を追加する方法」\(30ページ\)](#)を参照してください。

主要 CI の変更履歴を表示するには、次の手順に従います。

1. Service Manager にログインします。
2. **[問題管理]** に移動し、検索を実行して、主要 CI が UCMDB から同期されている問題レコードを開きます。
3. **[UCMDB の主要 CI 履歴]** タブをクリックします。

Service Manager チケットデータの UCMDB への連携

連携では、データを Service Manager (SM) から UCMDB に物理的にコピーしません。UCMDB での表示のために SM データを取得するだけです。出荷時設定では、UCMDB-SM 統合は、Incident、Problem、および RequestForChange 外部 CI タイプの UCMDB への連携をサポートします。統合ポイントを作成するときに、連携用にこれらの CI タイプを有効にした場合は、UCMDB でインシデント、問題、変更レコードデータを SM から取得できます。

出荷時設定で、各 CI タイプ (インシデント、問題、または変更) は、Service Manager でのそれらの属性のサブセットの連携をサポートしますが、統合をカスタマイズすれば、連携させる属性を増やすことができます。

本項の内容

- [「連携クエリ」\(47ページ\)](#)
- [「連携の使用例」\(47ページ\)](#)

- [「連携用のサポートされる CI タイプの属性を追加する方法」\(170ページ\)](#)

連携クエリ

連携では、クエリを使用して Service Manager から取得するデータを決定します。特定のチケットデータを Service Manager から取得するには、最初に TQL クエリを作成する必要があります。

ServiceManagerEnhancedGenericAdapter9-x が提供する多くのサンプル連携クエリを参考として使用できます。それらは、UCMDB の次のパスで使用できます。[モデリング]>[モデリングスタジオ]>[リソース]>[Integration]>[Service Manager]>[Federation]

連携の使用例

連携機能はさまざまな方法で使用できます。以下はこの機能の使用例です。

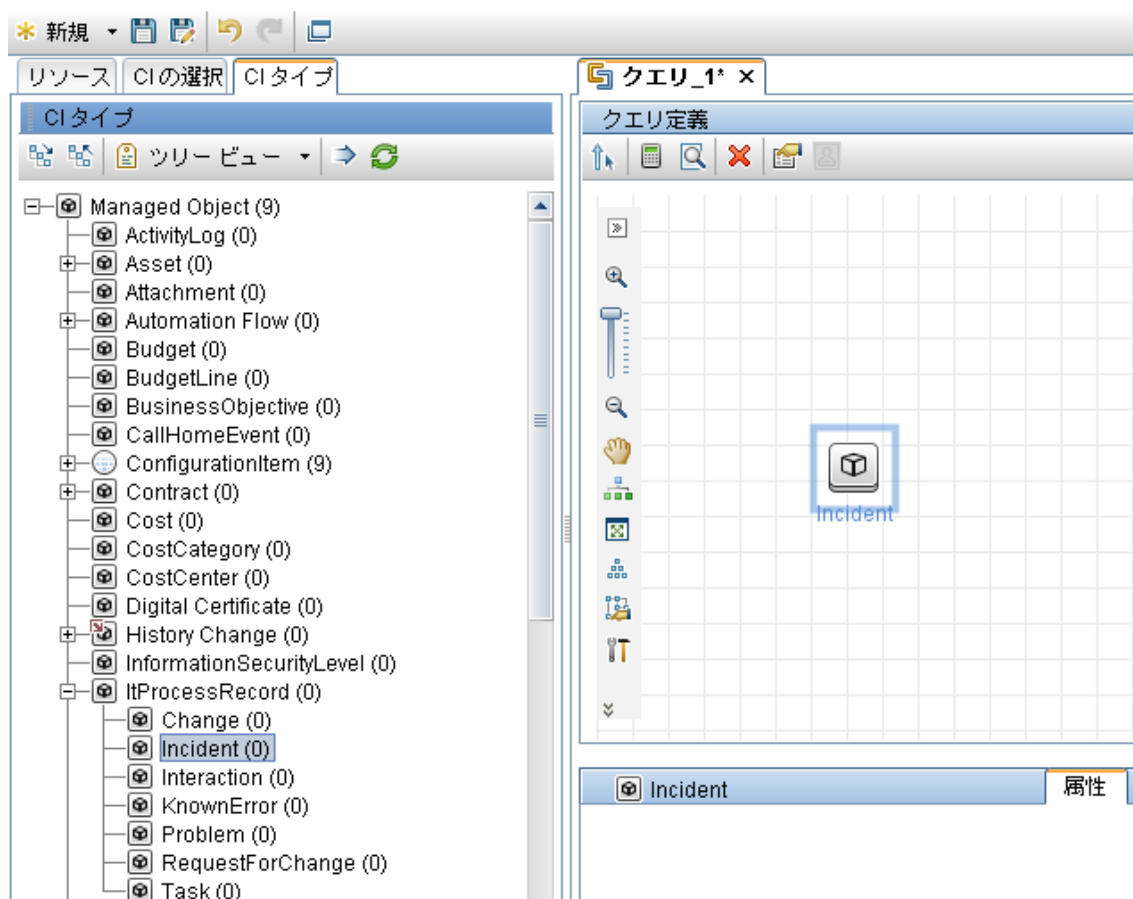
本項では、次の4つの例について説明します。

- [「例 1:すべての SM インシデントチケットを連携させる」\(47ページ\)](#)
- [「例 2:UCMDB ビジネスサービス CI に影響する SM インシデントレコードを連携させる」\(52ページ\)](#)
- [「例 3:UCMDB CI のインシデント、変更、および問題レコードのデータを Service Manager から連携させる」\(60ページ\)](#)
- [「例 4:UCMDB CI に関連する Service Manager レコードを取得する」\(63ページ\)](#)

例 1:すべての SM インシデント チケットを連携させる

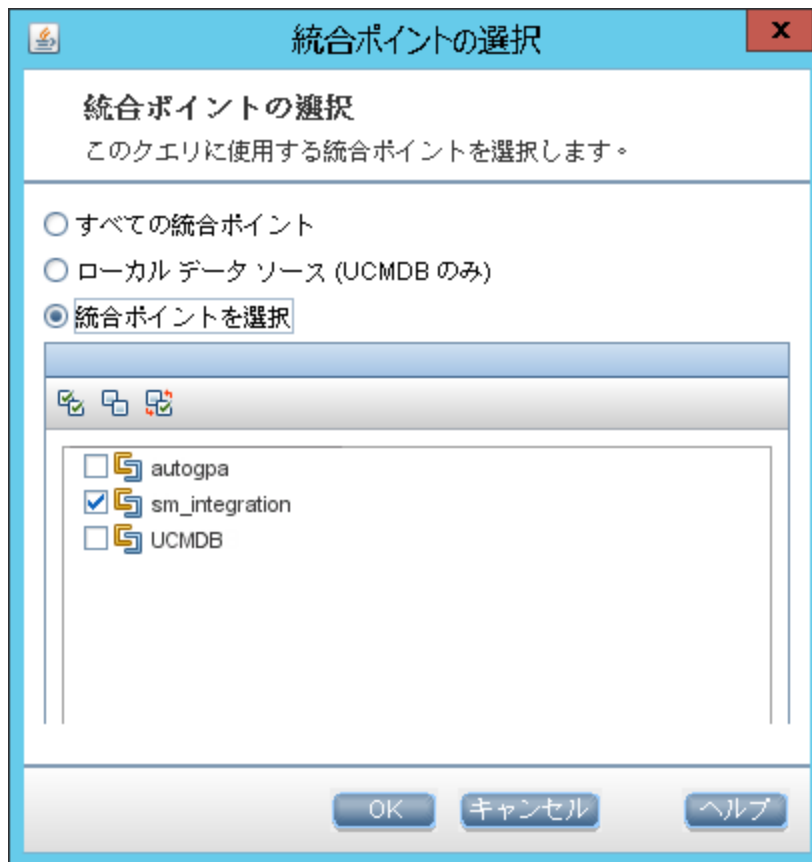
この例では、Service Manager に存在するすべてのインシデントレコードの情報を取得する方法について説明します。



1. 管理者として UCMDB にログインします。
2. [モデリング]>[モデリングスタジオ]>[リソース]に移動します。
3. [リソースタイプ]で、リストから[クエリ]を選択します。
4. [新規]>[クエリ]をクリックします。
5. [CI タイプ]タブで、[ItProcessRecord]>[Incident]に移動し、右側のクエリ表示枠にそれをドラッグします。

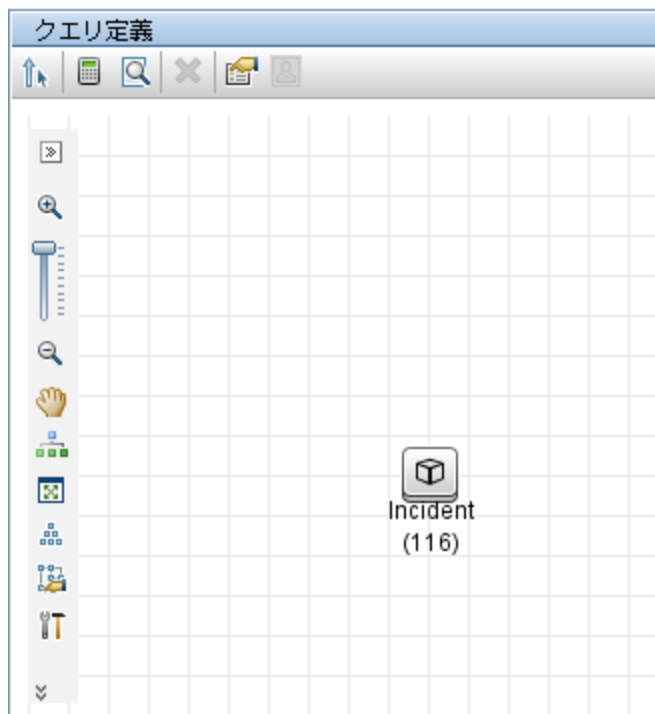


6. インシデントクエリノードのデータソースとして Service Manager を指定します。
 - a. インシデントクエリノードを選択し、右側の表示枠で[データソース]タブをクリックし、[編集]をクリックします。
 - b. [統合ポイントを選択]オプションを選択し、統合ポイント名 (sm_integration など) を選択しま

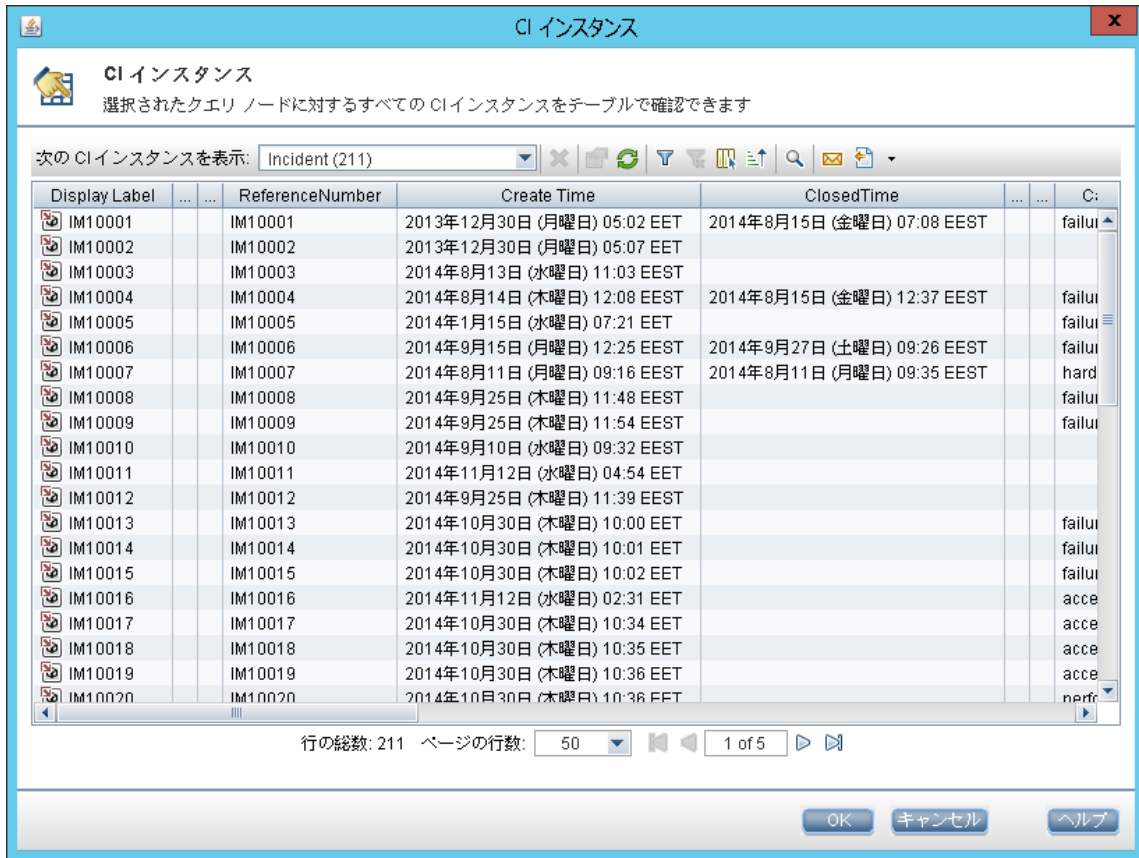
す。[OK]をクリックします。



7. [保存]アイコン  をクリックし、クエリ名を入力して、クエリの保存場所 ([ルート]フォルダなど) を選択します。
8. インシデントクエリノードを選択し、[クエリ結果数を計算する]アイコン  をクリックします。UCMDB がクエリ結果数を返します。たとえば、次の図は、Service Manager に合計で 116 のインシデントレコードがあることを示しています。



9. インシデントクエリノードを右クリックし、[要素インスタンスの表示]を選択します。UCMDB に、Service Manager に存在するすべてのインシデントレコードのリストが表示されます。



10. リストからインシデントレコードを選択し、[プロパティ]アイコン  をクリックして詳細を表示します。

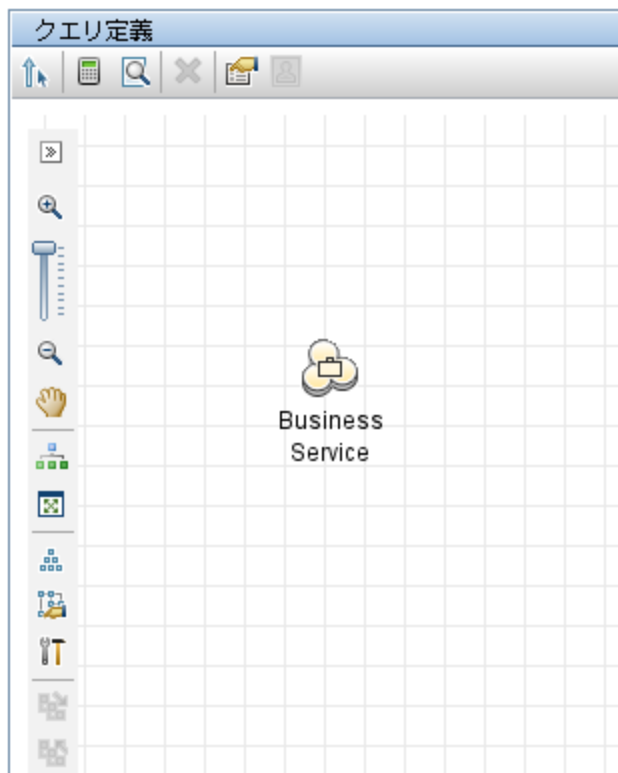


例 2 :UCMDB ビジネスサービスCI に影響する SM インシデントレコードを連携させる

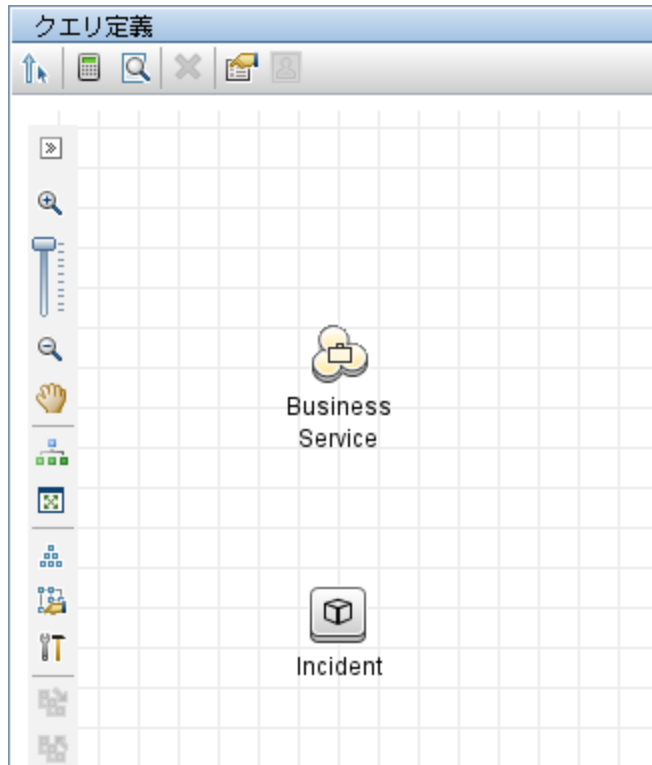
次の例では、[影響を受けるサービス]または[影響を受けるCI]フィールドにUCMDBのビジネスサービスCIが含まれているService Manager インシデントレコードのリストを連携させる方法について説明します。


この例では、SMのIM10005に**bizservice1**という名前のUCMDBからプッシュされた影響を受けるサービスがあります。

1. 管理者としてUCMDBにログインします。
2. [モデリング]>[モデリングスタジオ]>[リソース]に移動します。
3. [リソースタイプ]で、リストから[クエリ]を選択します。
4. [新規]>[クエリ]をクリックします。
5. [CIタイプ]タブで、[ConfigurationItem]>[BusinessElement]>[Service]>[BusinessService]に移動し、右側のクエリ表示枠にそれをドラッグします。



6. [ItProcessRecord]>[Incident]に移動し、クエリ表示枠にアイコンをドラッグします。

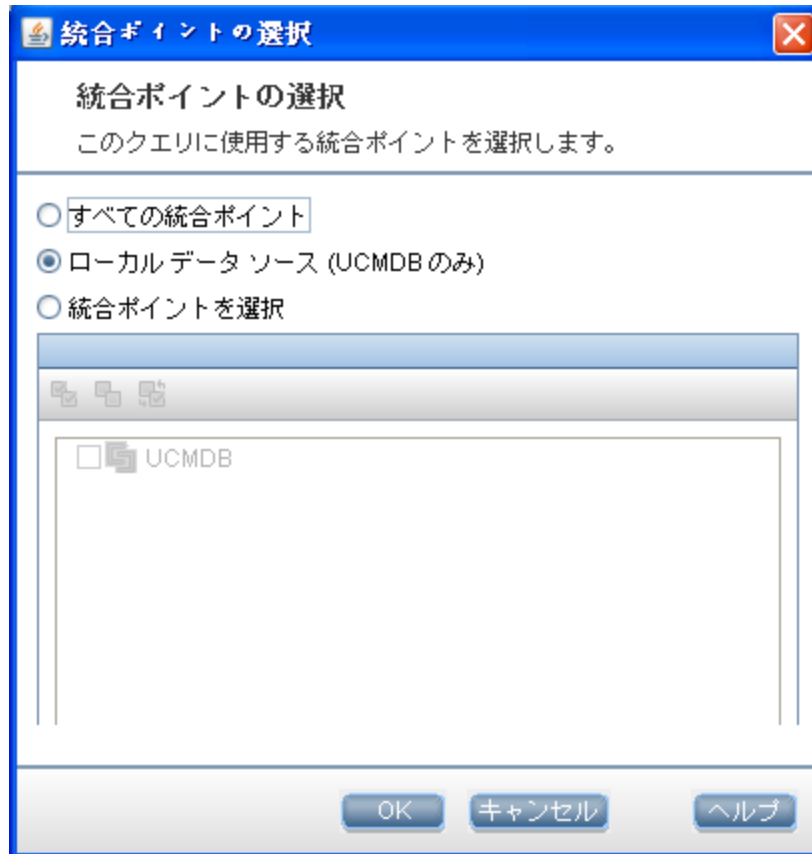


7. [関係を作成]アイコン  をクリックします。
8. インシデントクエリノードを選択し、このノードから BusinessService ノードに矢印をドラッグし、ノード間の通常関係を作成します。
 - a. [通常関係]を選択して、[OK]をクリックします。
 - b. [メンバシップ]を選択し、オプションで関係名 (**Membership** など)を入力します。[OK]をクリック

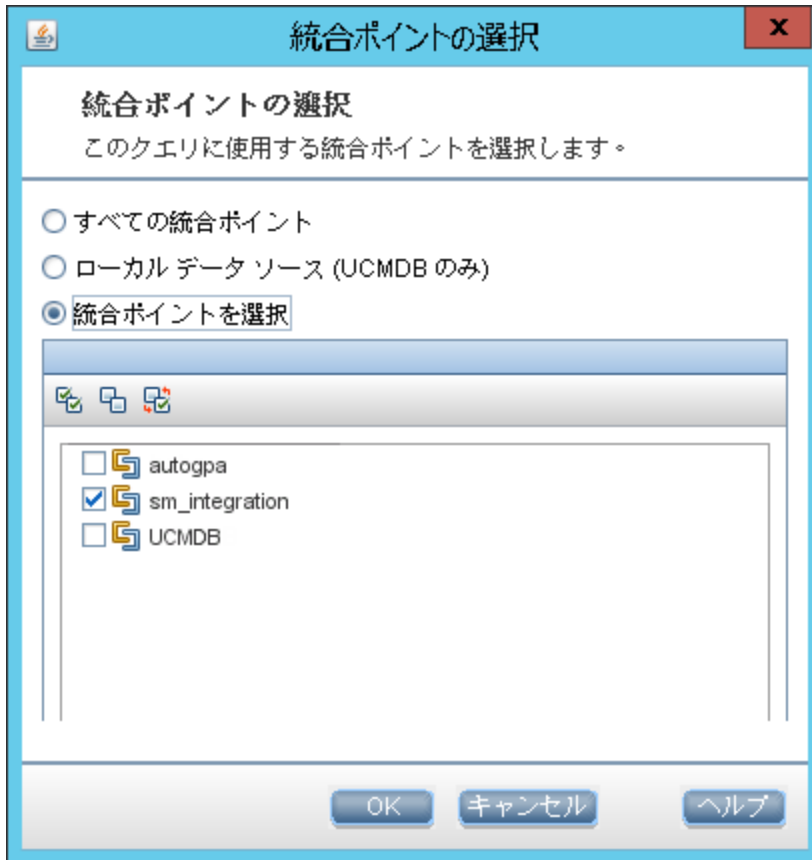
くします。




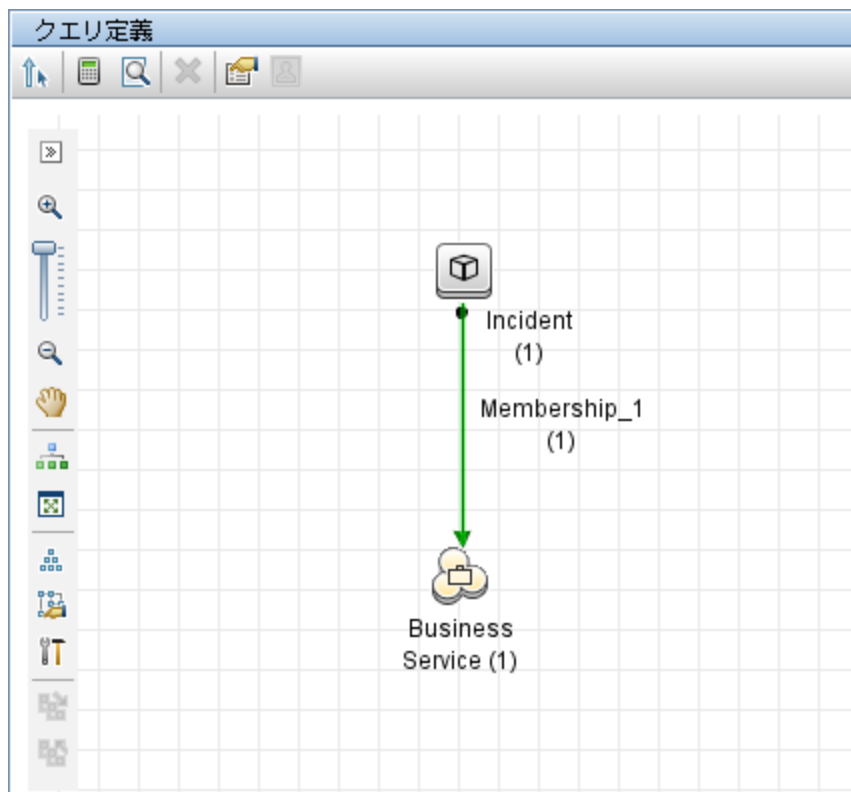
9. BusinessService クエリノードのデータソースとして UCMDDB を指定します。これを行うには、次の手順を実行します。
 - a. **BusinessService** クエリノードを選択します。
 - b. 右側の表示枠で[データソース]タブをクリックし、[編集]をクリックします。
 - c. [ローカル データ ソース (UCMDDB のみ)]オプションが選択されていることを確認します。




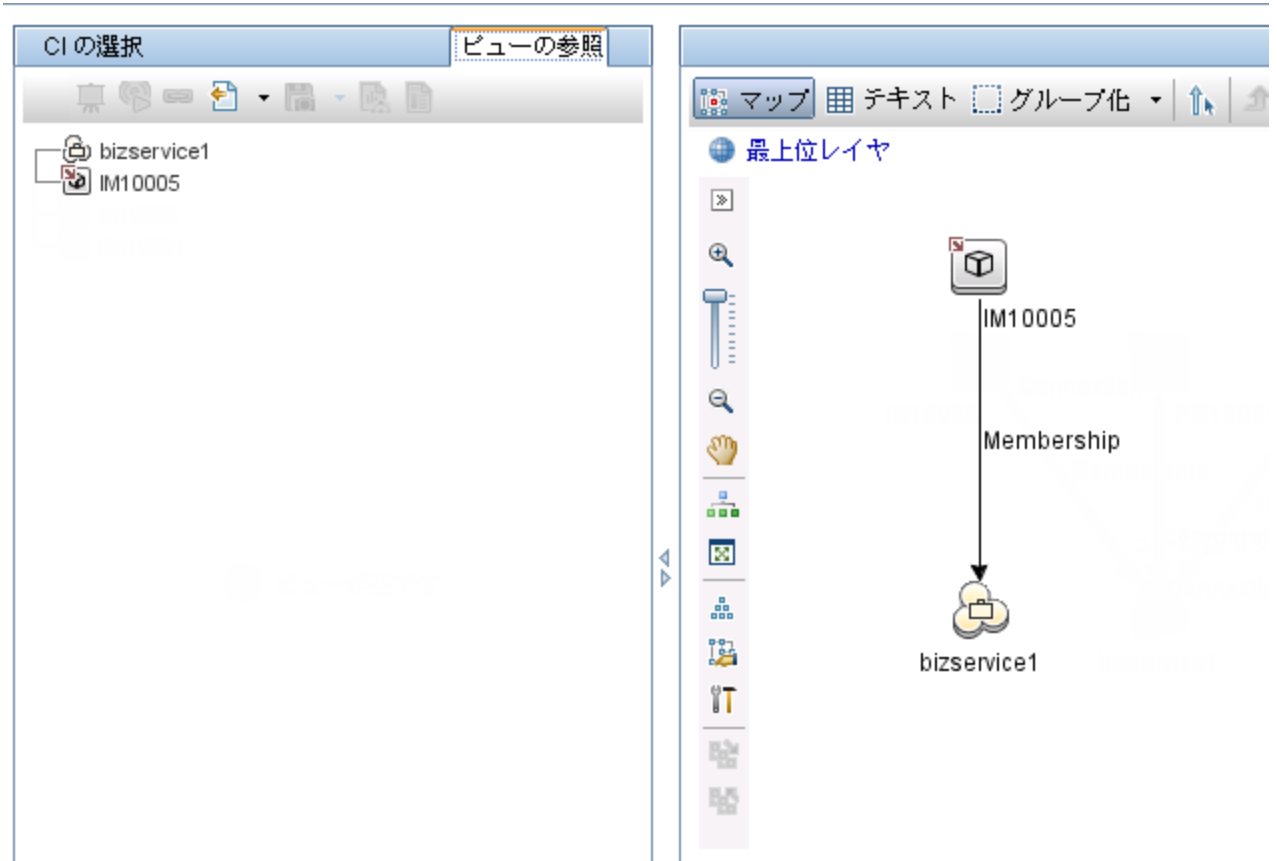
- d. [OK]をクリックします。
10. 上の手順を繰り返して、インシデントクエリノードのデータソースとして統合ポイント (**sm_integration** など)を指定します。




11. [クエリ結果数を計算する]アイコン  をクリックします。SM インシデントの数およびその影響を受ける UCMDB ビジネスサービスCI の数が表示されます。



12. [プレビュー]アイコン  をクリックしてクエリ結果を表示します。



13. [CI の選択] 表示 枠またはクエリ表示 枠から SM インシデントレコードを選択し、[プロパティ] アイコン  をクリックして詳細を表示します。

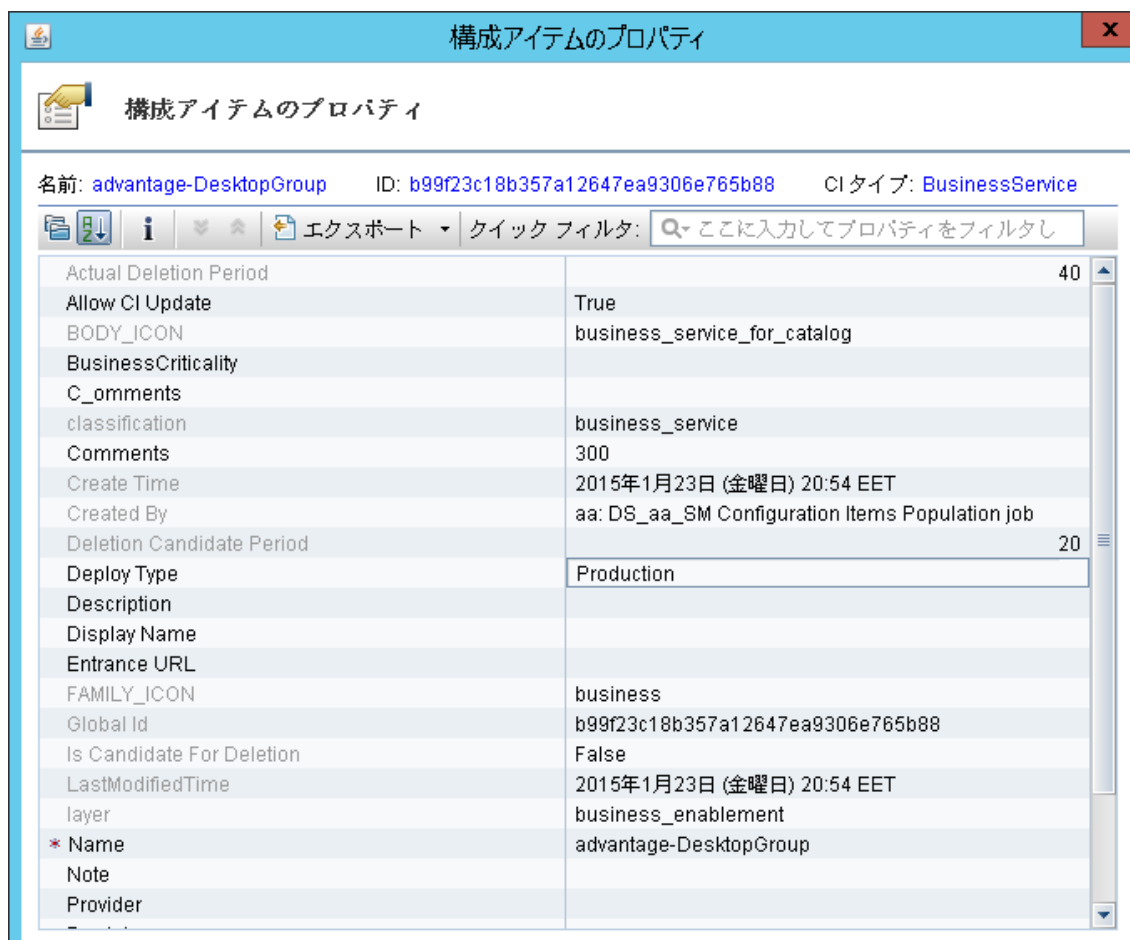
構成アイテムのプロパティ

名前: IM ID: 0reference_number%3DSTRING%3DIM10005%0Aurgency%3DSTRING%3D1_critical%0A CIタイプ: lr

エクスポート クイック フィルタ:

Actual Deletion Period		40
Category	failure	
ClosedTime		
Create Time	2014年1月15日 (水曜日) 07:21 EET	
Deletion Candidate Period		20
Details		
Display Label	IM10005	
ImpactScope	user	
IncidentStatus	Work_In_Progress	
LastModifiedTime	Mon Jan 5 2015 10:52 PM IST	
Name	E-mail attachments being blocked	
Priority	2_high	
ReferenceNumber	IM10005	
Urgency	1_critical	

14. [CI の選択] 表示 枠から各 UCMDB CI レコードを選択し、[プロパティ] アイコンをクリックして詳細を表示します。




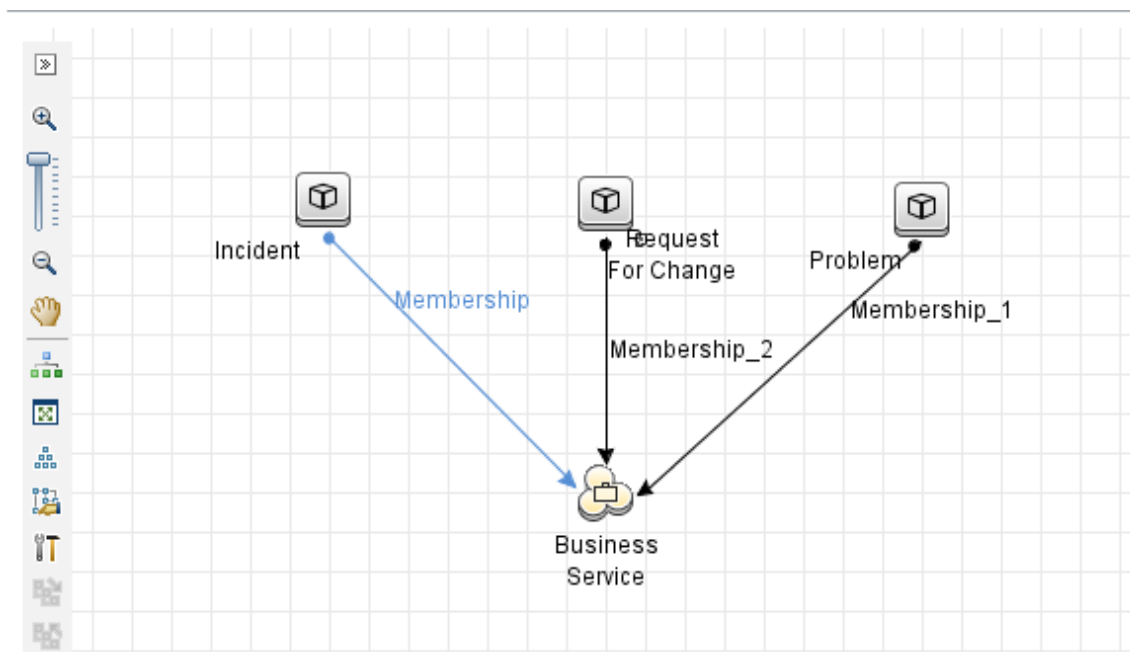
例 3 :UCMDB CI のインシデント、変更、および問題レコードのデータを Service Manager から連携させる

次の例では、UCMDB ビジネスサービス CI に影響する Service Manager 内のインシデント、変更、および問題レコードの情報を取得する方法について説明します。

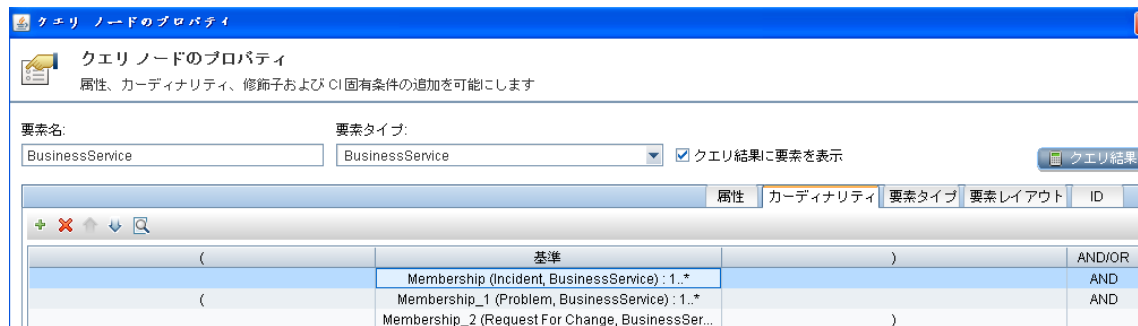
1. 管理者として UCMDB にログインします。
2. [モデリング]>[モデリングスタジオ]>[リソース]に移動します。
3. [リソースタイプ]で、リストから[クエリ]を選択します。
4. [新規]>[クエリ]をクリックします。
5. [CI タイプ]タブで、[ConfigurationItem]>[BusinessElement]>[Service]>[BusinessService]に移動し、右側のクエリ表示 枠にそれをドラッグします。

6. [ItProcessRecord]に移動し、[Incident]、[Problem]、および[RequestForChange]をクエリ表示枠にドラッグします。

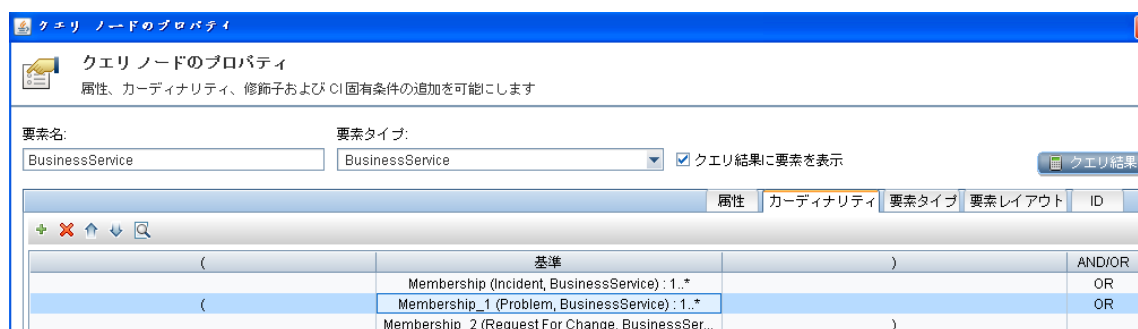
7. [関係を作成]アイコン  をクリックして、次の図に示すように BusinessService ノードと他のノードの間に通常関係を作成します。




