Agile Product Lifecycle Management

SDK 開発者ガイド

v9.2.2.3

部品番号: E06159-01
2008 年 5 月

著作権および商標について

目次

はじめに

Oracle|Agile マニュアル セットには Adobe® Acrobat™ PDF ファイルが含まれます。Oracle Technology Network (OTN) Web サイト (http://www.oracle.com/technology/documentation/agile.html) には、Oracle|Agile PLM の最新版の PDF ファイルがあります。この Web サイトのマニュアルは、その場で表示することもダウンロードして使用することもできます。また、使用しているネットワーク上の Oracle|Agile マニュアル フォルダに Oracle|Agile マニュアル (PDF) ファイルが格納されている場合もあります。詳細は、Agile 管理者にお問い合わせください。

注意 PDF ファイルを表示するには、Adobe Acrobat Reader™ のバージョン 7.0 以降 (無料) を使用する必要があります。このプログラムは、Adobe 社の Web サイト (http://www.adobe.com) からダウンロードできます。

Oracle Technology Network (OTN) Web サイト (http://www.oracle.com/technology/documentation/agile.html) は、Agile Web クライアントと Agile Java クライアントのいずれの場合も、[ヘルプ] > [マニュアル] の順に選択してアクセスできます。さらに疑問点がある場合やサポートが必要な場合は、サポート (http://www.oracle.com/agile/support.html) にお問い合わせください。

注意 Oracle|Agile PLM マニュアルに関する問題について Agile サポートにお問い合わせいただく前に、タイトル ページにある完全な部品番号をご準備ください。

Oracle サポート サービスへの TTY アクセス

アメリカ国内では、Oracle サポート サービスへ 24 時間年中無休でテキスト電話 (TTY) アクセスが提供されています。TTY サポートについては、(800) 446-2398 にお電話ください。アメリカ国外からの場合は、+1-407-458-2479 にお電話ください。

ドキュメントのアクセシビリティについて

オラクル社は、障害のあるお客様にもオラクル社の製品、サービスおよびサポート ドキュメントを簡単にご利用いただけることを目標としています。オラクル社のドキュメントには、ユーザーが障害支援技術を使用して情報を利用できる機能が組み込まれています。HTML 形式のドキュメントで用意されており、障害のあるお客様が簡単にアクセスできるようにマークアップされています。標準規格は改善されつつあります。オラクル社はドキュメントをすべてのお客様がご利用できるように、市場をリードする他の技術ベンダーと積極的に連携して技術的な問題に対応しています。オラクル社のアクセシビリティについての詳細情報は、Oracle Accessibility Program の Web サイト http://www.oracle.com/accessibility/ を参照してください。

Readme

Oracle|Agile PLM の最新情報は、すべて Oracle Technology Network (OTN) Web サイト (http://www.oracle.com/technology/documentation/agile.html) にある Readme ファイルに記載されています。

Agile トレーニング支援

Agile トレーニングの講義内容詳細については、Oracle University Web ページ (http://www.oracle.com/education/chooser/selectcountry_new.html) にアクセスしてください。

ドキュメント内のサンプル コードのアクセシビリティについて

スクリーン リーダーは、ドキュメント内のサンプル コードを正確に読めない場合があります。コード表記規則では閉じ括弧だけを行に記述する必要があります。しかしスクリーン リーダーは括弧だけの行を読まない場合があります。

外部 Web サイトのドキュメントのアクセシビリティについて

このドキュメントにはオラクル社およびその関連会社が所有または管理しない Web サイトへのリンクが含まれている場合があります。オラクル社およびその関連会社は、それらの Web サイトのアクセシビリティに関しての評価や言及は行っておりません。

第 1 章

導入

扱うトピックは次のとおりです。

Agile SDK とは

Agile SDK はソフトウェア開発キットで、一連のツール、アプリケーション プログラミング インターフェース (API)、サンプル アプリケーションおよびマニュアルが組み込まれています。この SDK は、Agile アプリケーション サーバの機能にアクセスするカスタム アプリケーションを構築するために使用します。Agile SDK を使用すると、Agile PLM システムに対してタスクを自動的に実行するプログラムを作成できます。

Agile SDK によって、次の操作が可能になります。

- Agile PLM システムと ERP アプリケーションまたは他のカスタム アプリケーションとの統合。
- 製品データを処理するアプリケーションの開発。
- Agile アプリケーション サーバに対するバッチ操作の実行。

Agile SDK には、次の 3 つのモジュールがあります。

- Agile API - Agile ビジネス オブジェクトを公開するインターフェースを備えた Java API。Agile API は、追加の Agile PLM クライアントを作成するために使用したり、WSX または PX を使用して開発した拡張機能の一部として使用します。
- プロセス拡張 (PX) - Agile PLM の顧客が Agile PLM クライアントの機能を拡張できるようにするフレームワーク。拡張するには、外部レポート、ユーザーおよびワークフロー主導型のカスタム アクション、カスタム ツールおよびカスタム自動採番ソースを追加します。
- Web サービス拡張 (WSX) - Agile PLM の顧客が、Web サービスを使用して Agile PLM サーバの機能を拡張し、顧客固有のソリューションを公開できるようにするフレームワーク。

SDK のコンポーネント

Agile PLM は Agile サーバ上で実行され、管理者とユーザーとのすべての相互作用には、クライアント側のコンポーネントとサーバ側のコンポーネントが組み込まれた SDK が使用されます。

クライアント側のコンポーネント

Agile SDK に組み込まれているクライアント側のコンポーネントは、次のとおりです。

- Agile SDK 開発者ガイド (このマニュアル)
- Agile API HTML リファレンス
- サンプル アプリケーション
- Agile API ライブラリ (`AgileAPI.jar`)
- プロセス拡張 API ライブラリ (`pxapi.jar`)
- Apache Axis ライブラリ (`axis.jar`)

サーバ側のコンポーネント

Agile アプリケーション サーバに組み込まれている Agile SDK のサーバ側のコンポーネントは、次のとおりです。

- Agile API 実装クラス
- プロセス拡張フレームワーク
- Web サービス拡張フレームワーク

アーキテクチャ

Agile SDK を使用すると、Agile アプリケーション サーバに接続するための様々なタイプのプログラムを簡単に開発できます。Agile API のみを使用している場合は、サーバに直接接続するプログラムを開発できます。WSX を使用して Web サービスの拡張機能を開発する場合は、Agile アプリケーション サーバのコンテナ内に Web サービスを配置できます。WSX に使用する Web サーバは、企業の非武装地帯 (DMZ) コンピューティング ネットワークや防衛ネットワークの内部または外部に配置できます。Agile PLM クライアントでカスタム アクションを開始する場合は、サーバに配置されているプログラムを実行するか、外部のリソースまたは URL に接続します。Agile API は、WSX および PX の拡張でも使用できます。これは、Agile SDK のすべての開発プロジェクトで使用できるツールです。拡張機能は、他社が提供する API を使用して開発することもできます。

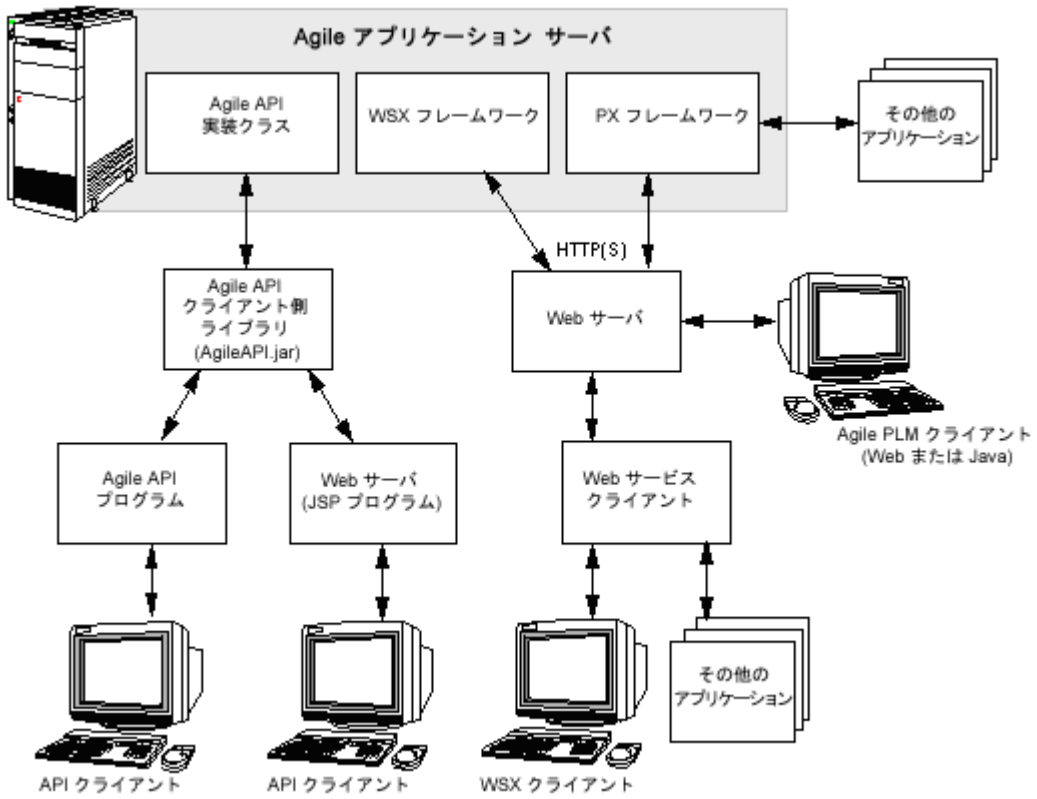
Agile XML (別名 aXML)

Agile XML 形式は、Agile のビジネス スキーマの XML 表現です。aXML には、Agile で管理される製品コンテンツがすべて含まれます。製品コンテンツには、アイテム、変更の詳細、製造元情報、コスト データ、図面および他のファイルが含まれています。aXML は Agile のすべての製品におけるスキーマ要素の表現であるため、Agile のビジネス スキーマとともに長期にわたって展開される予定です。

最新の aXML スキーマについては、次の Web ページを参照してください。
http://support.agile.com/misc/axml/2007/03/aXML.xsd

注意 Agile 9.2.1 から Agile 9.2.2 への移行で、aXML スキーマに変更がありました。詳細は、Agile 9.2.2 の Readme ファイルを参照してください。

図 1: Agile SDK のアーキテクチャ



注意 Agile API プログラムでは、保護されていない手段を使用して Agile アプリケーション サーバに接続します。したがって、Agile API プログラムは、企業のファイヤウォール内のみで実行する必要があります。一方、Web サービス クライアントは、標準的な HTTP(S) テクノロジを使用して企業のファイヤウォールを介してサーバに接続できます。

このリリースでの新機能と拡張機能

このリリースでの拡張機能と変更の概要は、次のとおりです。

- プログラム テンプレートを使用したプログラムの作成と所有権の変更 - `saveAs()` API 呼び出しを使用してテンプレートからプログラムを作成するときは、プログラムの所有者と新しいプログラムの子の所有者を変更できます。195 ページの「プログラムの作成および所有権の変更」を参照してください。
- コード例の変更 - マニュアルのコード サンプルが若干変更されました。これらの変更については、107 ページの「アイテムのリビジョンの取得および設定」のコード サンプルを参照してください。

新機能および Agile SDK リリース 9.2.2.2 の既存の機能に対する拡張の概要は、次のとおりです。

重要 AgileAPI.jar リリース 9.2.2.2/9.2.2.3 の PG&C 定数と関係テーブル機能は、前のバージョンの AgileAPI.jar と互換性がありません。

- Product Collaboration (PC) のユーザー二重識別サインオフ - PC での現在の電子承認プロセスを拡張するために、合計 7 個の新しい API が用意されました。この機能は、サインオフ プロセスで二重ユーザー識別が必要な FDA 規制会社やその他の企業の特定の要件をサポートします。166 ページの「[サインオフ ユーザー二重識別タイプ] プリファレンスの設定」を参照してください。
- ソーシング プロジェクトに対する PCM SDK の新機能と拡張機能 - 新しい 2 つの API と既存の API に対する拡張機能の組み合わせによって、SDK を使用してソーシング プロセスの開始から RFQ タスクまでの全体を完了できます。詳細は、次を参照してください。
  - 227 ページの「ソーシング プロジェクトでのアイテムの数量の設定」
  - 228 ページの「ソーシング プロジェクトでの数量ロールアップの実行」
  - 228 ページの「ソーシング プロジェクトでのパートナーの設定」
  - 230 ページの「ソーシング プロジェクトでのアイテムの目標価格の変更」
  - 230 ページの「ソーシング プロジェクトでのアイテムの最良回答の設定」
- PG&C の [適合性判定値] フィールドの値を設定する新規 API - 新規 API を使用すると、SDK を使用してこのフィールドの値を設定できます。詳細および例は、271 ページの「[適合性判定値] フィールドの値の設定」を参照してください。
- `ITable.getName()` の問題を解決する新規 API - この API をネストされた価格テーブルに対して使用すると、以前はヌル値が返されました。この問題は、価格テーブルの名前を PriceDetails としてハードコード化することで、リリース 9.2.2.2 では解決しています。

システム要件

Agile SDK システムの要件は、『Oracle|Agile PLM Capacity Planning and Deployment Guide』を参照してください。

Java の要件

Agile API は、アプリケーション サーバがサポートする Java のバージョンに依存しています。問題を回避するために、Agile API クライアントでは、接続先アプリケーション サーバが使用している Java バージョンと同一のバージョンを使用する必要があります。相互運用性と 2007 年以降の夏時間に準拠するために、Oracle Application Server 10g および BEA WebLogic Server 8.1 の両方で、Sun 社の Java Runtime Environment (JRE) 1.4.2_12 を使用する必要があります。

次の表に、Agile PLM がサポートする様々なアプリケーション サーバで Agile API クライアントを使用するために、推奨される Java Runtime Environment (JRE) を示します。

アプリケーション サーバ	オペレーティング システム	Agile API クライアントに必要な Java のバージョン
Oracle Application Server 10g	Windows 2003	Sun JRE 1.4.2
BEA WebLogic Server	Windows 2003	Sun JRE 1.4.2
IBM WebSphere 5.1	Windows/Solaris	IBM JDK 1.4.1

Agile SDK インストール フォルダ

コンピュータ上の Agile SDK ファイルには、次のフォルダ構造があります。

`lib - ¥agile_home¥integration¥sdk¥lib` フォルダには、次のライブラリが含まれます。

重要 axis.jar ファイルと AgileAPI.jar ファイルは同じクラスパスに格納しないでください。SDK クラスパスは、この設定をサポートしていないため、SDK が正しく機能しません。

- `AgileAPI.jar` – Agile API ライブラリ。このライブラリには、Agile API のクラスとインターフェースが含まれています。
- `pxapi.jar` – PX API ライブラリ。このライブラリには、カスタム自動採番ソースおよびカスタム アクションの開発に使用するインターフェースが含まれています。
- `axis.jar` – Apache Axis ライブラリの Agile 対応バージョン。このライブラリは Web サービス クライアントに必要です。

Agile PLM システムの確認

Agile PLM システムで Agile SDK クライアントを実行するには、その前に、そのシステムが正しく設定され、稼働していることを確認してください。特に、アプリケーション サーバの HTTP ポートが正しく設定されていることを確認してください。詳細は、『Agile PLM Installation Guide』を参照してください。

Agile PLM ビジネス オブジェクト

企業ソフトウェア システムでは、ビジネス オブジェクトを使用して企業のデータを管理します。次の表に、Agile PLM ビジネス オブジェクトとそれに関連する Agile API インターフェースを示します。

オブジェクト	関連する Agile API インターフェース
変更	IChange
顧客	ICustomer
デクラレーション	IDeclaration
ディスカッション	IDiscussion
ファイル フォルダ	IFileFolder
アイテム	IItem
製造元部品	IManufacturerPart
製造元	IManufacturer
パッケージ	IPackage
部品グループ (部品分類または部品ファミリ)	ICommodity
価格	IPrice
製品サービス依頼	IServiceRequest
プログラム	IProgram
ソーシング プロジェクト	IProject
品質変更依頼	IQualityChangeRequest
見積依頼 (RFQ)	IRequestForQuote
見積依頼回答	ISupplierResponse*
拠点	IManufacturingSite
含有基準	ISpecification
サブスタンス	ISubstance
サプライヤ	ISupplier

オブジェクト	関連する Agile API インターフェース
転送	ITransferOrder
ユーザー グループ	IUserGroup
ユーザー	IUser

* 現在のソフトウェア リリースでは、API インターフェースはサポートされていません。

表示できるビジネス オブジェクトや、これらのオブジェクトに対して実行できるアクションは、Agile アプリケーション サーバにインストールされているサーバ コンポーネントや、ライセンス、役割および割り当てられている権限によって決まります。権限のレベルは、フィールドごとに異なる場合があります。Agile PLM 管理者は、ユーザーおよびユーザー グループに加え、管理ノードや Agile PLM クラスなどの管理オブジェクトを使用します。

すべての Agile PLM ビジネス オブジェクトが Agile API で公開されているわけではありません。たとえば、レポート オブジェクトは Agile API 経由ではアクセスできません。

Licensing

Agile Software Corporation (Agile) requires any company or individual writing code to the Agile SDK to legally obtain an Agile SDK license. The Agile SDK license grants the licensee the right to use the Agile SDK in a design environment and to freely distribute the Agile SDK libraries (such as AgileAPI.jar) with any application written by the licensee that makes calls to the Agile SDK. The Agile SDK license prohibits the distribution of the Agile SDK documentation, sample code, and source code to any other party that has not legally obtained an Agile SDK license. It also explicitly prohibits the development of competing applications.

第 2 章

Agile API の開始

扱うトピックは次のとおりです。

Agile API の概要

この章では、Agile API が提供する機能の概要を説明します。説明する内容は、次のとおりです。

- Agile API のクラスとインターフェースのタイプ
- クラスのロード方法
- Agile API のスレッドセーフの仕組み
- Agile API アプリケーションのパッケージ方法
- サンプル プログラムの場所

Agile API のクラスとインターフェースのタイプ

Agile API (AgileAPI.jar ライブラリ) には、多様なクラスとインターフェースが含まれています。これらの内容は、関連するファイルを次の各グループに分類することで、より理解しやすくなります。

- 集約インターフェース - これらのインターフェースは、特定のオブジェクト タイプの関連する機能インターフェースを集約します。たとえば、`IItem` インターフェースは、`IDataObject`、`IRevisioned`、`IManufacturingSiteSelectable`、`IAttachmentContainer`、`IHistoryManager` および `IReferenced` を拡張します。ほとんどの SDK 機能は、これらのインターフェースに含まれます。これらのインターフェースは、Agile API の基礎となる実装クラス (非公開) によって実装されます。
- 機能ユニット インターフェース - これらのインターフェースは、他のインターフェースに拡張される機能ユニットを保持します。たとえば、`IAttachmentContainer` は、任意のオブジェクトの添付ファイル テーブルにアクセスする便利な手段を提供します。つまり、この `IAttachmentContainer` インターフェースは、`IChange`、`IItem` など、他の複数のインターフェースによって拡張されます。機能ユニットの役目を果たすもう 1 つのクラスは `IRoutable` です。このクラスは、別の Agile PLM ユーザーに送信可能なオブジェクトに対してメソッドを提供します。`IRoutable` は、`IChange`、`IPackage` および `ITransferOrder` のすべてによって拡張されます。

- メタデータ インターフェース - クラスのこのグループは、Agile アプリケーション サーバのメタデータ (およびメタメタデータ) を定義します。メタデータは、単に他のデータを説明するデータです。メタデータ インターフェースには、IAgileClass、INode、IRoutableDesc、ITableDesc、IWorkflow などのクラスが含まれます。
- ファクトリ クラス - AgileSessionFactory は、セッション (IAgileSession) を作成してトランザクション管理にアクセスするために使用するファクトリ クラスです。IAgileSession は、他のオブジェクトをインスタンス化できるファクトリ オブジェクトでもあります。多くの Agile API オブジェクトも同様に、テーブルまたは他の参照オブジェクトに対するファクトリ オブジェクトです。表もまた各行に対するファクトリです。
- 例外クラス - 例外クラスに該当するのは APIException のみです。
- 定数 - これらのクラスには、属性、テーブル、クラスなどの ID が含まれます。定数のみが格納されているクラスはすべて「Constants」で終了するクラス名 (ChangeConstants、ItemConstants、UserConstants など) が付いています。

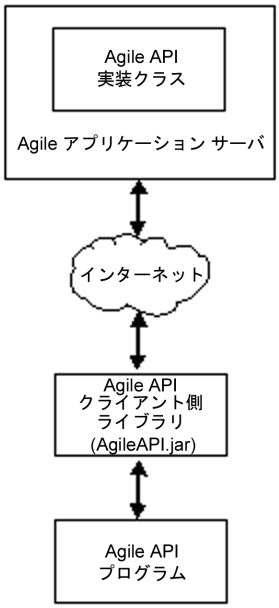
ネットワーク クラスのロード

Agile API には、2 つのメイン ソフトウェア コンポーネントがあります。

- クライアント側のライブラリ (AgileAPI.jar)
- サーバ側の実装クラス

サーバ側の実装クラスは、Agile アプリケーション サーバの各インスタンスとともに自動的にインストールされます。

Agile API のクライアント側のライブラリは、ほぼ全体が複数のインターフェースで構成されており、基本的にはクラス ローダです。Agile API プログラムを実行すると、Agile アプリケーション サーバに接続され、必要な実装クラスが自動的にダウンロードされます。たとえば、プログラムで IItem のメソッドが使用される場合は、実行時に IItem の実装がダウンロードされます。



ネットワーク クラス ロードには、サーバからクライアント実装クラスを自動的にダウンロードしてアップデートする機能も含めて、多くの利点があります。サーバからダウンロードした Agile API クラスは、ローカル ディスクに自動的にキャッシュされます。Agile API プログラムで特定のクラスをロードする必要がある場合は、ネットワークからクラスを再度ダウンロードするのではなく、キャッシュからそのクラスを取得します。キャッシュによって、クラスのロードが迅速化され、ネットワークの負荷が軽減されます。

ネットワーク クラス ローダによって、キャッシュが古い (つまり、サーバ内のクラスよりもキャッシュ内のクラスが古い) ことが検出された場合は、そのキャッシュは無効にされ、必要なクラスがサーバから再ロードされます。これによって、企業全体でアプリケーションを再配置せずに最新の実装クラスが使用されるように Agile SDK クライアントを更新できます。

シングルスレッド アプリケーションとマルチスレッド アプリケーションの対比

Agile API はスレッド互換であることが認定されています。Agile API は、シングルスレッドとマルチスレッドのいずれのアプリケーション開発にも使用できます。Agile API 呼び出しは、各メソッド呼び出し (または一連のメソッド呼び出し) を外部との同期で囲むことによって、安全にかつ同時に使用できます。

Agile API プログラムのパッケージ化

Agile API を呼び出すプログラムを開発した後は、そのファイルをパッケージ化してインストールできるようにする必要があります。多くの開発環境には、アプリケーションをパッケージ化して配置するためのツールが含まれています。

プログラムは手動でパッケージ化することもできます。この方法を選択した場合は、プロジェクトの依存関係を把握する必要があります。多くの開発環境には、依存関係ファイルを生成するためのツールも含まれています。依存関係ファイルには、プログラムのプロジェクト ファイルとともに配布する必要があるランタイム コンポーネントがリストされています。

Agile API Files You Are Allowed to Distribute

You can freely distribute any Java applications or applets that you create that make calls to the Agile API. You can include the Agile API library, `AgileAPI.jar`, when you package your application's files.

Your development environment might require you to distribute other class files or libraries with your program. Check the documentation for your development environment to see which runtime files should be distributed with your program. Consult the manufacturer's license agreement for each of the files you plan to distribute to determine whether you have the right to distribute the file with your application.

Agile API Files You Are Not Allowed to Distribute

Agile requires that any company or individual writing code to the Agile API must obtain an Agile SDK license. The Agile SDK license grants the right to use the Agile API in a design environment and to freely distribute the AgileAPI.jar with any application that makes calls to the API. The Agile SDK license explicitly prohibits distribution of the following files to any other party that has not legally obtained an Agile SDK license:

- Agile SDK documentation
- Sample code provided with the Agile SDK
- Source code

Note	The above list is not intended to be a complete list of Agile SDK files you are not allowed to distribute. For complete information, consult your Agile software license agreement.

サンプル プログラム

Agile SDK には、その API の使用方法を示す複数のサンプル プログラムが用意されています。サンプル プログラムは、SDK マニュアルに掲載されており、api、dx、px および wsx フォルダに格納されています。

各サンプル プログラムには、それぞれの Readme.txt があります。必ず Readme.txt ファイルを確認してからサンプル プログラムを実行してください。

Agile API プログラムの開始

Agile API を使用してプログラムを作成する場合は、次の一般的なアプローチに従ってプログラムを構築します。

1. Agile API クラスをインポートする import ステートメントを各クラス ファイルの先頭に追加します。

   ```
   import com.agile.api.*;
   ```

2. Agile アプリケーション サーバのインスタンスを取得します。
3. Agile セッションを作成します。
4. 1 つ以上のビジネス プロセスを完成します。プログラム コードの大半は、このビジネス プロセスに使用されます。
5. Agile セッションを閉じます。

Agile API ライブラリのクラス パスの設定

Java がソース内で参照するクラスを検索する際は、CLASSPATH 変数に指定されているディレクトリが確認されます。Agile API プログラムを作成するには、クラス パスに `AgileAPI.jar` を含める必要があります。

Java 開発環境を使用している場合は、通常、プロジェクトごとにクラス パスを変更できます。開発環境に対して Agile API ライブラリの位置を提示しないと、アプリケーションを構築できません。

Agile API クラスのインポート

プログラムでアクセス権が自動的に付与される唯一の Java パッケージは `java.lang` です。Agile API クラスを参照するには、各クラス ファイルの先頭で `com.agile.api` パッケージをインポートする必要があります。

```
import com.agile.api.*;
```

`com.agile.api` パッケージをインポートせずに、完全なパッケージ名で Agile API クラスを参照することもできます。

```
com.agile.api.IItem source =
(com.agile.api.IItem)m_session.getObject(com.agile.api.IIte
m.OBJECT_TYPE, "1000-02");
```

このように、com.agile.api パッケージをインポートしない場合は、そのクラスのいずれかを参照するたびに、完全なパッケージ名を入力する必要があるため煩雑になります。

com.agile.api パッケージがインポートされていない場合や、完全なパッケージ名で Agile API クラスが参照されない場合は、プログラムを構築しようとしたときに Java コンパイラからエラーが返されます。

セッションの作成およびログイン

Agile API プログラムを開始するには、次の 2 つのタスクを完了する必要があります。

1. Agile アプリケーション サーバのインスタンスを取得します。

   `AgileSessionFactory.getInstance()` メソッドを使用して、Agile サーバのインスタンスを取得します。サーバに対して接続 URL を指定する必要があります。指定する URL は、Agile サーバに直接接続するか、プロキシ Web サーバを介して接続するかによって異なります。

   Agile サーバに直接接続するには、次の URL を入力します。

   http://appserver:port/virtualPath

   プロキシ Web サーバを介して Agile サーバに接続するには、次の URL を入力します。

   protocol://*webserver*:*port*/*virtualPath*

   ここで、

   - *appserver* は Agile サーバのコンピュータ名です。
   - *webserver* は Web サーバのコンピュータ名です。
   - *virtualPath* は Agile PLM サーバの仮想パスです。デフォルト値は `Agile` です。仮想パスは、Agile PLM システムのインストール時に指定されます。詳細は、『Agile PLM Installation Guide』を参照してください。
   - *protocol* は HTTP または HTTPS です。
   - *port* は指定のプロトコルに使用されるポート番号です。このポートは、標準以外のポート番号が使用される場合のみ必要です。それ以外の場合は省略できます。

2. Agile PLM サーバ インスタンスのセッションを作成します。

   `AgileSessionFactory.createSession()` メソッドを使用してセッションを作成します。`createSession()` の `params` パラメータには、ログイン パラメータ (ユーザー名とパスワード) を含む `Map` オブジェクトを指定します。

次の例は、Agile API プログラムでセッションを作成し、Agile PLM サーバにログインする方法を示しています。

例: セッションの作成およびログイン

```
private IAgileSession login(String username, String password) throws
APIException {
  //Create the params variable to hold login parameters
  HashMap params = new HashMap();

  //Put username and password values into params
  params.put(AgileSessionFactory.USERNAME, username);
       params.put(AgileSessionFactory.PASSWORD, password);

       //Get an Agile server instance. ("agileserver" is the
     name of the Agile proxy server,
       //and "virtualPath" is the name of the virtual path
```

```
used for the Agile system.)
  AgileSessionFactory instance =
AgileSessionFactory.getInstance(
    "http://agileserver/virtualPath"
  );
  //Create the Agile PLM session and log in
  return instance.createSession(params);
}
```

注意 1 つのユーザー アカウントで Agile アプリケーション サーバに同時セッションを開くことができる最大数は、Agile PLM ライセンス キーによって決まります。セッションの最大数を超過することになる場合はログインできません。したがって、プログラムの実行を終了する際は、`IAgileSession.close()` メソッドを使用して適切にログアウトし、セッションを閉じることが重要です。Agile PLM システムが Oracle Application Server 上でホスティングされている場合は、各スレッド当たり 1 セッションのみに制限されます。

パスワードで保護された URL へのアクセスによるセッションの作成

ファイヤウォールを越えて Agile PLM にアクセスするユーザーに対して追加のセキュリティを提供するには、パスワードで保護された URL をプロキシ サーバに指定できます。この場合、サーバ インスタンスを取得してセッションを作成する標準的な方法は機能しません。かわりに、`AgileSessionFactory.createSessionEx()` メソッドを使用して、ログインに必要なユーザー名、パスワードおよび URL パラメータを指定する必要があります。`createSessionEx()` を使用する場合は、最初に `AgileSessionFactory.getInstance()` を呼び出してサーバ インスタンスを取得する必要がないため、ログイン コードはさらに簡単になります。`createSessionEx()` メソッドは、1 回の呼び出しでサーバ インスタンスを取得してセッションを作成します。

注意 この `createSessionEx()` メソッドはパスワードで保護されていない URL に対しても機能します。したがって、`createSession()` のかわりに使用することもできます。

例: パスワードで保護されたサーバ URL を使用したセッションの作成

```
private IAgileSession securelogin(String username, String password)
throws APIException {
  //Create the params variable to hold login parameters
  HashMap params = new HashMap();

  //Put username, password, and URL values into params
  params.put(AgileSessionFactory.USERNAME, username);
  params.put(AgileSessionFactory.PASSWORD, password);
  params.put(AgileSessionFactory.URL,
"http://agileserver.agilesoft.com/Agile");

  //Create the Agile PLM session and log in
  return AgileSessionFactory.createSessionEx(params);
}
```

Agile Web サービスからのセッションの作成

Web サービス拡張を使用して Web サービスを開発し、そのサービスを Agile PLM と同じコンテナに配置した場合は、Agile API を利用して Web サービス内から Agile PLM サーバの機能にアクセスできます。Web サービスの Agile PLM サーバ インスタンスを取得するには、AgileSessionFactory.getInstance() メソッドを使用しますが、url パラメータにはヌル値を指定します。

AgileSessionFactory オブジェクトを取得した後は、セッションを作成することもできます。Web サービス依頼には、ユーザー認証が用意されているため、Agile API セッションの作成時に、ユーザー名やパスワードを指定する必要はありません。したがって、AgileSessionFactory.createSession() の params パラメータには、必ずヌル値を指定してください。

```
AgileSessionFactory factory =
AgileSessionFactory.getInstance(null);
IAgileSession session = factory.createSession(null);
```

createSession() の params パラメータにヌル値を指定すると、Agile PLM サーバが Web サービス依頼をインターセプトしたときに実行されたユーザー認証が、Agile API セッションで再利用されます。再度ログインする必要はありません。セッションを閉じる際は IAgileSession.close() を使用しないでください。セッションは、認証ハンドラによって自動的に閉じます。

createSession() メソッドにヌル パラメータを指定すると、認証ハンドラによって作成されたセッションに対応する IAgileSession が作成されます。Web サービスで認証ハンドラを使用しない場合、または認証ハンドラに使用されているユーザーとは異なるユーザーのセッションを作成する場合も、createSession(params) を使用してセッションを作成できます。params パラメータには、ログイン パラメータ (ユーザー名とパスワード) を含む Map オブジェクトを指定します。セッションの作成に認証ハンドラを使用しない場合、そのセッションは自分で閉じる必要があります。IAgileSession.close() メソッドを呼び出してセッションを閉じます。Web サービス拡張の詳細は、323 ページの「Web サービス拡張の開発」を参照してください。

Agile PLM オブジェクトのロードおよび作成

すべての Agile API プログラムにおける基本的な要件は、オブジェクトを取得して作成する機能です。Agile API で使用できるオブジェクトには、次のインターフェースがマップされます。

- qIChange
- qICommodity
- qICustomer
- qIDeclaration
- qIDiscussion
- qIFileFolder
- qIFolder
- qIItem
- qIManufacturer
- qIManufacturerPart
- qIManufacturingSite
- qIPackage
- qIPrice
- qIProgram
- qIProject
- qIQualityChangeRequest
- qIQuery
- qIRequestForQuote
- qIServiceRequest
- qISpecification
- qISubstance
- qISupplier
- qISupplierResponse
- qITransferOrder
- qIUser
- qIUserGroup

これらの Agile PLM オブジェクトをロードして作成するには、最初に AgileSessionFactory オブジェクトのインスタンスを取得してから、Agile PLM セッションを作成する必要があります。次に、IAgileSession.getObject() を使用して Agile PLM オブジェクトをロードし、IAgileSession.createObject() を使用してオブジェクトを作成します。

注意 検索条件およびフォルダの作成方法の詳細は、37 ページの第 3 章「検索条件の作成およびロード」および 99 ページの「フォルダの使用」を参照してください。

オブジェクトのロード

Agile PLM オブジェクトをロードするには、次のいずれかの IAgileSession.getObject() メソッドを使用します。

- IAgileObject getObject(Object objectType, Object params)
- IAgileObject getObject(int objectType, Object params)

オブジェクト タイプの指定

これら 2 つの getObject() メソッドでは、次の値を使用して objectType パラメータを指定できます。

- いずれかの Agile PLM クラスを表す IAgileClass インスタンス。
- クラス ID (部品クラスに対応する ItemConstants.CLASS_PART など)。事前定義のクラス ID は、Agile API に付属している様々な「* Constants」ファイルで使用できます。
- OBJECT_TYPE 定数 (IItem.OBJECT_TYPE、IChange.OBJECT_TYPE など)。
- クラス名 (「部品」など)。ただし、クラス名は変更可能であり、一意性が保証されないため、クラス名を使用してオブジェクトをインスタンス化することはお薦めしません。

注意 getObject() メソッドを使用してオブジェクトをロードするときは、抽象または具象の Agile PLM クラスを指定できます。詳細は、282 ページの「具象クラスと抽象クラス」を参照してください。

オブジェクト パラメータの指定

getObject() メソッドの params パラメータは、Map または String の場合があります。

params パラメータに Map オブジェクトを指定する場合は、属性 (属性 ID または IAttribute オブジェクト) およびそれに対応する値を含める必要があります。Map には、すべての識別関連情報を指定する必要があります。たとえば、IManufacturerPart をロードするときは、製造元名と製造元部品番号の両方を指定する必要があります。

params パラメータに指定する Map オブジェクトに識別情報以外の追加属性が含まれている場合、それらの属性は無視されます。サーバは、識別情報のみを使用してオブジェクトを取得します。Agile PLM オブジェクトを一意に識別する際に使用される属性の完全なリストは、316 ページの「Agile PLM クラスの識別属性」を参照してください。

次の例は、属性 (ItemConstants.ATT_TITLE_BLOCK_NUMBER) と値を指定する Map パラメータを使用して、部品 1000-02 をロードする方法を示しています。

例: Map を使用した部品のロード

```
try {
  Map params = new HashMap();
  params.put(ItemConstants.ATT_TITLE_BLOCK_NUMBER, "1000-02");
  IItem item = (IItem)m_session.getObject(ItemConstants.CLASS_PART,
params);
} catch (APIException ex) {
   System.out.println(ex);
}
```

ロードするオブジェクトに一意識別子として機能する単一の属性がある場合は、その属性の String 値を params パラメータとして入力できます。たとえば、部品番号が部品の一意識別子であるとします。この場合は、部品番号をパラメータとして入力してオブジェクトをロードできます。

注意 すべてのオブジェクトに一意識別子として機能する単一の属性があるとはかぎりません。たとえば、製造元部品は、製造元名と製造元部品番号の両方で識別されます。この場合、製造元部品をロードするには、少なくとも 2 つの属性の値を指定する必要があります。

次の例は、一意の String 識別子を指定して部品 1000-02 をロードする方法を示しています。

例: String を使用した部品のロード

```
try {
  IItem item = (IItem)m_session.getObject(ItemConstants.CLASS_PART,
"1000-02");
} catch (APIException ex) {
  System.out.println(ex);
}
```