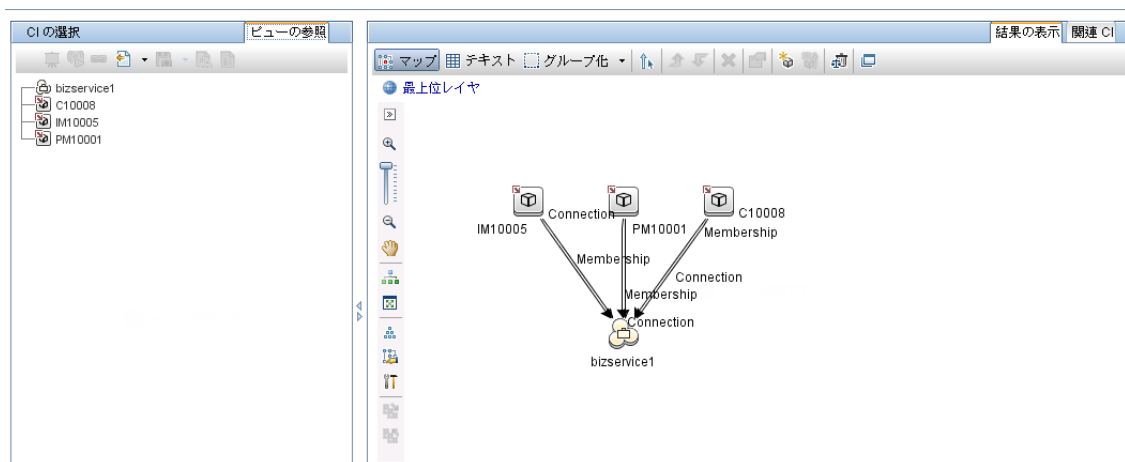
8. 各ノードを選択し、[データソース]タブをクリックして、各ノードのデータソースを指定します。
 - a. BusinessService ノードの場合、データソースとして **UCMDB** を指定します。
 - b. Incident、Problem、および RequestForChange ノードの場合、データソースとして統合ポイント (この例では **sm_integration**) を指定します。
9. クエリを保存します。
10. オプションで、必要に応じて BusinessService ノードのプロパティを編集します。
 - a. BusinessService ノードを選択し、右下の表示枠で[編集]をクリックします。
 - b. [カーディナリティ]タブをクリックします。カーディナリティのデフォルト設定が表示されます。




- c. 必要な場合は、どちらかまたは両方の **AND** オペレータを **OR** に変更します。これにより、フィルタ条件が変更され、そのためにクエリ結果が変更されます。



11. [プレビュー]アイコン  をクリックしてクエリ結果を表示します。



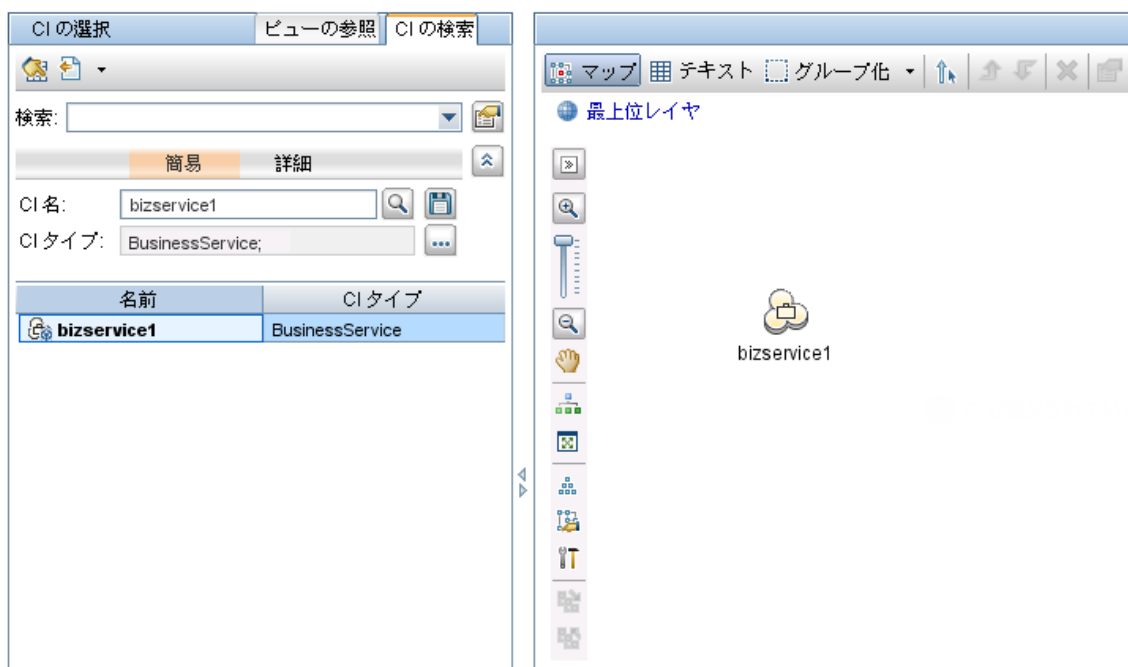
12. [CI の選択] 表示 枠またはクエリ表示 枠から SM インシデントレコードを選択し、[プロパティ]アイコン  をクリックして詳細を表示します。

13. [CI の選択] 表示 枠またはクエリ表示 枠から各 UCMDB CI レコードを選択し、[関連 CI] タブで、[関連 CI を表示] をクリックして、SM と UCMDB の両方の関連 CI を表示します。

例 4 :UCMDB CI に関連する Service Manager レコードを取得する

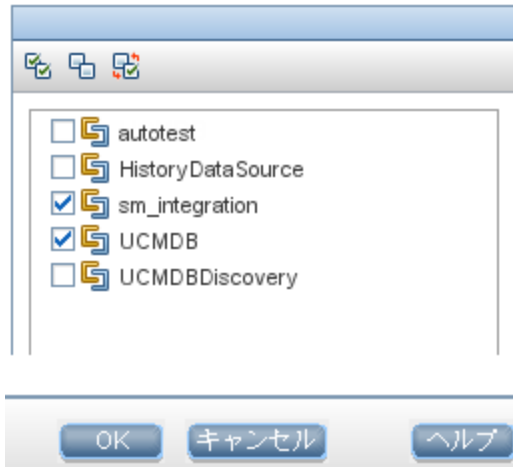
次の例では、[関連 CI を取得] 機能を使用して、UCMDB CI に関連する Service Manager レコードを取得する方法について説明します。

1. 管理者として UCMDB にログインします。
2. [モデリング]>[IT ユニバース マネージャ]に移動します。
3. [CI の検索] タブで、Service Manager 内に関連するレコードがある UCMDB CI を検索します。たとえば、[CI 名] フィールドに「bizservice1」と入力し、[検索] をクリックして、CI をダブルクリックして開きます。

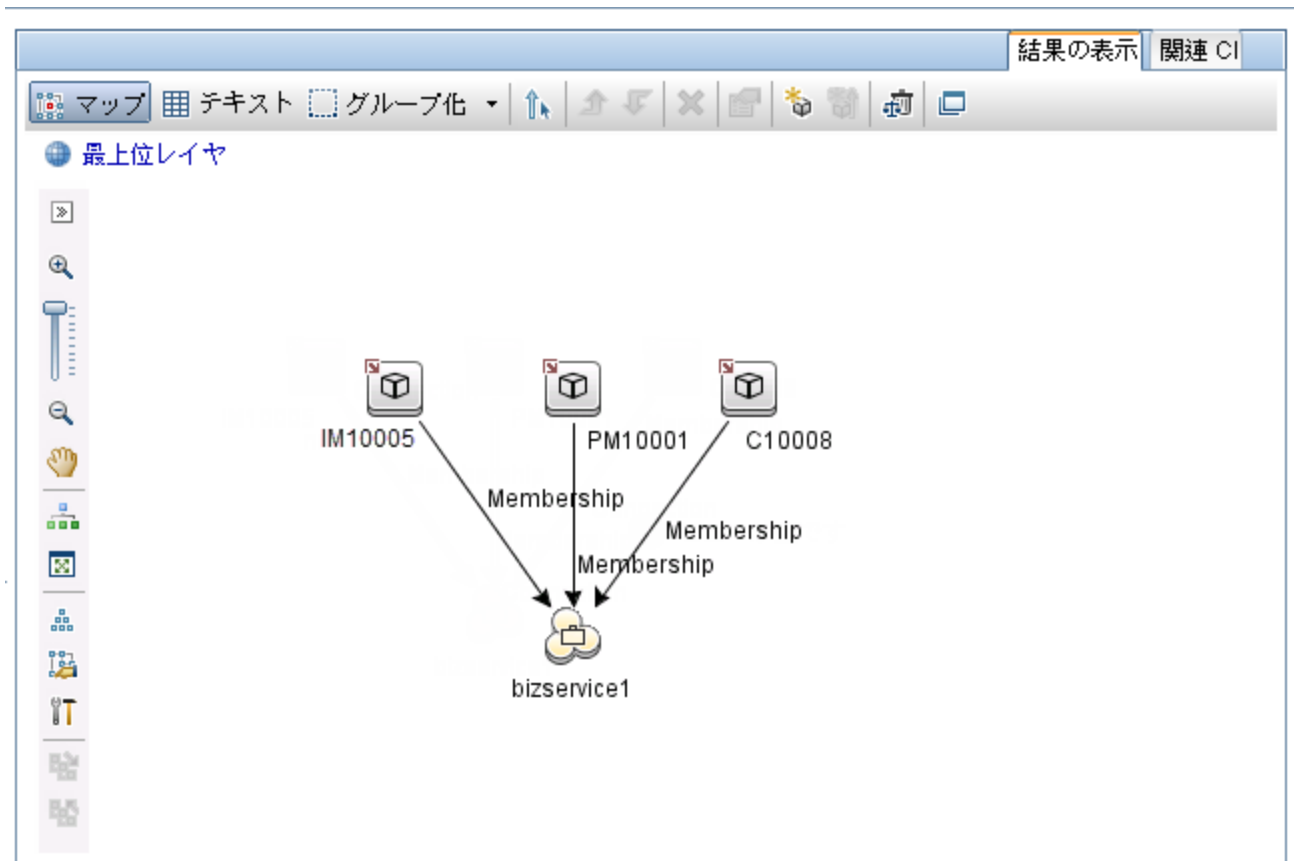


4. [関連 CI を取得] 表示 枠が表示されない場合は、[[関連 CI を取得] 表示 枠を表示] アイコン をクリックします。
5. [関連する CI のターゲット統合ポイントを選択します。] アイコン をクリックします。
6. [統合ポイントを選択] オプションを選択し、UCMDB と統合ポイントの両方を選択します。[OK] をクリックします。

- すべての統合ポイント
- ローカル データ ソース (UCMDB のみ)
- 統合ポイントを選択



7. [関連 CI を表示] をクリックします。CI の関連する SM レコードと UCMDB CI が表示されます。



8. クエリ表示枠から各 SMレコードを選択し、[プロパティ]アイコン  をクリックして詳細を表示します。

第3章: マルチテナンシー (マルチカンパニー) のセットアップ

UCMDB-SM 統合はマルチテナンシー構成をサポートしており、これにより、Service ManagerとUCMDBの両システムが、会社 ID で構成アイテム (CI) と構成アイテム関係 (CIR) を追跡します。マルチテナンシー構成では、統合をカスタマイズして、各テナントが自社の会社 ID に一致する CI と CIR のみを表示して作業できるようにできます。マルチテナンシーは、複数のテナントにサービスとして構成管理を提供する管理サービスプロバイダ (MSP) 向けです。

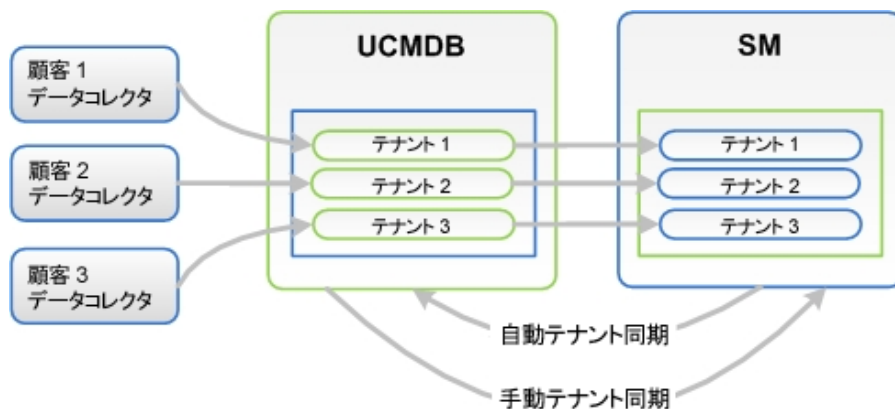
本章の内容

- 「マルチテナンシー (マルチカンパニー) のサポート」(66ページ)
- 「マルチテナンシーの要件」(73ページ)
- 「UCMDB でのマルチテナンシー統合のセットアップ」(73ページ)
- 「Service Manager でのマルチテナンシー統合のセットアップ」(76ページ)

マルチテナンシー (マルチカンパニー) のサポート

マルチテナンシーとは、ソフトウェアの単一のインスタンスが、サーバ上で実行され、複数のクライアント組織 (テナントとも呼ばれます) の要求を処理している状況です。マルチテナンシーとは対照的にマルチインスタンスアーキテクチャでは、個別のソフトウェアインスタンスまたはハードウェアシステムが異なるクライアント組織用にセットアップされます。

マルチテナントアーキテクチャを実装するときには、ソフトウェアアプリケーションはデータと構成を仮想的に分割するように設計され、各クライアント組織はカスタマイズされた仮想アプリケーションインスタンスを使用します。次の図は、マルチテナント統合のデプロイメントの例を示しています。



UCMDB で構成されるすべてのテナントは、SM 内の関連するテナントを使用します。UCMDB でテナントが構成されていない場合、SM から UCMDB に構成を自動的に転送するためにテナント構成をアクティブ化する必要があります。この手順は、システム管理者が1度だけ実行します。

UCMDB のテナント構成がすでに存在し、SM の構成が存在しない場合、UCMDB の構成に従って SM テナントを手動で構成する必要があります。

UCMDB-SM 統合でのマルチテナンシーの実装

SM は、マルチテナント構成での各テナントを記述する会社レコードを格納します。Service Manager システムは、会社レコードの確定的なソースであり、すべての新しい会社 ID を UCMDB システムにプッシュして UCMDB 内に同等のエンティティを作成します。

SM は、マルチテナント構成で、各 CI と関係の会社 ID を追跡します。CI レコードは、その CI レコードを検出した UCMDB フィーダの会社 ID を継承します。関係レコードは、関係の親 CI の会社 ID を継承します。

データ制限 SM セキュリティレイヤ

データ制限は、CI 情報からの顧客 ID をフィルタ処理するために使用される SM ソフトウェアレイヤです。SM は、データ制限を使用して、CI の会社 ID がオペレータの会社 ID に一致した場合にのみ CI および関係レコードがオペレータに表示されるようにします。データ制限で表示を制限すると、Service Manager での他のすべての関連レコード (変更依頼、インシデントなど) の表示も制限されます。

UCMDB に格納されるマルチテナント情報について

UCMDB システムは、各 CI と CIR の会社 ID 属性を格納します。会社 ID により、UCMDB システムが CI データの更新に使用するアダプタと同期スケジュールが決定します。各 CI と関係レコードは、1つの会社 ID のみを取ることができます。UCMDB システムは Service Manager システムから会社 ID を取得します。

複数のテナント (会社) が同一の CI を共有する場合、各テナントは CI を記述する独自の一意な CI レコードを持ちます。実際には、UCMDB システムは1つの管理資産を追跡するために複数の CI レコードを作成します。各テナントの CI レコードは、そのテナントに固有であり、その会社固有の会社 ID をリストします。

Service Manager に格納されるマルチテナント情報について

Service Manager は、マルチテナント構成での各テナントを記述する会社レコードを格納します。Service Manager システムは、会社 ID の確定的なソースであり、UCMDB システムに新規情報と更新された情報をプッシュします。

Service Manager は、マルチテナント構成で、各 CI と関係の会社 ID を追跡します。CI レコードは、その CI レコードを検出した UCMDB フィーダの会社 ID を継承します。関係レコードは、関係の親 CI の会社 ID を継承します。

ベストプラクティスの実装では、Service Manager は、データ制限を使用して、CI の会社 ID がオペレータの会社 ID に一致した場合にのみ CI および関係レコードがオペレータに表示されるようにします。データ制限で表示を制限すると、Service Manager での他のすべての関連レコード (変更依頼、インシデントなど) の表示も制限されます。

一意の論理名

Service Manager では、すべての CI に一意の論理名があることが必要です。論理名生成プロセスで重複する論理名の値が生成されると、Service Manager は、論理名の最後にアンダースコアと番号を付記して論理名を一意にします。たとえば、2 つの CI が論理名 **mytesthost** を取る場合、2 番目の CI の名前は **mytesthost_1** になります。重複する CI がもう 1 つあれば、その名前は **mytesthost_2** になります。

会社レコードの同期

システムが、マルチテナンシーサポートの条件をすべて満たす場合、Service Manager は UCMDDB システムに会社レコードの会社 ID をプッシュするためのスケジュールレコードを作成します。Service Manager は、UCMDDB システムに会社 ID をプッシュするかどうかの判定に次のルールを用います。

Service Manager が UCMDDB と会社 ID を同期する条件

条件	テナント情報が同期されるかどうか	作成されるスケジュールレコードと、UCMDDB で行われるアクション
<ul style="list-style-type: none"> UCMDDB-SM 統合が有効 Service Manager でマルチカンパニーモードが有効 Service Manager で新しい会社レコードを作成する 	はい	Synch Company with UCMDDB - <UCMDDB Company ID> <ul style="list-style-type: none"> 新しい会社 ID の追加
<ul style="list-style-type: none"> UCMDDB-SM 統合が有効 UCMDDB と同期していない既存会社レコードを更新する Service Manager でマルチカンパニーモードが有効 	はい	Synch Company with UCMDDB - <UCMDDB Company ID> <ul style="list-style-type: none"> 新しい会社 ID の追加
<ul style="list-style-type: none"> UCMDDB-SM 統合が有効 Service Manager でマルチカンパニーモードが有効 UCMDDB と同期している会社に対し、マルチカンパニーリストで会社を表示するオプションを無効にする 	はい	Inactivate Company with UCMDDB - <UCMDDB Company ID> <ul style="list-style-type: none"> 既存の会社 ID の非アクティブ化

Service Manager が UCMDB と会社 ID を同期する条件 (続き)

条件	テナント情報が同期されるかどうか	作成されるスケジュールレコードと、UCMDB で行われるアクション
<ul style="list-style-type: none"> UCMDB-SM 統合が有効 Service Manager でマルチカンパニーモードが有効 既存会社レコードに対し、UCMDB との再同期を行うオプションを選択する 	はい	Synch Company with UCMDB - <UCMDB Company ID> <ul style="list-style-type: none"> 新しい会社 ID の追加
<ul style="list-style-type: none"> UCMDB-SM 統合が有効 Service Manager でマルチカンパニーモードが有効 非アクティブな会社に対し、マルチカンパニーリストで会社を表示するオプションを有効にする 	はい	Synch Company with UCMDB - <UCMDB Company ID> <ul style="list-style-type: none"> 会社 ID の再アクティブ化
<ul style="list-style-type: none"> UCMDB-SM 統合が無効 Service Manager でマルチカンパニーモードが有効 UCMDB とすでに同期されている既存会社レコードを更新する 	いいえ	なし
<ul style="list-style-type: none"> UCMDB-SM 統合が無効 Service Manager でマルチカンパニーモードが有効 Service Manager で新しい会社レコードを作成する 	いいえ	なし
<ul style="list-style-type: none"> UCMDB-SM 統合が有効 Service Manager でマルチカンパニーモードが有効 UCMDB と同期していない会社に対し、マルチカンパニーリストで会社を表示するオプションを無効にする 	いいえ	なし

Service Manager が UCMDB と会社 ID を同期する条件 (続き)

条件	テナント情報が同期されるかどうか	作成されるスケジュールレコードと、UCMDB で行われるアクション
<ul style="list-style-type: none">UCMDB-SM 統合が有効Service Managerでマルチカンパニーモードが無効Service Manager で新しい会社レコードを作成する	いいえ	なし

UCMDB 顧客 ID

マルチテナンシーの統合を有効にすると、Service Manager で各会社レコードに[UCMDB 顧客 ID]という新しいフィールドが表示されます。UCMDB と会社レコードを同期するには、まずこのフィールドの値を入力する必要があります。UCMDB 顧客 ID 値を入力すると、このフィールドは読み取り専用になります。会社の UCMDB 顧客 ID は、一度設定すると変更できません。

このフィールドには、10 文字までの数字データのみを入力できます。Service Manager ではフィールド値は一意的な正の整数である必要があります。重複した値を入力したり、小数、負の数、ゼロを使用することはできません。

UCMDB システムは、単一テナントモードで実行時には UCMDB 顧客 ID として自動的に 1 を使用します。Service Manager の会社はこの UCMDB 顧客 ID 値を割り当てることで、マルチテナント実装でこのデフォルト値を再使用できます。出荷時設定では、UCMDB 顧客 ID が 1 の Service Manager 会社はありません。

UCMDB ユーザ ID とパスワード

マルチテナンシーの統合を有効にすると、Service Manager では各会社レコードに[UCMDB ユーザ ID]と[UCMDB パスワード]という2つの新しいフィールドが表示されます。これらのフィールドを使用することで、[実際のステータス]セクションの情報を要求するときに Service Manager が使用する接続情報を指定できます。これらのフィールドに入力するユーザ名とパスワードは、UCMDB システムで有効である必要があります。

会社情報レコードに入力されるユーザ名とパスワードは、システム情報レコードに入力されるユーザ名とパスワードより優先されます。これにより、管理サービスプロバイダは、テナント別ベースで UCMDB システムへのアクセスを制御できます。会社固有の UCMDB ユーザ名とパスワードを入力しない場合、Service Manager ではシステム情報レコードに入力された資格情報が使用されます。

会社コード

マルチテナンシーの統合では、各会社レコードが一意的な会社コード (会社フィールド) 値を取る必要があります。会社コードは必須フィールドであるため、既存の会社レコードには会社コード値が必要です。ただし、各会社レコードに一意的な会社コード値があることを確認する必要があります。

UCMDB-SM 統合用の Service Manager 拡張汎用アダプタを使用する場合は、ServiceManagerEnhancedAdapter9-x/mappings/scripts/SMUtils.groovy ファイルで次の形式を使用して会社コードを定義する必要があります (マッピングエントリの区切りにコンマを使用します)。

```
[ "<UCMDB Company 1 Code>":"<SM Company 1 Code>", "<UCMDB Company 2 Code>":"<SM Company 2 Code>" ]
```

例: ["1":"hp", "2":"sap"]

注意: マルチテナンシーの統合を有効にした後は、会社コード値は変更できません。変更すると、Service Manager データが同期されなくなるためです。

CI 調整ルール

マルチテナンシーが有効である場合、Service Manager は、データプッシュジョブの会社 ID と一致する会社 ID を持つ CI のみを調整します。たとえば、会社 2 から CI をプッシュする際は、会社番号 2 に対応する会社コードの Service Manager CI レコードにのみ調整ルールが適用されます。

CI および CI 関係レコードにプッシュされる会社情報

マルチテナンシーの統合を有効にすると、Service Manager がデータプッシュ時に SM 会社コード値を CI および関係レコードに挿入します。Service Manager は UCMDB 顧客 ID を使用して、一致する SM 会社コード値を検索します。

インシデントレコードに複製される会社情報

マルチテナンシーの統合を有効にし、UCMDB が新規 CI、更新された CI、削除された CI を検出したときにインシデントを作成するオプションを選択すると、Service Manager は、レプリケーション時にインシデントレコードに SM 会社コード値を挿入します。Service Manager は UCMDB 顧客 ID を使用して、一致する SM 会社コード値を検索します。

スケジュールレコード

Service Manager は、**problem** スケジュールプロセスを使用して、UCMDB システムへの会社 ID の同期を管理します。[システムステータス] フォームから **problem** スケジュールプロセスを手動で有効にすることができます。

表「[Service Manager が UCMDB と会社 ID を同期する条件](#)」(68ページ)に示した同期の条件が満たされる場合、Service Manager によって、「Synch Company with UCMDB - <UCMDB Company ID>」スケジュールレコード (“Synch Company with UCMDB - 1234567890” など) が作成されます。会社を非アクティブにすると、Service Manager によって、「Inactivate Company with UCMDB - <UCMDB Company ID>」スケジュールレコード (“Inactivate Company with UCMDB - 1234567890” など) が作成されます。**problem** スケジュールプロセスは、次のバックグラウンド処理の際に新しいスケジュールレコードを処理します。

Service Manager システムが何らかの理由で UCMDB システムに接続できない場合、次のスケジュール間隔 (出荷時設定での間隔は 5 分間) に会社の同期を再スケジュールします。problem スケジュールプロセスは、再スケジュールされたステータスでスケジュールレコードを更新します。Service Manager システ

ムが UCMDDB システムに接続中にその他のエラーメッセージを受信すると、スケジュールレコードは、「application failed due to error - check msglog for possible messages」ステータスに更新されます。

テナント固有の検出イベントマネージャ(DEM) ルール

マルチテナンシーの UCMDDB-SM 統合では、条件フィールド機能を実装することで、特定のテナントに固有の SM DEM ルールを作成できます。

テナントルールは、SM テナントの構成要件によって異なります。UCMDDB から SM にプッシュされる各レコード情報タイプ用に、異なるテナントで異なる DEM テナントルールを構成できます。

各テナントには独自の要件のセットが存在する可能性があり、そのために統合を介して異なるプロセスを実装する場合があります。

たとえば、SM に直接 CI を追加したいと考えるテナントもあれば、各 CI について変更をオープンしたいと考えるテナントもあります。

次の表に、これを実現する方法について説明するサンプルの DEM ルールセットを示します。

テナント固有 DEM ルール

DEM ルール ID	新規 CI に対するアクション	条件
ucmdbNode_advantage	CI の追加	company in \$.file="advantage"
ucmdbNode_hp	変更の作成	company in \$.file="HP"

注: DEM ルール

DEM ルールを作成するときには、必ず各テナント用に個別の DEM ルールを作成してください。

マルチテナンシーのユースケース

次の表では、さまざまなデプロイメントの状況で、マルチテナンシーの問題に対処するために実行する必要があるアクションについて説明します。

マルチテナンシーのユースケース

デプロイメント統合のタイプ	説明
マルチテナンシールールがある UCMDDB	UCMDDB には既存のマルチテナンシールールが存在し、SM ではマルチテナンシールールが構成されていない UCMDDB-SM のデプロイメントを実装するときには、ユーザが UCMDDB のルールに従い、手動で SM 内にマルチテナンシールールを作成します。
マルチテナンシールールがない SM	

マルチテナンシーのユースケース (続き)

デプロイメント統合のタイプ	説明
マルチテナンシールールがある SM マルチテナンシールールがない UCMDB	SM では既存のマルチテナンシールールが構成され、UCMDB ではマルチテナンシールールが構成されていない UCMDB-SM のデプロイメントを実装するときには、ユーザが前に SM で構成したルールに従い、手動で UCMDB 内にマルチテナンシールールを作成します。
マルチテナンシールールがない UCMDB マルチテナンシールールがない SM	UCMDB または SM でマルチテナンシールールが構成されていない UCMDB-SM のデプロイメントを実装するときには、ユーザが SM でルールを構成します。 ユーザは、SM マルチテナンシーウィザードを使用して構成を行うときに、UCMDB 内に対応するテナント構成を作成できます。ユーザは、SM 内に対応するテナント構成を作成することで、UCMDB 内に対応するテナントを作成することもできます。

マルチテナンシーの要件

統合がマルチテナンシーをサポートするには、システムが以下の要件を満たす必要があります。

- HP Universal CMDB バージョン 8.02 以降
- HP Service Manager バージョン 9.20 以降
- UCMDB と Service Manager間の統合が有効であること
- Service Manager システム上でマルチカンパニーモードが有効であること
- Service Manager 上で problem スケジュールプロセスが動作していること

マルチテナンシーの統合の詳細については、[HP ソフトウェアサポートオンライン Web サイト](#) または Service Manager のヘルプを参照してください。

UCMDB でのマルチテナンシー統合のセットアップ

マルチテナンシーの統合をセットアップするには、UCMDB で次のタスクを実行する必要があります。

1. 統合がサポートする各テナントに、個別の Data Flow Probe をインストールします。
[「各テナントに個別の Data Flow Probe をインストールする方法」\(74ページ\)](#)を参照してください。
2. テナント固有 Data Flow Probe を起動します。
[「テナント固有の Data Flow Probe を起動する方法」\(75ページ\)](#)を参照してください。

3. テナント固有 Data Flow Probe の IP アドレス範囲を設定します。
「[テナント固有の Data Flow Probe の IP 範囲を構成する方法](#)」(76ページ)を参照してください。

注意: 統合で Service Manager 拡張汎用アダプタを使用している場合、ポピュレーションではマルチテナンシーはサポートされません。

各テナントに個別の Data Flow Probe をインストールする方法

マルチテナント構成をサポートする場合、各テナントごとに個別のデータプローブをインストールする必要があります。出荷時設定では、UCMDB インストーラは 1 つの Data Flow Probe とサービスのみをインストールします。

追加の Data Flow Probe をインストールしてオペレーティングシステムのコマンドプロンプトから起動するには:

1. 管理者として UCMDB システムのホストにログインします。
2. システムディスクドライブに HP Universal CMDB Setup Windows DVD を挿入します。
3. Data Flow Probe インストーラ (HPUCMDB_DataFlowProbe_x.xx.exe) を起動します。
4. 画面の指示に従ってウィザードを完了します。この際、インストールする各 Data Flow Probe には次の値を使用します。
 - a. 各インストールフォルダの一意なパスを入力します。
 - b. 各 Data Flow Probe に、同一の UCMDB アプリケーションサーバアドレスを使用します。
 - c. 有効な Data Flow Probe アドレスを入力します。
 - d. 各 Data Flow Probe 識別子の一意な名前を入力します。
 - e. 各プローブに、一意な顧客の Data Flow Probe ドメインを作成します ([標準設定の UCMDB ドメインを使用] オプションをクリアします)。
 - f. 各プローブには、同じプローブゲートウェイとプローブマネージャ設定を使用します (結合プロセス、または独立プロセスの使用など)。

完全なインストールの手順については、『HP Universal CMDB デプロイメントガイド』を参照してください。

5. インストールする各 Data Flow Probe について、[手順 3](#)と[手順 4](#)を繰り返します。
6. テキストエディタで、プローブの DiscoveryProbe.properties ファイルを開きます。デフォルトで、このファイルは次のフォルダに配置されています:
<UCMDB インストールフォルダ>\<Data Flow Probe インストールフォルダ>\conf

たとえば、C:\hp\UCMDB\DataFlowProbe\conf などです。

注: <Data Flow Probe のインストールフォルダ>は、テナントごとに一意である必要があります。

7. 設定ファイルで次のプロパティを編集します。
各テナントに設定するディスカバリプローブプロパティ

プロパティ	値
serverName	UCMDB サーバの名前を確認します
customerId	この Data Flow Probe がサポートするテナントの顧客 ID を入力します
appilog.collectors.probe.name	プローブ名が、サーバ+テナント ID などのように一意であることを確認します
appilog.collectors.domain	Data Flow Probe 名を確認します
appilog.collectors.local.ip	Data Flow Probe ゲートウェイ名を確認します
appilog.collectors.probe.ip	Data Flow Probe マネージャ名を確認します
appilog.collectors.rmi.port	各プローブに対して一意なポートを入力します
appilog.collectors.rmi.gw.port	各プローブに対して一意なポートを入力します
appilog.collectors.probe.html.port	各プローブに対して一意なポートを入力します
appilog.collectors.local.html.port	各プローブに対して一意なポートを入力します
appilog.collectors.ProbeUseSpecific RMIPortFrom	各プローブに対して一意なポートを入力します。「0」を入力するとシステムが自動的にポートを選択します
appilog.collectors.bigBrother.port	各プローブに対して一意なポートを入力します

8. 設定ファイルを保存します。
9. 各テナントの Data Flow Probe について、手順 6 から手順 8 を繰り返します。

テナント固有の Data Flow Probe を起動する方法

テナント固有の Data Flow Probe を起動するには:

1. OS のコマンドプロンプトを開き、プローブの bin フォルダに移動します。
C:\hp\UCMDB\DataFlowProbe1\bin などです。


2. 「gateway console」と入力します。
3. 起動する各 Data Flow Probe について、手順 1 と手順 2 を繰り返します。

テナント固有の Data Flow Probe の IP 範囲を構成する方法

テナント固有の Data Flow Probe の IP 範囲を構成するには:

1. 構成する Data Flow Probe のテナントの会社 ID を使用し、管理者として UCMDDB にログインします。
2. [データフロー管理]>[Data Flow Probe 設定]を選択します。
3. 起動するプローブが含まれる Data Flow Probe ドメインを展開します。[Customer2]などです。
4. プローブノードを展開し、起動する Data Flow Probe を選択します。[Probe2Customer2]などです。



5. [IP 範囲を追加]アイコン  をクリックします。
6. Data Flow Probe がスキャンする IP 範囲を入力します。オプションで、除外する IP 範囲を追加します。
7. [OK]をクリックして、IP 範囲を保存します。
8. 構成する各 Data Flow Probe について、手順 1 から手順 7 を繰り返します。

Service Manager でのマルチテナンシー統合のセットアップ

マルチテナンシー統合をセットアップするには、Service Manager で次のタスクを実行する必要があります。

マルチテナンシーサポートは、テナントに対してサービスとして構成管理を行う管理サービスプロバイダ (MSP) を対象とした統合のオプション機能です。マルチテナンシー構成では、各 CI と CIR レコードは対応する会社 ID を持ちます。出荷時設定の Service Manager では、すべてのオペレータはどの会社 ID であっても CI データを表示できます。会社 ID で CI データへのアクセスを制限するには、データ制限を有効にし、[会社 ID] フィールドを制限用クエリで使用する必要があります。マルチカンパニーモードとデータ制限の詳細については、Service Manager のヘルプを参照してください。

統合でマルチテナンシーサポートを有効にするには、Service Manager で次のタスクを完了する必要があります。

1. スケジュールプロセスを開始します。
「[スケジュールプロセスを開始する方法](#)」(77ページ)を参照してください。
2. Service Manager システム情報レコードを構成します。
「[Service Manager システム情報レコードを構成する方法](#)」(78ページ)を参照してください。
3. 会社レコードにテナント固有の UCMDDB ID とパスワード値を追加します (オプション)。
「[テナント固有 UCMDDB ユーザ ID とパスワード値の追加方法](#)」(79ページ)を参照してください。
4. 既存の会社レコードに UCMDDB 顧客 ID を追加します。
「[UCMDDB 顧客 ID 値を既存の会社追加する方法](#)」(80ページ)を参照してください。
5. 既存の会社レコードを UCMDDB と同期します。
「[Service Manager から UCMDDB に既存の会社を同期する方法](#)」(80ページ)を参照してください。
6. Service Manager によって会社レコードが UCMDDB と同期したことを確認します (オプション)。
「[会社情報が UCMDDB にあるかどうかを確認する方法](#)」(81ページ)を参照してください。
7. 既存の会社レコードを UCMDDB と再同期します (必要な場合)。
「[既存の会社を UCMDDB と再同期する方法](#)」(82ページ)を参照してください。
8. 統合に含めない会社レコードを非アクティブ化します (必要な場合)。
「[同期された会社を非アクティブにする方法](#)」(82ページ)を参照してください。
9. 統合に含める非アクティブな会社レコードを再アクティブ化します (必要な場合)。
「[非アクティブな会社を再度アクティブにする方法](#)」(83ページ)を参照してください。
10. テナント固有 DEM ルールを追加します。
「[テナント固有 DEM ルールの追加方法](#)」(83ページ)を参照してください。

スケジュールプロセスを開始する方法

この統合では、Service Manager から UCMDDB に会社レコードを同期するために **problem** スケジュールプロセスを必要とします。会社レコードを同期する前に、このプロセスが開始されていることを確認します。

[**problem**] スケジュールプロセスを開始するには:

1. システム管理者として Service Manager にログインします。
2. システムナビゲータの[システムステータス]をクリックします。
現在開始されているスケジュールのリストが表示されます。
3. [表示のリフレッシュ]ボタンをクリックしてリストを更新します。
4. **problem** スケジュールプロセスがリストにない場合は、次の手順を実行します。
 - a. [スケジュールの開始]ボタンをクリックします。
 - b. [**problem**]スケジュールプロセスをダブルクリックします。**problem** スケジュールプロセスが開始されたことを示すメッセージが表示されます。

Service Manager システム情報レコードを構成する方法

統合でマルチテナンシーを有効にするには、Service Manager システム情報レコードに追加情報を入力する必要があります。

注意: マルチテナンシーのサポートを有効にするには、HP Universal CMDB バージョン 8.02 以降を使用する必要があります。それよりも前のバージョンの HP Universal CMDB では、マルチテナンシーモードで実行しようとする、エラーメッセージが表示されます。

Service Manager システム情報レコードで追加情報を入力するには:

1. システム管理者として Service Manager にログインします。
2. [システム管理]>[ベースシステム構成]>[その他]>[システム情報レコード]に移動します。
3. [全般]タブをクリックします。
4. [マルチカンパニーモードで実行する]オプションを有効にします。
5. [アクティブ統合]タブをクリックします。
6. [HP Universal CMDB]オプションを選択します。
フォームに[UCMDB Web サービス URL]フィールドが表示されます。
7. [UCMDB Web サービス URL]フィールドに、CI の同期 Web サービス API の URL を入力します。
URL、は次の形式です:
`http://<UCMDB server name>:<port>/axis2/services/ucmdbSMService`

<UCMDB server name>には UCMDB サーバのホスト名、<port>には UCMDB サーバが使用する通信ポートを入力します。
8. [ユーザID]と[パスワード]に、UCMDB システム上で CI を管理するのに必要なユーザ資格情報を入力します。たとえば、出荷時設定の資格情報は「admin/admin」です。

9. [マルチテナント Web サービス URL] フィールドに、会社 ID の同期 Web サービス API の URL を入力します。URL、は次の形式です:
http://<UCMDB server name>:<port>/axis2/services/UcmdbManagementService

<UCMDB server name>には UCMDB サーバのホスト名、<port>には UCMDB サーバが使用する通信ポートを入力します。
10. UCMDB システムで会社 ID を同期するのに必要なユーザ名とパスワードを入力します。たとえば、UCMDB の出荷時設定のシステム管理者用資格情報は「**sysadmin/sysadmin**」です。
11. [保存] をクリックします。Service Manager のメッセージが表示されます。「情報レコードは更新されました。」と表示されます。
12. Service Manager システムからログアウトし、管理者アカウントでログインし直します。
13. [システムステータス]>[表示オプション]>[すべてのタスク] をクリックします。
14. **problem** スケジュールプロセスの隣にある[コマンド] フィールドに「k」と入力し、[コマンドの実行] をクリックします。**problem** スケジュールプロセスが終了するまで、数分待ちます。
15. [スケジュールの開始] をクリックします。
16. [problem] スケジュールプロセスをダブルクリックします。これで、システムは UCMDB 向けマルチテナンシーをサポートするようになります。

テナント固有 UCMDB ユーザ ID とパスワード値の追加方法

[実際のステータス] セクションの情報を要求するときに Service Manager が使用する、テナント固有の UCMDB ユーザ名とパスワードを入力できます。資格情報がない場合、Service Manager はすべてのテナントに対して、システム情報レコードにある資格情報を使用します。

注: 会社レコードに入力する資格情報は、システム情報レコードに入力する資格情報より優先されます。[UCMDB ユーザ ID] と [UCMDB パスワード] フィールドは、マルチテナンシーの統合を有効にしている場合のみ利用可能です。

テナント固有 UCMDB ユーザ名とパスワードを追加するには:

1. システム管理者として Service Manager にログインします。
2. [システム管理]>[ベースシステム構成]>[会社] を選択します。
3. 会社レコードの検索に使用する検索基準を入力します。すべての会社レコードを検索するには、検索フォームを空欄にしておきます。
4. [検索] をクリックします。
5. [UCMDB ユーザ ID] フィールドに、この会社が UCMDB に接続するのに使用するユーザ名を入力します。

6. [UCMDB パスワード]フィールドに、このUCMDB ユーザ名のパスワードを入力します。
7. [保存]をクリックします。
8. 資格情報を入力する各会社について、手順 3 から手順 7 を繰り返します。