異なるタイプのオブジェクトのロード

次に示すように、部品グループ オブジェクトには ICommodity インターフェースを使用できます。次の例は、I Loading の異なるタイプのオブジェクトを使用して、様々なタイプの Agile PLM オブジェクトをロードする複数の異なる方法を示しています。

```
try {
  //Load a change
  IChange change = (IChange)m_session.getObject(IChange.OBJECT_TYPE,
"C00002");
  System.out.println("Change : " + change.getName());

  //Load a commodity
  ICommodity comm =
(ICommodity)m_session.getObject(ICommodity.OBJECT_TYPE, "Res");
  System.out.println("Commodity : " + comm.getName());

  //Load a customer
  ICustomer cust =
(ICustomer)m_session.getObject(ICustomer.OBJECT_TYPE,
"CUST00006");
  System.out.println("Customer : " + cust.getName());
```

```
  //Load a declaration
  IDeclaration dec =
(IDeclaration)m_session.getObject(IDeclaration.OBJECT_TYPE,
"MD00001");
  System.out.println("Declaration : " + dec.getName());

  //Load a discussion
  IDiscussion discussion =
(IDiscussion)m_session.getObject(IDiscussion.OBJECT_TYPE,
"D00002");
  System.out.println("Discussion : " + discussion.getName());

  //Load a file folder
  IFileFolder ff =
(IFileFolder)m_session.getObject(IFileFolder.OBJECT_TYPE,
"FOLDER00133");
  System.out.println("File Folder : " + ff.getName());

  //Load a folder
  IFolder folder = (IFolder)m_session.getObject(IFolder.OBJECT_TYPE,
    "/Personal Searches/MyTemporaryQueries");
  System.out.println("Folder : " + folder.getName());

  //Load an item
  IItem item = (IItem)m_session.getObject(IItem.OBJECT_TYPE,
"1000-02");
  System.out.println("Item : " + item.getName());

  //Load a manufacturer
  Map params = new HashMap();
  params.put(ManufacturerConstants.ATT_GENERAL_INFO_NAME, "World
Enterprises");
  IManufacturer mfr =

(IManufacturer)m_session.getObject(IManufacturer.OBJECT_TYPE,
params);
  System.out.println("Manufacturer : " + mfr.getName());
  //Load a manufacturer part
  params.clear();

params.put(ManufacturerPartConstants.ATT_GENERAL_INFO_MANUFACTURER
_NAME, "World Enterprises");

params.put(ManufacturerPartConstants.ATT_GENERAL_INFO_MANUFACTURER
_PART_NUMBER, "WE10023-45");
  IManufacturerPart mfrPart =

(IManufacturerPart)m_session.getObject(IManufacturerPart.OBJECT_TY
PE, params);
  System.out.println("ManufacturerPart : " + mfrPart.getName());
  //Load a manufacturing site
  IManufacturingSite siteHK =
(IManufacturingSite)m_session.getObject(
```

```
    ManufacturingSiteConstants.CLASS_SITE, "Hong Kong");
  System.out.println("ManufacturingSite : " + siteHK.getName());
  //Load a package
  IPackage pkg =
(IPackage)m_session.getObject(PackageConstants.CLASS_PACKAGE,
"PKG00010");
  System.out.println("Package : " + pkg.getName());

  //Load a price
  IPrice price = (IPrice)m_session.getObject(IPrice.OBJECT_TYPE,
"PRICE10008");
  System.out.println("Price : " + price.getName());

  //Load a program
  IProgram program =
(IProgram)m_session.getObject(IProgram.OBJECT_TYPE, "PGM10008");
  System.out.println("Program : " + program.getName());

  //Load a PSR
  IServiceRequest psr =
(IServiceRequest)m_session.getObject(IServiceRequest.OBJECT_TYPE,
    "NCR01562");
  System.out.println("PSR : " + psr.getName());
  //Load a QCR
  IQualityChangeRequest qcr =
(IQualityChangeRequest)m_session.getObject(
    IQualityChangeRequest.OBJECT_TYPE, "CAPA02021");
  System.out.println("QCR : " + qcr.getName());
  //Load a query
  IQuery query =
(IQuery)m_session.getObject(IQuery.OBJECT_TYPE,
    "/Personal Searches/Part Numbers Starting with P");
  System.out.println("Query : " + query.getName());
  //Load an RFQ
  IRequestForQuote rfq =
(IRequestForQuote)m_session.getObject(
    IRequestForQuote.OBJECT_TYPE, "RFQ01048");
  System.out.println("RFQ : " + rfq.getName());
  //Load an RFQ response
  params.clear();

params.put(SupplierResponseConstants.ATT_COVERPAGE_RFQ_NUMBER,
"RFQ01048");

params.put(SupplierResponseConstants.ATT_COVERPAGE_SUPPLIER,
"SUP20013");
  ISupplierResponse rfqResp =
(ISupplierResponse)m_session.getObject(
    ISupplierResponse.OBJECT_TYPE, params);
  System.out.println("RFQ Response : " + rfqResp.getName());
  //Load a sourcing project
```

```
  IProject prj = (IProject)m_session.getObject(IProject.OBJECT_TYPE,
"PRJACME_110");
  System.out.println("Project : " + prj.getName());

  //Load a specification
  ISpecification spec =
(ISpecification)m_session.getObject(ISpecification.OBJECT_TYPE,
    "WEEE");
  System.out.println("Specification : " + spec.getName());
  //Load a substance
  ISubstance sub =
(ISubstance)m_session.getObject(ISubstance.OBJECT_TYPE,
"Cadmium");
  System.out.println("Substance : " + sub.getName());

  //Load a supplier
  ISupplier supplier =
(ISupplier)m_session.getObject(ISupplier.OBJECT_TYPE,
"SUP20013");
  System.out.println("Supplier : " + supplier.getName());

  //Load a transfer order
  ITransferOrder to =
(ITransferOrder)m_session.getObject(TransferOrderConstants.
CLASS_CTO,
    "456602");
  System.out.println("TransferOrder : " + to.getName());
  //Load a user
  params.clear();
  params.put(UserConstants.ATT_GENERAL_INFO_USER_ID,
"OWELLES");
  IUser user =
(IUser)m_session.getObject(IUser.OBJECT_TYPE, params);
  System.out.println("User : " + user.getName());

  //Load a user group
  params.clear();
  params.put(UserGroupConstants.ATT_GENERAL_INFO_NAME,
"Designers");
  IUserGroup group =
(IUserGroup)m_session.getObject(IUserGroup.OBJECT_TYPE,
params);
  System.out.println("UserGroup : " + group.getName());

} catch (APIException ex) {
  System.out.println(ex);
}
```

オブジェクトの作成

Agile PLM オブジェクトを作成するには、次のいずれかの IAgileSession.createObject() メソッドを使用します。

- IAgileObject createObject(Object objectType, Object params)
- IAgileObject createObject(int objectType, Object params)

重要 SDK では、オブジェクトを作成する際に、そのオブジェクトのライフ サイクル フェーズ (LCP)/ワークフロー ステータス属性を設定することはできません。これは、オブジェクトが作成されるまでは、LCP に使用できる設定が有効にならないようにするためです。UI に対しても同様のルールが適用されます。たとえば、IChange は、ワークフローが選択されるまで LCP 値を取得しません。ただし、SDK を使用してオブジェクトを作成した後は、LCP/ワークフロー ステータス属性を設定したり、変更することが可能です。オブジェクトが作成され、オブジェクトに関連するアクションが実行されるまでは、このフィールドの値リストを取得できないことに注意してください。

objectType および params パラメータは、IAgileSession.getObject() メソッドで使用される場合と同じです。詳細は、16 ページの「オブジェクトのロード」を参照してください。IFolder および IQuery オブジェクト以外は、objectType パラメータに具象クラスを指定する必要があります。たとえば、部品を作成している場合、そのクラスはインスタンス化できない抽象クラスであるため、ItemConstants.CLASS_PARTS_CLASS を指定できません。ただし、事前定義またはユーザー定義の具象クラスのクラス ID (ItemConstants.CLASS_PART など) は指定できます。

ユーザー定義のサブクラスのオブジェクトを作成する場合、createObject() の objectType パラメータは、サブクラス ID に対応する Integer オブジェクトであることが必要です。Agile PLM システムで使用可能なすべてのユーザー定義サブクラスに対して定数を定義する場合もあります。

Map または String タイプに加え、IAgileSession.createObject() の params パラメータには、特定のオブジェクト クラスの自動採番ソースを表す INode オブジェクトを指定することもできます。Agile アプリケーション サーバは、次の番号に対する自動採番ソースを一連の番号の中で検索し、その番号が一意識別子として使用されます。

注意 使用可能な自動採番ソースがないオブジェクトの params パラメータには、INode オブジェクトを指定できません。

次の例は、属性 (ItemConstants.ATT_TITLE_BLOCK_NUMBER) と値を指定する Map パラメータを使用して、部品 1000-02 を作成する方法を示しています。

例: Map を使用した部品の作成

```
try {
  Map params = new HashMap();
  params.put(ItemConstants.ATT_TITLE_BLOCK_NUMBER, "1000-02");
  IItem item =
(IItem)m_session.createObject(ItemConstants.CLASS_PART, params);
} catch (APIException ex) {
    System.out.println(ex);
}
```

次の例は、一意の String 識別子を指定して部品 1000-02 を作成する方法を示しています。

例: String を使用した部品の作成

```
try {
  IItem item =
(IItem)m_session.createObject(ItemConstants.CLASS_PART,
"1000-02");
} catch (APIException ex) {
    System.out.println(ex);
}
```

Agile PLM クラスの使用

クラスは Agile アプリケーション サーバごとにカスタマイズされているため、特に複数の Agile アプリケーション サーバまたは異なるロケールでプログラムを使用する予定がある場合は、クラス名に対する参照のハードコード化は回避する必要があります。かわりに、実行時には各オブジェクト タイプのクラスを取得できます。プログラムによって、ユーザーがリストからクラスを選択できるユーザー インターフェースを提供できます。

次の例は、実行時に特定のオブジェクト タイプに対するクラスのリストを取得する方法を示しています。

例: クラスの取得

```
try {
  //Get the IAdmin interface for this session
  IAdmin m_admin = m_session.getAdminInstance();
  //Get the Item base class
  IAgileClass itemClass =

m_admin.getAgileClass(ItemConstants.CLASS_ITEM_BASE_CLASS);
  // Clear the Item Type combo box
  comboItemType.removeAllItems();

  // Get the Item subclass names and populate the Item Type combo box
  IAgileClass[] subclasses = itemClass.getSubclasses();
  for (int i = 0; i < subclasses.length; ++i) {
    comboItemType.addItem(subclasses[i].getName());
  }
} catch (APIException ex) {
    System.out.println(ex);
}
```

ユーザー定義サブクラスのオブジェクトの作成

ユーザー定義のサブクラスは、Agile PLM システムのために特別に作成されたクラスです。したがって、Agile API には、これらのサブクラスに対する事前定義のクラス ID は用意されていません。`createObject()` の `objectType` パラメータにユーザー定義のサブクラスを指定するには、クラス ID に対応する `Integer` を渡します。ユーザー定義クラスのクラス ID を取得するには、`IAgileClass.getId()` メソッドを使用します。

次の例は、レジスタ オブジェクトを作成する方法を示しています。この例で、レジスタは部品クラスのユーザー定義サブクラスです。

例: ユーザー定義サブクラスのオブジェクトの作成

```
try {
  //Define a variable for the Resistor subclass
  Integer classResistor = null;
  //Get the Resistor subclass ID
  IAgileClass[] classes =
m_admin.getAgileClasses(IAdmin.CONCRETE);
  for (int i = 0; i < classes.length; i++) {
    if (classes[i].getName().equals("Resistor")) {
      classResistor = (Integer)classes[i].getId();
      break;
    }
  }
  //Create a Resistor object
  if (classResistor != null) {
    IItem resistor =
(IItem)m_session.createObject(classResistor, "R10245");
  }
} catch (APIException ex) {
    System.out.println(ex);
}
```

もちろん、ユーザー定義サブクラスは、次の例のように名前で参照することもできます。ただし、クラス名は一意であるとはかぎりません。同じ名前のサブクラスが 2 つある場合は、最初に検出されたサブクラスが照合されますが、これは意図しているサブクラスではない可能性があります。

例: サブクラス名の参照によるオブジェクトの作成

```
try {
  IItem resistor =
(IItem)m_session.createObject("Resistor", "R10245");
} catch (APIException ex) {
  System.out.println(ex);
}
```

自動採番の使用

Agile PLM クラスには、1 つ以上の AutoNumber ソースを指定できます。AutoNumber ソースは、事前定義の連続番号で、オブジェクトの番号を自動的に割り当てます。AutoNumber ソースは、Agile Java クライアントの管理機能で定義されます。

注意 自動採番は、製造元クラス、製造元部品クラスおよびそれらのユーザー定義サブクラスでは、サポートされません。

特定のクラスのオブジェクトを作成する場合は、AutoNumber を使用するように Agile アプリケーション サーバを設定する必要があります。オブジェクトに自動採番が必要かどうかは、IAgileClass.isAutoNumberRequired() メソッドによって判断されます。ただし、このメソッドは、特定のクラスに自動採番が必要な場合でも Agile API ではオブジェクトの自動採番が実施されないため、お薦めできません。自社の環境でこの機能が必要な場合は、必要なルーチンを開発する必要があります。したがって、ユーザーによる Agile PLM オブジェクトの作成を可能にする GUI プログラムを開発する場合は、必要なときにユーザー インターフェースで自動採番が実施されることを確認してください。クライアント プログラムによる自動採番の実施方法の例では、Agile Web クライアントを使用して数個のオブジェクトを作成し、ユーザー インターフェースが機能する様子を確認してください。

一連の番号の中で次に使用可能な自動採番を取得する手順は、次のとおりです。

一連の番号の中で次に使用可能な AutoNumber を割り当てるには、IAutoNumber.getNextNumber(IAgileClass) メソッドを使用します。このメソッドは、該当する番号が別のオブジェクトで使用されていないことを確認します。このプロセスは、指定した Agile サブクラスに使用可能な最初の自動採番を検出して返すまで継続されます。次に使用可能な自動採番の取得に失敗した場合は、例外が発生します。該当する番号が別のオブジェクトですでに使用されている場合、IAutoNumber.getNextNumber() メソッドは、確認を実行せずにスキップします。

次の例は、次の AutoNumber を使用して部品を作成する方法を示しています。

例: 次に使用可能な自動採番の取得

```
private void createPart(String partNumber) throw APIException {
  IAdmin admin;
  IAgileClass cls;
  IItem part;
  IAutoNumber[] numSources;
   String nextAvailableAutoNumber;

  //Get the Admin instance
  admin = session.getAdminInstance();

  //Get the Part class
  cls = admin.getAgileClass(ItemConstants.CLASS_PART);

  //Check if AutoNumber is required
  if (isAutoNumberRequired(cls)) {

    // Get AutoNumber sources for the Part class
    numSources = cls.getAutoNumberSources();

    // Get the next available AutoNumber using the first
autonumber source
     nextAvailableAutoNumber =
numSources[0].getNextNumber(cls);
    // Create the part using the available AutoNumber
    part =
(IItem)session.createObject(ItemConstants.CLASS_PART,
nextAvailableAutoNumber);
  } else {
  // Create the part using the specified number
  // (if AutoNumber is not required)
  part = (IItem)session.createObject(ItemConstants.CLASS_PART,
partNumber);
}
public boolean isAutoNumberRequired(IAgileClass cls) throws
APIException {
  if (cls.isAbstract()) {
    return false;
  }
  IProperty p =
```

```
((INode)cls).getProperty(PropertyConstants.PROP_AUTONUMBER_
REQUIRED);
  if (p != null) {
    IAgileList value = (IAgileList)p.getValue();
    return ((Integer)(value.getSelection()[0]).getId()).intValue()
== 1;
  }
  return false;
}
```

一連の番号の中で次の自動採番を取得する手順は、次のとおりです。

一連の番号の中で次の自動採番を増分または検索するには、`IAutoNumber.getNextNumber(IAgileClass)` メソッドを使用します。このメソッドは、次の自動採番を生成しますが、その可用性は確認しません。つまり、生成した自動採番が別の Agile オブジェクトで使用されているかどうかは検証されません。次の自動採番の取得に失敗した場合は、例外が発生します。

たとえば、可用性を確認せずに次の自動採番を割り当てる場合は、前述の例を次のように変更します。

- `String nextAvailableAutoNumber` を `String nextAutoNumber` に置き換えます。
- `nextAvailableAutoNumber = numSources[0].getNextNumber(cls);` を `nextAutoNumber = numSources[0].getNextNumber();` に置き換えます。
- `part = (IItem)session.createObject(ItemConstants.CLASS_PART, nextAvailableAutoNumber);` を `part = (IItem)session.createObject(ItemConstants.CLASS_PART,nextAutoNumber);` に置き換えます。

必須フィールドの設定

クラスは、複数の必須属性を使用して定義できます。特定の属性を必須にするには、Agile PLM 管理者が属性の [表示] プロパティと [必須] プロパティを [はい] に設定します。Agile Java クライアントまたは Agile Web クライアントで、必須フィールドに値を設定せずにオブジェクトを作成しようとした場合、そのオブジェクトは、すべての必須フィールドに値を設定するまで保存できません。

Agile PLM 管理者は、属性がクラスに対して必須かどうかを定義できますが、ユーザーによる値の設定時に、Agile API で、必須フィールドへの値の入力を自動的に強制することはありません。したがって、すべての必須フィールドに値が設定されていない場合でも、API を使用してオブジェクトを作成し、保存できます。クライアント プログラムで必須フィールドへの値の入力を強制し、Agile Web クライアントおよび Java クライアントと同様に動作させる場合は、対応するコードを記述する必要があります。

必須フィールドを確認する手順は、次のとおりです。

1. `ITable.getAttributes()` または `ITableDesc.getAttributes()` を呼び出して、テーブルの属性リストを取得します。
2. 各属性に対して、`IAttribute.getProperty(PropertyConstants.PROP_REQUIRED).getValue()` を呼び出して、[必須] プロパティの値を取得します。

次の例は、クラスの [ページ 1]、[ユーザー定義 1] および [ユーザー定義 2] について、必須属性の配列を取得する方法を示しています。

例: クラスの必須属性の取得

```
/**
 * Returns true if the specified attribute is required and visible.
 */
public boolean isRequired(IAttribute attr) throws APIException {
  boolean result = false;
  IProperty required =
attr.getProperty(PropertyConstants.PROP_REQUIRED);
  if (required != null) {
    Object value = required.getValue();
    if (value != null) {
      result = value.toString().equals("Yes");
    }
  }
  IProperty visible =
attr.getProperty(PropertyConstants.PROP_VISIBLE);
  if (visible != null) {
    Object value = visible.getValue();
    if (value != null) {
      result &= value.toString().equals("Yes");
    }
  }
  return result;
}
/**
 * Returns an array containing the required attributes for the specified
class.
 */
public IAttribute[] getRequiredAttributes(IAgileClass cls) throws
APIException {
  //Create an array list for the results
  ArrayList result = new ArrayList();

  //Check if the class is abstract or concrete
  if (!cls.isAbstract()) {
    IAttribute[] attrs = null;
    //Get required attributes for Page One
    ITableDesc page1 =
cls.getTableDescriptor(TableTypeConstants.TYPE_PAGE_ONE);
    if (page1 != null) {
      attrs = page1.getAttributes();
      for (int i = 0; i < attrs.length; i++) {
        IAttribute attr = attrs[i];
        if (isRequired(attr)) {
          result.add(attr);
        }
      }
    }
    //Get required attributes for Page Two
    ITableDesc page2 =
cls.getTableDescriptor(TableTypeConstants.TYPE_PAGE_TWO);
    if (page2 != null) {
      attrs = page1.getAttributes();
      for (int i = 0; i < attrs.length; i++) {
        IAttribute attr = attrs[i];
```

```
        if (isRequired(attr)) {
          result.add(attr);
        }
      }
    }
    //Get required attributes for Page Three
    ITableDesc page3 =
cls.getTableDescriptor(TableTypeConstants.TYPE_PAGE_THREE);
    if (page3 != null) {
      attrs = page3.getAttributes();
      for (int i = 0; i < attrs.length; i++) {
        IAttribute attr = attrs[i];
        if (isRequired(attr)) {
          result.add(attr);
        }
      }
    }
  }
  return (IAttribute[])result.toArray(new IAttribute[0]);
}
```