UCMDB 顧客 ID 値を既存の会社に追加する方法

次のステップを使用して、既存のService Manager会社レコードにUCMDB顧客 ID を追加します。

1. システム管理者として Service Manager にログインします。
2. [システム管理]>[ベースシステム構成]>[会社]を選択します。
3. 会社レコードの検索に使用する検索基準を入力します。すべての会社レコードを検索するには、検索フォームを空欄にしておきます。
4. [検索]をクリックします。
5. [UCMDB 顧客 ID]フィールドに、この会社の数値を入力します。
6. [保存]をクリックします。
7. レコードをUCMDB と同期することを確認するプロンプトが表示されます。会社を今すぐ同期するには[はい]、会社を後で同期するには[いいえ]をクリックします。
8. [次へ]をクリックして、レコードリストの次の会社に移動します。
9. レコードリスト内の各会社について、手順 5 ~ 8 を繰り返します。

Service Manager から UCMDB に既存の会社を同期する方法

Service Manager システムに、マルチテナンシーの統合で使用する会社レコードがすでに含まれている場合があります。

Service Manager で UCMDB にまだ同期していない会社レコード内のフィールドを更新する場合、会社を UCMDB に同期するかどうかのプロンプトが表示されます。

注: マルチカンパニーリストに会社を表示するオプションを無効にしている場合、または会社に関連する保留中のスケジュールレコードがある場合、会社レコードの同期は求められません。詳細については、「[同期された会社を非アクティブにする方法](#)」(82ページ)を参照してください。

本タスクの手順は次のとおりです。

1. システム管理者として Service Manager にログインします。
2. [システム管理]>[ベースシステム構成]>[会社]を選択します。
3. 会社レコードの検索に使用する検索基準を入力します。すべての会社レコードを検索するには、検索フォームを空欄にしておきます。
4. [検索]をクリックします。
5. 更新する会社レコードを選択します。
6. 会社レコードを更新します。
7. [保存]をクリックします。レコードを UCMDB と同期することを確認するプロンプトが表示されます。

注: 同期を実行するかどうかにかかわらず、会社レコードは保存されます。

会社情報が UCMDB にあるかどうかを確認する方法

マルチテナンシーの統合が有効である場合、Service Manager の各会社レコードに、UCMDB 顧客 ID が UCMDB システムと同期済みかどうかを示す読み取り専用フィールドが表示されます。

注: [UCMDB 顧客 ID] フィールドは、マルチテナントの UCMDB 統合が有効である場合のみ表示されます。

UCMDB に会社情報があるかどうかを確認するには:

1. システム管理者として Service Manager にログインします。
2. [システム管理]>[ベースシステム構成]>[会社]を選択します。
3. 会社レコードの検索に使用する検索基準を入力します。すべての会社レコードを検索するには、検索フォームを空欄にしておきます。
4. [検索]をクリックします。
5. [UCMDB と同期]フィールドのステータスを確認します。
チェックボックスがオンである場合、Service Manager と UCMDB システムの会社 ID はすでに同期済みです。チェックボックスがオフである場合、Service Manager はまだ UCMDB システムにこの会社を追加していません。

注: 同期障害の詳細については、「[スケジュールレコード](#)」(71ページ)を参照してください。

既存の会社を UCMDB と再同期する方法

なんらかの理由で UCMDB データを失った場合、Service Manager では、会社レコードを UCMDB システムと再同期することができます。たとえば、統合テスト中に UCMDB データを意図的に削除した場合や、障害後にデータを復旧する必要がある場合などです。UCMDB との再同期オプションを使用することで、強制的に Service Manager の会社を UCMDB システムと同期させることができます。

既存の会社を UCMDB と再同期するには:

1. システム管理者として Service Manager にログインします。
2. [システム管理]>[ベースシステム構成]>[会社]を選択します。
3. 会社レコードの検索に使用する検索基準を入力します。すべての会社レコードを検索するには、検索フォームを空欄にしておきます。
4. [検索]をクリックします。
5. 同期する会社レコードを選択します。
6. [UCMDB と同期]チェックボックスの隣にある[再同期]ボタンをクリックします。

注: [再同期]ボタンが利用できる会社レコードは、UCMDB とすでに同期済みで、[UCMDB と同期]チェックボックスがオンである会社レコードのみです。UCMDB システムにこの ID 値の会社がすでに存在している場合、再同期要求は無視されます。Service Manager で会社を UCMDB と再同期する既存のスケジュールレコードが存在する場合にも、再同期要求は無視されます。この場合、「この会社を UCMDB と再同期させるためのスケジュールレコードはすでに追加されています。」というメッセージが表示されます。

同期された会社を非アクティブにする方法

会社レコードを UCMDB と同期すると、そのレコードは削除できなくなります。その代わりに、会社レコードを非アクティブ化することができます。これにより、UCMDB システムはそれ以降、その会社の CI の更新をすべて停止します。その会社のあらゆる既存の CI データはまだ UCMDB システムに存在し、非アクティブな UCMDB 顧客 ID に関連付けられていますが、これ以降は UCMDB システムにおいてこの会社とその関連 CI は両方とも表示されなくなります。

同期された会社を非アクティブ化するには:

1. システム管理者として Service Manager にログインします。
2. [システム管理]>[ベースシステム構成]>[会社]を選択します。
3. 会社レコードの検索に使用する検索基準を入力します。すべての会社レコードを検索するには、検索フォームを空欄にしておきます。
4. [検索]をクリックします。
5. 非アクティブ化する会社レコードを選択します。

6. [マルチカンパニーモード時に会社をリストに表示]で[いいえ]を選択します。
7. [保存]をクリックします。
8. この会社が以前に UCMDB と同期された場合、Service Manager では非アクティブにすることを確認するプロンプトが表示されます。
9. 非アクティブにするには[はい]、変更をキャンセルするには[いいえ]をクリックします。

非アクティブな会社を再度アクティブにする方法

Service Manager システムで非アクティブな会社を再度アクティブにして、マルチテナンシーの統合に含めることができます。UCMDB がこの会社の任意の CI 更新を処理できるようにするため、会社を UCMDB と同期する必要もあります。

非アクティブな会社を再度アクティブにするは:

1. システム管理者として Service Manager にログインします。
2. [システム管理]>[ベースシステム構成]>[会社]を選択します。
3. 会社レコードの検索に使用する検索基準を入力します。すべての会社レコードを検索するには、検索フォームを空欄にしておきます。
4. [検索]をクリックします。
5. 再度アクティブにする会社レコードを選択します。
6. [マルチカンパニーモード時に会社をリストに表示]で[はい]を選択します。
7. [保存]をクリックします。Service Manager で UCMDB の会社を再度アクティブにすることを確認するプロンプトが表示されます。
8. [はい]をクリックします。Service Manager により、会社を再度アクティブにするスケジュールレコードが作成されます。

テナント固有 DEM ルールの追加方法

UCMDB-SM マルチテナンシー統合では、条件フィールドを使用することにより、特定のテナントに固有の DEM ルールを作成できます。たとえば、Service Manager に直接 CI を追加したいと考えるテナントもあれば、各 CI について変更をオープンしたいと考えるテナントもあります。次のサンプル DEM ルールで、この方法を示します。

テナント固有 DEM ルール

DEM ルール ID	新規 CI に対するアクション	条件
ucmdbNode_advantage	CI の追加	company in \$L.file="advantage"

テナント固有 DEM ルール (続き)

DEM ルール ID	新規 CI に対するアクション	条件
ucmdbNode_hp	変更の作成	company in \$L.file="HP"

ヒント: 各テナントごとに独立した DEM ルールを作成することがベストプラクティスです。

第4章: 標準およびベストプラクティス

本章の内容

- 「UCMDB-SM 構成のベストプラクティス」(85ページ)
- 「よく寄せられる質問」(97ページ)

UCMDB-SM 構成のベストプラクティス

本項では、さまざまな環境でこの統合の実装を成功させるためのベストプラクティスと推奨事項について説明します。本項では、UCMDB-SM 統合の性能を高めるための重要な知識と手法を紹介し、さらに一般的な問題の解決策および回避策についても説明します。具体的なシステム要件と設定はシステム環境によって異なるので、これらのベストプラクティスと推奨事項は実装ごとに少々異なる場合があります。

本項の内容

- 「CI 名のマッピングに関する考慮事項」(85ページ)
- 「双方向データ同期に関する推奨事項」(86ページ)
- 「プッシュのスケジュールに関する推奨事項」(88ページ)
- 「クラスタ化された環境でのプッシュ」(89ページ)
- 「初期ロードの構成」(91ページ)
- 「差分ロード DEM ルールを構成する方法」(95ページ)
- 「プッシュの障害検出およびリカバリ」(96ページ)
- 「Lightweight Single Sign-On (LW-SSO) 構成を有効にする方法」(97ページ)

CI 名のマッピングに関する考慮事項

UCMDB では重複する CI 名が許可されますが、Service Manager では一意の論理名が必要です。UCMDB CI をプッシュする前に、それらの CI 名の正しいマッピングを定義する必要があります。たとえば、多くの UCMDB CI (実行中のソフトウェア、CRG、スイッチ、ルータタイプの CI など) に同じ表示ラベルが付けられます。

UCMDB CI をプッシュするときに Service Manager で重複する CI 名が発生しないようにするために、出荷時設定で次のマッピングが提供されます。

CRG マッピング

出荷時設定の UCMDB CRG レコードは、次のように Service Manager にマップされます。

- CRG のクラスタが存在する場合は、CRG は次の CI の論理名にマップされます。<Cluster display label>_<CRG display label>;
- CRG のクラスタは存在しないが、複数の IP アドレスがある場合、CRG は次の論理名にマップされます (ここで IP アドレスはアルファベット順でソートされます)。
<IpAddress1>_..._<IpAddressN>.<authoritativeDnsName>_<CRG display label>
(IpAddress.authoritativeDnsName が存在する場合)

<IpAddress1>_..._<IpAddressN>_<CRG display label> (IpAddress.authoritativeDnsName が存在しない場合)
- CRG のクラスタも IP アドレスも存在しない場合、<CRG display label> に直接マップされます。

実行中のソフトウェアのマッピング

実行中のソフトウェア CI は、Service Manager CI にマップされる時に次のようにルートコンテナノード表示ラベルのプリフィックスが付けられます。<Node display label>_<Running Software display label>

スイッチとルータのマッピング

UCMDB のスイッチまたはルータタイプの CI レコードは、Service Manager CI にマップされる時に次のように MAC アドレスのプリフィックスが付けられます。<MACAddress1>_..._<MACAddressN>_<Switch or Router display label>。ここで MAC アドレスはアルファベット順でソートされます。

双方向データ同期に関する推奨事項

UCMDB-SM 統合は、UCMDB と Service Manager (SM) 間の双方向データ同期をサポートします。データプッシュおよびポップレーションの機能の誤った使用が原因で発生する不要な問題を回避するために、以下のベストプラクティスに従うことをお勧めします。

- UCMDB が自動的に検出できる CI または CI 関係の場合、UCMDB をデータソースとして使用します。それらを Service Manager で変更しないでください。代わりに常に UCMDB でそれらの変更を検出して変更を SM にプッシュするようにします。
- UCMDB が自動的に検出できない CI または CI 関係の場合、SM をデータソースとして使用します。それらを UCMDB で変更しないでください。代わりに常に SM でそれらを変更し、変更を UCMDB にポップレートします。
- SM ですでに作成され、UCMDB が自動的に検出できる CI または CI 関係の場合、ポップレーションを 1 回実行し、それらを UCMDB に同期してから UCMDB をデータソースとして使用します。

注意: これらのベストプラクティスに従わない場合、次のような問題が発生する可能性があります。

問題 1:

(ポピュレーションアダプタ): CI または CI 関係が UCMDb からプッシュされた後に、最初にそれらをポピュレーションで UCMDb に戻さずに SM でそれらのレコードを直接変更した場合、変更を UCMDb にポピュレートできません。

回避策:

これらの UCMDb レコードを SM で変更することは推奨されません。ただし、この操作が必要な場合は、次の手順を実行してこの問題を解決できます。すべてのレコードが SM にプッシュされた後で、最初に UCMDb にそれらをポピュレーションに戻してから、SM で変更を加えます。このようにすることで、変更を UCMDb にポピュレートできるようになります。

問題 2:

(ポピュレーションアダプタ): ノード CI (node 1) と実行中のソフトウェア CI の間の Composition 関係が SM にプッシュされた後で、この関係のアップストリーム CI を node 1 から node 2 に変更し、変更のポピュレーションを実行してこの変更をポピュレートした場合、実行中のソフトウェア CI が UCMDb で削除されます。

回避策:

SM で実行中のソフトウェアのコンテナを直接置き換える代わりに、UCMDb で実行中のソフトウェアを削除して、新しい実行中のソフトウェアを作成することをお勧めします。問題が発生する操作を回避できない場合は、次の手順を実行します。

関係のアップストリーム CI を node 1 から node 2 に変更した後で、変更のポピュレーションを直接実行しないようにします。この問題を回避するには、次の手順に従います。

1. SM で実行中のソフトウェア CI を更新します (または更新済みとしてマークするために単に保存します)。
2. 実行中のソフトウェア CI の変更のポピュレーションを実行します。これにより、node 2 (UCMDb 内にまだ存在していない場合) および node 2 とこの実行中のソフトウェア CI の間の新しい Composition 関係が作成されます。
3. 差分ポピュレーションを実行して、関係の変更を UCMDb に同期します。node 1 と実行中のソフトウェア CI の間の関係が削除され、手順 2 で作成した新しい関係が残ります。

関係のアップストリーム CI を node 1 から node 2 に変更した後で差分ポピュレーションを実行し、結果として実行中のソフトウェア CI が UCMDb で削除された場合は、次の手順に従って問題を解決します。

1. SM で実行中のソフトウェア CI を更新します (または更新済みとしてマークするために単に保存します)。
2. 実行中のソフトウェア CI の変更のポピュレーションを実行します。これにより、実行中のソフトウェア CI、node 2 (UCMDb 内にまだ存在していない場合)、および node 2 とこの実行中のソフトウェア CI の間の新しい Composition 関係が作成されます。

プッシュのスケジュールに関する推奨事項

プッシュジョブには2つの主要な実行方法があります。プッシュジョブを手動で実行する方法とプッシュジョブをスケジュールする方法です。

すべてのプッシュジョブは潜在的にUCMDB および SM システムに負荷を発生させる可能性があるため、HP では、次のガイドラインに従うことを推奨しています。

スケジューラの時間枠

スケジューラの「時間枠」の概念の役割を理解することが重要です。プッシュジョブを実行すると、システムのアクティビティが増加し、アプリケーションの応答時間に影響する可能性があります。ユーザが効果的にアプリケーションを対話操作できるようにするために、HP では次のガイドラインを推奨しています。

システムの負荷を減らすために、ピーク時以外の時間帯、できればシステムの使用率が最も低い時間帯にUCMDB から SM へのプッシュが実行されるようにスケジュールします。

スケジュールの頻度

スケジュールの頻度を構成するときには、ビジネス要件を認識することが重要です。スケジュールの頻度は、UCMDB と SM の間で同期する必要があるインフラストラクチャ環境の変更に依存します。

スケジュールの頻度は、最新のCI情報の利用に関するビジネス要件を基にして決定します。ほとんどの実装では、日ごとに更新する必要があります。頻繁に変更される傾向がある小規模なITシステムでスケジュールする際には、スケジュールの頻度の増加が必要になることがあります。

プッシュジョブの依存関係

UCMDB プッシュジョブは、相互の依存関係をサポートしません。各「プッシュジョブ」は、個別のタスクと見なされ、ユーザはジョブの依存関係を定義できません。たとえば、1つのジョブが別のジョブに依存したり、完了を待ってから次のジョブを実行したりすることはできません。

Service Manager に関係がプッシュされないことを回避するために、CI クエリとそれらが依存する関係クエリの両方が同じジョブ内に存在することが重要です。リスト内での各クエリの位置を変更して適切な実行順序を定義することができます。次の図に例を示します。



クラスタ化された環境でのプッシュ

クラスタ化された SM 環境は、並列的に実行される複数のサブレットと、利用可能なサブレットにユーザの依頼をディスパッチするロードバランサで構成されます。SM ロードバランサではなく、特定のサブレットをポイントするように UCMDB-SM 統合を構成する必要があります。これを行うには、最初に専用の Web サービスリスナを作成する必要があります。

本項の内容

- [「専用 Web サービス」\(89ページ\)](#)
- [「クラスタ構成プロセスの手順」\(89ページ\)](#)
- [「複数の SM プロセスの接続」\(90ページ\)](#)

専用 Web サービス

水平スケーリングまたは垂直スケーリング用に構成された Service Manager システムでは、ロードバランサを使用して、クライアントの接続要求を利用可能な SM プロセスにリダイレクトします。ただし、ほとんどの Web サービスクライアントはリダイレクト要求を処理できないため、SM ロードバランサをエンドポイント URL として使用した場合は失敗します。

HP では、Web サービス要求専用に関係する 1 つ以上の SM プロセスを作成することをお勧めします。ユーザは、専用の Service Manager プロセスに直接接続する関連する外部 Web サービスクライアントを構成する必要があります。

クラスタ構成プロセスの手順

本項では、統合のクラスタ環境を構成する手順について説明します。

Web クライアントの構成方法

関連する外部 Web クライアントを構成するには:

1. Service Manager サービスを停止します。
2. `sm.cfg` ファイルを開き、`-debugnode` パラメータを使用して、Web サービス要求をリスンする専用の SM プロセスを作成します。
次のエントリは、ポート 13085 および 13445 上でリスンする専用のプロセスを作成します。

```
01 sm -httpPort:13080 -loadbalancer
02 sm -httpPort:13081 -httpsPort:13443
03 sm -httpPort:13083 -httpsPort:13444
04 sm -httpPort:13085 -httpsPort:13445 -debugnode
```

解説

この部分的なコードは、SM クライアントが SM サービスに接続できるようにする各 SM プロセスリスタ (Web サービス) のさまざまな設定を示しています。

行 01 は、ロードバランサポート (13080) を定義します。

行 02 および 03 は、非専用 SM クライアントが SM ロードバランサによってリダイレクトされる SM ポートを定義します。

行 04 は、専用の SM クライアントによって使用される debugnode ポートを定義します。

注: debugnode パラメータ

debugnode パラメータは、この Service Manager プロセスへのクライアント接続要求を転送しないように SM ロードバランサに指示します。このプロセスに直接接続するクライアントのみがこのポートにアクセスできます。

debugnode の構成方法

debugnode を構成するには:

1. SM サービスを開始します。
2. debugnode で実行されている SM プロセスに直接接続する外部 Web サービスクライアントを構成します。UCMDB を使用して統合を実行しているときには、SM の Service Manager アダプタが debugnode ポートに接続するように構成する必要があります。
たとえば、通常の接続の場合、エンドポイント URL を次のように設定します。

```
http://<fully qualified host name>:13085/SM/9/rest/<Service Name>
```

SSL で暗号化された接続の場合、URL を次のように設定します。

```
https://<fully qualified host name>:13445/SM/9/rest/<Service Name>
```

これらのクライアントには、UCMDB (プッシュのため)、Connect-It、その他のアプリケーションが含まれます。

複数の SM プロセスの接続

パフォーマンスを向上させたい場合は、複数の Service Manager プロセスに接続できます。統合は、Service Manager の垂直ロードバランサ環境と水平ロードバランサ環境の両方をサポートします。

Web サービス要求専用の複数の SM プロセスを作成し、それらの専用の SM プロセスを使用して、統合ポイントの[URL 上書き]フィールドを構成することができます。このフィールドの値 (存在する場合は、[ホスト名/IP]および[ポート])の設定を上書きします。

次に、このフィールドの値の例を示します。2つの SM プロセスに接続します。

```
http://<fully qualified host name1>:13080/SM/9/rest;http://<fully qualified host name2>:13082/SM/9/rest
```

初期ロードの構成

構成プロセスを開始する前に、UCMDB から SM に転送される CI および関係のデータ量を評価し、データ量に基づいて必要になる反復プロセスを確認する必要があります。

最初に、すべてのデータを1回の処理でプッシュできるかどうかを確認する必要があります。これは、プッシュクエリに含まれるデータの量、およびこのデータをプッシュするのにかかる時間によって確認できます。

本項の内容

- ・ [「シングルスレッド環境でのプッシュのパフォーマンス」\(91ページ\)](#)
- ・ [「マルチスレッドの実装」\(92ページ\)](#)
- ・ [「マルチスレッド環境でのプッシュのパフォーマンス」\(93ページ\)](#)
- ・ [「複数 SM プロセス環境でのプッシュのパフォーマンス」\(93ページ\)](#)
- ・ [「初期ロード用の SM DEM ルールを設定する方法」\(94ページ\)](#)

注: この文書で示されるパフォーマンスデータは、HP で実行されたテストを基にしており、参考データとしてのみ提供されます。統合のパフォーマンスは、実際の環境ではハードウェア構成によって大きく異なる場合があります。

シングルスレッド環境でのプッシュのパフォーマンス

シングルスレッド環境で 22,500 の UCMDB ルート CI (クエリ内のルート) および関係をプッシュする場合、約 1 時間かかり、直線的に処理が実行されます。次の表を参照してください。

シングルスレッド環境でのパフォーマンスデータ

1 時間の間にプッシュされるルート CI/CI 関係の数	sm.properties でのマルチスレッドの設定
22,500	<code>number.of.concurrent.sending.threads=1</code> <code>min.objects.for.concurrent.sending=50</code> <code>number.of.chunks.per.thread=3</code> <code>recommended.min.cis.per.chunk=50</code>

UCMDB で sm.properties ファイルを表示または編集するには、[\[データフロー管理\]](#) > [\[アダプタ管理\]](#) > [\[ServiceManagerEnhancedAdapter9-x\]](#) > [\[構成ファイル\]](#) > [\[sm.properties\]](#) に移動します。

ルート CI と関係の数/22,500

特定の環境でのプッシュの時間 (時間単位) は次のように計算されます。

単一の計画済みクエリのプッシュが許可されている時間枠に違反する可能性がある場合は、データを複数のクエリに分割する必要があります。各クエリは個別にプッシュする必要があります。

このクエリの分割は、複数のクエリを作成し、データのフィルタリングを可能にする異なるノード条件を各クエリに設定することによって行います。すべてのクエリの最初のプッシュが完了したときに、初期ロードプロセスが完了します。

注: ノード条件の適用

さまざまな SM の同期クエリにノード条件を適用するときには、すべての関連データが SM にコピーされるようにすべての情報がクエリに含まれていることを確認する必要があります。

マルチスレッドの実装

パフォーマンスを向上させるために、Service Manager アダプタは、CI および関係のデータを SM にプッシュする際に複数のスレッドを使用します。次のセクションでは、これらの設定、およびパフォーマンスを最大限に向上させるための設定の構成方法について説明します。

マルチスレッドの構成は、UCMDB サーバ上の `sm.properties` ファイルで定義します。UCMDB でこのファイルを表示または編集するには、[\[データフロー管理\]](#) > [\[アダプタ管理\]](#) > [\[ServiceManagerEnhancedAdapter9-x\]](#) > [\[構成ファイル\]](#) > [\[sm.properties\]](#) に移動します。

次に、`sm.properties` ファイルのマルチスレッド定義の例を示します。

```
01 number.of.concurrent.sending.threads=6
02 min.objects.for.concurrent.sending=50
03 number.of.chunks.per.thread=3
04 recommended.min.cis.per.chunk=50
```

解説

この部分的なコードは、UCMDB サーバでの関連するマルチスレッドの設定を示しています。

- 行 01 は、CI プッシュのために UCMDB が SM に対して開く並列スレッドの数を定義します。このパラメータを 1 に設定すると、マルチスレッドが無効になり、2 以上の値にすると、マルチスレッドが有効になります。
- 行 02 は、シングルスレッドとは異なるマルチスレッドを使用するために必要な SM オブジェクトの最小数を定義します。
- 行 03 は、スレッドごとのチャンク数を定義します。この数にスレッド数を乗算すると、CI データチャンクの合計数になります。
- 行 04 は、CI データチャンクごとの推奨される CI の最小数を定義します。

チャンクの合計数 = `number.of.chunks.per.thread * number.of.concurrent.sending.threads`

統合は、次のようにキューメカニズムを実装します。

UCMDB から SM に渡されるデータは、均等なチャンクに分割され、これらのチャンクはキューに入れられます。

すべてのスレッドが利用可能になるまで、利用可能な各スレッドがキューから次のチャンクを取り出します。このプロセスが完了すると、プッシュが完了します。

このメカニズムは、各スレッドのアイドル時間が最小限になるように設計されています。各プロセスが並列でチャンクを処理するので、一部のスレッドが他のスレッドより前に完了し、そのためにスレッドが相互に待機して非効率的になる場合があります。

注意: 定義するスレッドが多すぎる場合

スレッドの数を増やしすぎると、SM サーバが過負荷状態になり、効率が低下します。非常に高性能な SM サーバでプッシュデータを処理しているエンタープライズ環境では、スレッドの数を 10 に増やすことが可能であり、場合によっては 20 に増やすこともできます。しかし、スレッドの数を増やすと、プッシュ中の SM サーバの CPU 使用率が高くなり、アプリケーションのパフォーマンスが低下する可能性があることを考慮する必要があります。

マルチスレッド環境でのプッシュのパフォーマンス

マルチスレッド環境で 60,000 の UCMDB ルート CI (クエリ内のルート) および関係をプッシュする場合、約 1 時間かかり、直線的に処理が実行されます。次の表を参照してください。

出荷時設定のマルチスレッド環境でのパフォーマンスデータ

1 時間の間にプッシュされるルート CI/CI 関係の数	sm.properties でのマルチスレッドの設定 (標準設定)
60,000	<code>number.of.concurrent.sending.threads=6</code> <code>min.objects.for.concurrent.sending=50</code> <code>number.of.chunks.per.thread=3</code> <code>recommended.min.cis.per.chunk=50</code>

特定の環境でのプッシュの時間 (時間単位) は次のように計算されます。

ルート CI と関係の数 / 60,000

複数 SM プロセス環境でのプッシュのパフォーマンス

複数 SM プロセス環境で 190,000 の UCMDB ルート CI (クエリ内のルート) および関係をプッシュする場合、約 1 時間かかり、直線的に処理が実行されます。次の表を参照してください。

複数 SM プロセス環境でのプッシュのパフォーマンスデータ

1 時間の間にプッシュされるルート CI/CI 関係の数	SM プロセス	sm.properties でのマルチスレッドの設定 (標準設定)
160,000	各ホストで4つのプロセスが実行されている2つのサーバホスト	number.of.concurrent.sending.threads=60 min.objects.for.concurrent.sending=50 number.of.chunks.per.thread=3 recommended.min.cis.per.chunk=50 データプッシュチャンクサイズ = 4000 (UCMDBでの統合設定)

統合用の複数の SM プロセスの定義の詳細については、「[UCMDB 内に統合ポイントを作成する方法](#)」(31ページ)を参照してください。

特定の環境でのプッシュの時間 (時間単位) は次のように計算されます。

ルート CI と関係の数 / 160,000

初期ロード用の SM DEM ルールを設定する方法

SM 検出イベントマネージャールール (DEM ルール) を使用すると、ユーザが SM にレポートされる各イベントタイプに対して実行する適切なアクションを定義できます。

UCMDB から SM にプッシュされる各 CI および関係レコードは、既存の SM レコードと比較して分析され、変更依頼がオープンされます。SM ルールは、SM に送信される各タイプの CI データ更新に対して実行する適切なアクションを定義します。

SM 検出イベントマネージャールールを表示または更新するには:

1. システム管理者として Service Manager にログインします。
2. [カスタマイズ]>[Web サービス]>[検出イベントマネージャールール]に移動します。
3. [Enter] キーを押すか、[検索] ボタンをクリックします。
すべての検出イベントマネージャールールのリストが表示されます。通常、各ルールは1つの CI タイプまたは同じタイプの CI のサブセットにリンクされます。
4. 個別の CI 検出イベントマネージャールールをクリックすると、詳細が表示されます。

初期ロード用の DEM ルールを設定するには:

ヒント: 初期ロードを実行するときには、以下に説明するように、SM 検出イベントマネージャールールを設定して、新しくレポートされた CI を追加することをお勧めします。これにより、膨大な数の変更またはインシデントを作成する可能性がある初期ロードの「ノイズ」が最小限に抑えられます。

各検出イベントマネージャールールについて、次の手順を実行します。

1. 関連する検出イベントマネージャールールを選択します。
2. [一致するレコードが存在しない場合のアクション]セクションに移動し、[レコードの追加]オプションを選択します。
3. [レコードは存在するが、予期しないデータが検出された場合のアクション]セクションに移動し、[結果のログの記録とレコードの更新]オプションを選択します。
4. [レコードが削除予定である場合のアクション]で、[レコードの削除]オプションを選択します。
5. 検出イベントマネージャールールレコードを保存します。

差分ロード DEM ルールを構成する方法

ヒント: CI データの「初期ロード」または「データロード」が完了したら、差分ロードの設定を適用することをお勧めします。これらの設定は、UCMDB から SM にロードされるすべてのデータに適用されます。

これらのロードでは、IT インフラストラクチャ内で検出された変更に関する更新のみを UCMDB から SM に送信します。

差分ロードの SM DEM ルールをセットアップするには:

1. システム管理者として Service Manager にログインします。
2. [カスタマイズ]>[Web サービス]>[検出イベントマネージャールール]に移動します。
3. [Enter]キーを押すか、[検索]ボタンをクリックします。
SM 内のすべての検出イベントマネージャールールのリストが表示されます。
4. 各検出イベントマネージャールールについて、次の手順を実行します。
 - a. 関連する検出イベントマネージャールールを選択します。
 - b. [一致するレコードが存在しない場合のアクション]で、新しく検出される各 CI に必要なアクションを選択します。よくわからない場合は、[レコードの追加]オプションを選択します。
 - c. [レコードは存在するが、予期しないデータが検出された場合のアクション]セクションで、変更されたために予期しないまたは正しくない結果になった各 CI に対する適切なアクションを選択します。ベストプラクティスとして、[変更のオープン]オプションを選択することをお勧めします。
 - d. [レコードが削除予定である場合のアクション]で、削除された各 CI に必要なアクションを選択します。ベストプラクティスとして、CI 関係については[レコードの削除]オプションを選択し、CI については[レコードを選択したステータスに更新]オプションを選択することをお勧めします。
 - e. 検出イベントマネージャールールレコードを保存します。

プッシュの障害検出およびリカバリ

Universal CMDB のバージョン 9.05 以降では、障害検出およびリカバリのメカニズムが提供されています。これにより、個別の CI の障害が原因でプッシュ全体が失敗することはなくなりました。Universal CMDB Studio ですべての失敗した CI を確認し、それらを再プッシュすることができます。

重複する logical.name の問題

一般的に発生する障害として、重複する論理名の問題があります。これは、Universal CMDB と Service Manager で異なる固有キーを使用することが原因になっています。Service Manager の CI logical.name は一意であり、通常は Universal CMDB の CI 表示名にマップされます (こちらは一意ではありません)。HP では、この問題を解決するために次のガイドラインに従うことをお勧めします (優先順位の高いものから順番に示します)。

- UCMDB の各表示ラベルフィールドの値が一意であることを確認します。
- これを確認できない場合は、アダプタマッピング構成ファイルで、Universal CMDB 表示ラベルと SM 論理名の直接のマッピングを使用しないようにします。
- SM 論理名を一意的な別の Universal CMDB フィールドにマップします。
- UCMDB 表示ラベルの値にプリフィックスまたはサフィックスを追加します。

注: 出荷時設定では、実行中のソフトウェアの SM 論理名は DNS 名のプレフィックスを付けてマップされます。

```
<target_mapping datatype="STRING" name="CIIdentifier"
  value="SMPushFunctions.getCIIdentifier(Root['display_label'],Root.Node*.
  getAt('display_label'))"/>

public static String getCIIdentifier(String name, def arr){
    if( fIsEmpty(arr) ){ return name;}else{
        return covertArray2String(arr)+name;
    }
    return name;
}
```

- 上記のどの方法も実行できない場合は、以下に説明するように、UCMDB の障害検出およびリカバリメカニズムと、DEM ルールの [重複ルール] 設定を合わせて使用します。

重複する論理名に対する DEM ルールをセットアップするには:

1. システム管理者として Service Manager にログインします。
2. [カスタマイズ]>[Web サービス]>[検出イベントマネージャールール]>[重複ルール] タブに移動します。
3. 各検出イベントマネージャールールについて、次の手順を実行します。
 - a. [論理名が重複している場合のアクション] セクションに移動し、[エラーを返す] オプションを選択します。

- b. 検出イベントマネージャールールレコードを保存します。

注: プッシュジョブを実行した後で、重複名の例外が発生して失敗した CI として重複する論理名を持つ CI が報告されます。Universal CMDB Studio で失敗した CI を確認し、Universal CMDB またはアダプタマッピング構成ファイル (XML および Groovy) でデータを変更してエラーを修正し、失敗した CI を再プッシュします。

Lightweight Single Sign-On (LW-SSO) 構成を有効にする方法

この統合の LW-SSO を有効にすると、ユーザは、UCMDB のユーザ名とパスワードを入力しなくとも、Service Manager Web クライアントから **[UCMDB のビュ]** ボタンをクリックして、UCMDB CI レコードを直接表示できるようになります。

注: LW-SSO は、Service Manager Windows クライアントではサポートされません。

統合の LW-SSO を有効にするには:

1. LW-SSO が必要な各 Service Manager ユーザアカウントについて、UCMDB に同じユーザ名のユーザアカウントを作成します。2つのシステムでパスワードは異なってもかまいません。
2. Service Manager Web Tier で LW-SSO を有効にします。詳細については、Service Manager のヘルプの「Service Manager Web Tier での LW-SSO の構成」を参照してください。
3. UCMDB で LW-SSO を有効にします。詳細については、『HP Universal CMDB デプロイメントガイド』を参照してください。

よく寄せられる質問

本項では、UCMDB-SM 統合に関してよく寄せられる質問の答えを示します。

本項の内容

- 「Service Manager で新しい CI が作成されるのはいつですか。」(98ページ)
- 「SM で CI が削除された理由を分析できますか。」(98ページ)
- 「UCMDB と SM の間で関係の変更を監視するにはどうすればよいですか。」(99ページ)
- 「UCMDB から SM にどのような関係がプッシュされますか。」(99ページ)
- 「ルート CI ノードとは」(99ページ)
- 「ルート関係とは」(100ページ)

- 「UCMDB-SM 統合クエリで使用される「仮想 - 複合」関係タイプとは」(100ページ)
- 「ポピュレーション機能が必要になる状況」(100ページ)
- 「物理的に削除された CI を SM から UCMDB にポピュレートできますか。」(100ページ)
- 「SM で CI 関係の停止の依存関係の設定を維持するにはどうすればよいですか。」(101ページ)
- 「XML 構成ファイルを作成する方法を教えてください。」(103ページ)
- 「[フィールドのロード] ボタンを使用して複数の管理フィールドを追加する方法を教えてください。」(104ページ)
- 「ポピュレーション構成ファイル (smPopConf.xml) の <container> 要素の目的」(104ページ)
- 「サブアイテムの削除をポピュレートできますか。」(105ページ)
- 「ポピュレーションジョブが失敗または完了した場合の処理」(105ページ)

Service Manager で新しい CI が作成されるのはいつですか。

SM で CI が作成されるのは、次のような状況です。

- 構成管理モジュールを介して CI が SM に手動で追加される場合。
- 次の条件に従って UCMDB が新しく検出された CI をレポートする場合。
 - 新しい CI がレポートされ、かつ検出イベントマネージャールールが[レコードの追加]に設定されている場合。
 - 新しい CI がレポートされ、かつ検出イベントマネージャールールが[インシデントのオープン]に設定され、インシデントがクローズされている場合。
 - 新しい CI がレポートされ、かつ検出イベントマネージャールールが[変更のオープン]に設定され、変更が検証されている場合。

SM で CI が削除された理由を分析できますか。

いいえ。

SM は、削除された CI に関する変更依頼をオープンし、次の情報を含めます。

「検出によって CI "CI 名" の削除イベントがトリガされました。」

回避策

SM 変更依頼には、削除の理由の説明は含まれませんが、UCMDB 履歴データベースから CI の削除に関する具体的な情報を抽出することができます。UCMDB データは、CI の削除を開始したユーザまたは検出パターンに関する情報を提供します。

UCMDB と SM の間で関係の変更を監視するにはどうすればよいですか。

SM での関係の変更を理解するには、次のような異なるタイプの変更を区別する必要があります。

- 関係の 2 番目のエンドポイントが変更され、関係を介して CI X が CI Y にリンクされる代わりに、CI X が CI Z にリンクされるようになる。
- 関係の属性が変更される。

最初のタイプの変更は、UCMDB-SM 統合によってサポートされるので、そのような「関係の変更」で CI 関係の更新を起動するか、インシデントまたは変更の作成を実行し、その後で確認および監視することができます。

2 つ目の関係の変更もサポートされますが、出荷時設定では対応していません。ユーザが、そのような関係の属性を公開するように Universal CMDB クエリを構成し、マップされたフィールドを公開するように Service Manager WSDL を構成して、その後で XML および Groovy でアダプタのマッピングを構成できます。ただし、そのような関係の属性の変更では、インシデントまたは変更の作成は実行できず、CI 関係の更新の直接の起動のみがサポートされます。

UCMDB から SM にどのような関係がプッシュされますか。

次の条件の下で、あらゆる種類の関係が UCMDB から SM にプッシュされます。

- この関係は、UCMDB クエリ マネージャの[Service Manager]>[Push]フォルダにあるプッシュクエリに表示されます。
- この関係は、プッシュクエリ内の **Root** という名前の関係です。
- この関係は、UCMDB 構成ファイル (XML および Groovy ファイル) で SM 内の適切なターゲットにマップされます。

UCMDB から SM にプッシュされる出荷時設定の関係は、次のような 2 つの CI 間の関係です。

- ビジネスサービスとアプリケーションの間
- ビジネスサービスとホストの間
- アプリケーションとネットワークコンポーネントの間
- ホスト、ネットワークコンポーネント、およびプリンタの間

ルート CI ノードとは

ルートノードは、TQL クエリ構造から SM へのプッシュによって作成される CI タイプを表す TQL クエリノードです。残りの TQL クエリ構造は、ルート CI タイプに組み込むことができる情報を含み、追加の情報や属性で SM 内のレコードの情報を充実させるために使用されます。

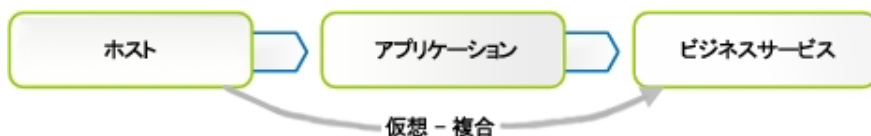
ルート関係とは

ルート関係は、クエリ内の関係で、プッシュによりSMに作成されます。2つのルート CI 間の関係を表します。ルートとしてマークされた関係のみがSMにプッシュされます。

UCMDB-SM 統合クエリで使用される「仮想 - 複合」関係タイプとは

複数のUCMDB CI エンティティが連続して接続される場合、「仮想 - 複合」関係は、最初と最後のエンティティ間の関係を表します。これは仮想関係であり、物理的な表現は存在しません。

「仮想 - 複合」関係タイプは、論理的な関係がある2つのCIタイプエンティティをリンクする関係です。次の図を参照してください。



解説

この図は、仮想 - 複合関係の例を示しています。SMの関係は、ホストとビジネスサービスの間で直接作成されます。

ポピュレーション機能が必要になる状況

次の状況で、ポピュレーション機能が必要になります。

- SMでモデリングを実行した場合。特に計画およびデザインフェーズで、モデルをUCMDBに反映する必要がある場合。
- UCMDB-SM統合を実装する必要があるが、SM CMDBにすでに投資していて、その投資を無駄にしたくない場合。
- UCMDB/ディスカバリの実装を発展させながら、SM CMDBの一部を引き続き維持する必要がある場合。

物理的に削除されたCIをSMからUCMDBにポピュレートできますか。

いいえ。

CIの物理的な削除はSMで許可されますが、SMはそのような「削除変更」を取得できず、ポピュレーション機能はそのような変更をUCMDBに同期しません。

CI の物理的な削除は、CI を誤って作成した後にのみ発生する例外と考えることができます。通常、CI の削除は、ステータスを[廃棄済み/消費済み]などに設定することで行います。そのような CI が UCMDDB にポピュレートされた場合、ユーザが UCMDDB からそれらを手動で削除する必要があります。

SM で CI 関係の停止の依存関係の設定を維持するにはどうすればよいですか。

出荷時設定では、UCMDDB から SM にプッシュされる CI 関係には、デフォルトの停止の依存関係はありません。そのような情報が必要な場合は、次のようにして CI 関係 WSDL の DEM ルールを設定できます。

1. システム管理者として Service Manager にログインします。
2. [カスタマイズ]>[Web サービス]>[検出イベントマネージャールール]に移動します。
3. ucmdbRelationship レコードを開きます。
4. [ルール]タブで、[レコードを追加して、依存関係を True に設定]を選択します。

ID:

テーブル名:

条件:

ルール **管理フィールド** インシデントのカスタマイズ 変更のカスタマイズ

一致するレコードが存在しない場合のアクション

- レコードを追加して、依存関係を True に設定
- レコードの追加
- 変更のオープン
- インシデントのオープン

これにより、各 CI 関係の停止の依存関係が true に設定され、依存するダウンストリーム CI の数が 1 に設定されます (これは、UCMDDB が 1 対 1 の CI 関係のみをサポートするためです)。

たとえばビジネスサービスから開始される関係の停止の依存関係を構成する場合など、一部の関係にのみ停止の依存関係を設定する場合は、アダプタ構成ファイル(XML)とWSDL定義を構成できます。関係タイプごとに停止の依存関係を構成することもできます(UCMDDB クエリ)。

1. WSDL 定義で、**outage.dependency** フィールドと **outage.threshold** フィールドを公開します。

外部アクセス定義

サービス名: リリース済み:
 名前: 廃止予定:
 オブジェクト名:

許可されるアクション 式 フィールド RESTful

フィールド	キャプション	タイプ
relationship.name	RelationshipName	
logical.name	ParentCI	
related.cis	ChildCIs	
relationship.type	RelationshipType	
relationship.subtype	RelationshipSubtype	
outage.dependency	OutageDependency	
outage.threshold	OutageThreshold	

2. XML ファイルで、公開される停止フィールドを設定します。たとえば、ビジネスサービス関係の停止の依存関係を true に設定し、しきい値を 1 に設定する場合、XML マッピングファイル **SM Business Service Relations Push 2.0.xml** の変更のみが必要です。この XML マッピングファイルで、次の OutageDependency および OutageThreshold の設定を使用します。

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<integration>
  <info>
    <source name="UCMDB" vendor="HP" version="10.20"/>
    <target name="SM" vendor="HP" version="9.40"/>
  </info>
  <import>
    <scriptFile path="mappings.scripts.SMPushFunctions"/>
  </import>
  <target_entities>
    <source_instance root-element-name="Root_directly" query-name="EA_SM Business Service Relations Push">
      <target_entity name="Relationship">
        <target_mapping datatype="STRING" name="RelationshipType"
          value="SMPushFunctions.getDisplayName(Root_directly['element_type'],ClassModel)"/>
        <target_mapping datatype="STRING" name="ParentCI"
          value="SMPushFunctions.getEndId(OutputCI.getExternalId().getEnd1Id())"/>
        <target_mapping datatype="STRING_LIST" name="ChildCIs"
          value="[SMPushFunctions.getEndId(OutputCI.getExternalId().getEnd2Id())]"/>
      </target_entity>
    </source_instance>
  </target_entities>
</integration>
```


```
        <target_mapping datatype="BOOLEAN" name="OutageDependency"
value="true"/>
        <target_mapping datatype="NUMBER" name="OutageThreshold"
value="1"/>

    </target_entity>
</source_instance>
</target_entities>
</integration>
```

XML 構成ファイルを作成する方法を教えてください。

XML 構成ファイルはアダプタ管理で作成します。既存の XML 構成ファイルの内容を新しいファイルにコピーし、その後で必要な編集を行うことができます。

XML 構成ファイルを作成するには:

1. 管理者として UCMDB にログインします。
2. [データフロー管理]>[アダプタ管理]に移動します。[ServiceManagerEnhancedAdapter9-x]>[構成ファイル]を選択します。
3. [リソースの新規作成]アイコン  をクリックします。
4. [新しい構成ファイル]を選択します。
5. ファイルの名前を入力します。ファイル名は、<AdapterID>/mappings/<Synch Type>/<filename> の形式にする必要があります。例: ServiceManagerEnhancedAdapter9-x/mappings/push/SM Computer Push.xml
6. [OK]をクリックします。ビジュアルマッピングツールエディタでファイルを開くかどうかを尋ねるメッセージが表示されます。
7. [はい]または[いいえ]をクリックして続行します。
UCMDB によって、アダプタの構成ファイルフォルダに新しい XML 構成ファイルが作成されます。
8. 既存の XML 構成ファイルの内容を新しいファイルにコピーします。
9. 新しいファイルに必要な編集を加えます。

注意: 無効な XML

XML ファイルから XML 要素を削除するときには、残りの要素が、UCMDB クエリ定義を変換するために使用される有効な XML ファイルを構成するように注意する必要があります。

[フィールドのロード]ボタンを使用して複数の管理フィールドを追加する方法を教えてください。

Service Managerは、ucmdbIntegration Web サービスに管理フィールドの一覧を格納します。このWebサービスは複数のWebサービスオブジェクトで構成されます。DEM ルールに管理フィールドを追加し、Service ManagerがUCMDBのより多くのCI属性の変更を監視して、関連するDEMルールで定義されたアクションをトリガできるようにすることができます。

関連付けられているWSDL定義で公開される管理フィールドをDEMルールに手動で追加することもできますが、[フィールドのロード]ボタンを使用すると、管理フィールドをDEMルールに自動的に(そのため正しく)追加できます。

1. DEM ルールの[管理フィールド]タブをクリックします。
2. [フィールドのロード]ボタンをクリックします。
3. DEM ルールレコードのテーブル([テーブル名]フィールド内)に1つのWSDL定義のみが関連付けられている場合、WSDL定義で公開されるすべてのフィールドは直ちに[管理フィールド]リストに追加されます。
次のようなメッセージが表示されます。<XX> 個の新規フィールドがロードされました。
4. テーブルに複数のWSDL定義が関連付けられている場合、管理フィールドのインポートウィザードが開き、WSDL定義(ucmdbIntegration Web サービスオブジェクト)のリストが表示されます。
 - a. 1つ以上のオブジェクトを選択し、[次へ]をクリックします。選択したWebサービスオブジェクトから追加できるすべての新しいフィールドが表示されます。
 - b. すべてのフィールドを追加する場合は、[終了]をクリックします。一部のフィールドを無視する場合は、それらの[アクション]の値を[追加]から[無視]に変更して[終了]をクリックします。
次のようなメッセージが表示されます。<XX> 個の新規フィールドがロードされました。
5. DEM ルールレコードを保存します。

ポピュレーション構成ファイル(smPopConf.xml)の<container>要素の目的

出荷時設定では、smPopConf.xmlファイルにはcontainer要素があります。

```
<sql name="SM RunningSoftware Population 2.0" citype="running_software">
  <request type="Retrieve" dataType="ci"
    resourceCollectionName="ucmdbRunningSoftwares"
    resourceName="ucmdbRunningSoftware"
    basicQueryCondition="type='runningsoftware';"
    fullQueryCondition="istatus~='Retired/Consumed';"
    changedCreationQueryCondition="created.by.date> '{fromDate}' and
istatus~='Retired/Consumed';"
    changedUpdateQueryCondition="created.by.date<= '{fromDate}' and
devicemodtime> '{fromDate}' and istatus~='Retired/Consumed';"
    changedDeletionQueryCondition="devicemodtime> '{fromDate}' and
```



```
istatus="Retired/Consumed"/>
  <container tq1="SM Computer Population 2.0"
    keyFields="CIIdentifier"
    linkTq1="SM Computer Composition Software 2.0"
    linkRetrieveCondition="downstreamci.logical.name="$$$"; and
    upstreamci.type="computer"; and
    downstreamci.type="runningsoftware"; and
    relationship.subtype="Composition";"
    linkRetrieveConditionKey="CIIdentifier"
    linkValueFields="upstreamci.logical.name"/>
</tq1>
```

- UCMDDB では、RunningSoftware CI はルートコンテナ(ノード)とともに存在する必要がありますが、Service Manager では、ノードを伴わない RunningSoftware CI が許可されます。
- 統合アダプタは、CI と関係を別々に同期します。そのため、RunningSoftware CI のポピュレーションを行うときに、統合が CI とノードの間に関係が存在するかどうかを確認する機会はありません。

統合は、<container> 要素を使用して、RunningSoftware CI を container とともにポピュレートします。

サブアイテムの削除をポピュレートできますか。

はい。

Service Manager と UCMDDB は CI 情報を異なるデータ構造に格納するため、1つの SM CI が複数の CI として UCMDDB に同期される場合があります。たとえば、ポピュレーション時に、SM のコンピュータ CI レコードが UCMDDB のノード CI に同期され、コンピュータ CI の属性が、IP、インターフェース、場所など(これらはノード CI のサブアイテムと呼ばれます)に同期されることがあります。この場合、Node CI はルート CI になります。

統合では、サブアイテムの削除を UCMDDB にポピュレートできます。たとえば、コンピュータの IP アドレス属性値を削除した場合、UCMDDB 内の対応する IP CI レコードも削除されます。

ポピュレーションジョブが失敗または完了した場合の処理

ポピュレーションジョブが失敗した場合

失敗した場合、残りのポピュレーションタスクは実行されません。次のジョブの実行は、最終成功時間から開始されます。ページネーションが発生した場合(つまりタスクが複数のページに分割された場合)、最終成功時間から最初のページ内でタスクが何度も実行されます(最初のページの末尾に到達すると、新しいタスクは実行されなくなります)。

ポピュレーションジョブが完了した場合

ジョブのステータスが[正常に完了]ではなく[完了]になると、警告が表示されます。警告が表示されても、残りのポピュレーションタスクは実行されます。次のジョブの実行では、最終成功時間から開始してすべてのタスクが再び実行されます。ページネーションが発生した場合(タスクが複数のページに分割された場合)、すべてのページ上のタスク(前回正常に完了したタスクを含む)が再実行されます。

第5章: 統合のカスタマイズ

UCMDB-SM 統合をカスタマイズすることで、管理 CI タイプ、属性、関係タイプの追加や削除を行い、ビジネスニーズに合わせることができます。本章では、統合アーキテクチャ、およびデータプッシュ、ポピュレーション、連携のオプションのカスタマイズについて説明します。

本項の内容

- [「統合のアーキテクチャ」\(106ページ\)](#)
- [「統合カスタマイズオプション」\(120ページ\)](#)

統合のアーキテクチャ

統合をカスタマイズする前に、出荷時の統合に含まれる次のコンポーネントがどのように機能するのかを把握しておく必要があります。

- [「統合のクラスモデル」\(106ページ\)](#)
- [「統合クエリ」\(106ページ\)](#)
- [「Service Manager Web サービス」\(111ページ\)](#)
- [「Service Manager 調整ルール」\(116ページ\)](#)
- [「Service Manager 検出イベントマネージャールール」\(117ページ\)](#)

統合のクラスモデル

UCMDB 9 以降では、統合 CI を管理するのに、以前のバージョンで必要だった CI のプライベートクラスモデルは使用しません。代わりに標準的なUCMDB管理オブジェクトを使用して、それらのオブジェクトをクエリと変換ファイルで Service Manager CI タイプと属性にマップします。

統合クエリ

本項では、データプッシュ、実際のステータス、およびポピュレーションに使用される出荷時設定のクエリについて説明します。

- [「プッシュのクエリ」\(107ページ\)](#)
- [「実際のステータスのクエリ」\(109ページ\)](#)

- 「ポピュレーションのクエリ」(110ページ)
- 「クエリの要件」(111ページ)

プッシュのクエリ

プッシュ機能の場合、統合は、一連のクエリを使用して、Universal CMDB から CI 属性情報を収集し、それを Service Manager システムに送信します。

これらの出荷時設定のデータプッシュクエリにアクセスするには、[モデリング]>[モデリングスタジオ]に移動し、リソースタイプの[クエリ]を選択して、[Root]>[Integration]>[Service Manager]>[Push]フォルダに移動します。

統合に含める CI タイプ、関係タイプ、または属性を変更する場合、更新された CI タイプ、CI 関係タイプ、および属性をサポートする統合クエリも編集する必要があります。

クエリ名	説明
SM Local Printer Push 2.0	このクエリは、プリンタ CI から CI 属性を収集します。Node CI タイプからも関連する CI 属性を収集します。
SM Net Printer Push 2.0	このクエリは、NodeRole に “printer” が含まれている Node CI タイプから CI 属性を収集します。 また、コンテナとリンクを経由して、次の CI タイプからも関連 CI 属性を収集します。 IPAddress、Interface、CPU、FileSystem、DiskDevice、Location
SM Mainframe Push 2.0	このクエリは、次の Node CI タイプから CI 属性を収集します。Mainframe Logical Partition および Mainframe CPC また、コンテナとリンクを経由して、次の CI タイプからも関連 CI 属性を収集します。 IPAddress、Interface、CPU、FileSystem、DiskDevice、Location
SM Mobile Device Push 2.0	このクエリは、NodeRole に “pda_handheld” が含まれている Node CI タイプから CI 属性を収集します。 また、コンテナとリンクを経由して、次の CI タイプからも関連 CI 属性を収集します。 IPAddress、Interface、CPU、FileSystem、DiskDevice、Location
SM Network Component Push 2.0	このクエリは、NodeRole に router、adsl_modem、appletalk_gateway、bandwidth_manager、cable_model、csu_dsu、ethernet、fdi、firewall、hub、kvm_switch、load_balancer、multicast_enabled_router、nat_router、token_ring、undefined_network_component、voice_gateway、voice_switch、または vpn_gateway が含まれている Node CI タイプから CI 属性を収集します。 また、コンテナとリンクを経由して、次の CI タイプからも関連 CI 属性を収集します。 IPAddress、Interface、CPU、FileSystem、DiskDevice、Location
SM Cluster Push 2.0	このクエリは、次の Node CI タイプから CI 属性を収集します。ClusterResourceGroup また、コンテナとリンクを経由して、次の CI タイプからも関連 CI 属性を収集します。 IPAddress、Interface、CPU、FileSystem、DiskDevice、Location、Cluster

クエリ名	説明
SM Computer Push 2.0	このクエリは、NodeRole に “desktop”、 “server”、 “virtualized_system” が含まれているかまたは NodeRole が設定されていない Node CI タイプから CI 属性を収集します。また、コンテナとリンクを経由して、次の CI タイプからも関連 CI 属性を収集します。 IPAddress、 Interface、 CPU、 FileSystem、 DiskDevice、 Location
SM Storage Push 2.0	このクエリは、NodeRole に san_switch、 san_gateway、 san_router が含まれているか、表示名がストレージレイである Node CI タイプから CI 属性を収集します。 また、コンテナとリンクを経由して、次の CI タイプからも関連 CI 属性を収集します。 IPAddress、 Interface、 CPU、 FileSystem、 DiskDevice、 Location
SM Running Software Push 2.0	このクエリは、実行中のソフトウェア CI から CI 属性を収集します。
SM Switch Push 2.0	このクエリは、NodeRole に atm_switch、 frame_relay_switch、 または lan_switch が含まれている Node CI タイプから CI 属性を収集します。 また、コンテナとリンクを経由して、次の CI タイプからも関連 CI 属性を収集します。 IPAddress、 Interface、 CPU、 FileSystem、 DiskDevice、 Location
SM Service Element Push 2.0	このクエリは、ビジネス要素 CI タイプから CI 属性を収集します。
SM Business Service Push 2.0	このクエリは、ビジネスサービス CI タイプから CI 属性を収集します。
SM Layer2 Topology Relations Push 2.0	このクエリは、次のコンポーネント間の関係を収集します。複数のノード。グループを通じて関係が拡張するため、クエリには複合関係が含まれます。
SM Business Service Relations Push 2.0	このクエリは、次のコンポーネント間の関係を収集します。 <ul style="list-style-type: none"> • ビジネスサービス CI と実行中のソフトウェア CI • ビジネスサービス CI と Node CI • 複数のビジネスサービス グループを通じて関係が拡張するため、クエリには複合関係が含まれます。
SM CRG Relations Push 2.0	このクエリは、次のコンポーネント間の関係を収集します。ノード CI とクラスターソースグループ CI グループを通じて関係が拡張するため、クエリには複合関係が含まれます。

クエリ名	説明
SM Node Relations Push 2.0	このクエリは、次のコンポーネント間の関係を収集します。 <ul style="list-style-type: none"> • ノード CI とプリンタ CI • ノード CI と実行中のソフトウェア CI 関係のルートクラスは composition です。

注: Service Manager 拡張汎用アダプタは、UCMDB バージョン 10.20 から導入されました。このアダプタは、古い XSLT アダプタと共存できます。XSLT アダプタのクエリと拡張アダプタのクエリを区別するために、拡張アダプタのクエリにはすべて **2.0** のサフィックスが付けられています。

実際のステータスのクエリ

出荷時設定では、UCMDB から Service Manager 構成アイテム (CI) フォームの[実際のステータス]セクションに CI 情報を取得する場合、次の表のクエリが使用されます。Service Manager では、これらのクエリに従って CI データを取得する UCMDB Web サービスを呼び出すことによって CI の実際のステータス情報を取得します。

これらのクエリは、UCMDB モデリングスタジオの[Integration]>[SM Query]フォルダにあります。

クエリ名	説明
localPrinterExtendedData	このクエリは、UCMDB のプリンタ CI からのリアルタイム拡張情報を収集します。
applicationExtendedData	このクエリは、UCMDB の実行中のソフトウェア (RunningSoftware) CI からのリアルタイム拡張情報を収集します。
businessServiceExtendedData	このクエリは、UCMDB のビジネスサービス CI からのリアルタイム拡張情報を収集します。
hostExtendedData	このクエリは、UCMDB のノード CI タイプからリアルタイム拡張情報 (Asset、Party、Location、LogicalVolume、WindowsService、Printer、InstalledSoftware、FileSystem、IPAddress、Interface、DiskDevice、Cpu など) を収集します。

連携のクエリ

汎用アダプタフレームワークの技術的な制限のために、UCMDB CI タイプと連携 CI タイプ (インシデント、問題、および Service Manager の RFC) の間の関係について記述するクエリが必要です。モデリングスタジオの[Integration]>[Service Manager]>[Federation]フォルダに連携用に定義された約 300 のクエリが用意されているので、UCMDB の出荷時設定の CI タイプ用にユーザがクエリを開発する必要はありません。通常、これらのクエリは変更する必要はありませんが、UCMDB 内の独自のカスタム CI タイプについてのみ、新しいクエリを定義する必要があります。

連携用のクエリを定義する方法については、『HP Universal CMDB 開発者向け参照情報ガイド』の「Achieving Data Federation Using the Generic Adapter (汎用アダプタを使用したデータ連携の実行)」を参照してください。

注: この技術的な制限は、UCMDB 10.20 から発生しました。将来のリリースで修正される可能性があります。

ポピュレーションのクエリ

CI/CI 関係のポピュレーションの場合、統合は以下のクエリを使用して CI/CI 関係の属性情報を UCMDB に保存します。

クエリ名	説明
SM Business Service Population 2.0	ビジネスサービス CI の CI 格納構造を定義します。
SM Business Application Population 2.0	アプリケーションサービス CI の CI 格納構造を定義します。
SM Infrastructure Service Population 2.0	インフラストラクチャサービス CI の CI 格納構造を定義します。
SM Running Software Population 2.0	実行中のソフトウェア CI の CI 格納構造を定義します。
SM Computer Population 2.0	コンピュータ CI の CI 格納構造を定義します。
SM CLIP Down Time Population 2.0	ScheduledDowntime CI の CI 格納構造を定義します。
SM Biz Containment Biz 2.0	このクエリは、bizservice CI が別の CI を含んでいる CI 関係の CI 格納構造を定義します。
SM Biz Usage Biz 2.0	bizservice CI が別の CI を使用する CI 関係の CI 格納構造を定義します。
SM Biz Containment Computer 2.0	このクエリは、bizservice CI が computer CI を含んでいる CI 関係の CI 格納構造を定義します。
SM Biz Containment Software 2.0	bizservice CI が RunningSoftware CI を含んでいる CI 関係の CI 格納構造を定義します。
SM Biz Usage Computer 2.0	bizservice CI がコンピュータ CI を使用する CI 関係の CI 格納構造を定義します。

クエリ名	説明
SM Biz Usage Software 2.0	bizservice CI が RunningSoftware CI を使用する CI 関係の CI 格納構造を定義します。
SM Computer Connects Computer 2.0	computer CI が別の CI に接続する CI 関係の CI 格納構造を定義します。
SM Computer Composition Software 2.0	RunningSoftware CI が computer CI 内に含まれ、RunningSoftware CI がコンテナなしで存在できない CI 関係の CI 格納構造を定義します。
SM CI Connection Down Time CI 2.0	ScheduledDowntime CI が影響を受ける CI に接続する CI 関係の CI 格納構造を定義します。

クエリの要件

統合では、ユーザが作成するカスタムクエリがフォーマット条件を満たす必要があります。統合に含めるとのクエリもこれらの条件を満たす必要があります:

- CI をクエリするには、クエリに Root という名前の CI タイプを 1 つ含める必要があります。この Root ノードが UCMDDB が同期するメイン CI です。その他の CI はすべて、Root CI に含まれます。
- 関係をクエリするには、クエリに Root という名前の 1 つまたは複数の関係を含める必要があります。
- クエリには、Root CI と、Root CI に直接接続している CI のみを含める必要があります。Root CI は、常にクエリ階層の最上位ノードです。
- クエリレイアウトには循環を含めることはできません。
- 関係を同期するクエリにカーディナリティがある場合、そのカーディナリティは 1...* である必要があります。カーディナリティエントリを追加する場合、エントリ間は OR 条件で連結する必要があります。
- 統合で特定 CI のみを同期する場合、その CI をフィルタ処理するクエリの条件を構成する必要があります。

Service Manager Web サービス

Service Manager は、Web サービスメッセージを使用して UCMDDB システムから CI 情報を取得して受信します。出荷時設定では、UCMDDB は、Service Manager システムが実際に管理するより多くの CI 属性情報を送信します。Service Manager ユーザは、CI レコードの[実際のステータス]セクションで、UCMDDB システムが送信する CI 属性情報をすべて確認できます。

Service Manager は、UCMDDB-SM 統合が使用するための Web サービスをいくつか公開します。UCMDDB システムは、Web サービスを使用して、UCMDDB CI タイプと CI 属性を Service Manager システムが認識する Web サービスオブジェクトにマップします。ビジュアルマッピングツールを使用して、Service Manager で管理する UCMDDB CI タイプや CI 属性を追加できます。このツールは、自動的に 1 つまたは複数の Web サービスを更新して Web サービスオブジェクトを定義します。

管理フィールド

注: 管理フィールドは、データプッシュ機能にのみ使用されます。

Service Manager 管理フィールドとは、受信する UCMDb Web サービスメッセージ内の CI 属性値と、Service Manager CI レコード内の値とをシステムが比較するフィールドのことです。Web サービスメッセージ内の値が CI レコード内の値と一致しない場合、Service Manager は検出イベントマネージャ (DEM) ルールを実行して、実行するアクションを決定します。DEM ルールは、Web サービスオブジェクトとして公開され、統合により管理されるフィールドを決定します。管理フィールドの値を変更するだけで、DEM ルールがトリガされません。

ucmdbIntegration Web サービスは、Web サービスオブジェクトのセットで構成され、それぞれのオブジェクトは Web サービスフィールドのリストを定義します。出荷時設定では、統合はそれらの一部のみを使用します (「[Service Manager Web サービスオブジェクト、テーブル、および DEM ルール間のマッピング](#)」の表を参照してください)。それらの一部 (およびそれらに関連する DEM ルール) は廃止され (「[廃止されたデータプッシュ用の ucmdbIntegration Web サービスオブジェクト](#)」の表を参照してください)、一部はポピュレーションまたは連携で使用されます (「[ポピュレーションまたは連携で使用される ucmdbIntegration Web サービスオブジェクト](#)」の表を参照してください)。

Service Manager Web サービスオブジェクト、テーブル、および DEM ルール間のマッピング

Web サービスオブジェクト	フィールドの公開元のテーブル	使用する DEM ルール ID
Relationship	cirelationship	ucmdbRelationship
ucmdbRunningSoftware	device	ucmdbRunningSoftware
ucmdbBusinessService	joinbizservice	ucmdbBusinessService
ucmdbNode	joinnode	ucmdbNode

廃止されたデータプッシュ用の ucmdbIntegration Web サービスオブジェクト

Web サービスオブジェクト	フィールドの公開元のテーブル	推奨される置換 (オブジェクト)
ucmdbApplication	device	ucmdbRunningSoftware
ucmdbComputer	ucmdbComputer	ucmdbNode
UcmdbDevice	device	ucmdbRunningSoftware
ucmdbNetwork	joinnetworkcomponents	ucmdbNode
ucmdbPrinter	joinofficeelectronics	ucmdbNode

ポピュレーションまたは連携で使用される ucmdbIntegration Web サービスオブジェクト

Web サービスオブジェクト	フィールドの公開元のテーブル	用途	DEM ルールが必要か
cirelationship1to1	cirelationship1to1	ポピュレーション	いいえ

ポピュレーションまたは連携で使用される ucmdbIntegration Web サービスオブジェクト (続き)

Web サービスオブジェクト	フィールドの公開元のテーブル	用途	DEM ルールが必要か
ucmdbIDPushBack	device	ポピュレーション	いいえ
UcmdbChange	cm3r	連携	いいえ
UcmdbChangeTask	cm3t	連携	いいえ
UcmdbIncident	Probsummary	連携	いいえ
UcmdbProblem	rootcause	連携	いいえ

以下の項では、データプッシュで使用される Web サービスオブジェクトとして公開されるフィールドの一覧を示し(「[Service Manager Web サービスオブジェクト、テーブル、および DEM ルール間のマッピング](#)」の表を参照してください)、それらが出荷時設定の Service Manager システムで管理フィールドかどうかを示します。このリファレンスを参考にして、Web サービスオブジェクトとしてフィールドを公開する必要があるかどうか、また、オブジェクトの DEM ルールを作成する必要があるかどうかを判断できます。

オブジェクト名 : Relationship

Service Manager は、cirelationship テーブルから次のフィールドを公開します。

Relationship オブジェクトの Web サービスと管理フィールド

Web サービスオブジェクトとして公開されるフィールド	Web サービスメッセージで使用されるキャプション	フィールドは管理フィールドである
relationship.name	RelationshipName	
logical.name	ParentCI	
related.cis	ChildCIs	はい
relationship.subtype	RelationshipSubtype	

オブジェクト名 : ucmdbRunningSoftware

Service Manager は、device テーブルの次のフィールドを公開します。

ucmdbRunningSoftware オブジェクトの Web サービスと管理フィールド

Web サービスオブジェクトとして公開されるフィールド	Web サービスメッセージで使用されるキャプション	フィールドは管理フィールドである
ucmdb.id	UCMDBId	
ci.name	ApplicationName	はい
type	Type	

ucmdbRunningSoftware オブジェクトの Web サービスと管理フィールド (続き)

Web サービスオブジェクトとして公開されるフィールド	Web サービスメッセージで使用されるキャプション	フィールドは管理フィールドである
subtype	Subtype	
company	CompanyId	
logical.name	CIIdentifier	はい
product.version	ProductVersion	
vendor	Vendor	
version	Version	
id ¹	CIName	

オブジェクト名 : ucmdbBusinessService

Service Managerは、joinbizservice テーブルの次のフィールドを公開します。

ucmdbBusinessService オブジェクトの Web サービスと管理フィールド

Web サービスオブジェクトとして公開されるフィールド	Web サービスメッセージで使用されるキャプション	フィールドは管理フィールドである
ucmdb.id	UCMDBId	
ci.name	ServiceName	はい
type	Type	
subtype	Subtype	
company	CustomerId	
logical.name	CIIdentifier	はい
vendor	ServiceProvider	
id ²	CIName	

オブジェクト名 : ucmdbNode

Service Managerは、joinnode テーブルの次のフィールドを公開します:

¹この属性は、ポピュレーション機能でのみ使用されます。

²この属性は、ポピュレーション機能でのみ使用されます。

ucmdbNode オブジェクトの Web サービスと管理フィールド

Web サービスオブジェクトとして公開されるフィールド	Web サービスメッセージで使用されるキャプション	フィールドは管理フィールドである
ucmdb.id	UCMDBId	
type	Type	
subtype	Subtype	
company	CustomerId	
logical.name	CIIdentifier	はい
default.gateway	DefaultGateway	はい
network.name	DNSName	はい
building	Building	はい
room	Room	はい
floor	Floor	はい
location	Location	
addIIPAddr[addIIPAddress]	AddIIPAddress	はい
addIIPAddr[addISubnet]	AddISubnet	はい
addIMacAddress	AddIMacAddress	はい
bios.id	BIOSId	はい
operating.system	OS	はい
os.version	OSVersion	はい
physical.mem.total	PhysicalMemory	はい
serial.no.	SerialNo	
vendor	Vendor	
cpu[cpu.id]	CpuID	
cpu[cpu.name]	CpuName	

ucmdbNode オブジェクトの Web サービスと管理フィールド (続き)

Web サービスオブジェクトとして公開されるフィールド	Web サービスメッセージで使用されるキャプション	フィールドは管理フィールドである
cpu[cpu.clock.speed]	CpuClockSpeed	
file.system[mount.point]	MountPoint	
file.system[disk.type]	DiskType	
file.system[file.system.type]	FilesystemType	
file.system[disk.size]	DiskSize	
asset.tag	AssetTag	
machine.name	HostName	はい
disk.device[model.name]	ModelName	
disk.device[disk.vendor]	DiskVendor	
disk.device[disk.name]	DiskName	
id ¹	CIName	
isVisualization	IsVisualization	
istatus	AssetStatus	

Service Manager 調整ルール

Service Manager 調整ルールにより、統合は UCMDB システム内の CI に一致する Service Manager システム内の CI レコードを特定できます。Service Manager は、UCMDB システムからの CI 属性の各ブッシュで CI レコードを調整します。統合は、次のワークフローを使用して UCMDB CI を Service Manager CI と照合します。

1. UCMDB システムは、Service Manager に最新の CI 属性データを含む Web サービスメッセージを送信します。
2. Service Manager は、Web サービスメッセージをスキャンして CI ucmdb.id 値を取得します。

注: 出荷時設定の Service Manager では、ユーザが値を変更できないようにするため、CI レコードフォームに ucmdb.id フィールドは表示されません。フォームにこの値を追加するには、

¹この属性は、ポピュレーション機能でのみ使用されます。

device テーブルで定義されている `umcdb.id` フィールドを検索します。HP では、このフィールドを読み取り専用フィールドにしておくことを推奨します。

3. Service Manager は、同じ `umcdb.id` 値の既存の CI レコードを検索します。
4. `umcdb.id` 値が一致する CI が見つかった場合は、調整は必要ありません。Service Manager は UCMDB CI 属性と Service Manager 管理フィールドとを比較し、必要に応じて適切な検出イベントマネージャ (DEM) ルールを実行します。
5. `umcdb.id` 値が一致する CI が見つからなかった場合は、調整ルールを実行します。
6. Service Manager は、同じ調整フィールド値の既存の CI レコードを検索します。
7. 調整フィールド値が一致する CI が見つかったら、Service Manager は一致する UCMDB CI の `umcdb.id` 値で CI レコードを更新します。Service Manager は UCMDB CI 属性と Service Manager 管理フィールドとを比較し、必要に応じて適切な検出イベントマネージャ (DEM) ルールを実行します。
8. 調整フィールド値が一致する CI が見つからなかった場合、Service Manager は「一致するレコードが存在しない場合のアクション」の DEM ルールを実行します。出荷時設定の DEM ルールでは、Service Manager は新規 CI レコードを作成します。CI レコードの作成には、受信した UCMDB CI の `umcdb.id` 値が使用されます。

パフォーマンスについて

Service Manager は各プッシュで CI の調整を試みるので、調整フィールドの数が統合のパフォーマンスに影響を与えます。調整ルールが多いほど、Service Manager が CI を照合するのに必要となる検索が多くなります。調整の検索パフォーマンスを改善するには、基礎となる Service Manager テーブルの一意なキーである調整フィールドを選択する必要があります。たとえば、**device** テーブルの CI レコードを調整する場合、一意なキーである `logical.name` フィールドを調整フィールドとして使用します。調整ルールの作成方法の詳細については、「[DEM 調整ルールの追加方法](#)」(125ページ)を参照してください。

DEM ルールの使用

Service Manager は、CI を調整できないときには常に「一致するレコードが存在しない場合のアクション」DEM ルールを使用します。UCMDB から Service Manager に最初の CI のプッシュを行う前に、DEM 設定を確認し、DEM 設定が会社でのビジネス基準を満たしていることを確認してください。たとえば、Service Manager で「[変更のオープン](#)」オプションを選択することで、最初の CI プッシュで各 CI の変更依頼が作成されるようにします。

Service Manager 検出イベントマネージャルール

検出イベントマネージャ (DEM) ルールを作成するだけで、次のカスタムアクションを実行できるようになります。

- [「DEM ルールが実行される条件の変更」\(118ページ\)](#)
- [「DEM ルールが実行するアクションの変更」\(118ページ\)](#)
- [「変更またはインシデントのレコードをオープンするカスタム JavaScript の作成」\(119ページ\)](#)

DEM ルールが実行される条件の変更

Service Manager は、条件フィールドが true に評価される場合にのみ DEM ルールを実行します。出荷時設定では、どの DEM ルールにもルールが実行される状況を限定する条件文はなく、すべての統合 DEM ルールはデフォルトで常に実行されます。

Service Manager が DEM ルールを実行する状況を限定するには、DEM ルールの条件文を更新します。たとえば、ucmdbNode DEM ルールに次の条件を追加すると、ルールはデスクトップ CI のみに限定されます。

```
subtype in $L.file="Desktop"
```

条件フィールドを使用して、同じテーブル名に適用される複数の DEM ルールを作成することもできます。たとえば、次の DEM ルールは両方とも joinnode テーブルに適用されます。

異なる条件を使用して、同じテーブルに影響を与える DEM ルール

DEM ルール ID	テーブル名	条件
ucmdbNode	joinnode	subtype in \$L.file!="Desktop"
ucmdbDesktop	joinnode	subtype in \$L.file="Desktop"

通常、条件を追加しなければならないのは、ビジネスプロセスにより統合で特定の CI タイプや SLA に異なるアクションが必要となる場合のみです。

DEM ルールが実行するアクションの変更

出荷時設定では、統合の DEM ルールは次のアクションを実行します:

- UCMDDB データが既存の Service Manager CI レコードに一致しない場合に CI レコードを追加
- UCMDDB CI 属性データが Service Manager CI レコードの CI 属性データと一致しない場合に、変更をオープン、または結果をログ記録して CI レコードを更新
- UCMDDB データに CI が削除されたことが示されている場合に CI レコードを削除

ビジネスプロセスに合わせて、統合の DEM ルールを変更できます。たとえば、統合が予期しないデータのある非デスクトップ CI を検出したときに ucmdbNode DEM ルールを使用して変更をオープンするように設定したり、統合が予期しないデータのあるデスクトップ CI を検出したときに ucmdbDesktop DEM ルールを使用して結果をログに記録し、レコードを更新するように設定することができます。

注意: 変更管理の検証と統合の変更管理の検証機能を使用する場合は、DEM ルールで「レコードは存在するが、予期しないデータが見つかった場合のアクション」イベントに[変更のオープン]オプションを使用する必要があります。

変更またはインシデントのレコードをオープンするカスタム JavaScript の作成

Service Manager が変更またはインシデントのレコードをオープンするときには、**discoveryEvent** JavaScript を使用して CI 名を作成し、必須フィールドの値を設定します。出荷時設定では、このスク립トは次のデフォルト値を使用します。

新規 CI を作成するためのデフォルト値

createCIName と populateNewCI 関数を更新して、次の CI の値を設定できます。

新規 CI を作成するのに使用されるデフォルト値

CI 属性	discoveryEvent で定義されているデフォルト値
record.logical_name	システムが生成する ID 番号
record.assignment	AUTO
record.istatus	Installed
record.os_name	record.operating_system の値

新規変更を作成するためのデフォルト値

populateChange 関数を更新して、次の変更の値を設定できます。

新規変更を作成するのに使用されるデフォルト値

CI 属性	discoveryEvent で定義されているデフォルト値
change.category	Unplanned Change
change.reason	reason の値
change.initial_impact	3
change.severity	3
change.coordinator	Change.Coordinator
change.requested_by	discovery
change.status	initial

新規インシデントを作成するためのデフォルト値

populateIncident 関数を更新して、次のインシデントの値を設定できます。

新規インシデントを作成するのに使用されるデフォルト値

CI 属性	discoveryEvent で定義されているデフォルト値
incident.category	incident
incident.subcategory	hardware
incident.product_type	missing または stolen
incident.assignment	Hardware
incident.initial_impact	3
incident.severity	3
incident.logical.name	ID の値
incident.site_category	C
incident.contact_name	ANALYST, INCIDENT
incident.affected_item	MyDevices

統合カスタマイズオプション

統合には、次のカスタマイズオプションがあります:

- 「[統合アダプタ構成ファイル\(sm.properties\)を更新する方法](#)」(121ページ)
- 「[DEM調整ルールの追加方法](#)」(125ページ)
- 「[検出イベントマネージャールールを追加する方法](#)」(127ページ)
- 「[データプッシュ用のCI属性を統合に追加する方法](#)」(131ページ)
- 「[データプッシュ用のCIタイプを統合に追加する方法](#)」(145ページ)
- 「[データプッシュ用のCI関係タイプを統合に追加する方法](#)」(160ページ)
- 「[統合ジョブにカスタムクエリを追加する方法](#)」(166ページ)
- 「[ポピュレーション用のCIタイプ、属性、または関係タイプを統合に追加する方法](#)」(167ページ)
- 「[CIタイプのUCMDB IDプッシュバックを有効または無効にする方法](#)」(168ページ)
- 「[連携用のサポートされるCIタイプの属性を追加する方法](#)」(170ページ)

統合アダプタ構成ファイル(sm.properties)を更新する方法

統合では、アダプタの構成ファイルとしてプロパティファイル(sm.properties)を使用します。出荷時設定でこのファイルはベストプラクティスに基づいてセットアップされているので、通常はデフォルトのパラメータ値をそのまま使用できます。オプションで、ニーズに合わせてパラメータ値を更新することもできます。

sm.properties ファイルを更新するには:

1. 管理者として UCMDB にログインします。
2. [データフロー管理]>[アダプタ管理]>[ServiceManagerEnhancedAdapter9-x]>[構成ファイル]に移動します。
3. プロパティ構成ファイル sm.properties をクリックします。
4. 必要に応じて、カーネルパラメータを更新します。パラメータのリストについては、下の表を参照してください。

#true の場合は、SM 統合で、ucmdbId の代わりに globalId を使用する。

```
use.global.id=true
```

#タイプ0: 機能が無効になる

#タイプ1: 列挙型が "{value}" に展開される

#タイプ2: 列挙型が "{index}-{value}" に展開される

```
type.of.expand.enum=2
```

#false の場合、型のラベルの代わりに型を直接使用する。

```
use.type.label=true
```

#Service Manager からデータをポピュレートするためのパラメータ

```
pop.pagination.switch=on
```

```
pop.pagination.recordcount=1000
```

```
pop.createci.key=sm_id
```

xslt ファイルでポピュレーションのための UCMDB ID キーとして使用されるプロパティ。global.id が true の場合はアダプタによって "global_id" として扱われ、それ以外の場合は CMDB ID として扱われる。