注意 オブジェクトの作成に使用されるプライマリ キー フィールドは、[必須] プロパティの設定に関係なく必須です。たとえば、新しいアイテムを作成する場合、アイテムの [タイトル ブロック.番号] フィールドは、それが必須フィールドかどうかに関係なく指定する必要があります。

異なるタイプのオブジェクトの作成

次の例は、様々なタイプの Agile PLM オブジェクトを作成する複数の異なる方法を示しています。コードを簡単にするために自動採番は使用していません。

例: 異なるタイプのオブジェクトの作成

```
try {
  //Create a Map object to store parameters
  Map params = new HashMap();
  //Create a change
  IChange eco =
(IChange)m_session.createObject(ChangeConstants.CLASS_ECO,
"C00002");
  System.out.println("Change : " + eco.getName());

  //Create a commodity
  ICommodity comm =
(ICommodity)m_session.createObject(CommodityConstants.CLASS_COMMOD
ITY,"RES");
  System.out.println("Commodity : " + comm.getName());

  //Create a customer
  params.clear();

params.put(CustomerConstants.ATT_GENERAL_INFO_CUSTOMER_NUMBER,
"CUST00006");

params.put(CustomerConstants.ATT_GENERAL_INFO_CUSTOMER_NAME,
"Western Widgets");
```

```
  ICustomer customer =
(ICustomer)m_session.createObject(CustomerConstants.CLASS_CUSTOMER,
params);
  System.out.println("Customer : " + customer.getName());
  //Create a declaration
  params.clear();
  ISupplier supplier =
(ISupplier)m_session.getObject(ISupplier.OBJECT_TYPE, "SUP20013");
  params.put(DeclarationConstants.ATT_COVER_PAGE_NAME, "MD00001");
  params.put(DeclarationConstants.ATT_COVER_PAGE_SUPPLIER,
supplier);
  IDeclaration dec = (IDeclaration)

m_session.createObject(DeclarationConstants.CLASS_SUBSTANCE_DECLAR
ATION, params);
  System.out.println("Declaration : " + dec.getName());
  //Create a discussion
  params.clear();
  params.put(DiscussionConstants.ATT_COVER_PAGE_NUMBER,
"D000201");
  params.put(DiscussionConstants.ATT_COVER_PAGE_SUBJECT,
"Packaging issues");
  IDiscussion discussion =
(IDiscussion)m_session.createObject(
    DiscussionConstants.CLASS_DISCUSSION, params);
  System.out.println("Discussion : " + discussion.getName());
  //Create a file folder
  IFileFolder ff = (IFileFolder)m_session.createObject(
    FileFolderConstants.CLASS_FILE_FOLDER, "FOLDER00133");
  System.out.println("File Folder : " + ff.getName());
  //Create a folder
  params.clear();
  IFolder parentFolder =
(IFolder)m_session.getObject(IFolder.OBJECT_TYPE,
"/Personal Searches");
  params.put(FolderConstants.ATT_FOLDER_NAME,
"MyTemporaryQueries");
  params.put(FolderConstants.ATT_PARENT_FOLDER,
parentFolder);
  IFolder folder =
(IFolder)m_session.createObject(IFolder.OBJECT_TYPE,
params);
  System.out.println("Folder : " + folder.getName());

  //Create an item
  IItem part =
```

```
(IItem)m_session.createObject(ItemConstants.CLASS_PART,
"1000-02");
  System.out.println("Item : " + part.getName());

  //Create a manufacturer
  params.put(ManufacturerConstants.ATT_GENERAL_INFO_NAME,
"World Enterprises");
  IManufacturer mfr =
(IManufacturer)m_session.createObject(
    ManufacturerConstants.CLASS_MANUFACTURER, params);
  System.out.println("Manufacturer : " + mfr.getName());
  //Create a manufacturer part
  params.clear();

params.put(ManufacturerPartConstants.ATT_GENERAL_INFO_MANUFACTURER
_NAME, "World Enterprises");

params.put(ManufacturerPartConstants.ATT_GENERAL_INFO_MANUFACTURER
_PART_NUMBER, "WE10023-45");
  IManufacturerPart mfrPart =
(IManufacturerPart)m_session.createObject(
    ManufacturerPartConstants.CLASS_MANUFACTURER_PART, params);
  System.out.println("ManufacturerPart : " + mfrPart.getName());
  //Create a manufacturing site
  IManufacturingSite siteHK =
(IManufacturingSite)m_session.createObject(
    ManufacturingSiteConstants.CLASS_SITE, "Hong Kong");
  System.out.println("ManufacturingSite : " + siteHK.getName());
  //Create a package
  IPackage pkg =
(IPackage)m_session.createObject(PackageConstants.CLASS_PACKAGE,
"PKG00010");
  System.out.println("Package : " + pkg.getName());

  //Create a price
  params.clear();
  params.put(PriceConstants.ATT_GENERAL_INFORMATION_NUMBER,
"PRICE10008");

params.put(PriceConstants.ATT_GENERAL_INFORMATION_CUSTOMER,
"CUST00006");

params.put(PriceConstants.ATT_GENERAL_INFORMATION_ITEM_NUMBER,
"1000-02");

params.put(PriceConstants.ATT_GENERAL_INFORMATION_ITEM_REV, "B");
```

```
params.put(PriceConstants.ATT_GENERAL_INFORMATION_PROGRAM,
"PROGRAM0023");

params.put(PriceConstants.ATT_GENERAL_INFORMATION_MANUFACTURING_SI
TE, "San Jose");
  params.put(PriceConstants.ATT_GENERAL_INFORMATION_SUPPLIER,
"SUP20013");
  IPrice price =
(IPrice)m_session.createObject(PriceConstants.CLASS_PUBLISHED_PRIC
E, params);
  System.out.println("Price : " + price.getName());
  //Create a program
  DateFormat df = new SimpleDateFormat("MM/dd/yy");
  IAttribute attr =
m_admin.getAgileClass(ProgramConstants.CLASS_PROGRAM).

getAttribute(ProgramConstants.ATT_GENERAL_INFO_DURATION_TYPE);
  IAgileList list = attr.getAvailableValues();
  list.setSelection(new Object[] {"Fixed"});
  params.clear();
  params.put(ProgramConstants.ATT_GENERAL_INFO_NAME, "Wingspan
Program");

params.put(ProgramConstants.ATT_GENERAL_INFO_SCHEDULE_START_DATE,
df.parse("06/01/05"));

params.put(ProgramConstants.ATT_GENERAL_INFO_SCHEDULE_END_DATE,
df.parse("06/30/05"));

params.put(ProgramConstants.ATT_GENERAL_INFO_DURATION_TYPE, list);
  IProgram program =
(IProgram)m_session.createObject(ProgramConstants.CLASS_PROGRAM,
params);
  System.out.println("Program : " + program.getName());
  //Create a PSR
  IServiceRequest psr =
(IServiceRequest)m_session.createObject(
    ServiceRequestConstants.CLASS_NCR, "NCR01562");
  System.out.println("PSR : " + psr.getName());
  //Create a QCR
  IQualityChangeRequest qcr =
(IQualityChangeRequest)m_session.createObject(
    QualityChangeRequestConstants.CLASS_CAPA, "CAPA02021");
  System.out.println("QCR : " + qcr.getName());
  //Create a query
  params.clear();
  IFolder parent =
```

```
(IFolder)m_session.getObject(IFolder.OBJECT_TYPE,
"/Personal Searches");
  String condition = "[Title Block.Number] starts with 'P'";
  params.put(QueryConstants.ATT_CRITERIA_CLASS,
ItemConstants.CLASS_ITEM_BASE_CLASS);
  params.put(QueryConstants.ATT_CRITERIA_STRING, condition);
  params.put(QueryConstants.ATT_PARENT_FOLDER, parent);
  params.put(QueryConstants.ATT_QUERY_NAME, "Part Numbers Starting
with P");
  IQuery query =
(IQuery)m_session.createObject(IQuery.OBJECT_TYPE, params);
  System.out.println("Query : " + query.getName());

  //Create a specification
  ISpecification spec = (ISpecification)

m_session.createObject(SpecificationConstants.CLASS_SPECIFICATION,
"WEEE");
  System.out.println("Specification : " + spec.getName());
  //Create a substance
  ISubstance sub =
(ISubstance)m_session.createObject(SubstanceConstants.CLASS
_SUBSTANCE,
    "Cadmium");
  System.out.println("Substance : " + spec.getName());
  //Create a transfer order
  ITransferOrder to =
(ITransferOrder)m_session.createObject(
    TransferOrderConstants.CLASS_CTO, "456602");
  System.out.println("TransferOrder : " + to.getName());
  //Create a user
  params.clear();
  params.put(UserConstants.ATT_GENERAL_INFO_USER_ID, "OWELLES");
  params.put(UserConstants.ATT_LOGIN_PASSWORD, "agile");
  IUser user =
(IUser)m_session.createObject(UserConstants.CLASS_USER, params);
  System.out.println("User : " + user.getName());

  //Create a user group
  params.clear();
  params.put(UserGroupConstants.ATT_GENERAL_INFO_NAME,
"Designers");
  IUserGroup group =
(IUserGroup)m_session.createObject(UserGroupConstants.CLASS
_USER_GROUP,
  params);
  System.out.println("UserGroup : " + group.getName());

} catch (APIException ex) {
    System.out.println(ex);
}
```

注意 SupplierResponse の作成に Agile API を使用することはできません。

Agile PLM オブジェクトの状態の確認

IStateful インターフェースは、Agile ワークフローまたは Agile ライフサイクルのいずれかの状態を保持する Agile オブジェクトをサポートしています。このインターフェースをサポートするオブジェクトは、アイテムおよび送信可能なオブジェクトです。

送信可能なオブジェクトは、次のとおりです。

- IChange
- IDeclaration
- IFileFolder
- IPackage
- IProgram
- IQualityChangeRequest
- IServiceRequest
- ITransferOrder

次の例では、オブジェクトのすべての状態を示す配列 (状態が未定義の場合はヌル) が返されます。

例: オブジェクトの様々な状態を定義する配列の取得

```
public interface IStateful {
public IStatus[] getStates()
throws APIException;
}
```

次の例では、オブジェクトの現在の状態 (状態が未定義の場合はヌル) が返されます。

例: オブジェクトの現在の状態の取得

```
public interface IStateful {
public IStatus getStatus()
throws APIException;
}
```

関連オブジェクトへの値の継承

Agile PLM のいくつかのオブジェクトには、関連オブジェクトがあります。たとえば、問題レポートや不具合レポートには [関連 PSR] テーブルがあります。[関連 PSR] テーブルには、関連オブジェクト (別の問題レポートや不具合レポートなど) での特定の結果をワークフロー イベントによってトリガーすることを指定できます。トリガーされた結果は、即時には発生しません。実際には、Agile PLM が値を関連オブジェクトに継承する際は、数秒程度の明らかな遅延が生じます。

オブジェクトを新規オブジェクトとして保存

Agile API では、既存のオブジェクトを新規オブジェクトとして保存できます。たとえば、プログラムのダイアログ ボックスには、[保存] ボタンに加えて、データを新しいオブジェクトとして保存する [名前を付けて保存] ボタンがあります。IDataObject.saveAs() メソッドを使用する場合は、オブジェクトとオブジェクト番号の保存に使用するサブクラスを指定する必要があります。自動採番は、サブクラスがサポートしている場合に使用できます。

次の例は、指定したサブクラスの次の自動採番を使用して、現在のオブジェクトを新しいオブジェクトとして保存する方法を示しています。

例: オブジェクトを新規オブジェクトとして保存

```
private void saveAsObject(IDataObject obj, IAgileClass sub) {
  String nextNum;
  try {
    // Get the next autonumber for the sublass
    IAutoNumber[] numSources = sub.getAutoNumberSources();
    nextNum = numSources[0].getNextNumber();

    // Save the object
    obj.saveAs(sub, nextNum);
  } catch (APIException ex) {
    System.out.println(ex);
  }
}
```

オブジェクトの共有

IShareable インターフェースは、Agile API が公開するすべての Agile PLM ビジネス オブジェクトによって実装されます。したがって、すべてのビジネス オブジェクトは共有可能です。共有機能を利用すると、自分が持っている 1 つ以上の役割を、別の Agile PLM ユーザーまたはユーザー グループに付与できます (対象とするオブジェクトは限定されます)。オブジェクトを共有している場合に割り当てることができる役割には、自分に与えられた役割または永続的な役割と、ユーザー グループのメンバーシップから割り当てられている役割があります。

オブジェクトを共有しているユーザーが実行できるアクションは、そのプロジェクトの役割で許可されているアクションのみです。これらのユーザーが、グローバルな方法で役割を取得することはありません。

IShareable インターフェースのメソッドは、getUsersAndRoles() と setUsersAndRoles() の 2 つのみです。getUsersAndRoles() メソッドは Map オブジェクトを返します。Map 内の各ユーザーには、関連付けられている役割の配列があります。setUsersAndRoles() メソッドのパラメータは 1 つで、Map オブジェクトです。このオブジェクトは、getUsersAndRoles() から返される Map と同様に、各ユーザーを役割の配列にマップします。各ユーザーには、選択した様々な役割を割り当てることができます。

例: オブジェクトの共有

```
private void getDataForSharing() throws Exception {
  //Get item
  IItem item =
(IItem)m_session.getObject(ItemConstants.CLASS_ITEM_BASE_CLASS,
"P10011");

  //Get users
  IUser user1 = (IUser)m_session.getObject(UserConstants.CLASS_USER,
"albert1");
  IUser user2 = (IUser)m_session.getObject(UserConstants.CLASS_USER,
"peter1");
  IUser[] users = new IUser[]{user1, user2};

  //Get roles
  INode nodeRoles =
(INode)m_session.getAdminInstance().getNode(NodeConstants.NODE_ROL
ES);
  IRole role1 = (IRole)nodeRoles.getChildNode("Component Engineer");
  IRole role2 = (IRole)nodeRoles.getChildNode("Incorporator");
  IRole[] roles = new IRole[]{role1, role2};

  //Share the item
  shareItem(item, users, roles);
}
private void shareItem(IItem item, IUser[] users, IRole[] roles) throws
Exception {
  Map map = new HashMap();
  for (int i = 0; i < users.length; i++) {
    map.put(users[i], roles);
  }
  IShareable shareObj = (IShareable)item;
  shareObj.setUsersAndRoles(map);
}
```

注意 ユーザーとユーザー グループには [共有] テーブルがあり、共有されているオブジェクトと、それらのオブジェクトに付与されている役割がリストされます。

オブジェクトの削除および削除取消

オブジェクトは、Agile Web クライアントなどの Agile API を使用して削除したり、削除を取り消すことができます。オブジェクトの削除と削除取消には、特定のオブジェクト タイプに対する削除権限と削除取消権限がそれぞれ必要です。

Agile API では、ソフト削除とハード削除をサポートしています。オブジェクトを初めて削除する場合の削除は、ソフト削除です。データベース内で削除マークが付きますが、完全に削除されたわけではありません。ソフト削除されたオブジェクトは、その後も取得可能です。たとえば、削除したオブジェクトは、`IAgileSession.getObject()` メソッドを使用して取得できます。検索を実行した場合、ソフト削除されたオブジェクトはその検索結果には表示されません。ただし、削除されたオブジェクトを検索できる事前定義の検索条件 ([変更分析者検索] フォルダの [削除されたアイテム] 検索条件など) が用意されています。

オブジェクトを完全に削除するには、削除を 2 回実行します。これがハード削除です。オブジェクトをハード削除すると、IDataObject.undelete() メソッドを使用してもオブジェクトを復元できません。

すべての Agile PLM オブジェクトが削除できるわけではありません。たとえば、次のオブジェクトは削除できません。これらのいずれかのオブジェクトを削除しようとした場合は、delete() メソッドで例外が発生します。

- 保留中の変更があるアイテム
- リビジョン履歴があるアイテム
- キャンセルされた変更があるアイテム
- リリース済みの変更
- 1 つ以上の製造元部品がある製造元
- 別のオブジェクトの [製造元] タブで現在使用されている製造元部品

別のアイテムの [BOM] タブで使用されているアイテムを削除しようとすると、Agile PLM サーバにより、ExceptionConstants.APDM_DELETECOMPINUSE_WARNING という ID の例外が発生します。次の例は、この警告を無効にしてアイテムを削除する方法を示しています。

例: アイテムの削除

```
private void deleteItem(IDataObject obj) {
  try {
    // Delete the Item
    obj.delete();
  } catch (APIException ex) {
    // Check for "Item is Used" warning
    if (ex.getErrorCode() ==
ExceptionConstants.APDM_DELETECOMPINUSE_WARNING) {
     int i = JOptionPane.showConfirmDialog(null, "This Item is used
by another Item. " +
     "Would you still like to delete it?", "Item is Used Warning",
JOptionPane.YES_NO_OPTION);
    }
    if (i == 0) {
      try {
        // Disable "Item is Used" warning

m_session.disableWarning(ExceptionConstants.APDM_DELETECOMPINUSE_W
ARNING);
        // Delete the object
        obj.delete();
        // Enable "Item is Used" warning

m_session.enableWarning(ExceptionConstants.APDM_DELETECOMPINUSE_WA
RNING);
      } catch (APIException exc) {
        System.out.println(exc);
      }
    } else {
      System.out.println(ex);
    }
  }
}
```

ソフト削除されたオブジェクトを復元するには、IDataObject.undelete() メソッドを使用します。再度、オブジェクトの削除を取り消すには、そのオブジェクト タイプに対する削除取消権限が必要です。ただし、[対象アイテム] タブにアイテムがあるソフト削除された変更は、ユーザーの権限に関係なく復元できません。次の例は、削除したオブジェクトの削除を取り消す方法を示しています。

例: オブジェクトの削除取消

```
private void undeleteObject(Object obj) throws APIException {
  // Make sure the object is deleted before undeleting it
  if (obj.isDeleted()) {
      // Restore the object
      obj.undelete();
  }
}
```

セッションを閉じる

Agile PLM の各ユーザーは、同時セッションを最大 3 つまで開くことができます。したがって、Agile API を使用して開いた各セッションは、適切に閉じる必要があります。セッションを適切に閉じないと、いずれかの同時セッションがタイムアウトするまで、新しいセッションでログインできない可能性があります。

例: セッションを閉じる

```
public void disconnect(IAgileSession m_session) {
  m_session.close();
}
```

第 3 章

検索条件の作成およびロード

扱うトピックは次のとおりです。

検索条件について

IQuery は、Agile PLM データの検索方法を定義するオブジェクトです。Agile Web クライアントで使用できる検索と同様の検索を定義します。検索には、複数の検索条件 (Agile Web クライアントの詳細検索など) を指定できます。1 つの条件のみを指定する簡易検索もあります。

検索条件の作成

検索条件を作成して実行するには、最初に IQuery オブジェクトを作成する必要があります。このオブジェクトは、他の Agile API オブジェクトと同様に、IAgileSession.createObject() メソッドを使用して作成します。

最も簡単な形式の場合、検索条件を作成する createObject() メソッドとともに渡すパラメータは、IQuery オブジェクト タイプと検索に使用する検索クラスです。次の例では、検索クラスはアイテム クラスです。

例: 検索条件の作成

```
try {
  IQuery query =
(IQuery)session.createObject(IQuery.OBJECT_TYPE,

  ItemConstants.CLASS_ITEM_BASE_CLASS);
  query.setCaseSensitive(false);
  query.setCriteria("[Title Block.Number] starts with 'P'");
  ITable results = query.execute();
} catch (APIException ex) {
   System.out.println(ex);
}
```

createObject() メソッドで指定する検索クラスには、そのサブクラスのオブジェクトもすべて含まれます。たとえば、アイテム クラスのオブジェクトを検索すると、その結果には部品とドキュメントが含まれます。変更クラスのオブジェクトを検索すると、その結果には、すべての変更サブクラス (期限付き設計変更、ECO、ECR、MCO、PCO、SCO および出荷停止) のオブジェクトが含まれます。特定のサブクラスのみを検索する場合は、そのクラスを明示的に指定する必要があります。次の例は、Foobar という名前のサブクラスのオブジェクトを検索する検索条件の作成方法を示しています。

例: 検索クラスの指定

```
IAdmin admin = m_session.getAdminInstance();
IAgileClass cls = admin.getAgileClass("Foobar");
IQuery query =
(IQuery)m_session.createObject(IQuery.OBJECT_TYPE, cls);
```

フォルダへの検索条件の保存

IQuery.setName() メソッドを使用して検索条件に名前を付けた後は、その検索条件をフォルダに追加できます。次の例は、検索条件に名前を付けて [パーソナル検索] フォルダに追加する方法を示しています。検索条件は、後でフォルダから取得して再利用できます。

例: 検索条件に名前を付けてフォルダに追加

```
try {
  IQuery query = (IQuery)session.createObject(IQuery.OBJECT_TYPE,
    ItemConstants.CLASS_ITEM_BASE_CLASS);
  query.setCaseSensitive(false);
  query.setCriteria("[Title Block.Number] starts with 'P'");
  query.setName("Items Whose Number Starts with P");
  IFolder folder = (IFolder)m_session.getObject(IFolder.OBJECT_TYPE,
"/Personal Searches");
  folder.addChild(query);
} catch (APIException ex) {
  System.out.println(ex);
}
```

検索条件は、IQuery.saveAs() メソッドを使用して名前を付けてフォルダに保存することもできます。

例: IQuery.saveAs() を使用して検索条件をフォルダに保存

```
try {
  IQuery query =
(IQuery)session.createObject(IQuery.OBJECT_TYPE,
    ItemConstants.CLASS_ITEM_BASE_CLASS);
  query.setCaseSensitive(false);
  query.setCriteria("[Title Block.Number] starts with
'P'");
  IFolder folder =
(IFolder)m_session.getObject(IFolder.OBJECT_TYPE,
"/Personal Searches");
  query.saveAs("Items Whose Number Starts with P", folder);
} catch (APIException ex) {
  System.out.println(ex);
}
```

注意 フォルダに明示的に保存せずに作成した検索条件は、一時的な検索条件とみなされます。すべての一時的な検索条件は、ユーザー セッションを閉じる際に Agile アプリケーション サーバによって削除されます。

パラメータ検索の作成

検索条件を指定するときは、パーセント記号 (%) の後ろに数字を指定して、パラメータ プレースホルダを示すことができます。このパラメータ値は、後で (通常は実行時に) 指定できます。パラメータには、検索条件に値を渡す便利な方法が用意されており、これによって、時間と余分なコーディング労力を削減できます。パラメータ検索は、保存して後で再利用できます。

注意 右オペランドの検索パラメータは、検索条件の各演算子に対して 1 つのプレースホルダをサポートしています。このため、検索条件に 3 つの検索条件演算子がある場合、検索条件には、3 つの演算子に対応する合計 3 つのプレースホルダを指定できます。between と not between の検索条件演算子には違いがあります。たとえば、`[2091] contains none of (%0,%1);` は使用できませんが、`[2091] contains none of (%0);` は使用できます。また、`query.execute(new Object[]{new Object[]{"B", "C"}});` も使用できません。

検索パラメータのインデックスは、0 が基準となります。パラメータには、0、1、2 のように番号が設定されます。パラメータは、常に昇順で指定する必要があります。

次の例は、IQuery.execute(Object[]) メソッドを使用して値が指定された 3 つのパラメータを持つ検索条件を示しています。

例: IQuery.execute(Object[]) を使用するパラメータ検索

```
public ITable runParameterizedQuery() throws Exception {
  String condition = "[Title Block.Number] starts with %0
and" +
                  "[Title Block.Part Category] == %1
and" +
                  "[Title Block.Description] contains %2";
  IQuery query = (IQuery)
m_session.createObject(IQuery.OBJECT_TYPE,
ItemConstants.CLASS_PART);
  query.setCriteria(condition);
  ITable table = query.execute(new Object[] {"1", "Electrical",
"Resistor"});
  return table;
}
```

検索パラメータは、次の例のように、IQuery.setParams() メソッドを使用して指定することもできます。検索パラメータ値は、IQuery.execute() を呼び出す前に設定してください。呼び出す前に設定しないと、検索を実行したときに前回のパラメータ値が使用されます。パラメータが設定されていない場合は、ヌル値が使用されます。同様に、検索にパラメータを渡さない場合、IQuery.getParams() メソッドはヌルを返します。

例: IQuery.setParams() を使用するパラメータ検索

```
public ITable runParameterizedQuery() throws Exception {
  String condition = "[Title Block.Number] starts with %0
and" +
                "[Title Block.Part Category] == %1
and" +
                "[Title Block.Description] contains %2";
  IQuery query = (IQuery)
m_session.createObject(IQuery.OBJECT_TYPE,
ItemConstants.CLASS_PART);
  query.setCriteria(condition);
  query.setParams(new Object[] {"1", "Electrical", "Resistor"});
  ITable table = query.execute();
  return table;
}
```

複数のパラメータが 1 つの特定の値のみを参照している場合は、パラメータ検索を引用符で囲まないでください。これは、これらの引用符によって、検索条件に対して一連の値 (複数の要素) が作成されるためです。次の例は、パラメータ検索の作成における引用符の正しい使用方法を示しています。

例: パラメータ検索での引用符の正しい使用方法

```
String criteria = "[NUMBER] == %0";
query.execute(new Object[]{"P1000-02"});
String criteria = "[P2.LIST01] in %0";
query.execute(new Object[]{new Object[]{"A1", "B2"}});
```

検索条件作成時の検索属性の指定

検索条件を作成するときは、検索クラスのみを渡すかわりに、より高度な形式の createObject() メソッドを使用して、1 つ以上の属性値を含む Map オブジェクトを渡すことができます。QueryConstants クラスには、検索条件の作成時に設定できる検索属性に対する定数がいくつか含まれています。これらの定数は、仮想属性です。この属性は、Agile PLM データベースには存在していませんが、実行時に検索条件を定義する際に使用できます。

属性の定数	説明
ATT_CRITERIA_CLASS	検索クラス
ATT_CRITERIA_PARAM	(パラメータ検索条件に対する) 検索条件パラメータ値
ATT_CRITERIA_STRING	検索条件文字列
ATT_PARENT_FOLDER	検索条件が格納されている親フォルダ
ATT_QUERY_NAME	検索名

次の例は、検索条件の作成時に検索クラス、検索条件、親フォルダおよび検索名を設定する方法を示しています。

例: 検索条件作成時の検索属性の指定

```
try {
String condition = "[Title Block.Number] starts with 'P'";
  IFolder parent =
(IFolder)m_session.getObject(IFolder.OBJECT_TYPE,
"/Personal Searches");
  HashMap map = new HashMap();
  map.put(QueryConstants.ATT_CRITERIA_CLASS,
ItemConstants.CLASS_ITEM_BASE_CLASS);
  map.put(QueryConstants.ATT_CRITERIA_STRING, condition);
  map.put(QueryConstants.ATT_PARENT_FOLDER, parent);
  map.put(QueryConstants.ATT_QUERY_NAME, "Part Numbers
Starting with P");
  IQuery query =
(IQuery)m_session.createObject(IQuery.OBJECT_TYPE, map);
  ITable results = query.execute();
} catch (APIException ex) {
  System.out.println(ex);
}
```

ワークフロー検索の指定

注意 次の機能は、SDK の現在のリリースではサポートされていません。

Agile SDK では、ワークフロー関連の検索条件を指定して、ワークフロー検索を開始できます。その際は、一連のワークフロー属性すべてを括弧で囲みます。

次の例は、ワークフロー検索を指定する方法を示しています。この例で、すべてのワークフロー属性が括弧で囲まれていることに注目してください。

例: 検索条件作成時のワークフロー属性の指定

```
private void testWorkflowQuery(IAgileSession session)
throws Exception {
  IQuery query =
(IQuery)session.createObject(IQuery.OBJECT_TYPE,
ChangeConstants.CLASS_ECO);
  String criteria = "([Workflow.Workflow Status] equal to
'Default Change Orders.CCB'"
  + " and [Workflow.Approver] contains ([" +
UserConstants.ATT_GENERAL_INFO_USER_ID + "] == 'yvonnec')"+ "
and [1047] starts with 'C'";
  query.setCriteria(criteria);
  ITable result = query.execute();
  System.out.println(result.size());
}
```

検索条件の指定

検索から返されるオブジェクトの数は、検索条件を指定して絞り込むことができます。検索条件を指定しないと、指定した検索クラスのすべてのオブジェクトに対する参照が返されます。返されるデータ量が膨大になると、パフォーマンスが低下する可能性があるため、検索条件は、できるかぎり限定することをお薦めします。

検索条件の指定には、3 つの異なる `setCriteria()` メソッドを使用できます。

- `setCriteria(ICriteria criteria)` - [条件] 管理ノードに格納されているデータから検索条件を設定します。[条件] 管理ノードにはワークフローに対する再利用の条件を定義しますが、このノードは、通常の検索条件として使用することもできます。

注意 ワークフロー検索は、リリース 9.2.2 ではサポートされていませんでした。

- `setCriteria(java.lang.String criteria)` - 指定の `String` から検索条件を設定します。
- `setCriteria(java.lang.String criteria, java.lang.Object[] params)` - 1 つ以上のパラメータを参照する指定の `String` から検索条件を設定します。

最初に `setCriteria()` メソッド (パラメータとして `ICriteria` オブジェクトが使用される) を使用しないかぎり、検索条件は `String` として解析されます。

検索条件

Agile API には、検索条件を指定するための簡単で強力なクエリ言語が用意されています。クエリ言語は、フィルタ、条件、属性参照、関係演算子、論理演算子およびその他の要素に適した構文を定義します。

検索条件は、1 つ以上の検索条件で構成されています。各検索条件には、次の要素が含まれます。

1. 左オペランド - 左オペランドは、[Title Block.Number] など、常に大括弧で囲まれた属性です。属性は、属性名 (完全修飾名または略式名称) または属性 ID 番号で指定できます。属性は、検索で使用するオブジェクトの特性を指定します。
2. 関係演算子 - 関係演算子は、「equal to」、「not equal to」など、指定した値と属性との関係を定義します。
3. 右オペランド - 左オペランドに指定した属性に対する照合値です。右オペランドは、単一の定数式または一連の定数式になります。一連の定数式は、関係演算子が「between」、「not between」、「in」または「not in」の場合に必要です。

次に、検索条件の例を示します。

```
[Title Block.Description] == 'Computer'
```

次に、右オペランドが一連の定数式である別の例を示します。

```
[Page Two.Numeric01] between ('1000', '2000')
```

クエリ言語のキーワード

検索条件を指定するときは、適切なキーワードを使用してステートメントを作成する必要があります。使用できるキーワードは、次のとおりです。

and	does	less	or	to
asc	equal	like	order	union
between	from	minus	phrase	where
by	greater	none	select	with
contain	in	not	start	word
contains	intersect	null	starts	words
desc	is	of	than	

クエリ言語のキーワードはローカライズされていません。ロケールに関係なく英語のキーワードを使用する必要があります。キーワードには小文字または大文字を使用できます。Agile API 検索条件には、キーワードの他に、$USER (現在のユーザー) や $TODAY (今日の日付) などの Agile PLM 変数を使用できます。

注意 in 演算子は、検索条件のマルチリストではサポートされていません。

検索属性の指定

検索できる各 Agile PLM オブジェクトには、一連の属性も関連付けられています。これはオブジェクト固有の特性です。これらの属性は、検索条件の左オペランドとして使用できます。検索条件の右オペランドには、属性の値を指定します。

検索属性は、[Title Block.Number] のように大括弧で囲む必要があります。大括弧が必要な理由は、多くの属性名に空白が使用されているためです。検索属性を大括弧で囲まないと、検索に失敗します。

検索属性は、次のように指定できます。

属性参照	例
属性 ID 番号	[1001]
完全修飾属性名	[Title Block.Number]
略式属性名	[Number]

注意 属性名は変更される場合があるため、属性は ID 番号または定数で参照することをお薦めします。ただし、この章の多くの例では、読み易さの観点から、属性を名前で参照しています。属性を名前で参照する場合は、略式名称ではなく完全修飾属性名を使用してください。略式属性名は一意性が保証されないため、検索に失敗したり、予期しない結果となる可能性があります。

属性名は、使用する形式 (長い名前または短い名前) に関係なく、大文字と小文字の区別はありません。たとえば、[Title Block.Number] と [TITLE BLOCK.NUMBER] は両方とも使用できます。また、属性名はローカライズされています。Agile PLM 属性の名前は、Agile アプリケーション サーバのロケールによって異なります。異なるロケールのサーバで使用される検索条件を作成する場合は、名前のかわりに、ID 番号 (または同等の定数) で属性を参照する必要があります。

引用符や円記号などの特殊文字が属性名に使用されている場合は、円記号 (**¥**) をエスケープ文字として使用して、これらの文字を入力できます。たとえば、文字列に引用符を指定するには、**¥'** と入力します。円記号を記述する場合は、2 つの円記号 (**¥¥**) を入力します。属性名に大括弧が使用されている場合は、名前全体を引用符で囲みます。

```
['Page Two.Unit of Measure [g or oz]']
```

属性を指定する方法は他にもありますが、直感性は低くなります。たとえば、setCriteria() メソッドのパラメータを使用して IAttribute 参照で渡すことができます。次の例では、「%0」で Object 配列パラメータの属性を参照します。

```
query.setCriteria("[%0] == 'Computer'", new Object[] { attr });
```

文字列 (String) の連結を使用して属性定数を参照することもできます。

```
query.setCriteria("[" +
ItemConstants.ATT_TITLE_BLOCK_DESCRIPTION + "] ==
'Computer'");
```

検索可能属性の取得

検索条件の検索可能属性は、指定した検索クラスまたはサブクラスによって異なります。ただし、サブクラスの検索可能属性は、その親クラスの検索可能属性とは大きく異なる場合があります。

データベース側の理由から、すべての属性が検索可能なわけではありません。通常、選定された数個の [ページ 1] 属性 (つまり、[タイトル ページ]、[カバー ページ] および [一般情報]) が、各クラスの検索可能属性です。

Java クライアントに表示するタブが設定されていない場合でも、そのタブの属性は、Agile SDK で検索できます。ただし、タブ名に対応するテーブル名を検索する必要があります。

注意 IQuery の設定にはテーブル名を使用するため、Agile Java クライアントに指定されているタブ名が Agile 管理者によって変更されても問題はありません。タブ名の変更が、SDK テーブル名に影響を与えることはありません。

検索条件の検索可能属性を見つけるには、IQuery.getSearchableAttributes() メソッドを使用します。

注意 属性が検索可能でない場合でも、その属性が検索結果の列として含まれる場合があります。詳細は、53 ページの「検索条件の結果属性の設定」を参照してください。

関係演算子の使用

次の表に、Agile API クエリ言語でサポートされている関係演算子を示します。

演算子 (英語)	表記法	説明
equal to	==	指定の値と完全に一致する内容のみを検索します。
not equal to	!=	指定の値と完全に一致する内容以外の値を検索します。
greater than	>	指定の値より大きい値を検索します。
greater than or equal to	>=	指定の値以上の値を検索します。
less than	<	指定の値未満の値を検索します。
less than or equal to	<=	指定の値以下の値を検索します。
contains、contains all		指定の値を含む値を検索します。
does not contain、 does not contain all		指定の値を含まない値を検索します。
contains any		指定の値を含む値を検索します。
does not contain any		指定の値を含まない値を検索します。
contains none of		指定の値すべてを含まない値を検索します。
does not contain none of		does not contain any と同様に機能します。
starts with		指定の値が先頭にある値を検索します。
does not start with		指定の値で始まらない値を検索します。
is null		選択した属性に値が含まれていないオブジェクトを検索します。
is not null		選択した属性に値が含まれているオブジェクトを検索します。
like		単一の文字または文字列と一致するオブジェクトを検出するワイルドカード検索を実行します。
not like		単一の文字または文字列と一致しないオブジェクトを検出するワイルドカード検索を実行します。
between		指定の値の範囲に含まれるオブジェクトを検索します。
not between		指定の値の範囲に含まれないオブジェクトを検索します。
in		指定の値のいずれかと一致するオブジェクトを検索します。
not in		指定の値のすべてに一致しないオブジェクトを検索します。
contains phrase		指定の語句が含まれているオブジェクトを検索します。
contains all words		指定の単語のすべてが含まれているオブジェクトを検索します。
contains any word		指定の単語のいずれかが含まれているオブジェクトを検索します。
contains none of		指定の単語すべてが含まれていないオブジェクトを検索します。

関係演算子は、ローカライズされていません。ロケールに関係なく英語のキーワードを使用する必要があります。関係演算子は、他のクエリ言語キーワードと同様に、小文字または大文字で使用できます。

Unicode エスケープ シーケンスの使用

Agile SDK クエリ言語は、Unicode エスケープ シーケンスをサポートしています。クエリ文字列で Unicode エスケープ シーケンスを使用するのは、主に変換不可能な文字セット、または外国のローカル文字セットを検索する場合です。Unicode 文字は、Unicode エスケープ シーケンス `¥uxxxx` で表されます (`xxxx` は 4 桁の 16 進数)。

たとえば、Unicode 3458 のアイテムを検索する場合は、次の検索条件を使用します。

```
Select * from [Items] where [Description] contains '¥u3458'
```

マルチリストの場合は、「contains」のかわりに使用する別の検索演算子があります。

Between、Not Between、In および Not In 演算子の使用

between、not between、in および not in 関係演算子は、Agile Web クライアントなどの Agile PLM クライアントでは直接サポートされていません。関係演算子には、equal to、not equal to、greater than or equal to または less than or equal to 演算子を一連の値で指定する簡便な方法があります。