```
pop.ucmdb.id.key=ucmdb_id
```

#UCMDB ID をプッシュバックするための SM Web サービス

```
ucmdbid.pushback.request=UpdateucmdbIDPushBackRequest
```

ucmdbid.pushback.xslt=ucmdbid_pushback.xslt

#ジョブを実行する前に SM インスタンスへの SOAP 接続を確認するかどうか

check.sm.connections=false

#外部 Web サービスのインタフェースタイプ。restful または soap

connector.type=restful

sm.properties ファイルのパラメータ

パラメータ	デフォルト値	コメント
timeout.minutes	10	統合の接続タイムアウト値 (分)
number.of.concurrent.sending.threads	6	データプッシュ機能に使用される同時スレッドの数 <ul style="list-style-type: none"> • 1: 無効 • 2 以上: 有効 <div style="border: 1px solid gray; padding: 5px; margin-top: 10px;"> 注: CI データプッシュのパフォーマンスを向上させるために複数の Service Manager インスタンスに接続する場合 (「UCMDB 内に統合ポイントを作成する方法」(31ページ)の URL 上書きの設定を参照してください)、パフォーマンスを最適化するためにこの値を増やすことが推奨されます。たとえば、2つの Service Manager インスタンスに接続する場合は、12に設定します。 </div>
min.objects.for.concurrent.sending	50	シングルスレッド送信の代わりに同時送信を使用するために必要な Service Manager オブジェクトの最小数 <div style="border: 1px solid gray; padding: 5px; margin-top: 10px;"> 注: プッシュ機能で使用されます。 </div>
number.of.chunks.per.thread	3	プッシュ機能に使用されるスレッドごとのチャンクの数 <p>チャンクの合計数 = number.of.chunks.per.thread * number.of.concurrent.sending.threads</p>

sm.properties ファイルのパラメータ (続き)

パラメータ	デフォルト値	コメント
recommended.min.cis.per.chunk	50	<p>CI データチャンクごとの許可される CI の最小数。この値は、データチャンクが小さくなりすぎるリスクを軽減するために使用されます。</p> <p>例えば、システムで次の設定を使用すると仮定します。</p> <ul style="list-style-type: none"> • number.of.chunks.per.thread=3 • number.of.concurrent.sending.threads=3 • recommended.min.cis.per.chunk=50 <p>1000 個の CI をプッシュする場合、各チャンク内の CI の数は 1000/3/3 として計算されます。計算結果が recommended.min.cis.per.chunk 値よりも大きいので、計算結果が使用されません。その一方、270 個の CI をプッシュする場合は、計算結果が recommended.min.cis.per.chunk 未満なので計算結果が使用されます。</p>
max.running.hours.for.multi.threaded	20	<p>マルチスレッド環境でプッシュ要求がタイムアウトになるまでの最大時間数。</p>
number.of.cis.per.request	1000	<p>ID ごとに Service Manager から取得されるオブジェクトの最大数</p> <div style="border: 1px solid gray; padding: 5px; margin-top: 10px;"> <p>注意: ポピュレーションおよび連携機能で使用されます。依頼に 64 K の制限がある場合は、1000 より大きい値に設定しないでください。</p> </div>
federation.request.pagination.switch	on	
federation.request.max.size	2000	<p>連携のクエリ条件のサイズ</p>
federation.isin.max.count	500	<p>isin{} サイズのサイズ</p>

sm.properties ファイルのパラメータ (続き)

パラメータ	デフォルト値	コメント
type.of.expand.enum	2	<p>UCMDB の列挙型の値のマッピングルールを構成します</p> <ul style="list-style-type: none"> 0: 機能が無効になります 1: 列挙型が“{value}”に展開されます。 2: 列挙型が“{index}-{value}”に展開されます。 <p>注: プッシュ機能で使用されます。</p>
op.pagination.switch	on	<p>ページネーション (クライアント駆動) が有効かどうかを示します</p> <ul style="list-style-type: none"> on: 有効。 off: 無効。 <p>注: ポピュレーション機能で使用されます。</p>
pop.pagination.recordcount	1000	<p>ページネーションが有効な場合に各ページに表示されるレコードの最大数。</p> <p>注: ポピュレーション機能で使用されます。</p>
pop.createci.key	sm_id	<p>Service Manager CI ID を格納する CI レコードの UCMDB フィールド。</p> <p>注: ポピュレーション機能による UCMDB ID のプッシュバックで使用されます。</p>

sm.properties ファイルのパラメータ (続き)

パラメータ	デフォルト値	コメント
ucmdbid.pushback.request	UpdateucmdbID PushBackRequest	UCMDB ID を Service Manager にプッシュバックするための Web サービス依頼。 注: ポピュレーション機能で使用されます。
check.sm.connections	false	ジョブを実行する前に Service Manager インスタンスへの SOAP 接続を確認するかどうかを示します。 次のいずれかの状況で、このオプションを有効にすることができます。 <ul style="list-style-type: none"> Service Manager が高可用性モードで (ロードバランスを使用して) 実行され、UCMDB を複数の Service Manager インスタンスに接続する必要がある場合。 利用可能な統合接続がないときには、ジョブを実行してから失敗をレポートするのではなく、UCMDB でジョブを実行しないようにする場合。

DEM 調整ルールの追加方法

Service Managerシステムには、UCMDBシステム内の CI と一致する CI レコードがすでに含まれている場合があります。Service Manager システムに重複する CI レコードを追加するのではなく、Service Manager を構成して、特定のフィールド値を基に両システム間で CI レコードが調整されるようにします。

Service Managerは必ず、Service Managerテーブルの一意のキーフィールドおよびucmdb.idフィールドを基にして、CI レコードの調整を試みます。[DEM 調整ルール]フォームで、調整の基準とする追加フィールドを指定できます。Service Manager がこれらのフィールドのいずれかに一致する値を検出すると、受信する UCMDB レコードの属性で Service Manager CI レコードを更新します。

マルチテナンシーが有効である場合、Service Manager は、データプッシュジョブの会社 ID と一致する会社 ID を持つ CI のみを調整します。たとえば、会社 2 から CI をプッシュする際は、会社番号 2 に対応する会社コードの Service Manager CI レコードにのみ調整ルールが適用されます。

調整フィールドを指定するには、Service Manager と UCMDB システムの両方でテーブルとフィールド名を理解する必要があります。UCMDBシステムの特定の属性について調整する場合、その属性に対応する Service Manager 管理フィールドが存在することを確認する必要があります。このようなマッピングが存在しない場合、Service Managerは、CI レコード内で一致する値を検索することができません。

注: Service Manager内に対応するフィールドが存在しないUCMDB属性があります。対応するフィールドが存在しない場合、Service Managerシステムをカスタマイズして、一致フィールドの追加が必要となる場合があります。

調整での結合テーブルの使用

調整ルールを設定するときに、調整するデバイスタイプに (**devtype** テーブルで定義される) **joindef** 定義がある場合は、**device** テーブルではなく結合テーブル名を使用します。たとえば、コンピュータ CI を調整する場合は、**device** テーブルではなく **joincomputer** テーブルを使用します。

調整の順序

調整ルールでは、一致する CI 値を検索する Service Manager テーブルとフィールドを指定します。また、Service Manager が調整ルールを処理する順序も指定します。デフォルトで、Service Manager はフィールド名のアルファベット順にルールを処理します。たとえば、Service Manager は、**ci.name** フィールドで CI を調整する前に、**asset.tag** フィールドで CI を調整します。

Service Manager が CI を調整する順序を変更するには、シーケンスフィールドに数値を追加します。たとえば、次の調整ルールを使用することで、Service Manager は **ci.name** フィールドで CI を処理してから、**asset.tag** フィールドで CI を調整するようになります。

シーケンスで順番付けられた調整ルールのサンプル

テーブル名	フィールド名	シーケンス
joincomputer	ci.name	1
joincomputer	asset.tag	2

検出イベントマネージャ (DEM) 調整ルールでは、既存の CI レコードが UCMDB システムにある CI と一致するかどうかを判断するのに使用する Service Manager フィールドを指定できます。管理者は通常、Service Manager によって重複する CI レコードが作成されないように、UCMDB レプリケーションジョブを開始する前に調整ルールを指定します。

DEM 調整ルールを作成するには:

1. システム管理者として Service Manager にログインします。
2. [カスタマイズ]>[Web サービス]>[DEM 調停ルール]をクリックします。[DEM 調停レコード]フォームが表示されます。
3. [テーブル名]に、調停するフィールドを含む Service Manager テーブルの名前を入力します。
4. [フィールド名]に、調停する値を含む Service Manager フィールドの名前を入力します。
5. [シーケンス]に、Service Manager がこのルールを実行する順序を指定します。

注: シーケンス値を指定しないと、Service Manager はフィールド名のアルファベット順に処理します。

6. [新規]をクリックします。調整ルールが作成されます。

検出イベントマネージャルールを追加する方法

Service Manager は、検出イベントマネージャ (DEM) を使用して、受信する構成アイテム (CI) レコードの実際のステータスが HP Service Manager 内の管理ステータスと異なる場合に、システムが実行すべきアクションを定義します。DEM ルールを使用することで、Service Manager システムが受信する UCMDDB データを基にして CI レコードの追加、更新、または削除を行うかどうかを定義できます。

CI レコードのみの場合、DEM ルールを使用して、Service Manager で重複する論理名を処理する方法を定義することもできます。

Service Manager で DEM ルールにアクセスするには、[カスタマイズ]>[Web サービス]>[検出イベントマネージャルール]に移動し、[検索]をクリックして既存のルールを表示するか、[新規]をクリックして新しいルールを作成します。

本項の内容

- [「DEM ルール」\(127ページ\)](#)
- [「重複ルール」\(129ページ\)](#)
- [「変更およびインシデントのレコードに表示される CI 属性」\(130ページ\)](#)
- [「統合によってオープンされた変更およびインシデントのレコードの検索」\(131ページ\)](#)

DEM ルール

Service Manager では、以下のルールオプションを利用できます：

一致するレコードが存在しない場合のアクション

これは、一致する CI レコードが見つからない場合に Service Manager が実行するアクションです。

- **レコードの追加** : (デフォルト) Service Manager は、一致するレコードが見つからないときに CI レコードを追加します。Service Manager が CI レコードの照合に使用するフィールドを定義するには、[「DEM 調整ルール追加方法」\(125ページ\)](#)を参照してください。
- **レコードを追加して、依存関係を True に設定** : このオプションは、CI 関係データの同期にのみ利用可能です。Service Manager は、CI 関係レコードを追加し、次の処理を実行して、レコードの停止の依存関係を有効にします。
 - [停止の依存関係] チェックボックスをオンにします。
 - 依存するダウンストリーム CI の数を 1 に設定します。これは、UCMDDB が 1 対 1 の CI 関係のみを

サポートするためです。

構成アイテム関係

アップストリーム CI	* E-mail / Webmail (Europe)
関係名	* E-mail / Webmail (Europe) Service
関係タイプ	<input checked="" type="radio"/> 論理 <input type="radio"/> 物理
関係サブタイプ	* Containment
ダウンストリーム CI	* adv-eur-server-mail Microsoft Outlook

停止の依存関係

停止の依存関係

この構成アイテムは

* 個以上のサポート構成アイテムがダウンした場合にダウンとみなされます

- インシデントのオープン:** Service Manager は、ユーザが新規 CI レコードを確認できるように、インシデントをオープンします。このインシデントにより、ユーザは新規 CI レコードがビジネスプラクティスに準拠しているかどうかを調査できます。
- 変更のオープン:** Service Manager は、ユーザが新規 CI レコードを確認できるように、計画外の変更をオープンします。この変更により、ユーザは新規 CI レコードがビジネスプラクティスに準拠しているかどうかを調査できます。CI レコードが準拠している場合、変更を承認できます。CI レコードが準拠していない場合、変更を却下して CI レコードを削除できます。変更レコードには、現在の属性値と推奨される属性値の両方が一覧されます。

レコードは存在するが、予期しないデータが検出された場合のアクション

これは、一致する CI 属性値が見つからない場合に Service Manager が実行するアクションです。

- 変更のオープン:** (デフォルト) Service Manager は、CI レコードの実際のステータスを確認するために、予定外の変更をオープンします。この変更により、ユーザは新規属性値がビジネスプラクティスに準拠しているかどうかを調査できます。値が準拠している場合、変更を承認できます。値が準拠していない場合、変更を却下して、CI 属性値を元の管理ステータスに戻すことができます。
- 結果のログの記録とレコードの更新:** Service Manager は、CI レコードの実際のステータスの結果をログに記録し、CI レコードを更新します。
- インシデントのオープン:** Service Manager がインシデントを開いて、CI レコードの実際のステータスを調査し、レコードを Service Manager に適合させるために実行または開始する必要があるアクションを決定します。

レコードが削除予定である場合のアクション

これは、外部イベントがレコードを削除する必要があると指定している場合に、Service Manager が実行するイベントです。

- **レコードの削除**: (CI 関係レコードのデフォルト) このオプションは、CI および CI 関係レコードの両方の同期で利用可能です。Service Manager は、CI/CI 関係レコードを自動的に削除します。
- **インシデントのオープン**: このオプションは、CI 関係レコードの同期にのみ利用可能です。Service Manager は、削除されるレコードを調査するためにインシデントを開き、レコードを Service Manager に準拠させるために実行や開始が必要となるアクションを判断します。
- **変更のオープン**: このオプションは、CI 関係レコードの同期にのみ利用可能です。Service Manager は、削除される CI レコードを確認させるため、計画外の変更をオープンします。この変更により、ユーザは削除されるレコードがビジネスプラクティスに準拠しているかどうかを調査できます。レコードが準拠している場合、変更を承認できます。CI レコードが準拠していない場合、変更を却下して CI レコードをシステムに戻すことができます。
- **レコードを選択したステータスに更新**: (CI レコードのデフォルト) このオプションは、CI レコードの同期にのみ利用可能です。Service Manager は、レコードを完全に削除する代わりに、CI レコードのステータスをドロップダウンリストから選択された値 ([**廃棄済み**]/**消費済み**) などに更新します。

注: ドロップダウンリストで選択できる値は、[ICM ステータス] グローバルリストで定義されます。

- **変更をオープンして、レコードを選択したステータスに更新します**: このオプションは、CI レコードの同期にのみ利用可能です。Service Manager は、予定外の変更をオープンし、CI レコードのステータスをドロップダウンリストから選択された値 ([**廃棄済み**]/**消費済み**) などに更新します。この変更により、ユーザは要求されたステータス変更がビジネスプラクティスに準拠しているかどうかを調査できます。変更が承認されてクローズされると、Service Manager が CI レコードを選択されたステータスに自動的に変更します。変更が否認された場合は、Service Manager は CI レコードを変更しません。
- **インシデントをオープンして、レコードを選択したステータスに更新します**: このオプションは、CI レコードの同期にのみ利用可能です。Service Manager は、インシデントをオープンし、レコードのステータスをドロップダウンリストから選択された値 ([**廃棄済み**]/**消費済み**) などに更新します。インシデントが承認されてクローズされると、Service Manager が CI レコードを選択されたステータスに自動的に更新します。

重複ルール

UCMDB は、完全に別々でありながら偶然同じ「名前」になった 2 つの有効な CI レコードを作成することがあります。UCMDB の名前フィールドは、Service Manager の logical.name フィールド (こちらは一意にする必要があります) にマップされます。この 2 つの CI レコードを Service Manager にプッシュすると、重複する論理名の問題が発生します。この問題を回避する方法がいくつかあります。次の表を参照してください。

重複する論理名の問題の解決策

製品側	解決策
UCMDB	<p>UCMDB で直接名前を変更するか、UCMDB で調整ルールを変更して、同じ名前にならないようにします。</p> <p>この方法を強くお勧めします。</p> <p>次のいずれかの方法を使用し、統合アダプタのマッピング構成 (xslt) ファイルで UCMDB の名前フィールドが SM の論理名フィールドに直接マッピングされることを回避します。</p> <ul style="list-style-type: none"> UCMDB の別の一意の属性を SM の logical.name フィールドにマップし、UCMDB の名前フィールドを別の SM フィールドにマップします。 名前にプリフィックスを追加します。以下に例を挙げます。 <ul style="list-style-type: none"> UCMDB のスイッチまたはルータには、単純に “Router” または “Switch” という名前を付け、基になる MAC アドレスで識別します。それらの「SM 論理名」が “<MAC> + <name>” になるように構成できます。 UCMDB では、複数のデータベースに同じ名前が付けられることがよくあります (クラスターおよび Oracle RAC の実装が原因になっています)。それらの「SM 論理名」が “<full DNS name> + <name>” になるように構成できます。
Service Manager	Service Manager で DEM ルールの重複ルールオプションを使用します。

Service Manager の[重複ルール]タブには以下の重複ルールオプションがあり、各 DEM ルールで “cirelationship” 以外のテーブル名を使用します。

- 論理名が重複している場合のアクション (異なる uCMDB ID の CI): これは、CI レコードが追加または更新されたときに、その論理名が別の CI レコードによってすでに使用されている場合に Service Manager が実行するアクションです。
 - <name>_[RENAMED]_1/2/3 に名前を変更: (デフォルト) Service Manager でサフィックスを追加して論理名を変更します。
 - エラーを返す: Service Manager から UCMDB に重複キーエラーを返します。

変更およびインシデントのレコードに表示される CI 属性

Service Manager では UCMDB-SM 統合を通じて CI 属性の変更を検出したときに、変更レコードやインシデントレコードを開くように DEM を構成すると、対応する変更の[変更の詳細]セクション、または対応するインシデントの[CMDB 変更]セクションが表示されます。Service Manager では、UCMDB-SM 統合が有効であり、検出イベントマネージャで、CI の追加、更新、または削除の際に変更レコードやインシデントレコードを作成するルールを定義している場合のみ、CI 属性のタブが表示されます。

[変更の詳細]セクションと[CMDB 変更]セクションの両方に、UCMDB によって検出された実際の属性値と一緒に現在の CI 属性値が表示されます。この情報を使用して、変更の承認や却下を行ったり、適切な担当グループにインシデントをエスカレートできます。

統合によってオープンされた変更およびインシデントのレコードの検索

次の検索基準を用いて、UCMDB-SM 統合によってオープンされた変更およびインシデントのレコードを検索できます。

変更レコードとインシデントレコードに利用できる検索オプション

レコードタイプ	利用できる検索オプション
変更	カテゴリが「unplanned change」のレコードを検索します。
インシデント	[UCMDB 統合により生成] オプションを使用してレコードを検索します。

データプッシュ用の CI 属性を統合に追加する方法

次の手順を用いて、統合に CI 属性を追加できます。

- UCMDB クラスモデルに CI 属性はすでに存在していますか?
はい。手順 3 に進みます。

いいえ。手順 2 に進みます。
- UCMDB クラスモデルに CI 属性を追加します。
「CI 属性を UCMDB クラスモデルに追加する方法」(131ページ)を参照してください。
- クエリのレイアウトに CI 属性を追加します。
「CI 属性をクエリレイアウトに追加する方法」(133ページ)を参照してください。
- Service Manager CI タイプ用の Web サービスフィールドを追加します。
「Service Manager CI タイプの Web サービスフィールドを追加する方法」(134ページ)を参照してください。
- Service Manager Web サービスフィールドに CI 属性をマップします。

「CI 属性を Service Manager Web サービスフィールドにマップする方法」(143ページ)を参照してください。

CI 属性を UCMDB クラスモデルに追加する方法

統合では、UCMDB システムで利用できる CI 属性のサブセットのみが使用されます。出荷時設定では、統合は、Service Manager システムで通常管理される CI 属性 (ホスト名やホスト DNS 名など) で構成されます。新規 UCMDB CI 属性を作成する前に、必要とするデータを提供する既存の CI 属性が UCMDB システムにあるかどうかを特定する必要があります。通常は、統合に追加する必要があるデータを追跡している既存の属性が存在します。たとえば、Node CI タイプには、統合に追加できる多くの属性が含まれています。




次の手順では、既存のCIタイプに新規CI属性を追加する方法を示します。このシナリオはあまり現実的な状況ではありません。通常は、統合に既存のCI属性を追加するので、これらの手順を実行する必要はありません。

注: 統合では、UCMDBクラスモデルにCI属性を追加するのに特別な手順は必要ありません。標準的なCI属性作成手順を用いて、CI属性を追加できます。CI属性の作成の詳細については、UCMDBヘルプセンターを参照してください。


次の手順では、例として、**comments** という属性をビジネスサービスCIタイプに追加する方法を示します。

UCMDBクラスモデルにCI属性を追加するには:

1. 管理者としてUCMDBにログインします。
2. [モデリング]>[CIタイプマネージャ]に移動します。
3. [CIタイプ]ナビゲーションツリーから、新規CI属性の追加先となるCIタイプを選択します。
[ConfigurationItem]>[BusinessElement]>[BusinessService]などです。
4. [属性]タブを選択します。
5. [追加]アイコンをクリックします。 
[属性の追加]ウィンドウが開きます。
6. [属性名]に、新規CI属性に使用する一意の名前を入力します。「comments」などです。

注意: 名前に次の文字を含めることはできません: ` / \ [] : | < > + = ; , ? *

7. [表示名]に、UCMDBのインターフェースに表示する名前を入力します。「Comments」などです。
8. (オプション)[説明]に、新規CI属性の説明を入力します。
9. [属性タイプ]で、属性のデータ型に応じて[プリミティブ]または[列挙/リスト]を選択します。たとえば、[プリミティブ]を選択して、[string]を選択します。

10. [値のサイズ]に、属性に格納できる最大文字数を入力します。「300」などです。
11. [標準設定値の有効化]オプションを選択し、デフォルト値を入力します。または、このオプションを選択せずに、デフォルト値を空白のままにします。
12. [OK]をクリックして、属性を保存します。
13. [保存]アイコン  をクリックして、CIタイプに属性の変更を保存します。

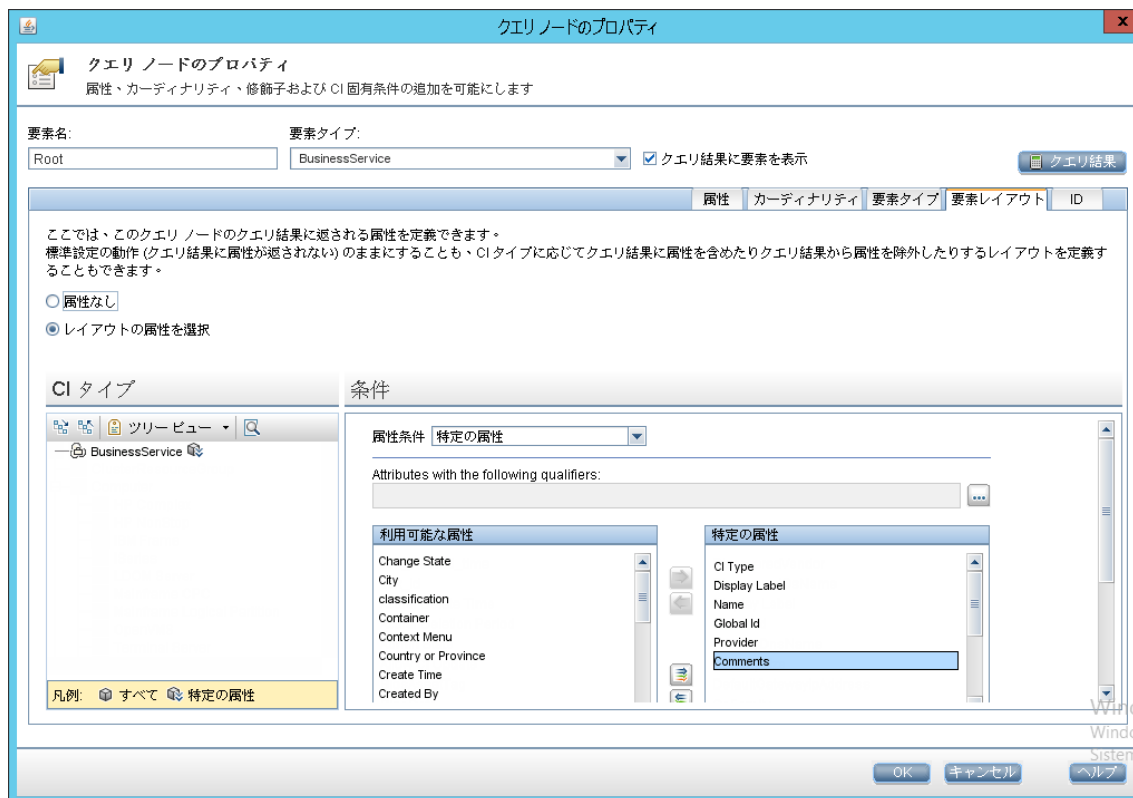
これで、CIタイプに属性が追加されました。次に、CIタイプを同期するクエリに属性を追加する必要があります。「[CI属性をクエリレイアウトに追加する方法](#)」(133ページ)を参照してください。

CI属性をクエリレイアウトに追加する方法


統合に既存のCI属性を追加するには、そのCIタイプを同期するクエリのレイアウト設定にこの属性を追加する必要があります。統合に追加するCI属性がどのCIタイプに含まれるのかを把握しておく必要があります。

クエリレイアウトにCI属性を追加するには:

1. 管理者としてUCMDBにログインします。
2. [モデリング]>[モデリングスタジオ]に移動します。
3. [リソースタイプ]で、[クエリ]を選択します。
4. [クエリ]ナビゲーションツリーから、[Integration]>[Service Manager]をクリックします。
5. 統合に追加する属性を持つCIタイプを管理するクエリを選択します。[Push]>[SM Computer Push 2.0]などです。UCMDBには、クエリのTQLレイアウトが表示されます。
6. 統合に追加するCI属性を含むノードをクエリレイアウトから選択します。「Root」などです。
7. ノードを右クリックし、[クエリノードのプロパティ]を選択します。[クエリノードのプロパティ]ウィンドウが表示されます。
8. CIタイプ(この例ではデータベース)を選択し、[要素レイアウト]タブをクリックします。
9. [利用可能な属性]リストから統合に含めるCI属性を選択し、[追加]アイコンをクリックして、[特定の属性]リストに追加します。[Comments]などです。



10. [OK]をクリックして、ノードのプロパティを保存します。

11. [保存]アイコン  をクリックして、クエリを保存します。

これで、対応するクエリのレイアウトにCI属性が追加されました。次に、このUCMDB CI属性にマップされるService Manager Web サービスフィールドを追加する必要があります。

Service Manager CIタイプのWeb サービスフィールドを追加する方法

統合では、Service Manager (SM) システムで利用できるCI属性のサブセットのみが使用されます。UCMDBとSMの間でCI属性をマップする必要があります。前に追加したUCMDB CI属性にマップされる新しいSM CI属性を作成する前に、必要なデータを提供する既存のCI属性がService Managerシステムにあるかどうかを確認する必要があります。通常は、統合に追加する必要があるデータを追跡している既存の属性が存在します。たとえば、コンピュータCIタイプには、統合に追加できる多くの属性が含まれています。

結合テーブル名

共通名

[共通名の編集](#)

ファイル名	フィールド名とキャプション	
テーブル名	フィールド名	フィールドキャプション
device	device.ac.category	Ac Category
device	device.admin.id	Admin Id
device	device.admin.password	Admin Password
device	device.admin.urlport	Admin Urlport
device	device.allow.subscription	Allow Subscription
device	device.asset.tag	Asset Tag
device	device.assignment	Assignment
device	device.auditDate	Audit Date
device	device.auditDiscrepancy	Audit Discrepancy
device	device.auditPolicy	Audit Policy
device	device.auditStatus	Audit Status
device	device.auditedBy	Audited by

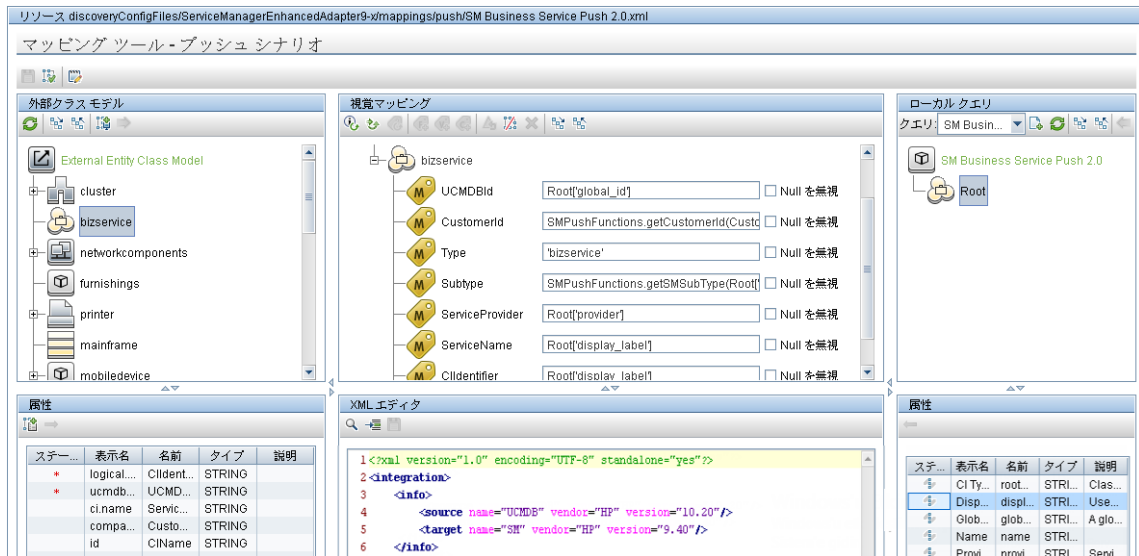
新しい属性を作成する代わりに既存の属性を使用する場合、Service Manager で Web サービス定義を直接変更してその属性を公開し、その後で UCMDB ビジュアルマッピングツールを使用して、この SM 属性を UCMDB 属性にマップすることができます ([「CI 属性を Service Manager Web サービスフィールドにマップする方法」](#)(143ページ)を参照してください)。

新しい属性を追加する場合は、次の手順に従います。手順は属性のデータ型によって異なります。つまり、単純な属性 (文字列など) か、複雑な属性 (構造体の配列、構造体) によって異なります。

SM CI タイプへの単純な属性の追加

次の手順では、例として、既存の CI タイプ (ビジネスサービス) に単純な属性 (**mycomments**) を追加する方法を示します。

1. システム管理者として UCMDB にログインします。
2. **[データフロー管理]** > **[アダプタ管理]** をクリックして、**[ServiceManagerEnhancedAdapter 9-x]** を選択して、対応する XML マッピングファイル (この例では **SM Business Service 2.0.xml**) をビジュアルマッピングツールエディタで開きます。
3. **[外部クラスモデル]** 表示枠で、CI タイプ (**bizservice**) を選択します。



4. 左側の[属性]表示枠 (Service Manager Web サービスオブジェクトから取得されるビジネスサービス CI タイプのフィールドが表示されます) で、[選択した外部ノードへの新規属性の追加]アイコンをクリックします。