短い形式	同等の長い形式
`[Number] between ('1','6')`	`[Number] >= '1' and [Number] <= '6'`
`[Number] not between ('1','6')`	`[Number] < '1' and [Number] > '6'`
`[Number] in ('1','2','3','4',5','6')`	`[Number] == '1' or [Number] == '2' or [Number] == '3' or [Number] == '4' or [Number] == '5' or [Number] == '6'`
`[Number] not in ('1','2','3','4','5','6')`	`[Number]!= '1' and [Number] != '2' and [Number] != '3' and [Number] != '4' and [Number] != '5' and [Number] != '6'`

前述の表に記載されているように、between、not between、in および not in 関係演算子を使用するときは、一連の値を構成する個々の値を引用符で囲み、カンマで区切る必要があります。次に、between 演算子と in 演算子を使用する条件の例を示します。

```
[Title Block.Number] in ('1000-02', '1234-01', '4567-89')
[Title Block.Effectivity Date] between ('01/01/2001', '01/01/2002')
[Page Two.Numeric01] between ('1000', '2000')
```

注意 関係演算子 `any`、`all`、`none of` および `not all` は、SDK ではサポートされていません。

ネストされた条件を使用したオブジェクト リスト tohere の値の検索

Agile PLM のいくつかのリストには、Agile PLM ユーザーなどのビジネス オブジェクトが含まれています。これらのリストにあるオブジェクトを検索するには、ネストされた検索条件を指定できます。ネストされた条件は、括弧で囲み、各条件を論理演算子 AND (&&) または OR (||) で区切ります。ネストされた条件を区切るには、カンマを使用することもできます。このカンマは論理 OR と等価です。

次の条件は、Christopher という姓または Nolan という名のユーザーを検索します。

```
[Page Two.Create User] in ([General Info.First Name] ==
'Christopher',
[General Info.Last Name] == 'Nolan')
```

次の条件は、Christopher という姓で、かつ Nolan という名のユーザーを検索します。

```
[Page Two.Create User] in ([General Info.First Name] ==
'Christopher' &&
[General Info.Last Name] == 'Nolan')
```

パラメータ検索は、ネストされた検索条件ではサポートされません。また、検索パラメータの 1 つのプレースホルダに対する複数の値は、二次元配列で指定する必要があります。次の例を参照してください。

例: ネストされた検索条件における適切なパラメータ検索と不適切なパラメータ検索

- 次のネストされた検索条件に指定されているパラメータ検索は、正しく実行できません。

  ```
  [Page Two.User1] in ([General Info.First Name] == %0)
  ```

- ただし、プレースホルダではなく文字列値として明示的に指定されている場合は、正常に実行されます。

  ```
  [Page Two.User1] in ([General Info.First Name] ==
  'Christopher')
  ```

添付ファイル内の単語または語句の検索

2 つの特別な属性 `[Attachments.File Document Text]` および `[Files.Document Text]` は、Agile ファイル管理サーバに格納されているファイルの内容にインデックスを付ける際に使用されます。Oracle でデータベースをホスティングしている場合は、添付ファイル内の単語または語句を検索する機能を利用できます。このいずれかの属性を使用する検索条件を作成するときは、さらに 4 つの関係演算子を使用できます。

- `contains phrase`
- `contains all words`
- `contains any word`
- `contains none of`

注意 これらの関係演算子は IBM 社の DB2 と Microsoft 社の SQL Server ではサポートされていません。

次の表は、添付ファイル内の単語または語句を検索する検索条件を示しています。

検索条件	検索対象
[Attachments.File Document Text] **contains phrase** 'adding new materials'	添付ファイルのいずれかに語句「adding new materials」が含まれているオブジェクト。
all [Attachments.File Document Text] **contains all words** 'adding new materials'	添付ファイルすべてに単語「adding」、「new」および「materials」が含まれているオブジェクト。
none of [Attachments.File Document Text] **contains any word** 'containers BOM return output'	いずれの添付ファイルにも単語「containers」、「BOM」、「return」または「output」が含まれていないオブジェクト。
[Attachments.File Document Text] **contains none of** 'containers BOM output'	いずれの添付ファイルにも単語「containers」、「BOM」および「output」が含まれていないオブジェクト。

検索条件の日付の書式設定

いくつかのタイプの検索条件には日付値が必要です。日付を文字列として渡す場合は、IAgileSession.setDateFormats() メソッドを使用して日付フォーマットを指定します。setDateFormats() メソッドは、setValue() メソッドで指定したすべての Agile API 値にも適用されます。

注意 setDateFormats() メソッドを使用して日付フォーマットを明示的に設定しなかった場合は、Agile PLM システムのユーザーの日付フォーマットが使用されます。Agile Web クライアントで日付フォーマットを調べるには、[設定] > [ユーザー プロファイル] の順に選択し、[プリファレンス] タブをクリックします。

例: 検索条件の日付フォーマットの設定

```
m_session.setDateFormats(new DateFormat[] {new
SimpleDateFormat("MM/dd/yyyy")});
query.setCriteria("[Title Block.Rev Release Date] between” +
"('9/2/2001', '9/2/2003')");
query.setCriteria("[Title Block.Rev Release Date]
between (%0,%1)", new String[] {"9/2/2001", "9/2/2003"} );
```

setCriteria(String criteria, Object[] params) メソッドを使用する場合は、Date オブジェクトをメソッドに対するパラメータとして渡すことができます。

例: Date オブジェクトを setCriteria() のパラメータとして渡す場合

```
DateFormat df = new SimpleDateFormat("MM/dd/yyyy");
query.setCriteria("[Title Block.Rev Release Date] between (%0,%1)",
new Object[] { df.parse("9/2/2001"), df.parse("9/2/2003") });
```

論理演算子の使用

論理演算子を使用すると、複数の検索条件を複雑なフィルタに組み合わせることができます。一連の検索条件に 2 つ以上の条件を定義する場合は、各条件の関係を and または or で定義します。

- **and** を使用すると、両方の条件が満たされる必要があるため、検索条件が絞り込まれます。検索結果の各アイテムは、必ず両方の条件を満たしています。and 論理演算子は、2 つのアンパサンド (&&) を使用して指定することもできます。
- **or** を使用すると、どちらかの条件を満たすオブジェクトが検索されるため、検索条件が拡大します。検索結果の各アイテムは必ず一方の条件に一致しますが、両方の条件に一致する場合もあります。or 論理演算子は、2 つの垂直バー (||) を使用して指定することもできます。

論理演算子では大文字と小文字が区別されません。たとえば、and と AND は両方とも使用できます。

次の検索条件は、部品カテゴリが [電気系] で、かつライフサイクル フェーズが [停止] である両方の条件を満たす部品を検索します。

```
[Title Block.Part Category] == 'Electrical' and
[Title Block.Lifecycle Phase] == 'Inactive'
```

「and」を「or」に置き換えると、部品カテゴリが [電気系] か、ライフサイクル フェーズが [停止] か、いずれかの条件を満たす部品が検索されます。その結果、より多くの部品が条件を満たすことになります。

```
[Title Block.Part Category] == 'Electrical' or
[Title Block.Lifecycle Phase] == 'Inactive'
```

注意 Agile API には、使用箇所検索用の 3 種類の集合演算子が用意されています。詳細は、62 ページの「使用箇所検索条件の作成」を参照してください。
論理演算子 (使用箇所検索の集合演算子を含む) はローカライズされていません。ロケールに関係なく英語のキーワードを使用する必要があります。

Like 演算子でのワイルドカード文字の使用

like 演算子を使用して検索条件を定義する場合は、2 つのワイルドカード文字を使用できます。使用できるワイルドカード文字は、アスタリスク (*) および疑問符 (?) です。アスタリスクは任意の長さの文字列に相当します。したがって、***at** では、「cat」、「splat」および「big hat」が検索されます。次に例を示します。

```
[Title Block.Description] like '*book*'
```

この例では、textbook、bookstore、books など、「book」という単語を含んだオブジェクトがすべて返されます。

疑問符は任意の 1 文字に相当します。したがって、**?at** では、「hat」、「cat」および「fat」が検索され、「splat」は該当しません。次に例を示します。

```
[Title Block.Description] like '?al*'
```

この例では、tall、wall、mall、calendar など、任意の 1 文字の後に「al」が続く単語が返されます。

検索条件での括弧の使用

使用箇所検索の集合演算子は、次の表に示すように、and および or 論理演算子より優先度が高く設定されています。

優先度	演算子
1	▫ union ▫ intersection ▫ minus
2	▫ and ▫ or

したがって、union、intersection および minus 演算子で結合された検索条件は、and または or で結合された条件より前に評価されます。

検索条件に使用箇所検索の集合演算子 (union、intersect または minus) を使用する場合は、括弧を使用して条件の評価順序を変更できます。検索条件に and または or 論理演算子のみが使用される場合は、条件の評価結果が変化しないため、括弧は不要です。

次の 2 つの条件には、同じ検索条件が含まれていますが、異なる位置に括弧が使用されているため、検索結果は異なります。

```
(([Title Block.Part Category] == 'Electrical' and
 [Title Block.Description] contains 'Resistor') union
([Title Block.Description] contains '400' and
 [Title Block.Product Line(s)] contains 'Taurus')
 [Title Block.Part Category] == 'Electrical' and
([Title Block.Description] contains 'Resistor' union
 [Title Block.Description] contains '400') and
 [Title Block.Product Line(s)] contains 'Taurus'
```

検索条件での SQL 構文の使用

Agile API では、標準的なクエリ言語に加え、検索条件に SQL 的な構文もサポートされています。SQL ステートメントの記述方法を理解している場合は、この拡張されたクエリ言語のほうが、より簡単に使用でき、柔軟性があって効果を発揮できる可能性があります。この記述方法では、検索結果属性、検索クラス、検索条件および並べ替え列の仕様が 1 つの操作に組み込まれます。

次に、構文の簡単な例を示します。

- 検索結果属性: `SELECT [Title Block.Number], [Title Block.Description]`
- 検索クラス: `FROM [Items]`
- 検索条件: `WHERE [Title Block.Number] starts with 'P'`
- 並べ替え列: `ORDER BY 1 asc`

可読性を考慮して、SELECT や FROM などの SQL キーワードはすべて大文字で入力し、ステートメントの各部は独立した行に記載することをお薦めします。これは慣例であり要件ではありません。SQL キーワードでは大文字と小文字が区別されず、クエリ文字列をすべて 1 行に記述することもできます。

SQL 構文の利点を示す最良の方法は、検索条件に Agile API の標準的な検索構文を使用する検索条件と、SQL 構文を使用する検索条件のコードを比較することです。次の例は、Agile API の標準的な検索構文を使用して作成した検索条件を示しています。

例: Agile API の標準的な検索構文を使用した検索条件

```
try {
  IQuery query = (IQuery)m_session.createObject(IQuery.OBJECT_TYPE,
"Items");
  query.setCriteria("[Page Two.Nummeric01] between (1000, 2000)");
  //Set result attributes
  String[] attrs = { "Title Block.Number", "Title Block.Description",
    "Title Block.Lifecycle Phase" };
  query.setResultAttributes(attrs);
  //Run the query
  ITable results = query.execute();
} catch (APIException ex) {
  System.out.println(ex);
}
```

次の例は、SQL 構文で書き換えた同様の検索条件を示しています。記載されているコード例は少ない行数ですが、SQL を理解しているユーザーは特に、Agile API の検索構文より可読性が高いことを確認できます。

例: SQL 構文を使用した検索条件

```
try {
  IQuery query = (IQuery)m_session.createObject(IQuery.OBJECT_TYPE,
    "SELECT " +
      "[Title Block.Number],[Title Block.Description], " +
      "[Title Block.Lifecycle Phase] " +
    "FROM " +
      "[Items] " +
    "WHERE " +
      "[Title Block.Number] between (1000, 2000)"
  );
  //Run the query
  ITable results = query.execute();
} catch (APIException ex) {
  System.out.println(ex);
}
```

次の例は、ATT_CRITERIA_STRING 検索属性を使用して検索条件を指定する SQL 構文で記述された検索条件を示しています。検索属性の使用方法の詳細は、40 ページの「検索条件作成時の検索属性の指定」を参照してください。

例: SQL 構文を使用した検索属性の指定

```
try {
  String statement =
    "SELECT " +
      "[Title Block.Number], [Title Block.Description] " +
    "FROM " +
      "[Items] " +
    "WHERE " +
      "[Title Block.Description] like %0";
  HashMap map = new HashMap();
  map.put(QueryConstants.ATT_CRITERIA_STRING, statement);
  map.put(QueryConstants.ATT_CRITERIA_PARAM, new Object[]
{ "Comp*" } );
  IQuery query = (IQuery)m_session.createObject(IQuery.OBJECT_TYPE,
map);
  ITable results = query.execute();
} catch (APIException ex) {
  System.out.println(ex);
}
```

注意 検索条件の FROM 部分には、検索クラスが指定されていることに注意してください。検索クラスの指定にも ATT_CRITERIA_CLASS 属性を使用した場合は、SQL 検索条件に指定されている検索クラスのほうが優先されます。

IQuery.setCriteria() メソッドを使用すると、検索条件を SQL 構文で指定できますが、IQuery.getCriteria() メソッドでは、常に Agile API の標準的な検索構文で検索条件が返されます。

SQL ワイルドカードの使用

SQL 構文を使用する検索条件では、アスタリスク (*) と疑問符 (?) の両方のワイルドカードを使用できます。Agile API の標準的なクエリ言語と同様に、アスタリスクは任意の文字列に相当し、疑問符は任意の 1 文字に相当します。ワイルドカードは、SELECT ステートメント (指定された検索結果属性) と WHERE ステートメント (検索条件) で使用できます。たとえば、「SELECT *」は、有効な検索結果属性すべてを指定します。

SQL 構文の使用による検索結果の並べ替え

Agile API の標準的なクエリ言語のかわりに SQL 構文を使用して検索条件を指定する場合は、ORDER BY キーワードを使用して検索結果を並べ替えることができます。SELECT ステートメントに指定されている属性に基づいて、結果を昇順または降順に並べ替えることができます。

ORDER BY ステートメントでは、SELECT ステートメントに記載されている順に 1 を基準とする番号で属性を参照します。昇順または降順に並べ替えるかを指定するには、属性番号の後に asc または desc を入力します。asc または desc を省略すると、昇順がデフォルトで使用されます。

例	説明
ORDER BY 1	最初の SELECT 属性で昇順にソートされます (デフォルト)。
ORDER BY 2 desc	2 番目の SELECT 属性で降順にソートされます。
ORDER BY 1 asc, 3 desc	最初の SELECT 属性で昇順にソートされ、さらに 3 番目の SELECT 属性で降順にソートされます。

SELECT ステートメントに指定されていない属性を使用して検索結果を並べ替えることはできません。また、「SELECT *」を使用して有効なすべての結果属性を選択する場合は、属性の順序が明示されないため、結果を並べ替えることはできません。

次の例では、SELECT ステートメントの 1 番目と 3 番目の属性である [Title Block.Number] と [Title Block.Sites] に従って結果を昇順に並べ替えています。

例: SQL 構文を使用した検索結果の並べ替え

```
IQuery query = (IQuery)m_session.createObject(IQuery.OBJECT_TYPE,
  "SELECT " +
    "[Title Block.Number],[Title Block.Description], " +
    "[Title Block.Sites],[Title Block.Lifecycle Phase] " +
  "FROM " +
    "[Items] " +
  "WHERE " +
    "[Title Block.Number] between (1000, 2000)" +
  "ORDER BY " +
    "1, 3"
);
```

検索条件の結果属性の設定

検索を実行すると、複数の出力フィールドが返されます。これらは結果属性とも呼ばれます。デフォルトでは、各検索クラスに対して数個の結果属性があります。結果属性は、IQuery.setResultAttributes() メソッドを使用して追加または削除できます。

次の表に、Agile PLM に事前定義されている各クラスに対するデフォルトの検索結果属性を示します。

検索クラス	デフォルトの結果属性
変更 設計変更 ECO 設計変更依頼 ECR 期限付き設計変更 期限付き設計変更 製造元変更 MCO 価格変更 PCO 拠点毎変更 SCO 出荷停止 出荷停止	カバー ページ.変更タイプ カバー ページ.番号 カバー ページ.説明 カバー ページ.ステータス カバー ページ.ワークフロー
顧客 顧客 顧客	一般情報.顧客タイプ 一般情報.顧客番号 一般情報.顧客名 一般情報.説明 一般情報.ライフサイクル フェーズ

検索クラス	デフォルトの結果属性
デクラレーション 均質材のデクラレーション 均質材のデクラレーション IPC 1752-1 デクラレーション IPC 1752-1 デクラレーション IPC 1752-2 デクラレーション IPC 1752-2 デクラレーション JGPSSI デクラレーション JGPSSI デクラレーション 部品のデクラレーション 部品のデクラレーション サブスタンスのデクラレーション サブスタンスのデクラレーション 適合のサプライヤ デクラレーション 適合のサプライヤ デクラレーション	カバー ページ.名前 カバー ページ.説明 カバー ページ.サプライヤ カバー ページ.ステータス カバー ページ.ワークフロー カバー ページ.適合性管理者 カバー ページ.締切日 カバー ページ.デクラレーション タイプ
ディスカッション ディスカッション ディスカッション	カバー ページ.件名 カバー ページ.ステータス カバー ページ.優先度 カバー ページ.タイプ
ファイル フォルダ ファイル フォルダ ファイル フォルダ	タイトル ブロック.タイプ タイトル ブロック.番号 タイトル ブロック.説明 タイトル ブロック.ライフサイクル フェーズ
アイテム 部品 部品 ドキュメント ドキュメント	タイトル ブロック.アイテム タイプ タイトル ブロック.番号 タイトル ブロック.説明 タイトル ブロック.ライフサイクル フェーズ タイトル ブロック.リビジョン

検索クラス	デフォルトの結果属性
製造元 製造元 製造元	一般情報.名前 一般情報.市町村区 一般情報.都道府県 一般情報.ライフサイクル フェーズ 一般情報.URL
製造元部品 製造元部品 製造元部品	一般情報.製造元部品番号 一般情報.製造元名 一般情報.説明 一般情報.ライフサイクル フェーズ
パッケージ パッケージ パッケージ	カバー ページ.パッケージ番号 カバー ページ.説明 カバー ページ.アセンブリ番号 カバー ページ.ステータス カバー ページ.ワークフロー
部品グループ 部品グループ 部品分類 部品ファミリ	一般情報.名前 一般情報.説明 一般情報.ライフサイクル フェーズ 一般情報.部品分類タイプ 一般情報.全体適合性
価格 公表価格 契約 公表価格 見積履歴 見積履歴	一般情報.価格番号 一般情報.説明 一般情報.リビジョン 一般情報.価格タイプ 一般情報.ライフサイクル フェーズ 一般情報.プログラム 一般情報.顧客 一般情報.サプライヤ
製品サービス依頼 不具合レポート NCR 問題レポート 問題レポート	カバー ページ.PSR タイプ カバー ページ.番号 カバー ページ.説明 カバー ページ.ステータス カバー ページ.ワークフロー

検索クラス	デフォルトの結果属性
プログラム アクティビティ プログラム フェーズ タスク ゲート ゲート	一般情報.名前 一般情報.説明 一般情報.ステータス 一般情報.ヘルス 一般情報.所有者 一般情報.ルートの親 一般情報.ワークフロー 一般情報.タイプ
プロジェクト ソーシング プロジェクト ソーシング プロジェクト	一般情報.プロジェクト タイプ 一般情報.番号 一般情報.説明 一般情報.製造元拠点 一般情報.出荷先の場所 一般情報.プログラム 一般情報.顧客 一般情報.ライフサイクル フェーズ
品質変更依頼 是正処置/予防処置 CAPA 検証 検証	カバー ページ.QCR タイプ カバー ページ.QCR 番号 カバー ページ.説明 カバー ページ.ステータス カバー ページ.ワークフロー
見積依頼回答 見積依頼回答 見積依頼回答	カバー ページ.見積依頼番号 カバー ページ.見積依頼説明 カバー ページ.ライフサイクル フェーズ カバー ページ.依頼済み カバー ページ.完了 カバー ページ.締切日

検索クラス	デフォルトの結果属性
RFQ RFQ RFQ	カバー ページ.見積依頼番号 カバー ページ.見積依頼説明 カバー ページ.製造元拠点 カバー ページ.出荷先の場所 カバー ページ.プログラム カバー ページ.顧客 カバー ページ.ライフサイクル フェーズ カバー ページ.見積依頼タイプ
拠点 拠点 拠点	一般情報.名前 一般情報.連絡先 一般情報.電話番号
含有基準 含有基準 含有基準	一般情報.名前 一般情報.説明 一般情報.ライフサイクル フェーズ 一般情報.管轄地域 一般情報.検証タイプ 一般情報.含有基準タイプ
サブスタンス マテリアル マテリアル サブパート サブパート サブスタンス グループ サブスタンス グループ サブスタンス サブスタンス	一般情報.名前 一般情報.説明 一般情報.CAS 番号 一般情報.ライフサイクル フェーズ 一般情報.サブスタンス タイプ
サプライヤ サプライヤ 部品メーカー 受託製造業者 ディストリビュータ メーカー代表者	一般情報.サプライヤ タイプ 一般情報.番号 一般情報.名前 一般情報.説明 一般情報.ステータス

検索クラス	デフォルトの結果属性
転送依頼 コンテンツ転送 CTO 自動転送 ATO	カバー ページ.転送タイプ (「コンテンツ転送作成者名の取得」を参照) カバー ページ.転送番号 カバー ページ.説明 カバー ページ.ステータス カバー ページ.ワークフロー

結果属性の指定

検索を実行して、結果の ITable オブジェクトに期待した属性が含まれていない場合は、結果属性を指定しなかったことが原因です。次の例は、検索条件に結果属性を指定する方法を示しています。

例: 検索結果属性の設定

```
private void setQueryResultColumns(IQuery query) throws APIException
{
  // Get Admin instance
  IAdmin admin = m_session.getAdminInstance();

  // Get the Part class
  IAgileClass cls = admin.getAgileClass("Part");

  // Get some Part attributes, including Page Two and Page Three
attributes
  IAttribute attr1 =
cls.getAttribute(ItemConstants.ATT_TITLE_BLOCK_NUMBER);
  IAttribute attr2 =
cls.getAttribute(ItemConstants.ATT_TITLE_BLOCK_DESCRIPTION);
  IAttribute attr3 =
cls.getAttribute(ItemConstants.ATT_TITLE_BLOCK_LIFECYCLE_PHASE);
  IAttribute attr4 =
cls.getAttribute(ItemConstants.ATT_PAGE_TWO_TEXT01);
  IAttribute attr5 =
cls.getAttribute(ItemConstants.ATT_PAGE_TWO_NUMERIC01);
  IAttribute attr6 =
cls.getAttribute(ItemConstants.ATT_PAGE_THREE_TEXT01);
  // Put the attributes into an array
  IAttribute[] attrs = {attr1, attr2, attr3, attr4, attr5, attr6};

  // Set the result attributes for the query
  query.setResultAttributes(attrs);

}
```

IQuery.setResultAttributes() メソッドでは、String、Integer または IAttribute の配列をサポートしている Object[] 値が、attrs パラメータとして使用されます。したがって、IAttribute オブジェクトの配列を指定するかわりに、属性名 ({"Title Block.Description", "Title Block.Number"} など) または属性 ID 定数の配列を指定することもできます。次の例は、ID 定数を使用して結果属性を指定する方法を示しています。

例: ID 定数の指定による検索結果属性の設定

```
private void setQueryResultColumns(IQuery query) throws APIException
{
  // Put the attribute IDs into an array
  Integer[] attrs = { ItemConstants.ATT_TITLE_BLOCK_NUMBER,
                      ItemConstants.ATT_TITLE_BLOCK_DESCRIPTION,
                      ItemConstants.ATT_TITLE_BLOCK_LIFECYCLE_PHASE,
                      ItemConstants.ATT_PAGE_TWO_TEXT01,
                      ItemConstants.ATT_PAGE_TWO_NUMERIC01,
                      ItemConstants.ATT_PAGE_THREE_TEXT01 };
  // Set the result attributes for the query
  query.setResultAttributes(attrs);
}
```

setResultAttributes() メソッドを使用する場合は、有効な結果属性を指定してください。そうでない場合、setResultAttributes() メソッドは失敗します。検索に使用できる有効な結果属性の配列を取得するには、次の例のように、getResultAttributes() を使用します。

例: 有効な結果属性の配列の取得

```
private IAttribute[] getAllResultAttributes(IQuery query) throws
APIException {
  IAttribute[] attrs = query.getResultAttributes(true);
  return attrs;
}
```

コンテンツ転送作成者名の取得

コンテンツ転送 (CTO) の [カバー ページ] には、コンテンツ転送を作成するユーザーの役割と拠点の割り当てを指定する [作成者] フィールドがあります。ユーザー名を取得する際、このフィールドは直接検索できないため、UserConstants のデータを取得する必要があります。たとえば、次のステートメントはユーザー名を直接取得しようとしていますが、機能しません。

```
QueryString = ("[Cover Page.Originator] equal to
'<Last_name>, <First_name>'");
```

一方、次のステートメントは UserConstants にデータを指定しており、正しく機能します。

```
QueryString = "[Cover Page.Originator] in
(["+UserConstants.ATT_GENERAL_INFO_USER_ID+"]=='<Use
rID>')";
```

または

```
QueryString = "[Cover Page.Originator] in
(["+UserConstants.ATT_GENERAL_INFO_LAST_NAME+"]=='<L
ast_name>'"+
                "&&
["+UserConstants.ATT_GENERAL_INFO_FIRST_NAME+"]=='<F
irst_name>');
```

注意 IItem、IChange など、数に制限がない属性タイプに対する検索条件は、ネストされた形式にする必要があります。Agile のすべてのユーザーを指す [作成者] 属性は、これに該当します。

拠点関連オブジェクトと AML の重複する結果

アイテムまたは変更を検索すると、Agile アプリケーション サーバの製造拠点機能が予想外の結果となる可能性があります。アイテムまたは変更を検索し、結果属性に拠点属性 (アイテムの場合は [タイトル ブロック.拠点]、変更の場合は [カバー ページ.拠点]) を指定すると、検索結果には、オブジェクトに関連付けられている各拠点に対して、重複するオブジェクトが挿入されます。同様に、アイテムを検索し、結果属性に AML 属性 ([製造元.製造元部品番号] など) を指定すると、検索結果には、アイテムの製造元テーブルにリストされている各製造元部品に対して、重複するアイテムが挿入されます。

たとえば、番号 1000-02 の部品が、5 箇所の拠点と関連付けられています。その部品を検索し、結果属性に [タイトル ブロック.拠点] を指定すると、IQuery.execute メソッドから返る結果の ITable オブジェクトには、5 つの行 (1 行ではなく) が含まれます。各行は同じオブジェクト (部品番号 1000-02) を参照しますが、[拠点] セルの値は異なります。ITable.getReferentIterator を使用して、検索結果の参照オブジェクトで処理を繰り返すと、重複するオブジェクトがより明確になります。この例では、同じアイテムが 5 回繰り返されます。

検索結果の使用

検索を実行すると、ITable オブジェクトが返されます。これは、java.Util.Collection を拡張したオブジェクトです。この結果を使用するには、ITable と java.Util.Collection のメソッドを使用できます。たとえば、次のコードは Collection.iterator() メソッドを使用する方法を示しています。

```
Iterator it = query.execute().iterator();
```

ITwoWayIterator インターフェースでは、next() および previous() メソッドを使用して、いずれの方向にも行のリストを移動できます。

```
ITwoWayIterator it = query.execute().getTableIterator();
ITwoWayIterator it = query.execute().getReferentIterator();
```

ITwoWayIterator の使用方法は、76 ページの「テーブル行の繰り返し処理」を参照してください。

検索結果の並べ替え

Agile API の他のテーブルとは異なり、検索結果には、ITable.ISortBy インターフェースを使用して、並べ替えた Iterator を作成することはできません。検索結果を並べ替えるには、SQL 構文を使用して、検索条件とともに ORDER BY ステートメントを指定します。詳細は、50 ページの「検索条件での SQL 構文の使用」を参照してください。

検索結果のデータ タイプ

検索結果テーブルの値は、その属性と同じデータ タイプになります。つまり、属性のデータ タイプが `Integer` の場合は、検索結果テーブルの値も `Integer` になります。

重要 Agile 9.0 SDK では、検索結果テーブルのすべての値が文字列 (String) であったことに注意してください。Agile 9.2 では、これらの値は整数 (Integer) になりました。

大量の検索結果の管理

Agile PLM には、[検索結果の最大表示数] という名前のシステム プリファレンスがあり、これによって検索から返すことができる最大行数の制限を設定します。ただし、このプリファレンスは Agile SDK クライアントには影響を与えません。Agile SDK クライアントから実行する検索では、常にすべての結果が返されます。

ユーザーは、返された `ITable` オブジェクトを使用して検索結果セット全体にアクセスできますが、Agile API では、必要に応じて結果を部分的に取得するように内部的に管理されます。たとえば、特定の検索が 5000 件のレコードを返すとします。ユーザーは、`ITable` インターフェースを使用して 5000 行のいずれかの行にアクセスできます。5000 行の中で実際にメモリにロードされる行数について懸念する必要はありません。

注意 他の Agile PLM クライアント (Agile Web クライアントなど) から実行する検索では、[検索結果の最大表示数] プリファレンスに指定された制限が厳守されます。

検索のパフォーマンス

検索を実行する際の応答時間は、Agile API プログラムの最大のボトルネックとなる可能性があります。パフォーマンスを改善するには、返される結果が最大でも数百件程度の検索条件を作成する必要があります。1000 件を超える結果を返す検索条件では、処理を終了するまでに数分の時間が必要です。このような検索条件では、Agile アプリケーション サーバでの有益な処理手続きが浪費され、場合によっては、すべてのユーザーに対してサーバのパフォーマンスが低下する可能性があります。

使用箇所検索条件の作成

この章の前述のセクションでは、Agile PLM オブジェクト (アイテム、変更など) を検索する検索条件の作成方法について説明しました。さらに、使用箇所検索条件も作成できます。使用箇所検索条件では、検索条件によってオブジェクトの BOM に表示されるアイテムを定義します。使用箇所検索条件を使用すると、特定の部品を使用するアセンブリを検索できます。

使用箇所検索条件のインターフェースは、標準的なオブジェクト検索条件とほぼ同じです。検索クラスがアイテム クラスである場合は、オブジェクト検索条件を使用箇所検索条件に変更できます。

注意 使用箇所検索条件は、アイテム クラスに対してのみ定義されます。

使用箇所検索条件を定義するには、`IQuery.setSearchType()` メソッドを使用します。次の論理演算子 (使用箇所検索の集合演算子とも呼ばれる) を使用すると、グループ化された検索条件同士の関係をより詳細に定義できます。各検索条件に使用できるのは、1 つの論理演算子のみです。

使用箇所検索の集合演算子	説明
intersect	検索条件の 2 つの異なるグループから両方の結果セットに表示されるレコードを作成します。
minus	検索条件の 1 つ目のグループからの結果には存在し、2 つ目のグループからの結果にはないレコードを作成します。
union	検索条件の 2 つのグループからの結果の組み合わせであるレコードを作成します。

注意 使用箇所検索の集合演算子は、他の論理演算子より優先度が高く設定されています。したがって、使用箇所検索の集合演算子で結合された検索条件は、and または or で結合された条件より前に評価されます。

例: 使用箇所検索条件

```
void btnFind_actionPerformed(ActionEvent e) {
  try {
    // Create the query
    IQuery wuquery =
      (IQuery)m_session.createObject(IQuery.OBJECT_TYPE,
ItemConstants.CLASS_ITEM_BASE_CLASS);
    // Set the where-used type

wuquery.setSearchType(QueryConstants.WHERE_USED_ONE_LEVEL_LATEST_
RELEASED);
    // Add query criteria
    wuquery.setCriteria(
      "[Title Block.Part Category] == 'Electrical'" +
      "and [Title Block.Description] contains 'Resistor'" +
      "union [Title Block.Description] contains '400'" +
      "and [Title Block.Product Line(s)] contains 'Taurus'");
    // Run the query
    ITable results = wuquery.execute();

    // Add code here to display the results
  }
  catch (APIException ex) {System.out.println(ex);}
}
```

検索条件のロード

検索条件をロードするには、2 つの方法があります。

- IAgileSession.getObject() メソッドを使用して、検索条件の完全パスを指定します。
- IFolder.getChild() メソッドを使用して、フォルダに相対する検索条件の場所を指定します。

次の例は、完全パスを指定して検索条件をロードする方法を示しています。

例: IAgileSession.getObject() を使用した検索条件のロード

```
try {
  //Load the "Changes Submitted to Me" query
  IQuery query =
(IQuery)m_session.getObject(IQuery.OBJECT_TYPE,
                    "/Workflow Routings/Changes Submitted
To Me");
} catch (APIException ex) {
  System.out.println(ex);
}
```

次の例は、フォルダ (ここではユーザーの [パブリック受信トレイ] フォルダ) に相対するパスを指定して検索条件をロードする方法を示しています。

例: IFolder.getChild() を使用した検索条件のロード

```
try {
  //Get the Workflow Routings folder
  IFolder folder =
(IFolder)m_session.getObject(IFolder.OBJECT_TYPE,
"/Workflow Routings");
  //Load the "Changes Submitted to Me" query
  IQuery query = (IQuery)folder.getChild("Changes Submitted To Me");
} catch (APIException ex) {
  System.out.println(ex);
}
```

検索条件の削除

保存されている検索条件を削除するには、IQuery.delete() メソッドを使用します。

ユーザー セッションを閉じると、一時的な検索条件 (つまり、作成した後フォルダに保存していない検索条件) は自動的に削除されます。長いセッションでは、一時的な検索条件の実行が完了した後に、delete() メソッドを使用してその検索条件を明示的に削除できます。

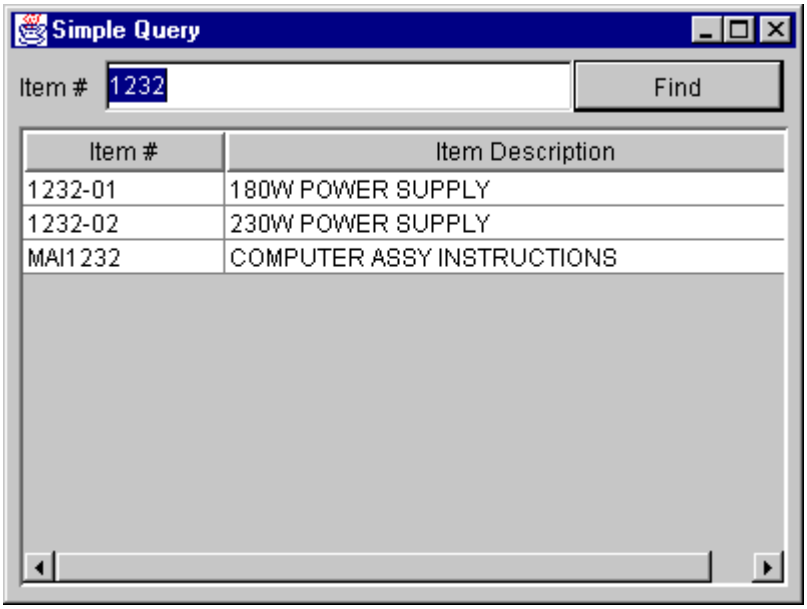
例: 検索条件の削除

```
void deleteQuery(IQuery query) throws APIException {
  query.delete();
}
```

簡単な検索条件の例

次の図は、簡単な検索条件を実行するダイアログ ボックスの例を示しています。

図 2: [Simple Query] ダイアログ ボックス



ユーザーは、[Simple Query] ダイアログ ボックスを使用して、検索するアイテム番号を指定します。[Find] ボタンをクリックすると、[Item Number] フィールドに指定したテキストが含まれたアイテムをすべて検索する検索条件が作成されます。次の例は、ユーザーが [Find] ボタンをクリックしたときに検索条件を実行するコードを示しています。

例: 簡単な検索条件のコード

```
        void btnFind_actionPerformed(ActionEvent e) {
          try {
            // Create the query
            IQuery query = (IQuery)m_session.createObject(IQuery.OBJECT_TYPE,
              ItemConstants.CLASS_ITEM_BASE_CLASS);
            // Turn off case-sensitivity
            query.setCaseSensitive(false);

            // Specify the criteria data
            query.setCriteria("[Title Block.Number] contains (%0)",
            new String[] { this.txtItemNum.getText().toString() });

           // Run the query
           ITable queryResults = query.execute();
           Iterator i = queryResults.iterator();

           // If there are no matching items, display an error message.
           if (!i.hasNext()) {
             JOptionPane.showMessageDialog(null, "No matching items.",
        "Error",
             JOptionPane.ERROR_MESSAGE);
```

```
      return;
    }

    // Define arrays for the table data
    final String[] names = {"Item Number", "Item Description"};
    final Object[][] data = new Object[resultCount][names.length];

    int j = 0;
    while (i.hasNext()) {
      IRow row = (IRow)i.next();
      data[j][0] =
row.getValue(ItemConstants.ATT_TITLE_BLOCK_NUMBER).toString();
      data[j][1] =
row.getValue(ItemConstants.ATT_TITLE_BLOCK_DESCRITPION).toString()
;
      j++;
    }
    catch (APIException ex) {
     System.out.println(ex);
    }

    // Create a table model
    TableModel newDataModel = new AbstractTableModel() {
      // Add code here to implement the table model
    };

    // Populate the table with data from the table model
    myTable.setModel(newDataModel);
}
```