5. 新しい属性の名前と表示ラベルを入力し、[OK]をクリックします。

注意: Service Manager CI タイプの属性テーブル内または **device** テーブル内の既存のフィールド名と重複する名前またはその一部になっている名前を入力した場合は、属性を作成できなかったことを示すエラーが表示されます。



6. [OK]をもう一度クリックして、属性の作成を確認します。

新しい属性 (**mycomments**) が、Service Manager の属性リストに表示されます。

属性				
ステータス	表示名	名前	タイプ	説明
*	logical.name	CIIdentifier	STRING	
*	ucmdb.id	UCMDBid	STRING	
	ci.name	ServiceName	STRING	
	company	CustomerId	STRING	
	id	CIName	STRING	
	mycomments	mycomments	STRING	
	subtype	Subtype	STRING	
	type	Type	STRING	
	vendor	ServiceProvider	STRING	

同時に、UCMDB が、Service Manager Web サービス API を呼び出して新しい属性を追加し、Web サービス API でその属性を公開して、管理フィールドとして設定します。Service Manager で Web サービスオブジェクトを開いて、新しい属性を確認できます ([カスタマイズ]>[Web サービス]>[Web サービス構成])。

外部アクセス定義

サービス名:

名前:

オブジェクト名:

許可されるアクション 式 フィールド RESTful

フィールド	キャプション	タイプ
ucmdb.id	UCMDBId	
ci.name	ServiceName	
type	Type	
subtype	Subtype	
company	CustomerId	
logical.name	CIIdentifier	
vendor	ServiceProvider	
id	CIName	
mycomments	MyComments	

7. 構成ファイルを保存します。

CIタイプへの構造体の配列または構造体の追加

Service Manager は、構造体の配列または構造体を使用して、複雑な属性 (コンピュータCI のポートなど) を格納します。UCMDB でビジュアルマッピングツールを使用してそのような複雑な属性を作成することもできます。

次の手順では、コンピュータCIタイプに **comport** という名前の属性を追加する方法を示します。

1. システム管理者として UCMDB にログインします。
2. ビジュアルマッピングツールエディタを使用して、対応する XML マッピングファイル(この例では **SM Computer Push 2.0.xml**)を開きます。
3. CI タイプ (**computer**) を選択し、[外部クラス モデルへの新規 CI タイプの追加] アイコンをクリックします。
4. 表示される子 CI タイプの作成ダイアログで、属性の名前と説明を入力し、[Relation with Parent] で、属性のデータ型が構造体の配列である場合は [Many to One] を選択し、データ型が構造体である場合は [One to One] を選択します。

新規子ノードの追加

新規子ノードの追加
外部クラス モデル用に新規ノード プロパティを定義する
必要があります。

General

* Name:

Description:

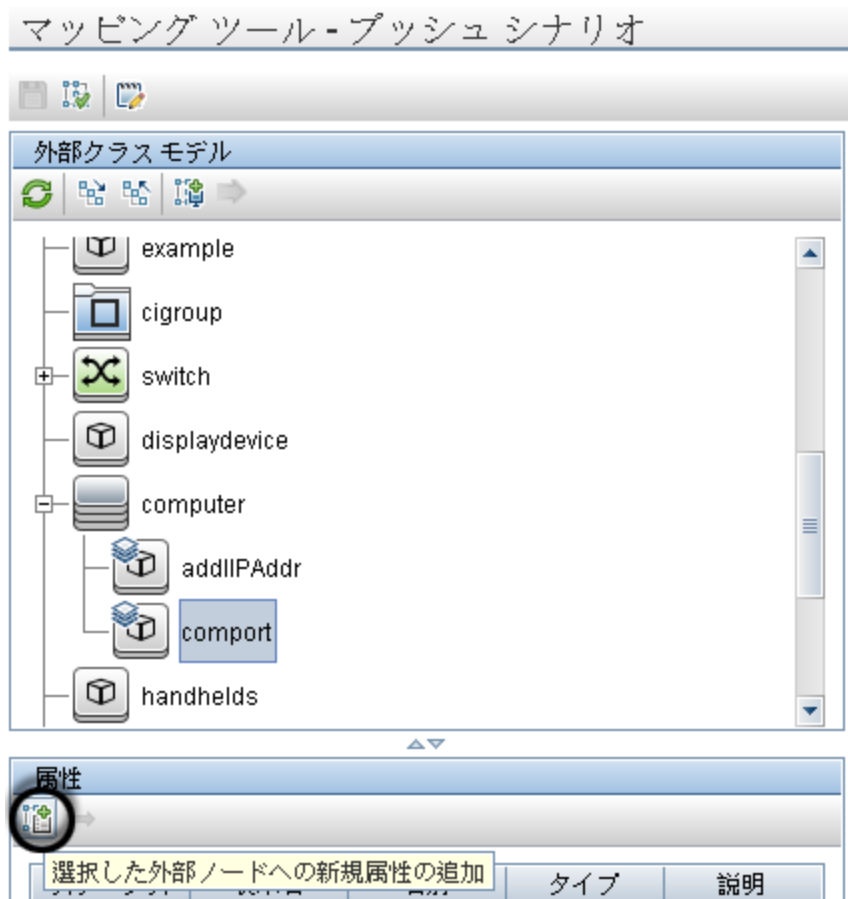
Parent Node:

Metadata

Relation with Parent

OK キャンセル

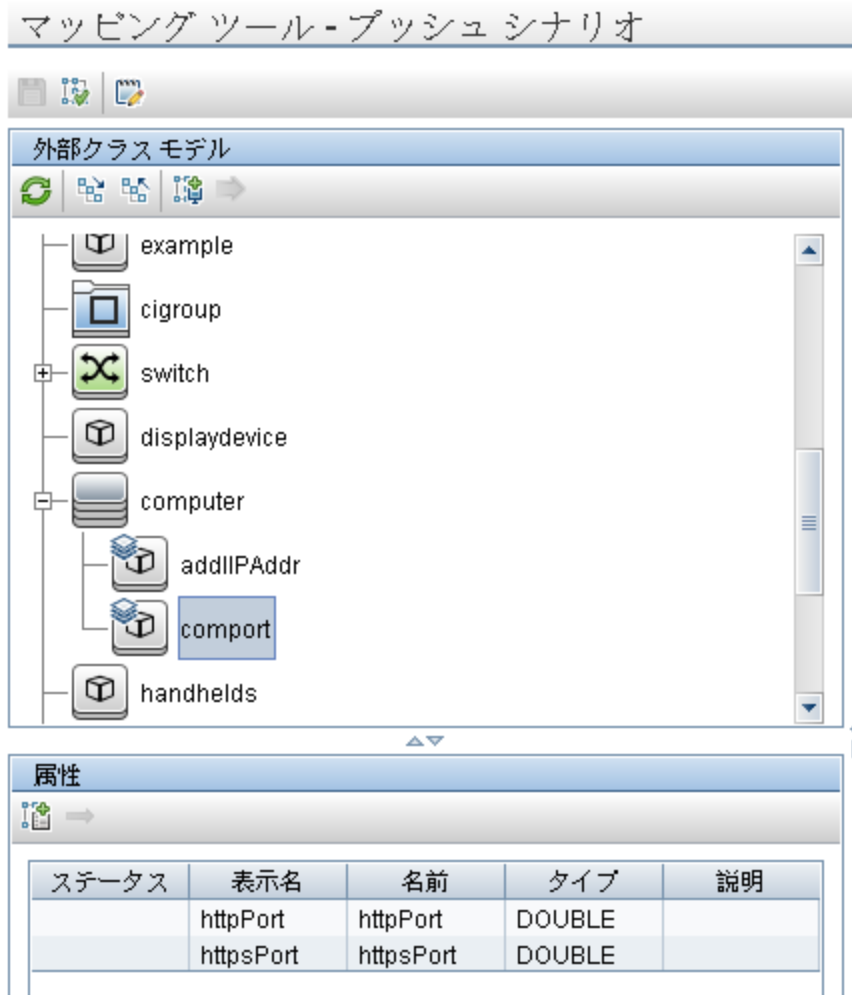
5. [OK]をクリックします。属性がService Manager側で作成されます。
6. [外部クラスモデル]表示枠で属性を選択し、[選択した外部ノードへの新規属性の追加]アイコンをクリックします。



7. 構造体の配列にフィールド (httpPort)を追加し、[OK]をクリックします。[OK]をもう一度クリックして、属性の作成を確認します。



8. 必要に応じて前の2つの手順を繰り返して、構造体の配列にフィールドを追加します。



さらに、この属性は、Service Manager 内の対応する Web サービスオブジェクトに自動的に追加されます。

外部アクセス定義

サービス名:	* ucmdbIntegration
名前:	* joinnode
オブジェクト名:	ucmdbNode

許可されるアクション	式	フィールド	RESTful
tie.system[disk.size]	disk.size		
asset.tag	AssetTag		
machine.name	HostName		
id	CIName		
disk.device[model.name]	ModelName		
disk.device[disk.vendor]	DiskVendor		
disk.device[disk.name]	DiskName		
isVisualization	IsVisualization		
istatus	AssetStatus		
comport[httpPort]	HttpPort		
comport[httpsPort]	HttpsPort		

9. 構成ファイルを保存します。

次に、この Service Manager CI タイプ属性を、UCMDB で前に追加した属性にマップする必要があります。

CI 属性を Service Manager Web サービスフィールドにマップする方法

統合では、アダプタを使用して、UCMDB CI 属性を Service Manager によって認識される Web サービスオブジェクトに変換します。アダプタは、UCMDB クエリを適切にフォーマットされた Service Manager Web サービスメッセージに変換するのに使用する XML 変換ファイルを指定します。

出荷時設定では、統合クエリごとに対応する XML 構成ファイルがあり、それにより UCMDB の特定の CI タイプがマップされます。さらに、計算を有効にする属性はそれぞれ、XML 構成ファイル内に独自のエン

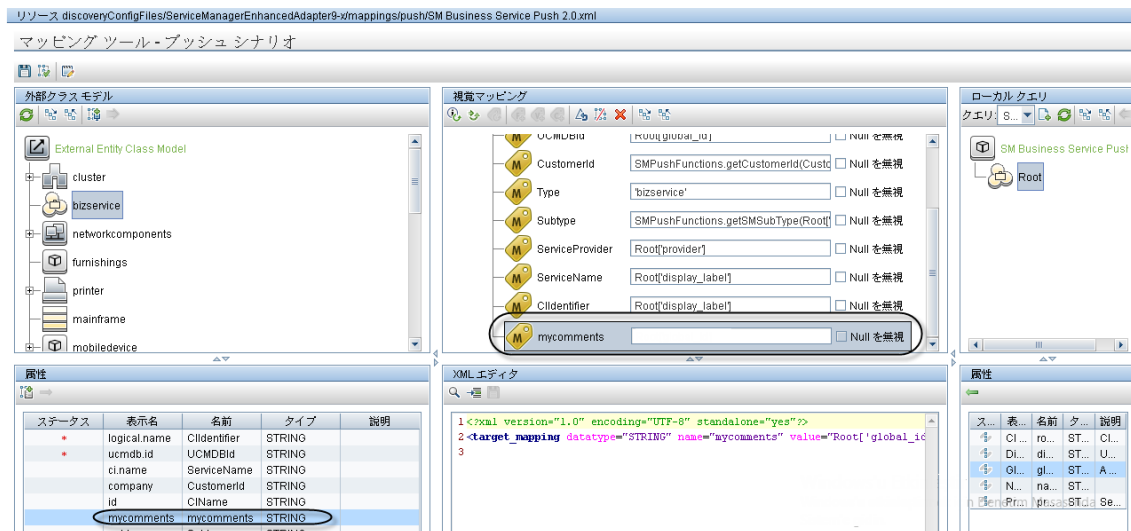
リを必要とします。XML 変換エントリがない場合、Service Manager は UCMDB システムから CI 属性の更新を受信できません。

統合に新規属性を追加する場合、上位 CI タイプの XML 構成ファイルを編集して、CI 属性のエントリを追加する必要があります。各クエリが管理する CI タイプの詳細については、「[プッシュのクエリ](#)」(107ページ)を参照してください。

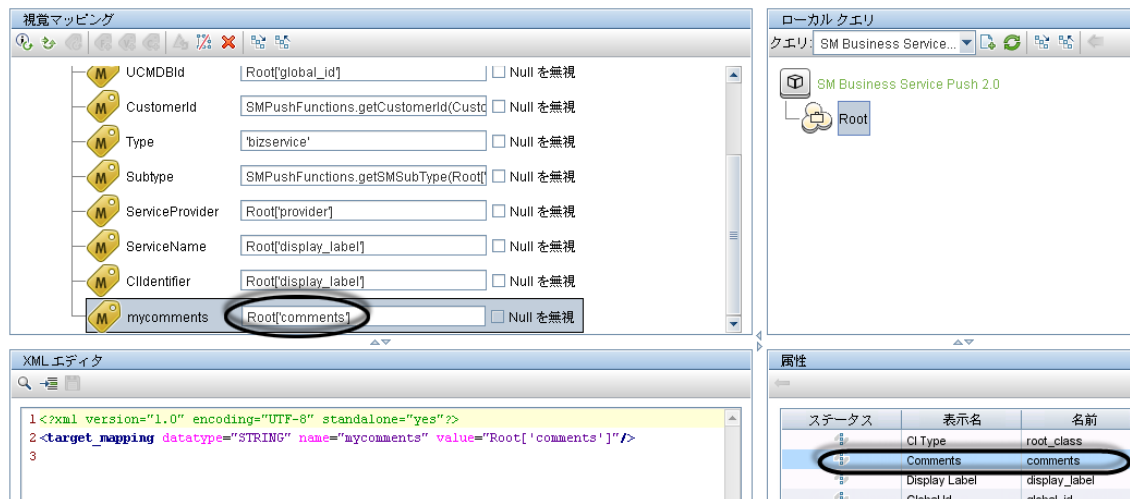
次の手順では、UCMDB CI 属性 **Comments** (前に追加しました) を **MyComments** Service Manager 属性 (前の手順で作成しました) にマップする方法を示します。

UCMDB CI 属性を SM CI 属性にマップするには:

1. 管理者として UCMDB にログインします。
2. [データフロー管理]>[アダプタ管理]>[ServiceManagerEnhancedAdapter9-x]>[構成ファイル]に移動します。
3. CI 属性の上位 CI タイプを管理する XML 構成ファイルをダブルクリックします。たとえば、**SM Business Service Push 2.0.xml** を開いて、**SM Business Service Push 2.0** クエリに属性を追加します。
4. [外部クラスモデル]表示枠で関連する CI タイプを選択し (この例では **bizservice**)、属性 (**mycomments**) を左側の [属性] 表示枠から [視覚マッピング] 表示枠にドラッグアンドドロップします。



5. [ローカルクエリ]表示枠で関連するノードを選択し (この例では **Root**)、属性 (この例では **comments**) を前の手順で作成したマッピングエントリにドラッグアンドドロップします。



- XML 構成ファイルを保存します。これで、UCMDB の **Comments** 属性が Service Manager の **MyComments** 属性にマップされました。

注: UCMDB のアダプタ管理で構成ファイルを作成または編集して保存すると、アダプタが新しい構成ファイルで自動的に再起動されます。

データプッシュ用の CI タイプを統合に追加する方法

次の手順を用いて、統合に CI タイプを追加できます。

- UCMDB クラスモデルに CI タイプはすでに存在していますか?
はい。手順 3 に進みます。

いいえ。手順 2 に進みます。
- UCMDB クラスモデルに CI タイプを追加します。
「[CI タイプを UCMDB クラスモデルに追加する方法](#)」(146ページ)を参照してください。
- CI タイプを同期するためのクエリを作成します。
「[CI タイプを同期するためのクエリの作成方法](#)」(148ページ)を参照してください。
- クエリのレイアウトに CI タイプの属性を追加します。
「[CI タイプの属性をクエリレイアウトに追加する方法](#)」(152ページ)を参照してください。
- Service Manager Service Manager に CI タイプを追加します。
「[Service Manager に CI タイプを追加する方法](#)」(154ページ)を参照してください。
- Web サービスフィールドに CI タイプの属性をマップします。
「[CI タイプの属性を Web サービスフィールドにマップする方法](#)」(156ページ)を参照してください。
- 統合のデータプッシュジョブにカスタムクエリを追加します。
「[統合ジョブにカスタムクエリを追加する方法](#)」(166ページ)を参照してください。


CI タイプを UCMDB クラスモデルに追加する方法

新規 UCMDB CI タイプを作成する前に、必要な CI 属性を提供する既存の CI タイプが UCMDB システムにあるかどうかを特定する必要があります。通常は、1 つまたは複数の既存の CI タイプへのリンクを作成することで、統合で使用される新しい論理 CI タイプを作成します。

次の手順では、database という既存の CI タイプを基にして、SM RDBMS という新規 CI タイプを作成する方法を示します。

注: 統合では、UCMDB クラスモデルに CI タイプを追加するのに特別な手順は必要ありません。標準的な CI タイプ作成手順を用いて、CI タイプを追加できます。CI タイプの作成の詳細については、UCMDB ヘルプセンタを参照してください。

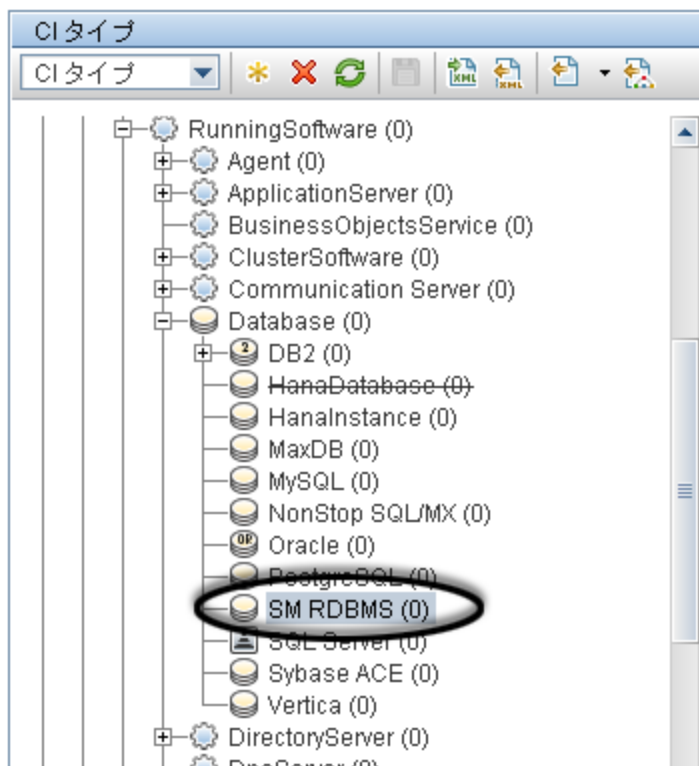
UCMDB クラスモデルに CI タイプを追加するには、次の手順を実行します。

1. 管理者として UCMDB にログインします。
2. [モデリング]>[CI タイプマネージャ]に移動します。
3. [CI タイプ]ナビゲーションツリーから、新規 CI タイプに対して使用するベースの CI タイプを選択します。[Managed Object]>[ConfigurationItem]>[Infrastructure Element]>[Running Software]>[Database]などです。
4. [新規]アイコン  をクリックします。
[構成アイテムタイプを作成]ウィンドウが開きます。
5. [名前]に、新規 CI タイプに使用する一意の名前を入力します。「sm_rdbms」などです。

注意: 名前に次の文字を含めることはできません: ` / \ [] : | < > + = ; , ? *

6. [表示名]に、インターフェイスに表示する名前を入力します。[SM RDBMS]などです。
7. [説明]に、新規 CI タイプの説明を入力します。これはオプションのフィールドです。「関係データベースを実行するホスト」などです。
8. [ベースの CI タイプ]で、適切なベース CI タイプが選択されていることを確認します。新規 CI タイプは、ここで選択するベースの CI タイプの属性を継承します。「Database」などです。
9. [次へ]をクリックします。ウィザードには、ベースの CI タイプの CI 属性のリストが表示されます。
10. 新規 CI タイプに対して、必要に応じて CI 属性を追加、編集、または削除します。たとえば、Database から継承したデフォルトの属性をそのまま使用します。
11. [次へ]をクリックします。ウィザードには、ベースの CI タイプの修飾子のリストが表示されます。
12. 新規 CI タイプの必要に応じて、修飾子を追加または削除します。たとえば、デフォルトの修飾子をそのまま使用します。
13. [次へ]をクリックします。ウィザードには、CI タイプに関連付けられたアイコンのリストが表示されます。

- このCIタイプに関連付けるアイコンを選択します。たとえば、デフォルトの抽象クラスのアイコンをそのまま使用します。
- [次へ]をクリックして、必要に応じてメニューアイテムのプロパティやラベル定義を追加します。たとえば、ベースのCIタイプのデフォルト設定をそのまま使用します。
- [完了]をクリックして、CIタイプを作成します。

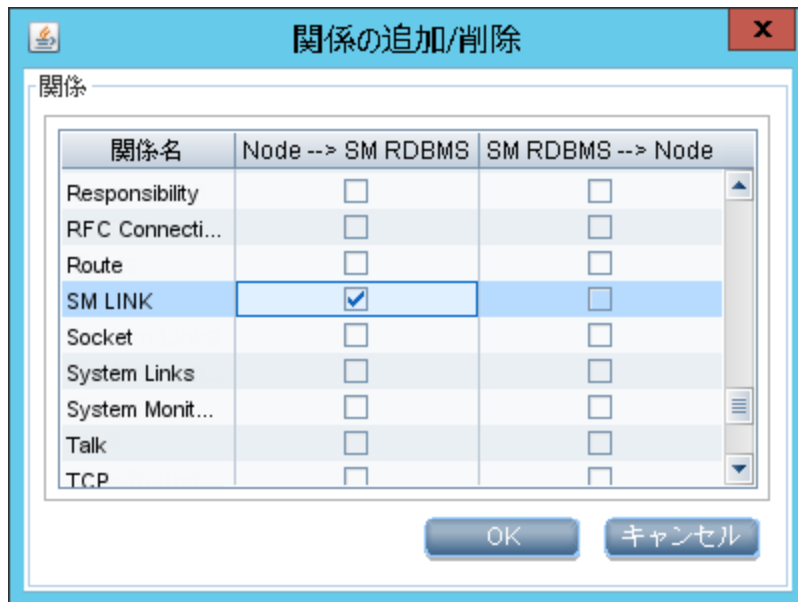


- ツリーから新規 CI タイプを選択します。[SM RDBMS]などです。
- リンク先とする既存のCIタイプを参照して、Ctrl キーを押しながらクリックし、そのCIタイプを選択に追加します。「Node」などです。

注: 新しい論理 CI タイプに追加する属性がある既存のCIタイプを選択します。

- 選択したCIタイプのいずれかを右クリックして、[関係の追加/削除]をクリックします。[関係]ウィンドウが開きます。
- 既存のCIタイプから新規CIタイプへのSMリンク関係を作成します。[Node]から[SM RDBMS]へのリンクなどです。

注: SMリンク関係が存在しない場合は、新しいSMリンク関係を作成する必要があります。



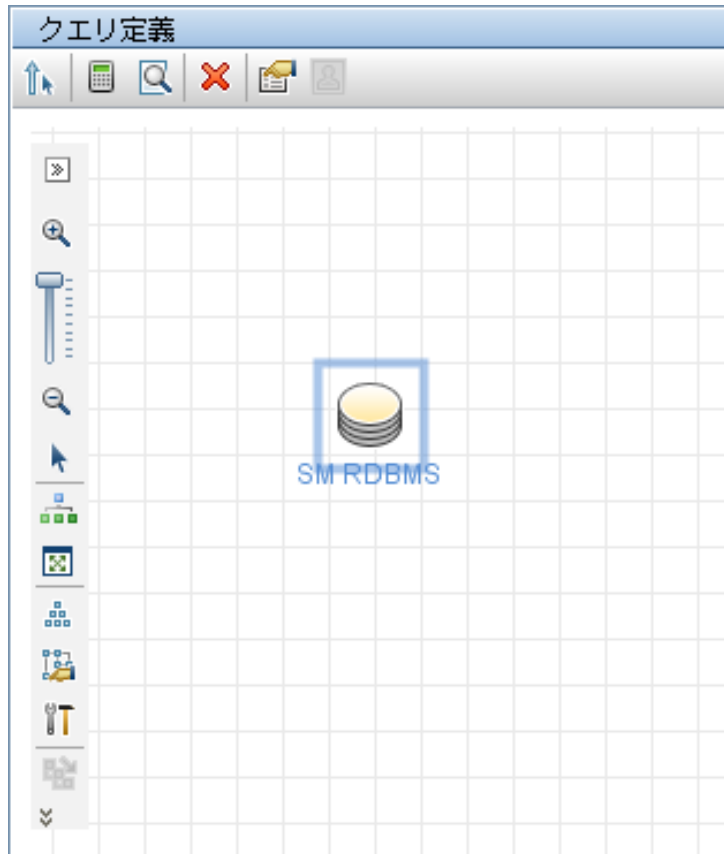
21. [OK]をクリックして、関係を作成します。
22. [保存]アイコンをクリックして、CI タイプを保存します。

CI タイプを同期するためのクエリの作成方法

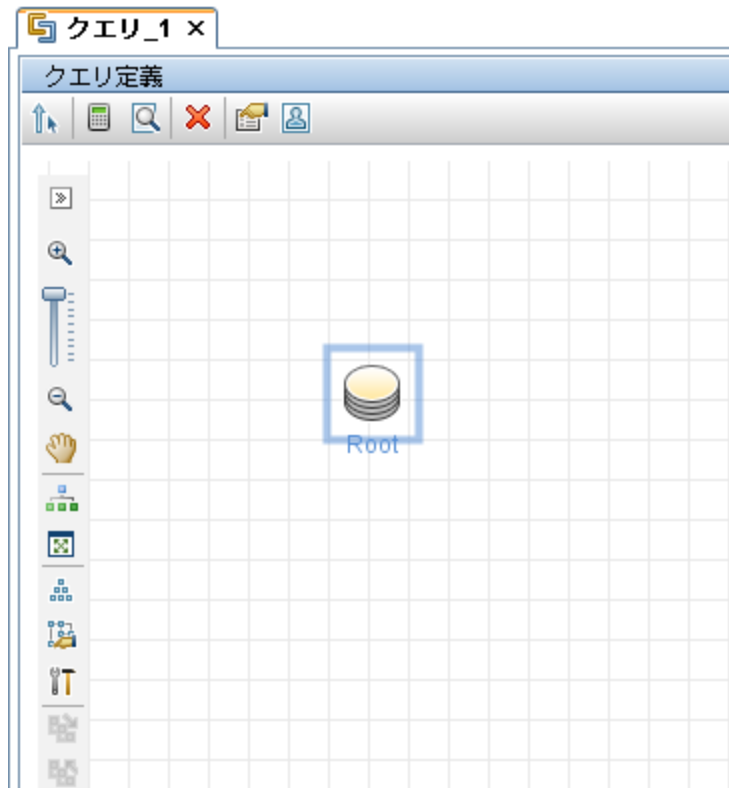
統合では、クエリを使用してCI 属性値を収集し、それを Service Manager システムに渡します。統合に追加する任意のCI タイプについて、クエリを作成する必要があります。作成するクエリは、いずれも「[クエリの要件](#)」(111ページ)に準拠する必要があります。

次の手順では、前の項で説明した **SM RDBMS** CI タイプに対して、**rdbmsData** という名前の新規クエリを作成する方法について説明します。

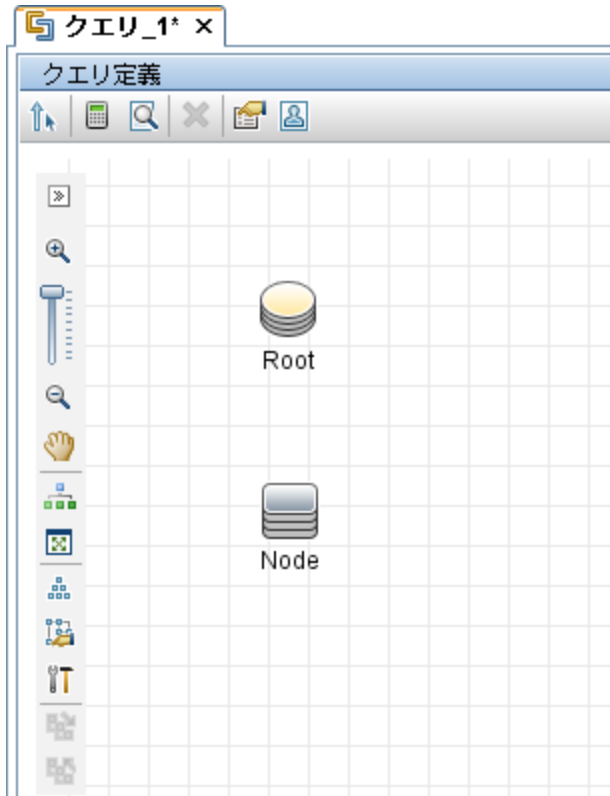
1. 管理者として UCMDb にログインします。
2. [モデリング]>[モデリングスタジオ]に移動します。
3. [クエリ]ナビゲーションツリーから、[Integration]>[Service Manager]をクリックします。
4. [Service Manager]を右クリックし、[新規]>[クエリ]を選択します。
[クエリ定義]ウィンドウが開きます。
5. [CI タイプセレクト]から、クエリのルートノードとなるCI タイプを探します。このCI タイプは通常、CI のほとんどの属性を提供するCI タイプです。たとえば、[Managed Object]>[ConfigurationItem]>[InfrastructureElement]>[RunningSoftware]>[Database]>[SM RDBMS]などです。
6. [CI タイプセレクト]からルート CI タイプをドラッグして、空の[編集]表示枠にドラッグします。
UCMDb にCI タイプのアイコンが表示されます。



7. CI タイプアイコンを右クリックし、[クエリノードのプロパティ]を選択します。[ノードのプロパティ]ウィンドウが表示されます。
8. 要素名を「**Root**」に変更します。
9. [OK]をクリックして、ノードのプロパティを保存します。

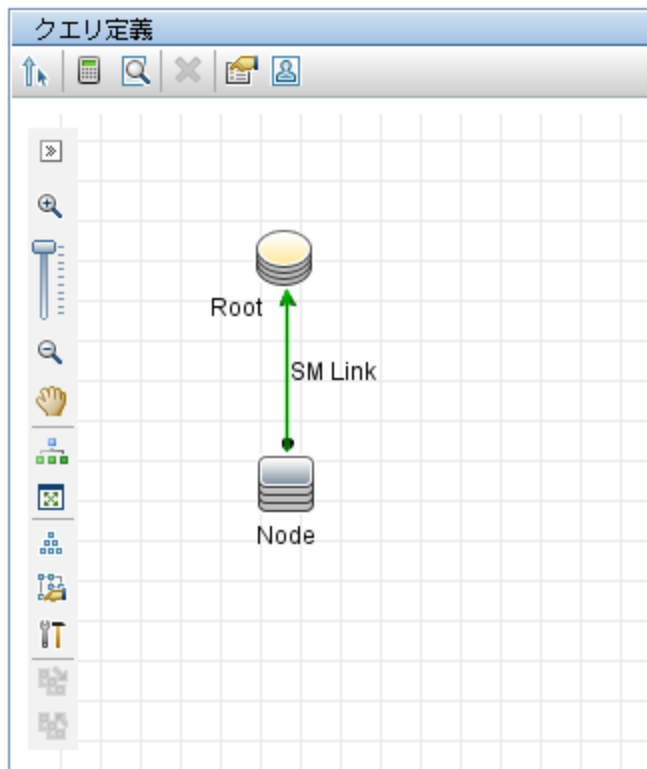



10. [CIタイプセレクタ]から、クエリに追加する追加のCIタイプを探します。これらのCIタイプは通常、追加のCI属性を提供します。たとえば、[Managed Object]>[ConfigurationItem]>[Infrastructure Element]>[Node]などです。
11. [CIタイプセレクタ]から追加のCIタイプをドラッグして、[クエリ定義]表示枠にドラッグします。UCMDBに追加のCIタイプのアイコンが表示されます。



12. 必要に応じて、ルート CI タイプと追加の CI タイプ間の関係を作成します。たとえば、**Root** と **Node** の間に「SM Link」を作成します。
 - a. [**Root**] を選択して、Ctrl キーを押しながら追加の CI タイプをクリックします。「**Node**」などです。
 - b. 選択した項目のいずれかを右クリックして、[**関係の追加**] をクリックします。[関係の追加] ウィンドウが開きます。
 - c. 関係の名前を入力します。たとえば、「SM Link」などです。
 - d. 関係の方向を選択します。

- e. [OK]をクリックして、関係を追加します。



13. クエリに追加する追加の各 CI タイプについて手順 10 ~ 12 を繰り返します。たとえば、SM RDBMS は追加の CI タイプは必要としません。
14. [保存]アイコン  をクリックして、クエリを保存します。
 - a. [クエリ名]に、新規クエリに使用する一意の名前を入力します。「rdbmsData」などです。
 - b. フォルダツリーで、クエリの保存先のフォルダを選択します。[Root]>[Integration]>[Service Manager]>[Push]などです。
 - c. [OK]をクリックします。UCMDB が[クエリ]リストに新規クエリを追加します。

CIタイプの属性をクエリレイアウトに追加する方法

統合にCI属性を追加するには、CIタイプを同期するクエリから計算レイアウト設定を有効にする必要があります。統合に追加する各属性で計算を有効にする必要があることから、統合のCIタイプとそれらの属性について理解しておく必要があります。

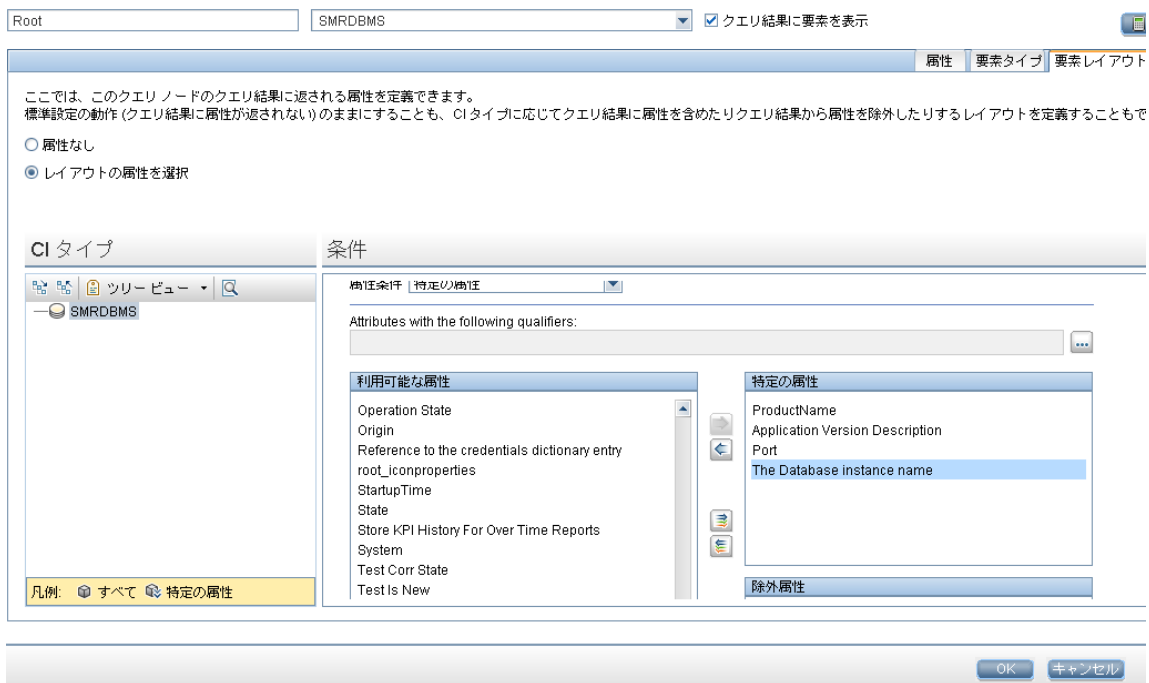
次の手順では、前の項で説明した **SM RDBMS** CIタイプの属性の計算を有効にする方法を示します。

クエリレイアウトにCIタイプの属性を追加するには:

1. 管理者として UCMDB にログインします。
2. [モデリング]>[モデリングスタジオ]に移動します。
3. [クエリ]ナビゲーションツリーから、[Integration]>[Service Manager]をクリックします。
4. 統合に追加する属性がある CI タイプを管理するクエリをダブルクリックします。[rdbmsData]などです。UCMDB には、クエリの TQL レイアウトが表示されます。
5. **Root** ノードを右クリックし、[クエリノードのプロパティ]を選択します。[クエリノードのプロパティ]ウィンドウが表示されます。


注意: 統合クエリには、**Root** という名前のノードが含まれている必要があります。詳細については、[「クエリの要件」\(111ページ\)](#)を参照してください。

6. [要素レイアウト]タブをクリックし、[レイアウトの属性を選択]オプションを選択します。
7. [属性条件]リストから[特定の属性]を選択し、[利用可能な属性]リストから必要な属性を選択して、[特定の属性]リストに移動します。たとえば、[製品名]、[アプリケーションバージョンの説明]、[データベースインスタンス名]、[ポート]などの属性を追加します。



8. [OK]をクリックして、クエリノードのプロパティを保存します。
9. 統合に追加する属性が含まれる追加のノードを選択します。「Node」などです。
10. 追加の各ノードについて手順 4 ~ 8 を繰り返します。
11. [OK]をクリックして、クエリノードのプロパティを保存します。

12. 統合に追加する属性が含まれる追加の各ノードについて手順 9 ~ 13 を繰り返します。

13. [保存]アイコン  をクリックして、クエリを保存します。

Service Manager にCI タイプを追加する方法

新規 Service Manager CI タイプを作成する前に、必要な CI 属性を提供する既存の CI タイプが Service Manager システムにあるかどうかを特定する必要があります。ほとんどの場合、統合用の既存の CI タイプを再利用できます。

UCMDB のビジュアルマッピングツールを使用して Service Manager に新しい CI タイプを追加することができます。つまり UCMDB をログアウトして Service Manager にログインする必要はありません。

次の手順では、**RDBMS** という名前の新しい CI タイプを作成する方法を示します。

注: この例は手順の説明のみを目的として提供されています。ベストプラクティスは、既存の Service Manager CI タイプ **RunningSoftware** を再利用して UCMDB CI タイプ **SM RDBMS** にマップすることです。

1. 管理者として UCMDB にログインします。
2. ビジュアルマッピングツールエディタを使用して、既存の XML マッピングファイル (**SM Computer Push 2.0.xml** など)を開きます。
3. [外部クラスモデル]表示枠でルートノードを選択し、[外部クラスモデルへの新規 CI タイプの追加]アイコンをクリックします。
4. 次のようにパラメータの値を入力します。

Name: rdbms

Description: RDBMS の CI タイプ

table: smrdbms

subtype: Oracle, SQL Server

新規ノードの追加

新規ノードの追加
外部クラス モデル用に新規ノード プロパティを定義する必要があります。

General

* Name: rdbms
Description: CI type for RDBMS

Metadata

table: smrdbms
subtype: [Oracle, SQL Server]

OK キャンセル

5. [OK]をクリックします。
6. [OK]をもう一度クリックして、CI タイプの作成を確認します。次の図に示すように、CI タイプは Service Manager 内に自動的に作成されます。

CI タイプの管理

CI タイプの説明:	Rdbms
CI タイプ:	rdbms
ビットマップ:	lbox
フォーマット名:	configurationItem
属性ファイル:	smrdbms
結合定義:	joinsmrdbms
印刷フォーマット名:	
一括更新フォーマット名:	
アクティブ:	<input checked="" type="checkbox"/>
	サブタイプ
	Oracle
	SQL Server

Service Manager (SM) 側での処理

フォーマットを除いて、新しい CI タイプに関連するすべての SM オブジェクト (DBDICT、結合定義、Web サービス API、DEM ルールなど) は、SM 側で自動的に作成されます。新しい CI タイプは、デフォルトのフォーマットとして **configurationItem** を使用します。

SM で新しい CI タイプにカスタムフォーマットを使用する場合は、それらを手動で作成する必要があります。フォームデザイナーで既存の表示フォームや一括更新フォームを基にしてフォーマットを作成できます。Service Manager でフォームデザイナーにアクセスするには、コマンドラインに「fd」と入力するか、**[カスタマイズ]>[フォームデザイナー]**に移動します。Service Manager でのフォームの作成の詳細については、Service Manager のオンラインヘルプおよび『Tailoring Best Practices Guide』を参照してください。


CI タイプの属性を Web サービスフィールドにマップする方法

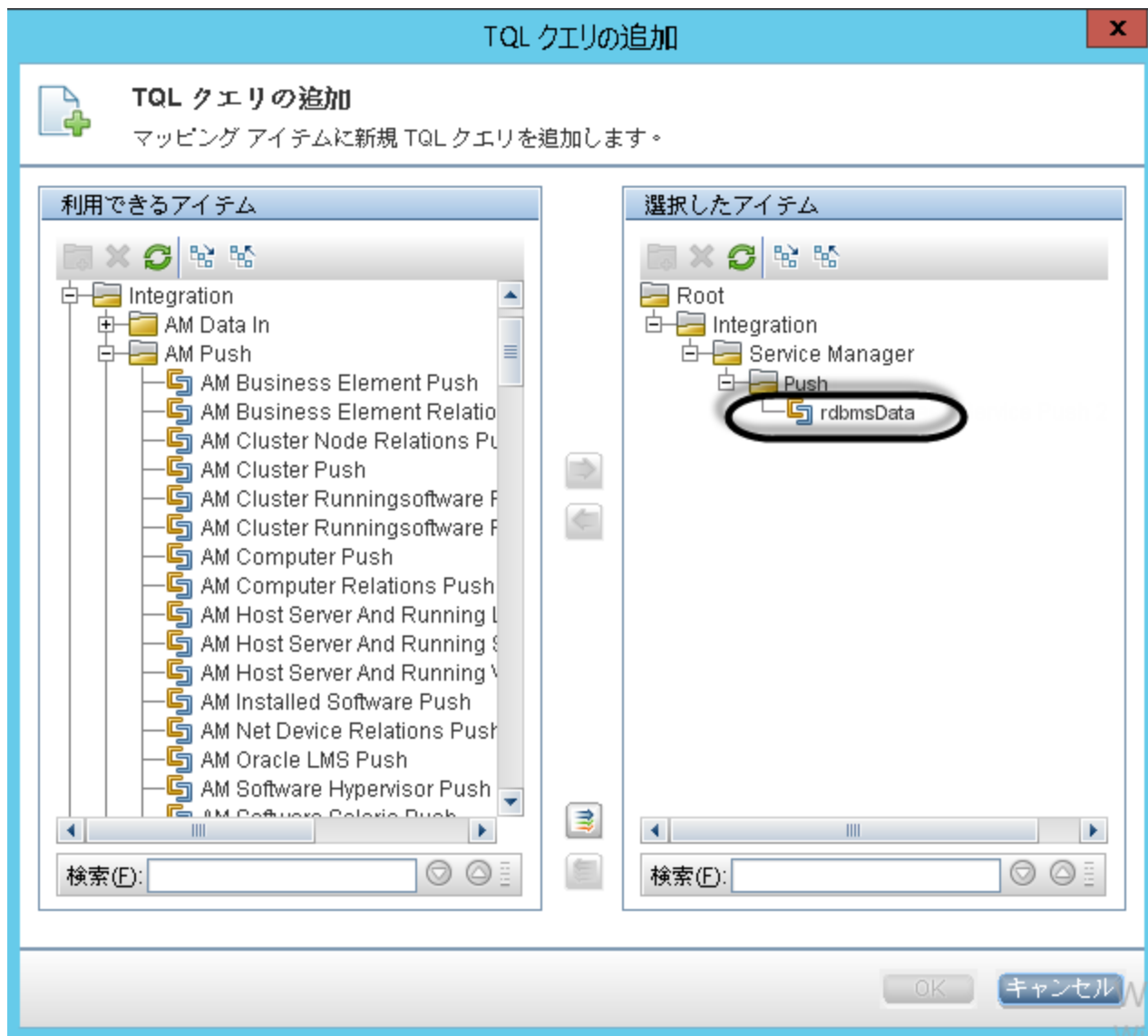
統合では、Service Manager アダプタを使用して、UCMDB CI 属性を Service Manager によって認識される Web サービスオブジェクトに変換します。Service Manager アダプタは、XML 構成ファイルを使用して、UCMDB クエリを適切にフォーマットされた Service Manager Web サービスメッセージに変換します。出荷時設定では、統合クエリごとに対応する XML 構成ファイルがあります。さらに、**[詳細レイアウト設定]**から同期を有効にする属性はそれぞれ、XML 構成ファイル内に独自のエントリを必要とします。

統合に CI タイプを追加する場合、Service Manager アダプタが Service Manager Web サービスオブジェクトに各 CI タイプを変換する方法を定義し、整合させる XML 構成ファイルを作成する必要があります。各クエリが管理する CI タイプの詳細については、**「統合クエリ」(106ページ)**を参照してください。

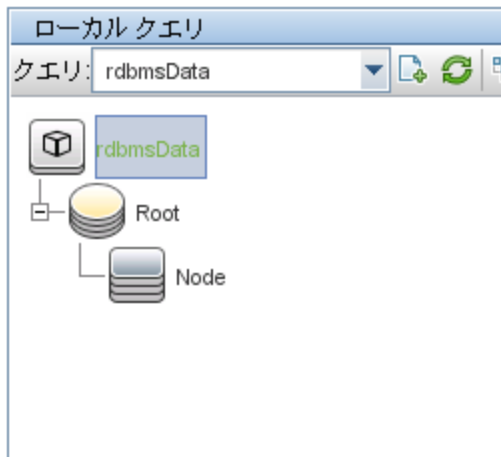
次の手順では、前の項で説明した rdbmsData クエリの XML 構成ファイルを作成する方法について説明します。

CI タイプの属性を Web サービスオブジェクトにマップするには:

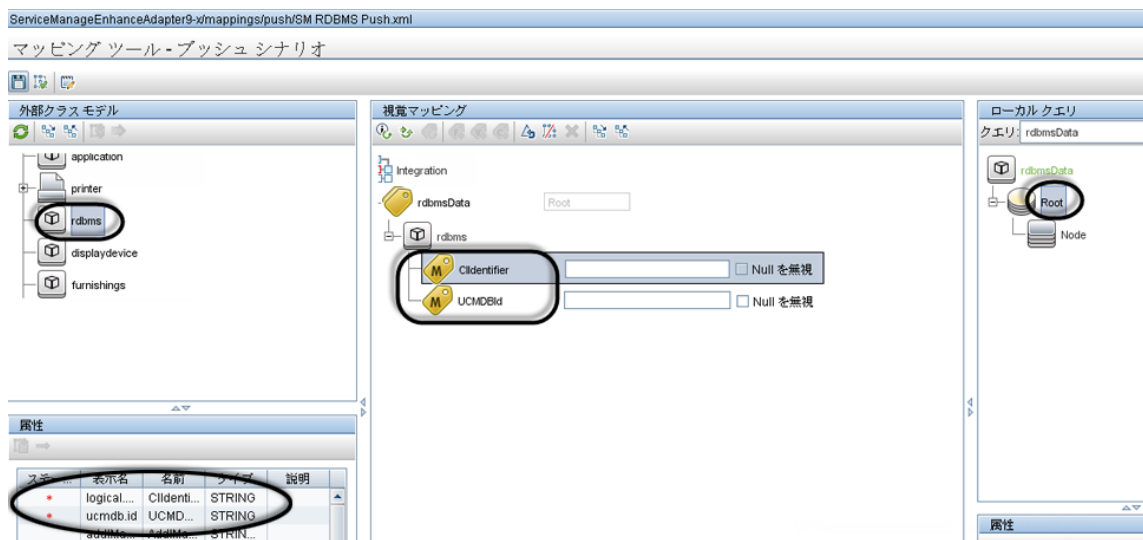
1. 管理者アカウントで UCMDB にログインします。
2. [データフロー管理]>[アダプタ管理]>[ServiceManagerEnhancedAdapter9-x]に移動し、アダプタを選択します。
3. [リソースの新規作成]アイコン  をクリックします。
4. [新しい構成ファイル]を選択します。
5. 完全なファイル名を入力します。<AdapterID>/mappings/push/<filename> のように入力します。たとえば、「ServiceManagerEnhancedAdapter9-x/mappings/push/SM RDBMS Push.xml」などです。
6. [OK]をクリックします。マッピングファイルが作成されます。
7. 新しいマッピングファイルをダブルクリックし、ビジュアルマッピングツールエディタで開きます。
8. [ローカルクエリ]表示枠で、[TQL クエリの追加]アイコンをクリックし、rdbmsData クエリを追加します。



9. [OK]をクリックします。



10. “Root” ノードを[ローカルクエリ]表示枠からマッピング領域にドラッグアンドドロップします。
11. “RDBMS” ノードを[外部クラスモデル]表示枠からマッピング領域にドラッグアンドドロップします。
12. 必要な SM 属性をマッピング領域にドラッグアンドドロップします。



13. 関連する UCMDB 属性をマッピング領域にドラッグアンドドロップします。



14. 新しいマッピングファイルを保存します。

注: UCMDB のアダプタ管理で構成ファイルを作成または編集して保存すると、アダプタが新しい構成ファイルで自動的に再起動されます。

XML 構成ファイルでの Groovy スクリプトの使用

Service Manager 拡張汎用アダプタは、次の4つのフィールドマッピングシナリオで、XML および Groovy マッピングスクリプトを使用します。1対1、1対多、多対多、および値の変換のシナリオです。Groovy スクリプトは、複雑なシナリオで使用する必要があります。詳細については、「[UCMDB でのカスタム CI タイプの構成の更新](#)」(22ページ)を参照してください。

データプッシュ用の CI 関係タイプを統合に追加する方法

新しい CI タイプを統合に追加し、そのタイプと他の CI タイプの間関係を UCMDB で作成した後に、これらの各関係タイプについて次の作業を実行し、UCMDB が各タイプの関係を Service Manager にプッシュできるようにする必要があります。

次の手順では、例として、データプッシュ用の **Ownership** という (**Cost** と **CostCategory** CI タイプの間) 関係タイプを統合に追加する方法について説明します。

注: これらの手順では、**Cost** および **CostCategory** CI タイプをすでに統合に追加済みであるものとします。

1. 関係タイプをプッシュするためのクエリを作成します。
「[関係タイプをプッシュするためのクエリの作成方法](#)」(161ページ)を参照してください。

2. 関係クエリを Service Manager Web サービスオブジェクトにマップします。
「[関係タイプクエリを Service Manager Web サービスオブジェクトにマップする方法](#)」(164ページ)を参照してください。
3. 新しい関係タイプの XML 構成ファイルを作成します。
「[関係タイプの XML 構成ファイルを作成する方法](#)」(165ページ)を参照してください。

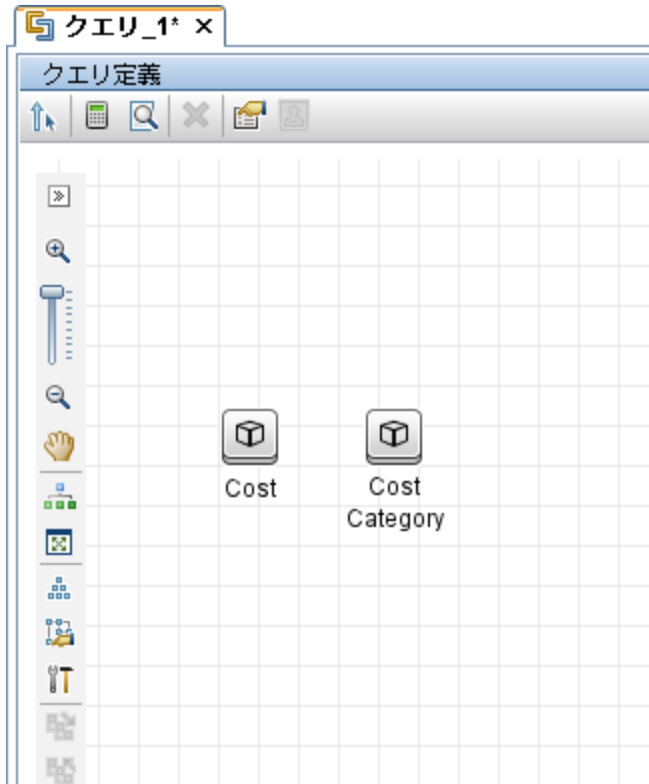
関係タイプをプッシュするためのクエリの作成方法

CI タイプ間の関係を作成したら、関係を Service Manager にプッシュするクエリを作成する必要があります。

注: 作成するクエリは、いずれも「[クエリの要件](#)」(111ページ)に準拠する必要があります。

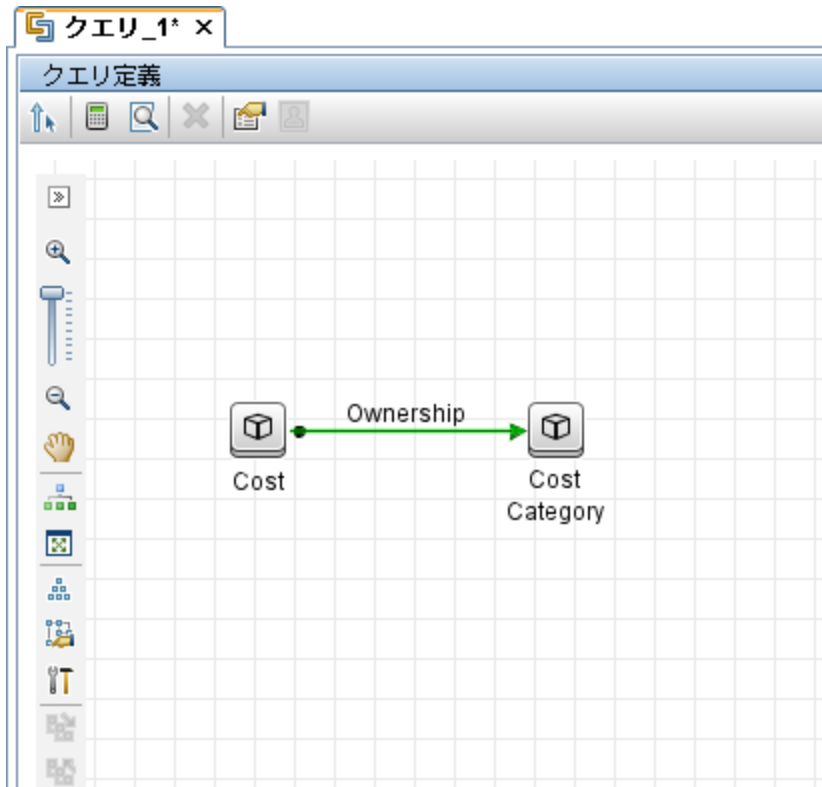
Cost と CostCategory CI タイプ間の所有関係のための **Cost CostCategory Ownership Relations** という新しいクエリを作成するには:

1. 管理者として UCMDDB にログインします。
2. [モデリング]>[モデリングスタジオ]に移動します。
3. [新規]>[クエリ]をクリックします。[クエリ定義]表示枠が表示されます。
4. CI タイプセレクタから、Cost および CostCategory CI タイプをクエリ表示枠にドラッグします。



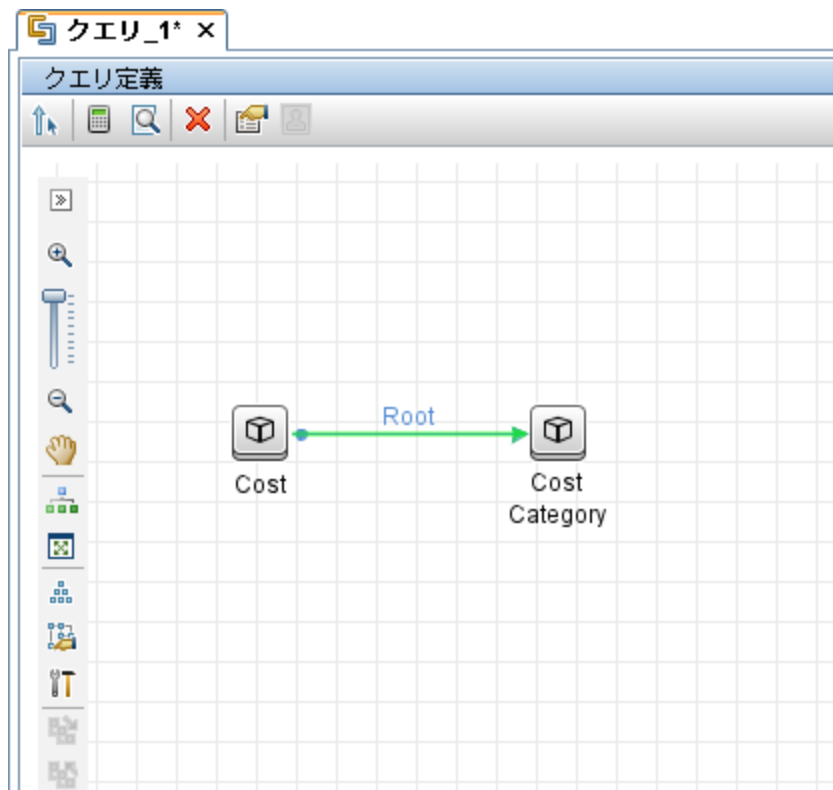
5. Cost から CostCategory への所有関係を作成します。
 - a. **[関係を作成]** アイコンをクリックします。
 - b. **[Cost]** ノードを選択し、このノードから CostCategory ノードへ矢印をドラッグします。
 - c. **[通常関係]** を選択して、**[OK]** をクリックします。
 - d. **[Connection]>[Ownership]** を選択し、**[関係の名前]** に「Ownership」と入力して、**[OK]**

をクリックします。CI タイプ間の所有関係が作成されます。



6. 関係の矢印を右クリックし、[関係のプロパティ]を選択します。
7. 要素名を「Ownership」から「Root」に変更し(または「Root_」から始まる名前を付け)、[OK]をクリックします。

The screenshot shows a dialog box titled '関係のプロパティ' (Relationship Properties). The title bar is blue. Below the title bar, there is a header area with a document icon and the text '関係のプロパティ' and '属性、カーディナリティ、修飾子およびCI固有条件の追加を可能にします'. Below this, there are two input fields: '要素名:' (Element Name) with the text 'Root' and '要素タイプ:' (Element Type) with a dropdown menu showing 'Connection'. To the right of these fields is a checked checkbox labeled 'クエリ結果に要素を表示' (Show elements in query results).



8. [保存]アイコンをクリックして、次のようにしてクエリを保存します。
 - a. クエリ名を入力します。「Cost CostCategory Ownership Relations Push」などです。
 - b. [Integration]>[Service Manager]>[Push]フォルダを選択します。
 - c. [OK]をクリックします。

これでクエリが作成されました。このクエリをプッシュ構成ファイル (smPushConf.xml) に追加できます。

関係タイプクエリを Service Manager Web サービスオブジェクトにマップする方法

関係タイプのクエリを作成したら、次の手順に従って、Service Manager でクエリを Relationship Web サービスオブジェクトにマップする必要があります。

1. [データフロー管理]>[アダプタ管理]>[ServiceManagerEnhancedAdapter9-x]>[構成ファイル]に移動します。
2. smPushConf.xml ファイルをクリックします。
3. 関係プッシュのための既存のマッピングエントリをコピーしてマッピングエントリを追加します。たとえば、次のマッピングエントリを追加します。

```
<tql name="SM Layer2 Topology Relations Push 2.0"
  resourceCollectionName="Relationships" resourceName="Relationship" />
```

4. TQL クエリの名前を関係タイプ用に作成したクエリの名前に変更します。「Cost CostCategory Ownership Relations Push」などです。

```
<tql name="Cost CostCategory Ownership Relations Push"
  resourceCollectionName="Relationships" resourceName="Relationship" />
```

5. [保存]をクリックして構成ファイルを保存します。

関係タイプのXML構成ファイルを作成する方法

関係タイプのXML構成ファイルを作成するには:

1. 管理者アカウントでUCMDBにログインします。
2. [データフロー管理]>[ServiceManagerEnhancedAdapter9-x]を選択し、アダプタを選択します。
3. [リソースの新規作成]アイコンをクリックします。
4. [新しい構成ファイル]を選択します。
5. 完全なファイル名を入力します。<AdapterID>/mappings/push/<filename> のように入力します。「ServiceManagerEnhancedAdapter9-x/mappings/push/SM Cost CostCategory Ownership.xml」などです。
6. [OK]をクリックして新しいファイルを保存します。新しいファイルが、構成ファイルの一覧に表示されます。
7. ビジュアルマッピングインタフェースのXMLエディタで新しいファイルを開きます。
8. XMLエディタ表示枠で、既存のマッピングファイルから内容をコピーして新しいファイルの内容を上書きし、クエリ名を作成したクエリの名前 (Cost CostCategory Ownership Relations Push)に更新します。

```
<?xml version="1.0" encoding="UTF-8"?>
  <integration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="../mappings_schema.xsd">
    <info>
      <source name="UCMDB" version="10.20" vendor="HP"/>
      <target name="SM" version="9.40" vendor="HP"/>
    </info>
    <import>
      <scriptFile path="mappings_scripts.SMPushFunctions"/>
    </import>
    <!--
      Push Cost to Cost Category Relations.
    -->
```

```
<target_entities>
  <source_instance query-name="Cost CostCategory Ownership Relations
Push" root-element-name="Root">
    <target_entity name="Relationship">
      <target_mapping name="RelationshipType" datatype="STRING"
value="SMPushFunctions.getDisplayName(Root['element_type'],ClassModel)"/>
      <target_mapping name="ParentCI" datatype="STRING"
value="SMPushFunctions.getEndId(OutputCI.getExternalId().getEnd1Id())"/>
      <target_mapping name="ChildCIs" datatype="STRING_LIST"
value="[SMPushFunctions.getEndId(OutputCI.getExternalId().getEnd2Id())]"/>
    </target_entity>
  </source_instance>
</target_entities>
</integration>
```

9. [保存]アイコンをクリックして新しい構成ファイルを保存します。

これで、新しい関係タイプが統合に追加されました。次に、下の図に示すように、新しい関係クエリをデータプッシュジョブに追加する必要があります。詳細な手順については、「[統合ジョブにカスタムクエリを追加する方法](#)」(166ページ)を参照してください。



注意: 関係プッシュジョブを実行する前に、既に関連するCIタイプ(この例では **Cost** と **CostCategory**)を統合に追加し、関連するCIをService Managerにプッシュしていることを確認してください。

統合ジョブにカスタムクエリを追加する方法



統合で、Service ManagerとUCMDBシステムの間で、カスタムCIタイプ、属性、および関係を同期するには、データプッシュジョブまたはポピュレーションジョブにカスタムクエリを追加する必要があります。

次の手順では、前に作成した **rdbmsData** という名前のカスタムクエリ ([「CI タイプを同期するためのクエリの作成方法」\(148ページ\)](#)) をデータプッシュジョブに追加する方法を示します。ポピュレーションクエリをポピュレーションジョブに追加する手順も同様です。

データプッシュジョブまたはポピュレーションジョブにカスタムクエリを追加するには:

1. 管理者として UCMDB にログインします。
2. **[データフロー管理]** > **[Integration Studio]** をクリックします。
3. Service Manager 統合ポイントの名前をクリックします。「**sm_integration**」などです。
4. **[データプッシュ]** タブを選択します。

注: ポピュレーションジョブにクエリを追加する場合は、代わりに **[ポピュレーション]** をクリックします。

5. 統合ジョブの名前をクリックします。「**SM Configuration Item Push job**」などです。
6. **[編集]** アイコン  をクリックします。
7. **[追加]** アイコンをクリックします。 
8. **[Integration]** > **[Service Manager]** > **[Push]** > **[rdbmsData]** をクリックします。

注: ポピュレーションの場合は、代わりに **[Integration]** > **[Service Manager]** > **[Population]** に移動し、ポピュレーションクエリをクリックします。

9. **[OK]** をクリックして、カスタムクエリを追加します。
10. クエリの **[削除を許可]** オプションを有効にし、統合ジョブで削除されたデータを消去できるようにします。
11. **[OK]** をクリックして、**[ジョブ定義の更新]** ウィンドウを閉じます。

ポピュレーション用の CI タイプ、属性、または関係タイプを統合に追加する方法

ポピュレーション用のすべての設定済みのマッピングファイルは、**[データフロー管理]** > **[アダプタ管理]** > **[ServiceManagerEnhancedAdapter9-x]** > **[ServiceManagerEnhancedAdapter9-x/mappings/population/<Name of Mapping File>]** を選択して表示できます。

ビジュアルマッピングツールを使用すると、データプッシュの場合と似た手順に従って、ポピュレーション用の CI タイプ、CI 属性、または関係を統合に追加できます。また、ポピュレーションクエリをポピュレーションジョブに追加する必要もあります。データプッシュに関する以下の手順を参照してください。

[「データプッシュ用の CI 属性を統合に追加する方法」\(131ページ\)](#)

[「データプッシュ用のCIタイプを統合に追加する方法」\(145ページ\)](#)

[「データプッシュ用のCI関係タイプを統合に追加する方法」\(160ページ\)](#)

[「統合ジョブにカスタムクエリを追加する方法」\(166ページ\)](#)

CIタイプのUCMDB ID プッシュバックを有効または無効にする方法

UCMDB で新しい CI が作成されると、UCMDB ID 値がその CI に割り当てられます。ポピュレーションを介して CI が Service Manager (SM) から UCMDB に同期される際に、UCMDB ID プッシュバック機能により、UCMDB ID 値をプッシュして SM に戻すことができます。その後、UCMDB ID フィールドは、ポピュレーションを介して CI が既に UCMDB に同期されているかどうかを示すフラグとして使用されます。

UCMDB ID のプッシュバック機能を有効にしたい場合もあります。例えば、UCMDB 内の `scheduled_downtime` CI タイプは、SM 内には物理的に存在しません。代わりに、統合が、SM 内の複数のエンティティから予定ダウンタイム情報を取得し、その情報を UCMDB 内の `scheduled_downtime` CI に同期します。このため、`scheduled_downtime` CI の UCMDB ID プッシュバック機能はデフォルトで無効になっています。無効になっていないとポピュレーション時にプッシュバックエラーが発生します。

CI タイプの UCMDB ID プッシュバックを有効にするには、次の手順に従います。

1. CI タイプのポピュレーション構成ファイル (**My Business Service Population.xml** など) に次のエンタリを追加します。

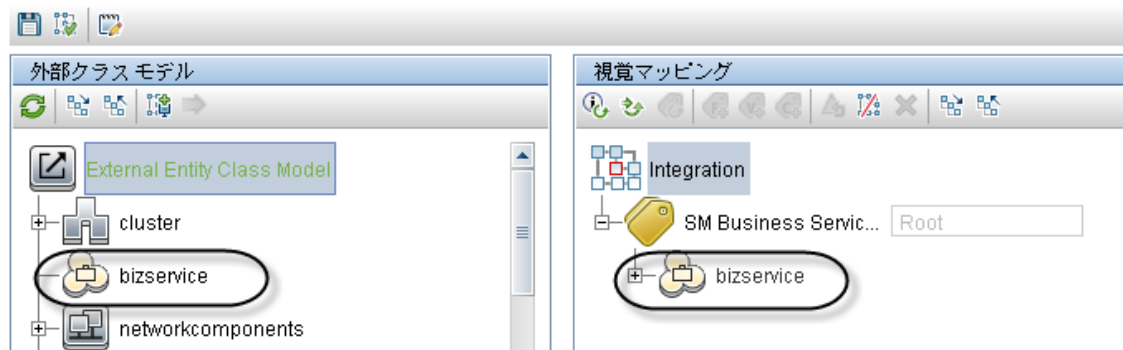
```
<target_mapping name="sm_id" datatype="STRING" value="bizservice['CIName']"/>
<target_mapping name="global_id" datatype="STRING" value="bizservice['UCMDBId']"/>
```

注: この例では、下の図に示すように、**bizservice** は、Service Manager で定義された CI タイプの表示名であり、UCMDB ビジュアルマッピングインタフェースに表示される外部クラスモデル名でもあります。

CIタイプの管理

CIタイプの説明:	Business Service
CIタイプ:	bizservice
ビットマップ:	lbox
フォーマット名:	configurationItem
属性ファイル:	bizservice
結合定義:	joinbizservice
印刷フォーマット名:	
一括更新フォーマット名:	device.businessservice.bulkupdate
アクティブ:	<input checked="" type="checkbox"/>
サブタイプ	
	Business Service
	Application Service
	Infrastructure Service

マッピング ツール - プッシュ シナリオ



2. XML 構成ファイルを保存します。
3. smPopConf.xml ファイルを開き、CI タイプの UCMDDB ID プッシュバックの設定が存在しないか、存在して true に設定されていることを確認します。

```
<pushback>
```

```
  <type name="business_service" enable="true"/>
```

```
</pushback>
```

ここで、<type name> は、UCMDDB 内の CI タイプの名前です。

CI タイプの UCMDDB ID プッシュバック機能を無効にするには、次の手順に従います。

1. smPopConf.xml ファイルを開きます。
2. <pushback> セクションに、CI タイプに関する次の 1 行を挿入します。

```
<config>
  <pushback>
    <type name="scheduled_downtime" enable="false"/>
    <type name="business_service" enable="false"/>
  </pushback>
  <mapping>
    <tql name="SM Business Service Population 2.0">
      ...
    </tql>
    ...
  </mapping>
</config>
```

連携用のサポートされる CI タイプの属性を追加する方法

初期設定状態の統合は、UCMDB の次の 3 つの外部 CI タイプの連携をサポートします。Incident、Problem、および RequestForChange です。UCMDB には、サポートされる CI タイプごとに属性のリストがあり、このリストを連携用の Service Manager Web サービスオブジェクトにマップすることができます。次の図は、Incident CI タイプで使用できる設定済みの UCMDB CI 属性を示しています。

サポートおよび選択された CI タイプ	CI タイプ取得モード
	<input type="radio"/> 選択した CI タイプの CI を取得 <input checked="" type="radio"/> 選択した属性の取得
	選択した属性は、統合から取得されます。CI は UCMDB に既に存在する必要があります。
	属性の選択
	<input type="checkbox"/> ActiveProcess <input type="checkbox"/> Actual Deletion Period <input type="checkbox"/> Allow CI Update <input type="checkbox"/> Category <input type="checkbox"/> ClosedTime <input type="checkbox"/> CompletionCode <input type="checkbox"/> Consumer Tenants <input type="checkbox"/> Container <input type="checkbox"/> Create Time <input type="checkbox"/> Created By <input type="checkbox"/> Deletion Candidate Period <input type="checkbox"/> Description <input type="checkbox"/> Details <input type="checkbox"/> Display Label <input type="checkbox"/> Enable Aging <input type="checkbox"/> Escalated <input type="checkbox"/> ExternalProcessReference <input type="checkbox"/> Global Id <input type="checkbox"/> ImpactScope <input type="checkbox"/> IncidentStatus <input type="checkbox"/> ...

たとえば、連携用の SM Incident 属性を追加するには、SM UcmdbIncident Web サービスオブジェクトでこのフィールドを公開し、その後でフィールドを適切な UCMDB 属性 (存在しない場合は先に UCMDB で作成する必要があります) にマップする必要があります。

次の図は、Service Manager の UcmdbIncident Web サービスオブジェクトで公開されたフィールドを示しています。

外部アクセス定義

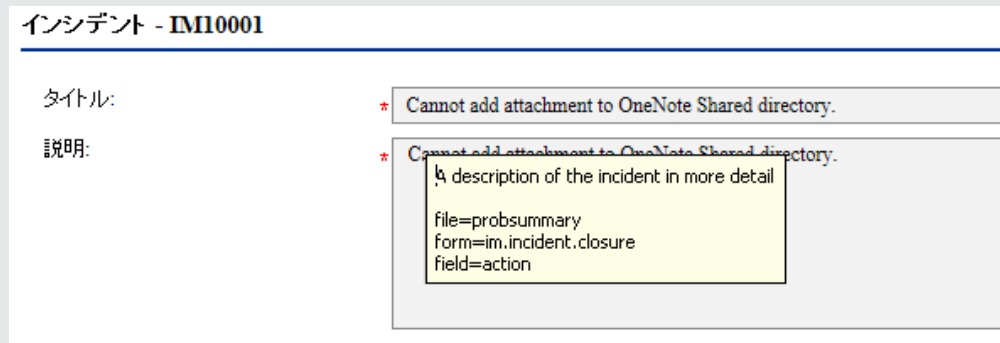
サービス名: * ucmdbIntegration リリース済み:
 名前: * probsummary 廃止予定:
 オブジェクト名: UcmdbIncident

許可されるアクション 式 フィールド RESTful

number	incidentID	
brief.description	BriefDescription	
subcategory	Category	
severity	Urgency	
open.time	OpenTime	DateTimeType
update.time	UpdatedTime	DateTimeType
close.time	ClosedTime	DateTimeType
problem.status	IMTicketStatus	
affected.item	Service	
priority.code	PriorityCode	
initial.impact	ImpactScope	

追加のフィールドを公開して、追加の Incident 属性を UCMDDB に連携させることができます。次に例として、連携のために Service Manager **probsummary** (Incident) ファイル内に“action”フィールドを追加する方法を説明します。これは、このフィールドを **details** という新しい UCMDDB 属性にマップすることで行います。

注: Service Manager 内のインシデントフォームで、“action”フィールドには、インシデントレコードをより詳しく説明するために使用される“Description”というラベルが付いています。次の図を参照してください。



連携用のサポートされる CI タイプの属性を追加するには:

1. **Web サービスオブジェクトに SM 属性を追加します。**
 次の例では、UcmdbIncident Web サービスオブジェクトでインシデントの SM “action” フィールドを公開する方法を説明します。

- a. システム管理者として Service Manager にログインします。
- b. [カスタマイズ]>[Web サービス]>[Web サービス構成]に移動します。
- c. 次のフィールドの値を入力し、[検索]をクリックします。
 - o サービス名:ucmdbIntegration
 - o 名前:Probsummary

UcmdbIncident Web サービスオブジェクトが表示されます。

- d. [フィールド]タブに以下の行を追加します。
 - o フィールド:action
 - o キャプション:Description

外部アクセス定義

サービス名: リリース済み:
 名前: 廃止予定:
 オブジェクト名:

許可されるアクション 式 フィールド RESTful

oner.description	OpenDescription	
subcategory	Category	
severity	Urgency	
open.time	OpenTime	DateTimeType
update.time	UpdatedTime	DateTimeType
close.time	ClosedTime	DateTimeType
problem.status	IMTicketStatus	
affected.item	Service	
priority.code	PriorityCode	
initial.impact	ImpactScope	
action	Description	

- e. Web サービスオブジェクトを保存します。
2. **SM 属性を UCMDDB 属性にマップします。**
 次の例では、SM の“action” 属性を details という名前の新しい UCMDDB 属性にマップする方法について説明します。

- a. 管理者として UCMDB にログインします。
- b. [モデリング]>[CI タイプマネージャ]に移動します。
- c. [ItProcessRecord]>[Incident]に移動し、そのプロパティ表示 枠を開きます。
- d. [追加]アイコンをクリックして、**details** という名前の新しい属性を Incident CI タイプに追加します。
 - o 名前: details
 - o 表示名: 詳細
 - o 説明: インシデントの詳細
 - o 属性タイプ: [プリミティブ]>[string] を選択します。
 - o 値のサイズ: 500

The screenshot shows a dialog box titled "属性の追加" (Add Property). It has three tabs: "詳細" (Details), "詳細設定" (Detailed Settings), and "UCMDB Browser 修飾子" (UCMDB Browser Decorator). The "詳細" tab is selected. The fields are as follows:

- 属性名: details
- 表示名: Details
- スコープ: CMS
- 説明: Incident details
- 属性タイプ: プリミティブ... 列挙/リスト
string
- 値のサイズ: 500
- 標準設定値ポリシー: 標準設定値の有効化
- 標準設定値: (empty)

Buttons: OK, キャンセル

- e. インシデント CI タイプレコードを保存します。

- f. [データフロー管理]>[アダプタ管理]>[ServiceManagerEnhancedAdapter9-x]>[構成ファイル]に移動します。
- g. 関係する連携マッピングファイル (**ServiceManagerEnhancedAdapter9-x/mappings/federation/SM Incident 2.0.xml**) を編集して、新しい属性のマッピングエントリを追加します。

```
<target_entities>
  <source_instance query-name="SM Incident 2.0" root-element-
name="ucmdbIncident">
    <target_entity name="incident">
      <target_mapping name="reference_number"
datatype="STRING" value="ucmdbIncident['IncidentID']"/>
      <target_mapping name="name"
datatype="STRING" value="ucmdbIncident['BriefDescription']"/>
      <target_mapping name="priority"
datatype="STRING" value="SMFederationFunctions.getEnumValue
(ucmdbIncident['PriorityCode'], 'Priority')"/>
      <target_mapping name="incident_status"
datatype="STRING"
value="SMFederationFunctions.firstLetterToLowerAndReplaceSpaceWithUnders
core(ucmdbIncident['IMTicketStatus'])"/>
      <target_mapping name="category"
datatype="STRING"
value="SMFederationFunctions.replaceSpaceWithUnderscore(ucmdbIncident
['Category'])"/>
      <target_mapping name="closed_time"
datatype="DATE" value="SMFederationFunctions.convertDate
(ucmdbIncident['ClosedTime'])"/>
      <target_mapping name="create_time"
datatype="DATE" value="SMFederationFunctions.convertDate
(ucmdbIncident['OpenTime'])"/>
      <target_mapping name="last_modified_time"
datatype="DATE" value="SMFederationFunctions.convertDate
(ucmdbIncident['UpdatedTime'])"/>
      <target_mapping name="impact_scope"
datatype="STRING" value="SMFederationFunctions.getEnumValue
(ucmdbIncident['ImpactScope'], 'ImpactScope')"/>
      <target_mapping name="urgency"
datatype="STRING" value="SMFederationFunctions.getEnumValue
(ucmdbIncident['Urgency'], 'Urgency')"/>
      <target_mapping name="details" datatype="STRING"
value="ucmdbIncident['Description']"/>
    </target_entity>
  </source_instance>
</target_entities>
```

- h. [保存]をクリックしてファイルを保存します。

次に、SM インシデントの説明 (フィールド名 : action) 属性が連携のために統合に追加されます。UCMDB モデリングスタジオでインシデント連携クエリを実行し、SM の説明データが正しく連携されているかどうかを確認します。連携クエリの実行方法の詳細については、「[連携の使用例](#)」(47ページ)を参照してください。

次の図は、SM インシデントレコードの説明が **Details** として UCMDB に連携されている例を示しています。



Property Name	Value
Actual Deletion Period	40
Category	failure
ClosedTime	
Create Time	2014年1月15日 (水曜日) 07:21 EET
Deletion Candidate Period	20
Details	[Virus scanner blocks e-mail attachments]
Display Label	IM10005
ImpactScope	user
IncidentStatus	Work_In_Progress
LastModifiedTime	Mon Jan 5 2015 10:52 PM IST
Name	E-mail attachments being blocked
Priority	2_high
ReferenceNumber	IM10005
Urgency	1_critical

第6章:トラブルシューティング

データプッシュやポピュレーションのエラーが発生した場合は、エラーメッセージと統合ログファイルを確認し、原因を特定してエラーを修正する必要があります。本章では、一般的なトラブルシューティング手順、一般的なエラー、および解決策について説明します。

本項の内容

- [「データプッシュの問題のトラブルシューティング」\(176ページ\)](#)
- [「ポピュレーションの問題のトラブルシューティング」\(185ページ\)](#)

データプッシュの問題のトラブルシューティング

データプッシュのエラーや問題が発生した場合は、エラーメッセージとログファイルを確認し、原因を特定してエラーを修正する必要があります。

この統合では、データプッシュに関して次のエラーコードを使用します。

SM エラーコード	説明	UCMDB エラーコード
-1	SM からのサーバアプリケーションエラー	900008
-4	SM からの 401 認証エラー	900001
3	データがロックされていることを示す SM からのエラー	900003
9	Restful 構成が見つからない	900009
51	別のプロセスによって変更されたことを示す SM からのエラー	900004
70	操作が無効なことを示す SM からのエラー	900005
71	検証に失敗したことを示す SM からのエラー	900002
881	データが無効なことを示す SM からのエラー	900007
882	関係操作の CI が SM で見つからない	900010
	SM でメッセージの作成に失敗した	900013
	SM との通信に失敗した	900014
	SM からのその他のエラー	900099

データプッシュジョブに失敗すると、ジョブのステータスが[失敗]になります。次のようにして、失敗したジョブのトラブルシューティングを行います。

- Universal CMDB Studio で失敗したジョブのエラーメッセージを確認します。
「失敗したプッシュジョブのエラーメッセージを確認する方法」(177ページ)を参照してください。
- ログファイルで詳細を確認します。
「プッシュのログファイルを確認する方法」(179ページ)を参照してください。

一部のレコードが失敗した状態でデータプッシュジョブが完了した場合、ジョブのステータスが[完了]になります。次のようにして、失敗したレコードのトラブルシューティングを行います。

- Universal CMDB Studio で失敗した CI のエラーメッセージを確認します。
「プッシュジョブで失敗した CI または関係のエラーメッセージを確認する方法」(178ページ)を参照してください。
- ログファイルで詳細を確認します。
「プッシュのログファイルを確認する方法」(179ページ)を参照してください。

失敗したプッシュジョブのエラーメッセージを確認する方法

失敗したプッシュジョブのエラーメッセージを確認するには:

1. 管理者として UCMDDB にログインします。
2. [データフロー管理]>[Integration Studio]をクリックします。
3. [統合ポイント]リストからこの統合の統合ポイントを選択します。
4. [データプッシュ]タブを選択します。
5. 統合ジョブからジョブを選択します。
6. [ジョブエラー]サブタブをクリックし、リストでメッセージの[重大度]をダブルクリックします。ポップアップウィンドウにこの失敗したジョブの詳細なエラーメッセージが表示されます。以下に示すのは、プッシュ TQL クエリのマッピング構成が見つからなかったことを示すエラーメッセージのサンプルの一部です。

```
java.lang.RuntimeException:No mapping is found for TQL: "SM Business Service
  Push 2.0", or Cannot retrieve the mapping from SM side by QueryNodeName
  [bizservice", please configure in smPushConf.xml or SM configuration
    at
  com.mercury.topaz.fcmdb.adapters.serviceDeskAdapter.push.SmGenericPusher.pus
  h(SmGenericPusher.java:119)
    ... 35 more
  --- End of probe-side exception ---

  at
  com.hp.ucmdb.discovery.probe.agents.probemgr.adhoctasks.AdHocProbeRequestOpe
  ration.convertThrowableToStringSafeException
```

```
(AdHocProbeRequestOperation.java:86)
    at
com.hp.ucmdb.discovery.probe.agents.probemgr.adhoctasks.AdHocProbeRequestOpe
ration.performAction(AdHocProbeRequestOperation.java:77)
    at
com.hp.ucmdb.discovery.probe.agents.probemgr.taskdispatcher.AdHocTaskDispatc
her.dispatchTask(AdHocTaskDispatcher.java:70)
    at sun.reflect.GeneratedMethodAccessor59.invoke(Unknown Source)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke
(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:601)
    at com.sun.jmx.mbeanserver.StandardMBeanIntrospector.invokeM2
(StandardMBeanIntrospector.java:111)
    at com.sun.jmx.mbeanserver.StandardMBeanIntrospector.invokeM2
(StandardMBeanIntrospector.java:45)
    at com.sun.jmx.mbeanserver.MBeanIntrospector.invokeM
(MBeanIntrospector.java:235)
    at com.sun.jmx.mbeanserver.PerInterface.invoke(PerInterface.java:138)
    at com.sun.jmx.mbeanserver.MBeanSupport.invoke(MBeanSupport.java:252)
    at javax.management.StandardMBean.invoke(StandardMBean.java:405)
    at com.sun.jmx.interceptor.DefaultMBeanServerInterceptor.invoke
(DefaultMBeanServerInterceptor.java:819)
    at com.sun.jmx.mbeanserver.JmxMBeanServer.invoke(JmxMBeanServer.java:792)
    at javax.management.MBeanServerInvocationHandler.invoke
(MBeanServerInvocationHandler.java:305)
    at org.springframework.jmx.access.MBeanClientInterceptor.doInvoke
(MBeanClientInterceptor.java:405)
    at org.springframework.jmx.access.MBeanClientInterceptor.invoke
(MBeanClientInterceptor.java:353)
    at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed
(ReflectiveMethodInvocation.java:172)
    at org.springframework.aop.framework.JdkDynamicAopProxy.invoke
(JdkDynamicAopProxy.java:202)
    at com.sun.proxy.$Proxy57.dispatchTask(Unknown Source)
    at
com.hp.ucmdb.discovery.probe.agents.probegw.managementtasks.adhoctasks.Adhoc
Thread.run(AdhocThread.java:54)
... 3 more
```

プッシュジョブで失敗したCIまたは関係のエラーメッセージを確認する方法

一部のレコードが失敗した状態でデータプッシュジョブが完了した場合、Universal CMDB Studio で、失敗した各レコードの詳細なエラーメッセージを確認できます。

データプッシュジョブで失敗したレコードのエラーメッセージを確認するには:

1. 管理者として UCMDB にログインします。
2. [データフロー管理]>[Integration Studio]をクリックします。
3. [統合ポイント]リストからこの統合の統合ポイントを選択します。
4. [データプッシュ]タブを選択します。
5. 統合ジョブからジョブを選択します。
6. [クエリのステータス]サブタブをクリックします。
7. 失敗したクエリをダブルクリックします。エラーメッセージおよび失敗した各 CI タイプの CI 数が表示されます。

重大度	CI数	CIタイプ	メッセージ
エラー	7	BusinessService	SMからのサーバアプリケーション...

8. エラーメッセージをダブルクリックします。失敗したレコードのリストが表示されます。
9. 失敗したレコードをダブルクリックします。レコードの詳細なエラーメッセージが表示されます。

プッシュのログファイルを確認する方法

UCMDB のプッシュの結果ツリーノードを確認し、Service Manager とメッセージを送受信できるようにするには、開発アダプタのログレベルを DEBUG または TRACE に設定する必要があります。または、ログレベルを TRACE に設定して、マッピングファイルを基にして変換時間を確認できます。

注: プッシュ、ポピュレーション、および連携の問題のトラブルシューティングを行うには、**fcmdb.properties** ファイルおよび **fcmdb.push.properties** ファイルで開発アダプタのログレベルを DEBUG または TRACE に設定する必要があります。これらのファイルは、Data Flow Probe または統合サービスのインストールフォルダの `\conf\log` フォルダにあります。さらに、**fcmdb.push.all.log** ファイルでプッシュ、ポピュレーション、および連携のログ情報を表示できます。このファイルは、Data Flow Probe または統合サービスのインストールフォルダの `\runtime\log` フォルダにあります。Data Flow Probe または統合サービスは、UCMDB サーバと同じホストにインストールすることも別のホストにインストールすることもできます。

開発アダプタのログレベルを DEBUG または TRACE に設定するには:

1. UCMDB Data Flow Probe または統合サービスがインストールされているホストに管理者としてログインします。
2. <UCMDB インストールフォルダ>\DataFlowProbe\conf\log フォルダまたは <UCMDB インストールフォ

ルダUCMDBServer\integrations\conf\logに移動します。例:

```
C:\hp\UCMDB\DataFlowProbe\conf\log\
```

または C:\hp\UCMDB\UCMDBServer\integrations\conf\log

3. テキストエディタで **fcmdb.properties** ファイルを開き、ログレベルを `DEBUG` または `TRACE` に変更します。例:

```
#loglevel can be any of TRACE DEBUG INFO WARN ERROR FATAL
loglevel=DEBUG
def.file.max.size=5000KB
def.files.backup.count=10
msg.layout=%d %-5p [%t] - %m%n
```

4. テキストエディタで **fcmdb.push.properties** ファイルを開き、log4j ログレベルを `DEBUG` または `TRACE` に変更します。例:

```
### UCMDB log4j Properties
log.file.path=log/${log.folder.path.output}
#loglevel can be any of TRACE DEBUG INFO WARN ERROR FATAL
loglevel=DEBUG
def.file.max.size=5000KB
def.files.backup.count=10
msg.layout=%d %-5p [%t] - %m%n
```

5. ファイルを保存します。

プッシュのログファイルを確認するには:

1. UCMDB Data Flow Probe または統合 サービスがインストールされているホストに管理者としてログインします。
2. <UCMDB インストールフォルダ>\DataFlowProbe\runtime\log フォルダまたは <UCMDB インストールフォルダ>UCMDBServer\integrations\runtime\log フォルダに移動します。
3. テキストエディタで **fcmdb.push.all.log** ファイルを開きます。

プッシュの一般的なエラーメッセージと解決策

本項では、データプッシュ時に発生する可能性がある一般的なエラーメッセージとそれらの解決策について説明します。

本項の内容

- 「クエリが `smPushConf.xml` で構成されていない」(181ページ)
- 「マッピングファイルが `well formed (整形式)` ではない」(182ページ)

クエリが smPushConf.xml で構成されていない

サンプル構成

関係をプッシュするために使用される TQL クエリは、smPushConf.xml ファイルで構成する必要があります。

```
<config>
  <mapping>
    <!--
      現在、マッピングを構成するときに TQL 名のワイルドカードがサポートされます。
      マッピングを構成するときに TQL 名の末尾に '*' を追加できます。
      OOTB でのマッピングにワイルドカードを使用すると、TQL に名前を付けて保存するたびに、
      マッピングを手動で変更しなくとも SM にプッシュできます。
      たとえば、<TQL_name> クエリを <TQL_name>_1 および <TQL_name>_2 に保存する場合、
      この構成ファイルでは TQL 名は <TQL_name>* として指定され、
      統合は自動的にこのマッピングエントリを 3 つの TQL すべてで使用します。
      ただし、正確な TQL 名を使用したマッピングの構成も引き続きサポートされます。
    -->
    <tql name="SM Business Service Relations Push 2.0"
      resourceCollectionName="Relationships" resourceName="Relationship"/>
    <tql name="SM CRG Relations Push 2.0"
      resourceCollectionName="Relationships" resourceName="Relationship"/>
    <tql name="SM Node Relations Push 2.0"
      resourceCollectionName="Relationships" resourceName="Relationship"/>
    <tql name="SM Layer2 Topology Relations Push 2.0"
      resourceCollectionName="Relationships" resourceName="Relationship"/>
  </mapping>
</config>
```

エラーメッセージ

「失敗」ステータスでプッシュジョブが失敗します。Universal CMDB Studio のログファイルと失敗したジョブの詳細エラーメッセージの両方に(「[失敗したプッシュジョブのエラーメッセージを確認する方法](#)」(177ページ)を参照してください)、次のようなエラーが表示されます。

```
java.lang.RuntimeException: No mapping is found for TQL: "SM Business Service Push
  2.0", or Cannot retrieve the mapping from SM side by QueryNodeName [bizservice",
  please configure in smPushConf.xml or SM configuration
  at
  com.mercury.topaz.fcldb.adapters.serviceDeskAdapter.push.SmGenericPusher.push
  (SmGenericPusher.java:119)
  ... 35 more
  --- End of probe-side exception ---

  at
  com.hp.ucmdb.discovery.probe.agents.probemgr.adhocktasks.AdHocProbeRequestOperati
  on.convertThrowableToStringSafeException(AdHocProbeRequestOperation.java:86)
  at
```

```
com.hp.ucmdb.discovery.probe.agents.probemgr.adhoctasks.AdHocProbeRequestOperation.performAction(AdHocProbeRequestOperation.java:77)
    at
com.hp.ucmdb.discovery.probe.agents.probemgr.taskdispatcher.AdHocTaskDispatcher.dispatchTask(AdHocTaskDispatcher.java:70)
    at sun.reflect.GeneratedMethodAccessor59.invoke(Unknown Source)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke
(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:601)
    at com.sun.jmx.mbeanserver.StandardMBeanIntrospector.invokeM2
(StandardMBeanIntrospector.java:111)
    at com.sun.jmx.mbeanserver.StandardMBeanIntrospector.invokeM2
(StandardMBeanIntrospector.java:45)
    at com.sun.jmx.mbeanserver.MBeanIntrospector.invokeM
(MBeanIntrospector.java:235)
    at com.sun.jmx.mbeanserver.PerInterface.invoke(PerInterface.java:138)
    at com.sun.jmx.mbeanserver.MBeanSupport.invoke(MBeanSupport.java:252)
    at javax.management.StandardMBean.invoke(StandardMBean.java:405)
    at com.sun.jmx.interceptor.DefaultMBeanServerInterceptor.invoke
(DefaultMBeanServerInterceptor.java:819)
    at com.sun.jmx.mbeanserver.JmxMBeanServer.invoke(JmxMBeanServer.java:792)
    at javax.management.MBeanServerInvocationHandler.invoke
(MBeanServerInvocationHandler.java:305)
    at org.springframework.jmx.access.MBeanClientInterceptor.doInvoke
(MBeanClientInterceptor.java:405)
    at org.springframework.jmx.access.MBeanClientInterceptor.invoke
(MBeanClientInterceptor.java:353)
    at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed
(ReflectiveMethodInvocation.java:172)
    at org.springframework.aop.framework.JdkDynamicAopProxy.invoke
(JdkDynamicAopProxy.java:202)
    at com.sun.proxy.$Proxy57.dispatchTask(Unknown Source)
    at
com.hp.ucmdb.discovery.probe.agents.probegw.managementtasks.adhoctasks.AdhocThread.run(AdhocThread.java:54)
... 3 more
```

解決策

No mapping is found for TQL というテキスト文字列を検索し、まだ構成されていないクエリの名前を見つけ、smPushConf.xml ファイルにそのクエリのマッピングエントリを追加します。

マッピングファイルが well formed (整形式) ではない

サンプル構成

"target_entity name" は、**bizservice** (Service Manager で定義される CI タイプの表示名、および UCMDb ビジュアルマッピングインタフェースに表示される外部クラスモデル名) にする必要がありますが、誤った名前 **businessservice** で構成しています。

```
<?xml version="1.0" encoding="UTF-8"?>
<integration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../mappings_schema.xsd">
  <info>
    <source name="UCMDB" version="10.20" vendor="HP"/>
    <target name="SM" version="9.40" vendor="HP"/>
  </info>
  <import>
    <scriptFile path="mappings_scripts.SMPushFunctions"/>
  </import>
  <!--
    uCMDB CIT を SM Business Service にプッシュする。
  -->
  <target_entities>
    <source_instance query-name="SM Business Service Push 2.0" root-element-
name="Root" >
      <target_entity name="businessservice">
        <target_mapping name="UCMDBId" datatype="STRING" value="Root
['global_id']"/>
        <target_mapping name="CustomerId" datatype="STRING"
value="SMPushFunctions.getCustomerId(CustomerInformation)"/>
        <target_mapping name="Type" datatype="STRING"
value="'bizservice'"/>
        <target_mapping name="Subtype" datatype="STRING"
value="SMPushFunctions.getSMSubType(Root['element_
type'],ClassModel,'BizService')"/>
        <target_mapping name="ServiceProvider" datatype="STRING"
value="Root['provider']"/>
        <target_mapping name="ServiceName" datatype="STRING" value="Root
['display_label']"/>
        <target_mapping name="CIIdentifier" datatype="STRING"
value="Root['display_label']"/>
      </target_entity>
    </source_instance>
  </target_entities>
</integration>
```

エラーメッセージ

「失敗」ステータスでデータプッシュジョブが失敗します。Universal CMDB Studio のログファイルと失敗したジョブのエラーメッセージの両方に(「[失敗したプッシュジョブのエラーメッセージを確認する方法](#)」(177ページ)を参照してください)、次のようなエラーが表示されます。

```
java.lang.RuntimeException:No mapping is found for TQL: "SM Business Service Push
2.0", or Cannot retrieve the mapping from SM side by QueryNodeName
[businessservice", please configure in smPushConf.xml or SM configuration
at
com.mercury.topaz.fcmdb.adapters.serviceDeskAdapter.push.SmGenericPusher.push
(SmGenericPusher.java:119)
... 35 more
```

```
--- End of probe-side exception ---  
  
    at  
com.hp.ucmdb.discovery.probe.agents.probemgr.adhocktasks.AdHocProbeRequestOperati  
on.convertThrowableToStringSafeException(AdHocProbeRequestOperation.java:86)  
    at  
com.hp.ucmdb.discovery.probe.agents.probemgr.adhocktasks.AdHocProbeRequestOperati  
on.performAction(AdHocProbeRequestOperation.java:77)  
    at  
com.hp.ucmdb.discovery.probe.agents.probemgr.taskdispatcher.AdHocTaskDispatcher.  
dispatchTask(AdHocTaskDispatcher.java:70)  
    at sun.reflect.GeneratedMethodAccessor59.invoke(Unknown Source)  
    at sun.reflect.DelegatingMethodAccessorImpl.invoke  
(DelegatingMethodAccessorImpl.java:43)  
    at java.lang.reflect.Method.invoke(Method.java:601)  
    at com.sun.jmx.mbeanserver.StandardMBeanIntrospector.invokeM2  
(StandardMBeanIntrospector.java:111)  
    at com.sun.jmx.mbeanserver.StandardMBeanIntrospector.invokeM2  
(StandardMBeanIntrospector.java:45)  
    at com.sun.jmx.mbeanserver.MBeanIntrospector.invokeM  
(MBeanIntrospector.java:235)  
    at com.sun.jmx.mbeanserver.PerInterface.invoke(PerInterface.java:138)  
    at com.sun.jmx.mbeanserver.MBeanSupport.invoke(MBeanSupport.java:252)  
    at javax.management.StandardMBean.invoke(StandardMBean.java:405)  
    at com.sun.jmx.interceptor.DefaultMBeanServerInterceptor.invoke  
(DefaultMBeanServerInterceptor.java:819)  
    at com.sun.jmx.mbeanserver.JmxMBeanServer.invoke(JmxMBeanServer.java:792)  
    at javax.management.MBeanServerInvocationHandler.invoke  
(MBeanServerInvocationHandler.java:305)  
    at org.springframework.jmx.access.MBeanClientInterceptor.doInvoke  
(MBeanClientInterceptor.java:405)  
    at org.springframework.jmx.access.MBeanClientInterceptor.invoke  
(MBeanClientInterceptor.java:353)  
    at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed  
(ReflectiveMethodInvocation.java:172)  
    at org.springframework.aop.framework.JdkDynamicAopProxy.invoke  
(JdkDynamicAopProxy.java:202)  
    at com.sun.proxy.$Proxy57.dispatchTask(Unknown Source)  
    at  
com.hp.ucmdb.discovery.probe.agents.probegw.managementtasks.adhocktasks.AdhocThre  
ad.run(AdhocThread.java:54)  
    ... 3 more
```

解決策

Service Manager で関連する CI タイプを検索し、正しい表示名を見つけて、正しい名前になるようにマッピングファイルを変更します。

ヒント: UCMDB ビジュアルマッピングツールを使用してマッピングファイルを生成すると、このような検証

の問題を簡単に修正できます。

ポピュレーションの問題のトラブルシューティング

ポピュレーションのエラーや問題が発生した場合は、エラーメッセージとポピュレーションログファイルを確認し、原因を特定して問題を解決する必要があります。

ポピュレーションジョブに失敗すると、ジョブのステータスが[失敗]になります。次のようにして、失敗したジョブのトラブルシューティングを行います。

- Universal CMDB Studio で失敗したジョブのエラーメッセージを確認します。
「失敗したポピュレーションジョブのエラーメッセージを確認する方法」(185ページ)および「ポピュレーションの一般的なエラーメッセージと解決策」(186ページ)を参照してください。
- ログファイルで詳細を確認します。
「ポピュレーションのログファイルを確認する方法」(185ページ)を参照してください。

失敗したポピュレーションジョブのエラーメッセージを確認する方法

ポピュレーションジョブが失敗した場合、Universal CMDB Studio で詳細なエラーメッセージを確認できません。

失敗したポピュレーションジョブのエラーメッセージを確認するには:

1. 管理者として UCMDDB にログインします。
2. [データフロー管理]>[Integration Studio]をクリックします。
3. この統合の統合ポイントを選択します。
4. [ポピュレーション]タブをクリックします。
5. 統合ジョブから失敗したジョブを選択し、[ジョブ エラー]サブタブをクリックします。
6. リストでエラーメッセージをダブルクリックします。
ポップアップウィンドウが開き、エラーの詳細が表示されます。

ポピュレーションのログファイルを確認する方法

開発アダプタのログレベルを DEBUG に設定し、UCMDDB のポピュレーションの結果ツリーノードを確認し、Service Manager とメッセージを送受信することができます。または、ログレベルを TRACE に設定して、マッピングファイルを基にして変換時間を確認できます。

注: プッシュ、ポピュレーション、および連携の問題のトラブルシューティングを行うには、**fcmdb.properties** ファイルおよび **fcmdb.push.properties** ファイルで開発アダプタのログレベルを DEBUG または TRACE に設定する必要があります。これらのファイルは、Data Flow Probe または統合サービスのインストールフォルダの \conf\log フォルダにあります。さらに、**fcmdb.push.all.log** ファイルでプッシュ、ポピュレーション、および連携のログ情報を表示できます。このファイルは、Data Flow Probe または統合サービスのインストールフォルダの \runtime\log フォルダにあります。Data Flow Probe または統合サービスは、UCMDB サーバと同じホストにインストールすることも別のホストにインストールすることもできます。

開発アダプタのログレベルを設定し、ポピュレーションログファイルを表示するには、「[プッシュのログファイルを確認する方法](#)」(179ページ)で説明されている手順に従います。

ポピュレーションの一般的なエラーメッセージと解決策

本項では、ポピュレーション時に発生する可能性がある一般的なエラーメッセージとそれらの解決策について説明します。

本項の内容

- 「[smPopConf.xml で TQL クエリが構成されていない](#)」(186ページ)
- 「[smPopConf.xml で TQL クエリのマッピングファイル名が定義されていない](#)」(189ページ)

smPopConf.xml で TQL クエリが構成されていない

エラーメッセージ

TQL クエリをジョブに追加していない場合、ジョブを作成または更新するときにこのクエリをリストから選択できません。

この TQL クエリをすでにジョブに追加していても、smPopConf.xml でこのクエリの構成を追加していない場合、このポピュレーションジョブを実行したときに「失敗」ステータスになります。さらに、Universal CMDB Studio で次のようなエラーメッセージが表示されます（「[失敗したポピュレーションジョブのエラーメッセージを確認する方法](#)」(185ページ)を参照してください）。

```
running population. Destination ID: sm, Failed during query: SM Business
Application Population 2.0, all queries: [SM Business Application Population
2.0, SM Business Service Population 2.0], finished queries: [].
ERROR: com.mercury.topaz.cmdb.shared.base.CmdbException: [ErrorCode [802]
General Integration Error{sm}]
appilog.framework.shared.manage.impl.MamResponseException: [ErrorCode [802]
General Integration Error{sm}]
CMDB Operation Internal Error: class
com.mercury.topaz.cmdb.shared.fcmdb.dataAccess.exception.AdapterAccessGeneralExc
eption : Unsupported Query [SM Business Application Population 2.0], only
population TQL is supported : operation Data Access Adapter Query: Retrieve
Changed Data
at
```

```

com.mercury.topaz.cmdb.shared.manage.operation.impl.AbstractCommonOperation.execute(AbstractCommonOperation.java:160)
    at
com.hp.ucmdb.dataAccess.manager.DataAccessAdapterManagerProbeImpl.executeOperation(DataAccessAdapterManagerProbeImpl.java:50)
    at
com.hp.ucmdb.discovery.probe.agents.probemgr.adapters.DataAccessAdaptersFacade.invokeOperation(DataAccessAdaptersFacade.java:406)
    at
com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.runChangesOnPopulateChangesAdapter(AdapterService.java:1262)
    at
com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.runQueriesOnPopulateChangesAdapter(AdapterService.java:1102)
    at com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.runQueries(AdapterService.java:354)
    at
com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.runDiscovery(AdapterService.java:198)
    at com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.discover(AdapterService.java:149)
    at
com.hp.ucmdb.discovery.probe.agents.probemgr.taskexecutor.JobExecutor.launchTask(JobExecutor.java:1188)
    at
com.hp.ucmdb.discovery.probe.agents.probemgr.taskexecutor.JobExecutor$JobExecutorWorker.launch(JobExecutor.java:945)

```

fcmdb.push.all.log ファイルで、次のような詳細を参照できます。

```

2014-11-20 10:40:50,949 [JobExecutorWorker-0:DS_sm_SM BS pop] ERROR - sm >> Fail to
Create or Return PopulationConnectorOutput
com.hp.ucmdb.federationspi.exception.DataAccessGeneralException: Unsupported
Query [SM Business Application Population 2.0], only population TQL is supported
    at com.hp.ucmdb.adapter.smpush.ServiceManagerGenericAdapter.populate
(ServiceManagerGenericAdapter.java:337)
    at com.hp.ucmdb.adapters.GenericAdapter.getChanges(GenericAdapter.java:881)
    at
com.hp.ucmdb.dataAccess.operations.DataAccessAdapterQueryRetrieveChanges.getChangesResult(DataAccessAdapterQueryRetrieveChanges.java:50)
    at
com.hp.ucmdb.dataAccess.operations.DataAccessAdapterQueryRetrieveChanges.doDataAccessQueryExecute(DataAccessAdapterQueryRetrieveChanges.java:38)
    at
com.hp.ucmdb.dataAccess.operations.AbstractDataAccessLifeCycleAdapterQuery.doLifeCycleExecute(AbstractDataAccessLifeCycleAdapterQuery.java:34)
    at
com.hp.ucmdb.dataAccess.operations.AbstractDataAccessLifeCycleAdapterOperation.doDataAccessExecute(AbstractDataAccessLifeCycleAdapterOperation.java:57)
    at

```

```
com.hp.ucmdb.dataAccess.operations.AbstractDataAccessAdapterOperation.dataAccess
Execute(AbstractDataAccessAdapterOperation.java:59)
    at
com.hp.ucmdb.dataAccess.operations.AbstractDataAccessAdapterOperation.doExecute
(AbstractDataAccessAdapterOperation.java:37)
    at
com.mercury.topaz.cmdb.shared.manage.operation.impl.AbstractFrameworkOperation.c
ommonExecute(AbstractFrameworkOperation.java:17)
    at
com.mercury.topaz.cmdb.shared.manage.operation.impl.AbstractCommonOperation$Oper
ationExecuteFlowTrackingCommand.execute(AbstractCommonOperation.java:87)
    at
com.mercury.topaz.cmdb.shared.manage.operation.impl.AbstractCommonOperation$Oper
ationExecuteFlowTrackingCommand.execute(AbstractCommonOperation.java:60)
    at com.mercury.topaz.cmdb.shared.manage.flowmanagement.api.FlowManager.execute
(FlowManager.java:227)
    at
com.mercury.topaz.cmdb.shared.manage.operation.flow.OperationInFlowDefaultExecut
or.execute(OperationInFlowDefaultExecutor.java:23)
    at
com.mercury.topaz.cmdb.shared.manage.operation.impl.AbstractCommonOperation.exec
ute(AbstractCommonOperation.java:158)
    at
com.hp.ucmdb.dataAccess.manager.DataAccessAdapterManagerProbeImpl.executeOperati
on(DataAccessAdapterManagerProbeImpl.java:50)
    at
com.hp.ucmdb.discovery.probe.agents.probemgr.adapters.DataAccessAdaptersFacade.i
nvokeOperation(DataAccessAdaptersFacade.java:406)
    at
com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.runChangesOnPo
pulateChangesAdapter(AdapterService.java:1262)
    at
com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.runQueriesOnPo
pulateChangesAdapter(AdapterService.java:1102)
    at com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.runQueries
(AdapterService.java:354)
    at
com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.runDiscovery
(AdapterService.java:198)
    at com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.discover
(AdapterService.java:149)
    at
com.hp.ucmdb.discovery.probe.agents.probemgr.taskexecuter.JobExecuter.launchTask
(JobExecuter.java:1188)
    at
com.hp.ucmdb.discovery.probe.agents.probemgr.taskexecuter.JobExecuter$JobExecute
rWorker.launch(JobExecuter.java:945)
    at
```

```
com.hp.ucmdb.discovery.probe.agents.probemgr.taskexecuter.JobExecuter$JobExecute
rWorker.executeTask(JobExecuter.java:867)
    at
com.hp.ucmdb.discovery.probe.agents.probemgr.taskexecuter.JobExecuter$JobExecute
rWorker.run(JobExecuter.java:728)
```

解決策

“Unsupported Query by this adapter” というテキスト文字列を検索し、まだ構成されていない TQL クエリ名を見つけて、smPopConf.xml ファイルでそのクエリを構成します。

smPopConf.xml で TQL クエリのマッピングファイル名が定義されていない

エラーメッセージ

ポピュレーションジョブを実行しているときに「失敗」ステータスになります。さらに、Universal CMDB Studio で次のようなエラーメッセージが表示されます。

```
running population. Destination ID: sm, Failed during query: SM Business Service
Population 2.0, all queries: [SM Business Application Population 2.0, SM
Business Service Population 2.0], finished queries: [SM Business Application
Population 2.0].
ERROR: com.mercury.topaz.cmdb.shared.base.CmdbException: [ErrorCode [802]
General Integration Error{sm}]
appilog.framework.shared.manage.impl.MamResponseException: [ErrorCode [802]
General Integration Error{sm}]
CMDB Operation Internal Error: class
com.mercury.topaz.cmdb.shared.fcldb.dataAccess.exception.AdapterAccessGeneralExc
eption : Query Definition [SM Business Service Population 2.0] has no matching
mapping. Make sure the mapping exist exists under folder 'mapping' and contains
the exact query name and root name. Available mappings:
[QueryRoot{queryName='SM Business Application Population 2.0',
rootName='bizservice'}] : operation Data Access Adapter Query: Retrieve Changed
Data
    at
com.mercury.topaz.cmdb.shared.manage.operation.impl.AbstractCommonOperation.exec
ute(AbstractCommonOperation.java:160)
    at
com.hp.ucmdb.dataAccess.manager.DataAccessAdapterManagerProbeImpl.executeOperati
on(DataAccessAdapterManagerProbeImpl.java:50)
    at
com.hp.ucmdb.discovery.probe.agents.probemgr.adapters.DataAccessAdaptersFacade.i
nvokeOperation(DataAccessAdaptersFacade.java:406)
    at
com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.runChangesOnPo
pulateChangesAdapter(AdapterService.java:1262)
    at
```

```
com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.runQueriesOnPopulateChangesAdapter(AdapterService.java:1102)
    at com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.runQueries(AdapterService.java:354)
    at
com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.runDiscovery(AdapterService.java:198)
    at com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.discover(AdapterService.java:149)
    at
com.hp.ucmdb.discovery.probe.agents.probemgr.taskexecutor.JobExecutor.launchTask(JobExecutor.java:1188)
    at
com.hp.ucmdb.discovery.probe.agents.probemgr.taskexecutor.JobExecutor$JobExecutorWorker.launch(JobExecutor.java:945)
    at
com.hp.ucmdb.discovery.probe.agents.probemgr.taskexecutor.JobExecutor$JobExecutorWorker.executeTask(JobExecutor.java:867)
    at
com.hp.ucmdb.discovery.probe.agents.probemgr.taskexecutor.JobExecutor$JobExecutorWorker.run(JobExecutor.java:728)
```

解決策

“has no matching mapping” というテキストを検索し、見つからないマッピングファイル名を確認して、アダプタパッケージにマッピングファイルを追加します。

連携の問題のトラブルシューティング

連携エラーが発生した場合は、連携ログファイルでエラーメッセージを確認し、原因を特定して問題を解決できます。

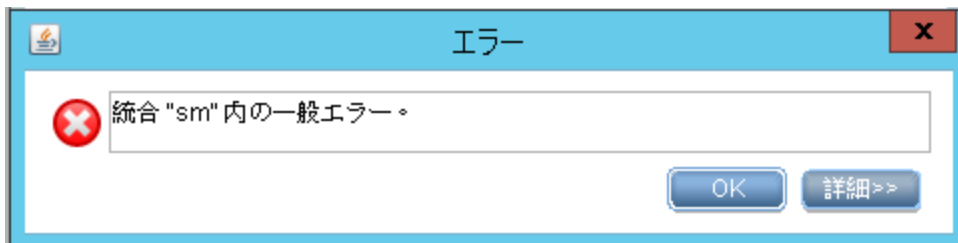
詳細については、以下を参照してください。

[「失敗した連携要求のエラーメッセージを確認する方法」\(190ページ\)](#)

[「連携の一般的なエラーメッセージと解決策」\(191ページ\)](#)

失敗した連携要求のエラーメッセージを確認する方法

連携クエリが失敗すると、UCMDB Studio に次のようなポップアップウィンドウが表示されます。



注: プッシュ、ポピュレーション、および連携の問題のトラブルシューティングを行うには、**fcmdb.properties** ファイルおよび **fcmdb.push.properties** ファイルで開発アダプタのログレベルを DEBUG または TRACE に設定する必要があります。これらのファイルは、Data Flow Probe または統合サービスのインストールフォルダの \conf\log フォルダにあります。さらに、**fcmdb.push.all.log** ファイルでプッシュ、ポピュレーション、および連携のログ情報を表示できます。このファイルは、Data Flow Probe または統合サービスのインストールフォルダの \runtime\log フォルダにあります。Data Flow Probe または統合サービスは、UCMDB サーバと同じホストにインストールすることも別のホストにインストールすることもできます。

開発アダプタのログレベルを設定し、失敗した連携クエリの詳細なエラーメッセージを表示するには、「[プッシュのログファイルを確認する方法](#)」(179ページ)で説明されている手順に従います。

連携の一般的なエラーメッセージと解決策

次の項では、連携中に発生する可能性がある一般的なエラーとそれらの解決策について説明します。

[「smFedConf.xml での連携 CI タイプの構成が正しくない」](#)(191ページ)

[「連携 TQL クエリのマッピングファイルが well formed \(整形形式\) ではない」](#)(193ページ)

smFedConf.xml での連携 CI タイプの構成が正しくない

エラーメッセージ

smFedConf.xml で CI タイプを正しく構成していない場合、UCMDB はこのタイプの CI を連携できません。関連する連携クエリを実行すると、UCMDB Studio にエラーウィンドウがポップアップ表示され、さらに、次のようなエラーメッセージが連携ログファイルに記録されます。

```
2014-11-20 13:22:52,055 [AdHoc:AD_HOC_TASK_PATTERN_ID-51-1416460969254] ERROR - sm
>> Failed to retrieve or parsing Root/parent CI message from SM side, exit from
federation!
java.lang.NullPointerException
    at
com.mercury.topaz.fcmdb.adapters.serviceDeskAdapter.federation.SmGenericFederato
r.processRootCiQueryByPage(SmGenericFederator.java:461)
    at
com.mercury.topaz.fcmdb.adapters.serviceDeskAdapter.federation.SmGenericFederato
r.generateRootRtnResult(SmGenericFederator.java:162)
    at
```

```
com.mercury.topaz.fcldb.adapters.serviceDeskAdapter.federation.SmGenericFederato
r.generateFederationRtnResult(SmGenericFederator.java:131)
    at
com.mercury.topaz.fcldb.adapters.serviceDeskAdapter.federation.SmGenericFederato
r.generateFederationOutput(SmGenericFederator.java:120)
    at
com.mercury.topaz.fcldb.adapters.serviceDeskAdapter.federation.SmGenericFederato
r.getNextResultChunk(SmGenericFederator.java:578)
    at com.hp.ucmdb.adapter.smpush.ServiceManagerGenericAdapter.populate
(ServiceManagerGenericAdapter.java:363)
    at com.hp.ucmdb.adapters.GenericAdapter.getDataResult(GenericAdapter.java:740)
    at
com.hp.ucmdb.discovery.probe.processor.FederationTopologyGetDataResultProbeReque
stProcessor.processRequest
(FederationTopologyGetDataResultProbeRequestProcessor.java:40)
    at
com.hp.ucmdb.discovery.probe.processor.FederationTopologyGetDataResultProbeReque
stProcessor.processRequest
(FederationTopologyGetDataResultProbeRequestProcessor.java:26)
    at com.hp.ucmdb.discovery.probe.processor.AbstractProbeProcessor.process
(AbstractProbeProcessor.java:56)
    at com.hp.ucmdb.discovery.probe.processor.AbstractProbeProcessor.process
(AbstractProbeProcessor.java:19)
    at
com.hp.ucmdb.discovery.probe.agents.probemgr.adhoctasks.AdHocProbeRequestOperati
on.performAction(AdHocProbeRequestOperation.java:63)
    at
com.hp.ucmdb.discovery.probe.agents.probemgr.taskdispatcher.AdHocTaskDispatcher.
dispatchTask(AdHocTaskDispatcher.java:70)
    at sun.reflect.GeneratedMethodAccessor75.invoke(Unknown Source)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke
(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:601)
    at com.sun.jmx.mbeanserver.StandardMBeanIntrospector.invokeM2
(StandardMBeanIntrospector.java:111)
    at com.sun.jmx.mbeanserver.StandardMBeanIntrospector.invokeM2
(StandardMBeanIntrospector.java:45)
    at com.sun.jmx.mbeanserver.MBeanIntrospector.invokeM
(MBeanIntrospector.java:235)
    at com.sun.jmx.mbeanserver.PerInterface.invoke(PerInterface.java:138)
    at com.sun.jmx.mbeanserver.MBeanSupport.invoke(MBeanSupport.java:252)
    at javax.management.StandardMBean.invoke(StandardMBean.java:405)
    at com.sun.jmx.interceptor.DefaultMBeanServerInterceptor.invoke
(DefaultMBeanServerInterceptor.java:819)
    at com.sun.jmx.mbeanserver.JmxMBeanServer.invoke(JmxMBeanServer.java:792)
    at javax.management.MBeanServerInvocationHandler.invoke
(MBeanServerInvocationHandler.java:305)
    at org.springframework.jmx.access.MBeanClientInterceptor.doInvoke
(MBeanClientInterceptor.java:405)
```



```
    at org.springframework.jmx.access.MBeanClientInterceptor.invoke
(MBeanClientInterceptor.java:353)
    at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed
(ReflectiveMethodInvocation.java:172)
    at org.springframework.aop.framework.JdkDynamicAopProxy.invoke
(JdkDynamicAopProxy.java:202)
    at com.sun.proxy.$Proxy51.dispatchTask(Unknown Source)
    at
com.hp.ucmdb.discovery.probe.agents.probegw.managementtasks.adhocktasks.AdhocThre
ad.run(AdhocThread.java:54)
    at org.eclipse.jetty.util.thread.QueuedThreadPool.runJob
(QueuedThreadPool.java:599)
    at org.eclipse.jetty.util.thread.QueuedThreadPool$3.run
(QueuedThreadPool.java:534)
    at java.lang.Thread.run(Thread.java:722)
```

解決策

smFedConf.xml ファイル (ServiceManagerEnhancedAdapter9-x アダプタの構成ファイルフォルダにあります) を開き、CI タイプの構成が正しいことを確認します。

連携 TQL クエリのマッピングファイルが well formed (整形式) ではない

エラーメッセージ

連携 TQL クエリのマッピングファイルが well formed (整形式) ではない場合があります。たとえば、次の例の連携マッピングファイル内の target_mapping_name “priority” は、誤った Groovy 関数名 (SMFederationFunctions.getEnumValue1) を指定しています。

```
<?xml version="1.0" encoding="UTF-8"?>
<integration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../mappings_schema.xsd">
  <info>
    <source name="SM"          version="9.40"    vendor="HP"/>
    <target name="UCMDB"      version="10.20"   vendor="HP"/>
  </info>
  <import>
    <scriptFile path="mappings.scripts.SMFederationFunctions"/>
  </import>
  <target_entities>
    <source_instance query-name="SM Incident 2.0" root-element-
name="ucmdbIncident">
      <target_entity name="incident">
        <target_mapping name="reference_number"    datatype="STRING"
value="ucmdbIncident['IncidentID']"/>
        <target_mapping name="name"                datatype="STRING"
value="ucmdbIncident['BriefDescription']"/>
        <target_mapping name="priority"           datatype="STRING"
```

```
value="SMFederationFunctions.getEnumValue1(ucmdbIncident
['PriorityCode'],'Priority')"/>
    <target_mapping name="incident_status"          datatype="STRING"
value="SMFederationFunctions.firstLetterToLowerAndReplaceSpaceWithUnderscore
(ucmdbIncident['IMTicketStatus'])"/>
    <target_mapping name="category"                datatype="STRING"
value="SMFederationFunctions.replaceSpaceWithUnderscore(ucmdbIncident
['Category'])"/>
    <target_mapping name="closed_time"             datatype="DATE"
value="SMFederationFunctions.convertDate(ucmdbIncident['ClosedTime'])"/>
    <target_mapping name="create_time"            datatype="DATE"
value="SMFederationFunctions.convertDate(ucmdbIncident['OpenTime'])"/>
    <target_mapping name="last_modified_time"     datatype="DATE"
value="SMFederationFunctions.convertDate(ucmdbIncident['UpdatedTime'])"/>
    <target_mapping name="impact_scope"          datatype="STRING"
value="SMFederationFunctions.getEnumValue(ucmdbIncident
['ImpactScope'],'ImpactScope')"/>
    <target_mapping name="urgency"                datatype="STRING"
value="SMFederationFunctions.getEnumValue(ucmdbIncident
['Urgency'],'Urgency')"/>
    </target_entity>
  </source_instance>
</target_entities>
</integration>
```

インシデントを連携しようとしたときに、Service Manager からレコードが取得されません。
fcmdb.push.all.log ファイルを確認すると、次のような連携クエリに関する詳細なエラーメッセージが見つかります。

```
2014-11-20 14:09:58,670 [AdHoc:AD_HOC_TASK_PATTERN_ID-66-1416463796366] ERROR - >>
Failed executing value [SMFederationFunctions.getEnumValue1(ucmdbIncident
['PriorityCode'],'Priority')] of mapping <target_mapping name="priority">
<target_ci_type name="incident"> , Root cmdbId [null]
groovy.lang.MissingMethodException: No signature of method: static
mappings.scripts.SMFederationFunctions.getEnumValue1() is applicable for
argument types: (java.lang.String, java.lang.String) values: [1, Priority]
Possible solutions: getEnumValue(java.lang.String, java.lang.String)
    at groovy.lang.MetaClassImpl.invokeStaticMissingMethod(MetaClassImpl.java:1359)
    at groovy.lang.MetaClassImpl.invokeStaticMethod(MetaClassImpl.java:1345)
    at org.codehaus.groovy.runtime.callsite.StaticMetaClassSite.call
(StaticMetaClassSite.java:50)
    at org.codehaus.groovy.runtime.callsite.CallSiteArray.defaultCall
(CallSiteArray.java:42)
    at org.codehaus.groovy.runtime.callsite.AbstractCallSite.call
(AbstractCallSite.java:108)
    at org.codehaus.groovy.runtime.callsite.AbstractCallSite.call
(AbstractCallSite.java:120)
    at Mapping_4a8761e7adc009e8a7d268c2f1349ab4.run(Mapping_
4a8761e7adc009e8a7d268c2f1349ab4.groovy:1)
    at
```

```
com.hp.ucmdb.adapters.instance.mapping.AbstractResultTreeNodeMapper.calculatePro
perties(AbstractResultTreeNodeMapper.java:231)
    at
com.hp.ucmdb.adapters.instance.mapping.AbstractResultTreeNodeMapper.processTarge
tEntity(AbstractResultTreeNodeMapper.java:301)
    at com.hp.ucmdb.adapters.instance.mapping.RtnToRtnMapper.processTargetEntities
(RtnToRtnMapper.java:214)
    at
com.hp.ucmdb.adapters.instance.mapping.RtnToRtnMapper.processSourceInstanceWrapp
er(RtnToRtnMapper.java:101)
    at com.hp.ucmdb.adapters.instance.mapping.RtnToRtnMapper.buildResultTreeNode
(RtnToRtnMapper.java:76)
    at
com.hp.ucmdb.adapters.GenericAdapter$PopulationFederationChunkProcessor.invoke
(GenericAdapter.java:1213)
    at com.hp.ucmdb.adapters.GenericAdapter.getDataResult(GenericAdapter.java:743)
    at
com.hp.ucmdb.discovery.probe.processor.FederationTopologyGetDataResultProbeReque
stProcessor.processRequest
(FederationTopologyGetDataResultProbeRequestProcessor.java:40)
    at
com.hp.ucmdb.discovery.probe.processor.FederationTopologyGetDataResultProbeReque
stProcessor.processRequest
(FederationTopologyGetDataResultProbeRequestProcessor.java:26)
    at com.hp.ucmdb.discovery.probe.processor.AbstractProbeProcessor.process
(AbstractProbeProcessor.java:56)
    at com.hp.ucmdb.discovery.probe.processor.AbstractProbeProcessor.process
(AbstractProbeProcessor.java:19)
    at
com.hp.ucmdb.discovery.probe.agents.probemgr.adhoctasks.AdHocProbeRequestOperati
on.performAction(AdHocProbeRequestOperation.java:63)
    at
com.hp.ucmdb.discovery.probe.agents.probemgr.taskdispatcher.AdHocTaskDispatcher.
dispatchTask(AdHocTaskDispatcher.java:70)
    at sun.reflect.GeneratedMethodAccessor75.invoke(Unknown Source)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke
(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:601)
    at com.sun.jmx.mbeanserver.StandardMBeanIntrospector.invokeM2
(StandardMBeanIntrospector.java:111)
    at com.sun.jmx.mbeanserver.StandardMBeanIntrospector.invokeM2
(StandardMBeanIntrospector.java:45)
    at com.sun.jmx.mbeanserver.MBeanIntrospector.invokeM
(MBeanIntrospector.java:235)
    at com.sun.jmx.mbeanserver.PerInterface.invoke(PerInterface.java:138)
    at com.sun.jmx.mbeanserver.MBeanSupport.invoke(MBeanSupport.java:252)
    at javax.management.StandardMBean.invoke(StandardMBean.java:405)
    at com.sun.jmx.interceptor.DefaultMBeanServerInterceptor.invoke
(DefaultMBeanServerInterceptor.java:819)
```

```
        at com.sun.jmx.mbeanserver.JmxMBeanServer.invoke(JmxMBeanServer.java:792)
        at javax.management.MBeanServerInvocationHandler.invoke
(MBeanServerInvocationHandler.java:305)
        at org.springframework.jmx.access.MBeanClientInterceptor.doInvoke
(MBeanClientInterceptor.java:405)
        at org.springframework.jmx.access.MBeanClientInterceptor.invoke
(MBeanClientInterceptor.java:353)
        at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed
(ReflectiveMethodInvocation.java:172)
        at org.springframework.aop.framework.JdkDynamicAopProxy.invoke
(JdkDynamicAopProxy.java:202)
        at com.sun.proxy.$Proxy51.dispatchTask(Unknown Source)
        at
com.hp.ucmdb.discovery.probe.agents.probegw.managementtasks.adhoctasks.AdhocThre
ad.run(AdhocThread.java:54)
        at org.eclipse.jetty.util.thread.QueuedThreadPool.runJob
(QueuedThreadPool.java:599)
        at org.eclipse.jetty.util.thread.QueuedThreadPool$3.run
(QueuedThreadPool.java:534)
        at java.lang.Thread.run(Thread.java:722)
```

解決策

fcmdb.push.all.log ファイルで "Exception" という単語を探し、誤ったメソッド名を見つけて、対応する連携マッピングファイルでそれを修正します。

ドキュメントのフィードバックを送信

本ドキュメントについてのご意見、ご感想については、電子メールでドキュメント制作チームまでご連絡ください。このシステムで電子メールクライアントが設定されていれば、このリンクをクリックすることで、以下の情報が件名に記入された電子メールウィンドウが開きます。

Feedback on Universal CMDB 統合ガイド (Service Manager 拡張汎用アダプタ使用) (Service Manager Service Manager 9.40、Universal CMDB 10.20 以降)

本文にご意見、ご感想を記入の上、[送信]をクリックしてください。

電子メールクライアントが利用できない場合は、上記の情報をコピーしてWebメールクライアントの新規メッセージに貼り付け、ovdoc-itsm@hp.com宛にお送りください。

お客様からのご意見、ご感想をお待ちしています。