第 4 章

テーブルの使用

扱うトピックは次のとおりです。

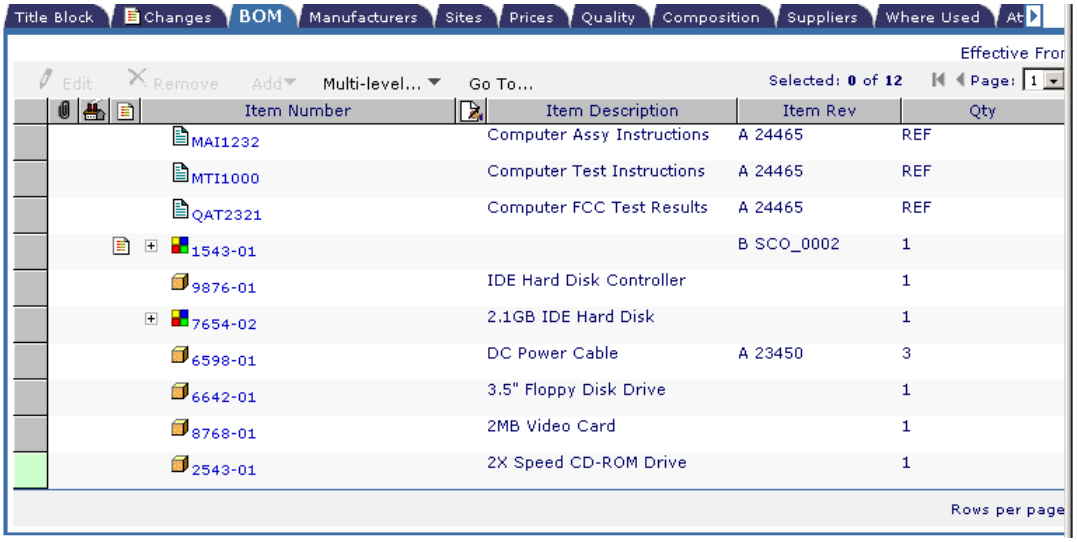
テーブルについて

プログラムで Agile PLM オブジェクトを使用するときは必ず、オブジェクトのデータを取得して表示する必要があります。データは、1 つ以上のテーブルに格納されています。Agile Web クライアントでは、これらのテーブルは、[製造元] タブや [BOM] タブなど、ウィンドウ内の個別のタブに相当します。

注意 場合によっては、Agile Web クライアントのタブに複数のテーブルが含まれることがあります。たとえば、アイテムに対する [変更] タブには、[保留中の変更] テーブルと [変更履歴] テーブルが含まれます。タブとそのタブに含まれるテーブルは、様々な Agile 製品について常に同じであるとはかぎりません。また、これらは、各 Agile PLM データオブジェクトについても同じではありません。たとえば、部品オブジェクト用のテーブルは、製造元オブジェクト用のテーブルとは異なります。68 ページの「テーブルの取得」を参照してください。

次の図は、Agile Web クライアントのアイテムに対する [BOM] タブを示しています。

図 3: アイテムの [BOM] タブ



Agile PLM テーブル内のデータを使用するには、次の基本手順に従います。

1. オブジェクト (例: アイテムまたは設計変更) を作成または取得します。
2. テーブル (例: [BOM] テーブル) を取得します。
3. テーブル行で処理を繰り返して、行を取得します。
4. 選択した行に対して 1 つ以上の属性値を取得または設定します。

`IFolder` などの `ITable` は、`java.util.Collection` を拡張し、そのスーパーインターフェースで提供されるすべてのメソッドをサポートします。したがって、Java の `Collection` と同様に `ITable` オブジェクトを使用できます。

インターフェース	継承メソッド
`java.util.Collection`	`add()`、`addAll()`、`clear()`、`contains()`、`containsAll()`、`equals()`、`hashCode()`、`isEmpty()`、`iterator()`、`remove()`、`removeAll()`、`retainAll()`、`size()`、`toArray()`、`toArray()`

テーブルの取得

オブジェクトの作成または取得後は、`IDataObject.getTable()` メソッドを使用して特定の Agile PLM テーブルを取得できます。`IDataObject` は、データのテーブルが含まれる Agile PLM オブジェクトを表す汎用オブジェクトです。これは、`IItem`、`IChange` および `IUser` など、他のいくつかのオブジェクトのスーパーインターフェースです。

注意 PG&C の適合のサプライヤ デクラレーション (SDOC) テーブルを取得すると、IDataObject.getTable() によって、この基本クラスに属する 14 個すべての SDOC テーブルが取得されます。ただし、これらの中で 6 つのテーブル ([アイテム]、[製造元部品]、[部品グループ]、[アイテム組成]、[製造元部品の組成]、[部品グループの組成]) は使用不可です。

テーブルは、各 Agile PLM データオブジェクトごとに異なります。変更オブジェクト用のテーブルは、アイテム用のテーブルとは異なります。特定のデータオブジェクト用の各テーブルは、そのデータオブジェクトに対する定数クラス内の定数で識別されます。アイテム定数は ItemConstants クラスに含まれ、変更定数は ChangeConstants クラスに含まれ、同様に他の定数も対応するクラスに含まれます。

これらのテーブルの使用に関する情報は、次の Agile 製品管理マニュアルを参照してください。

- 『Agile PLM ユーザー・ガイドおよびスタート・ガイド』
- 『Agile PLM 管理者ガイド』
- 『Agile PLM Product Governance & Compliance ユーザー・ガイド』
- 『Agile PLM Product Portfolio Management ユーザー・ガイド』

新規およびマージされた [関係] テーブルへのアクセス

[関係.影響元]、[関係.影響先] および [関係.参照] テーブルは、[関係] という 1 つのテーブルにマージされました。さらに、これらのテーブルで使用されていた TABLE_REFERENCES、TABLE_RELATIONSHIPSAFFECTS および TABLE_RELATIONSHIPSAFFECTEDBY 定数は、サポートされなくなりました。これらの定数が必要な場合は、ルーチン内に再度記述する必要があります。

これらのテーブルの使用に関する情報は、次の Agile マニュアルを参照してください。

- これらのテーブルを Agile PLM 製品で使用する場合は、『Agile PLM ユーザー・ガイドおよびスタート・ガイド』および『Agile PLM 管理者ガイド』を参照してください。
- これらのテーブルを Agile PPM 製品で使用する場合は、『Agile PLM Product Portfolio Management ユーザー・ガイド』を参照してください。

[関係] テーブルへのアクセス

このテーブルにアクセスするために、IRelationshipContainer インターフェースが実装されました。[関係] テーブルが含まれるすべての Agile ビジネス オブジェクトで、このインターフェースが実装されています。このテーブルにアクセスするには、IRelationshipContainer を使用するか、または CommonConstants.TABLE_RELATIONSHIPS 定数を指定して IDataObject.getTable() を使用します。

```
IRelationshipContainer container = (IRelationshipContainer) object;
ITable relationship = container.getRelationship();
```

マージされたテーブルへのアクセス

以前のリリースの Agile PLM でこれらのテーブルを使用していて、提供されていた機能が必要な場合は、コードを次のように変更してください。

マージされた [関係.影響元] テーブルへのアクセス

- 9.2.1.x 以前のリリースで使用されていたコード

```
ITable affectedBy =
object.getTable(ChangeConstants.TABLE_RELATIONSHIPSAFFECTEDBY);
```

- このリリースで推奨されるコード

```
ITable affectedBy =
object.getTable(CommonConstants.TABLE_RELATIONSHIPS)
.where("[2000007912] == 1", null);
```

マージされた [関係.影響先] テーブルへのアクセス

- 9.2.1.x 以前のリリースで使用されていたコード

```
ITable affects =
object.getTable(ChangeConstants.TABLE_RELATIONSHIPSAFFECTS);
```

- このリリースで推奨されるコード

```
ITable affects =
object.getTable(CommonConstants.TABLE_RELATIONSHIPS)
.where("[2000007912] == 2", null);
```

マージされた [関係.参照] テーブルへのアクセス

- 9.2.1.x 以前のリリースで使用されていたコード

```
ITable references =
object.getTable(ChangeConstants.TABLE_RELATIONSHIPS_REFERENCES);
```

- このリリースで推奨されるコード

```
ITable references =
object.getTable(CommonConstants.TABLE_RELATIONSHIPS)
.where("[2000007912] == 3", null);
```

重要 `ITable.where()` メソッドは、これらの 3 つのテーブルとともに配置される場合のみ動作が保証されており、SDK で他のテーブルにアクセスするために使用する場合は、失敗する可能性があります。

次の例は、アイテムに対する [BOM] テーブルを取得し、印刷する方法を示しています。

例: [BOM] テーブルの取得

```
//Load an item
private static IItem loadPart(String number) throws APIException {
  IItem item = (IItem)m_session.getObject(ItemConstants.CLASS_PART,
number);
  return item;
}
//Get the BOM table
private static void getBOM(IItem item) throws APIException {
```

```
    IRow row;
    ITable table = item.getTable(ItemConstants.TABLE_BOM);
    Iterator it = table.iterator();
    while (it.hasNext()) {
      row = (IRow)it.next();
      //Add code here to do something with the BOM table
    }
  }
```

読み取り専用テーブルの使用

いくつかの Agile PLM テーブルには、関連オブジェクトに関する履歴情報またはデータが格納されています。これらのテーブルは読み取り専用であるため、変更することはできません。テーブルにアクセスするコードを作成する場合は、ITable.isReadOnly() メソッドを使用して、テーブルが読み取り専用かどうかを確認します。

テーブルのメタデータの取得

ITableDesc は、テーブルのメタデータを表すインターフェースです。メタデータとは、テーブルのプロパティを説明する基礎データです。ITableDesc と ITable の関連は、IAgileClass と IDataObject の関連と同様です。特定のテーブルの属性、その ID、またはそのテーブル名を、データオブジェクトをロードせずに識別する必要がある場合があります。次の例は、ITableDesc インターフェースを使用して、テーブルに対するすべての属性 (非表示の属性も含む) のコレクションを取得する方法を示しています。

例: テーブルのメタデータの取得

```
private IAttribute[] getBOMAttributes() throws APIException {
  IAgileClass cls = admin.getAgileClass(ItemConstants.CLASS_PART);
  ITableDesc td = cls.getTableDescriptor(ItemConstants.TABLE_BOM);
  IAttribute[] attrs = td.getAttributes();
  return attrs;
}
```

Agile API を使用してメタデータを使用する方法の詳細は、273 ページの第 17 章「管理タスクの実行」を参照してください。

テーブル行の追加

テーブル行を作成するには、ITable.createRow(java.lang.Object) メソッドを使用します。このメソッドでは、新規行が作成され、paramパラメータで指定したデータで初期化されます。createRow の paramパラメータでは、次のデータを渡すことができます。

- 行のセルに対する一連の属性と値
- [添付ファイル] テーブルに追加するファイルまたは URL
- テーブルに追加する Agile PLM オブジェクト (例: IItem)

テーブルに行を追加するとき、テーブルの最後に追加されるとはかぎりません。

注意 空の行を作成する、パラメータなしのバージョンの createRow() メソッドもありますがお薦めできません。このメソッドは、Agile PLM の将来的なリリースでサポートされなくなる予定のため、使用しないでください。行を作成するときは、データで行を初期化する必要があります。

また、ITable.createRow() を使用して、テーブル行をバッチ形式で追加することもできます。74 ページの「複数のテーブル行の追加および更新」を参照してください。

[BOM] テーブルへのアイテムの追加

次の例は、ITable.createRow() メソッドを使用して、[BOM] テーブルにアイテムを追加する方法を示しています。

例: 行の追加および値の設定

```
private static void addToBOM(String number) throws APIException {
  IItem item = (IItem)m_session.getObject(ItemConstants.CLASS_PART,
number);
  ITable table = item.getTable(ItemConstants.TABLE_BOM);
  Map params = new HashMap();
  params.put(ItemConstants.ATT_BOM_ITEM_NUMBER, "1543-01");
  params.put(ItemConstants.ATT_BOM_QTY, "1");

item.setManufacturingSite(ManufacturingSiteConstants.COMMON_SITE);
  IRow row = table.createRow(params);
}
```

注意 [BOM] テーブルに拠点別の行を追加するには、ITable.createRow() を呼び出す前に、IManufacturerSiteSelectable.setManufacturingSite() を使用して特定の拠点を選択します。

[添付ファイル] テーブルへの添付ファイルの追加

次の例は、ITable.createRow(java.lang.Object) メソッドを使用して、[添付ファイル] テーブルに行を追加する方法を示しています。このコードでは、テーブルに行が追加され、指定したファイルで初期化されます。行の追加後、[ファイルの説明] フィールドの値も設定されます。

例: [添付ファイル] テーブルへの行の追加

```
private static void addAttachmentRow(String number) throws
APIException {
  File file = new File("d:/MyDocuments/1543-01.dwg");
  IItem item = (IItem)m_session.getObject(ItemConstants.CLASS_PART,
number);
  ITable table = item.getTable(ItemConstants.TABLE_ATTACHMENTS);
  IRow row = table.createRow(file);
}
```

[製造元] テーブルへの製造元部品の追加

次の例は、ITable.createRow(java.lang.Object) メソッドを使用して、アイテムの [製造元] テーブルに行を追加する方法を示しています。このコードでは、テーブルに行が追加され、指定した IManufacturerPart オブジェクトで初期化されます。

例: [製造元] テーブルへの行の追加

```
private static void addMfrPartRow(String number) throws APIException
{
  HashMap info = new HashMap();

info.put(ManufacturerPartConstants.ATT_GENERAL_INFO_MANUFACTURER_P
ART_NUMBER, "TPS100-256");

info.put(ManufacturerPartConstants.ATT_GENERAL_INFO_MANUFACTURER_N
AME, "TPS_POWER");
  IManufacturerPart mfrPart =
(IManufacturerPart)m_session.getObject(
    ManufacturerPartConstants.CLASS_MANUFACTURER_PART, info
  );
  IItem item = (IItem)m_session.getObject(ItemConstants.CLASS_PART,
number);

item.setManufacturingSite(ManufacturingSiteConstants.COMMON_SITE);
  ITable table = item.getTable(ItemConstants.TABLE_MANUFACTURERS);
  IRow row = table.createRow(mfrPart);
}
```

注意 [製造元] テーブルに拠点別の行を追加するには、ITable.createRow() を呼び出す前に、IManufacturerSiteSelectable.setManufacturingSite() を使用して特定の拠点を選択します。

[対象アイテム] テーブルへのアイテムの追加

次の例は、ITable.createRow(java.lang.Object) メソッドを使用して、設計変更の [対象アイテム] テーブルに行を追加する方法を示しています。このコードでは、テーブルに行が追加され、指定した IItem オブジェクトで初期化されます。

例: [対象アイテム] テーブルへの行の追加

```
private static void addItemRow(String number) throws APIException {
  IItem item = (IItem)m_session.getObject(ItemConstants.CLASS_PART,
"P522-103");
  IChange change =
(IChange)m_session.getObject(ChangeConstants.CLASS_ECO, number);
  ITable table =
change.getTable(ChangeConstants.TABLE_AFFECTEDITEMS);
  IRow row = table.createRow(item);
}
```

[BOM] テーブルも IItem オブジェクトを参照するため、前述の例と同様のコードを使用して、[BOM] テーブルに行を追加できます。

[スケジュール] テーブルへのタスクの追加

次の例は、ITable.createRow(java.lang.Object) メソッドを使用して、プログラムの [スケジュール] テーブルに行を追加する方法を示しています。このコードでは、テーブルに行が追加され、指定した IProgram オブジェクトで初期化されます。

例: [スケジュール] テーブルへの行の追加

```
private static void addTaskRow(IProgram program, IProgram task) throws
APIException {
  // Get the Schedule table of the program
  ITable table = program.getTable(ProgramConstants.TABLE_SCHEDULE);

  // Add the task to the schedule
  IRow row = table.createRow(task);
}
```

複数のテーブル行の追加および更新

ITable インターフェースには、1 回の API 呼び出しで複数のテーブル行を追加および更新するための便利なメソッドが 2 つ用意されています。

- ITable.createRows()
- ITable.updateRows()

これらのメソッドでは、複数のテーブル行が 1 つの API 呼び出しにグループ化されるため、リモート プロシージャによる呼び出し (RPC) の回数が削減され、パフォーマンスが向上します。特に、Wide Area Network (WAN) を通してサーバに接続している場合に有効です。ただし、これらのメソッドは、追加または更新対象の各行で単純に処理を繰り返している Agile アプリケーション サーバでは、効率的なバッチ操作になりません。

重要 ITable.createRows() および ITable.updateRows() メソッドがサポートされるのは、アイテムの [BOM] テーブル、または変更の [対象アイテム] テーブルで複数の行を追加または更新する場合のみです。

[BOM] テーブルへの複数アイテムの追加

次の例は、ITable.createRows() メソッドを使用して、[BOM] テーブルに複数のアイテムを追加する方法を示しています。

例: 複数行の追加および値の設定

```
private static void createBOMRows(String partNumber) throws
APIException {
  IItem[] child = new IItem [3];
  IItem parent = null;
  ITable tab = null;
```

```
  // Get the parent item
  parent = (IItem) m_session.getObject(IItem.OBJECT_TYPE,
partNumber);

  // Get the BOM table
  tab = parent.getTable(ItemConstants.TABLE_BOM);

  // Create child items
  child[0] = (IItem) m_session.createObject(ItemConstants.CLASS_PART,
partNumber + "-1");
  child[1] = (IItem) m_session.createObject(ItemConstants.CLASS_PART,
partNumber + "-2");
  child[2] = (IItem) m_session.createObject(ItemConstants.CLASS_PART,
partNumber + "-3");

  // Create a row array
  IRow[] rowArray = new IRow[3];

  // Add the items to the BOM
  rowArray = tab.createRows(new Object[]{child[0], child[1],
child[2]});
}
```

注意 [BOM] テーブルに拠点別の行を追加するには、`ITable.createRow()` を呼び出す前に、`IManufacturerSiteSelectable.setManufacturingSite()` を使用して特定の拠点を選択します。

複数の BOM 行の更新

複数の行を更新するには、`ITable.updateRows()` メソッドを使用します。このメソッドでは、複数の更新操作が 1 つの呼び出しにまとめられます。テーブル内の複数の行に対して `IRow.setValues()` を呼び出すかわりに、この API では、1 回のメソッド呼び出しでテーブル全体が更新されます。

`updateRow()` の `rows` パラメータを使用すると、キーとしての `IRow` インスタンスと値に対するインスタンスを含む `Map` を渡すことができます。値の Map オブジェクトには、キーとしての属性 ID と値に対する置換データが必要です。

例: 複数の BOM 行の更新

```
private static void updateBOMRows(String partNumber) throws
APIException {
  IItem parent = null;
  ITable tab = null;
  HashMap[] mapx = new HashMap[3];
  Map rows = new HashMap();
  IRow[] rowArray = new IRow[3];

  // Get the parent item
  parent = (IItem) m_session.getObject(IItem.OBJECT_TYPE,
partNumber);

  // Get the BOM table
  tab = parent.getTable(ItemConstants.TABLE_BOM);
```

```
  // Create three items
  IItem child1 = (IItem)
m_session.createObject(ItemConstants.CLASS_PART, partNumber + "-1");
  IItem child2 = (IItem)
m_session.createObject(ItemConstants.CLASS_PART, partNumber + "-2");
  IItem child2 = (IItem)
m_session.createObject(ItemConstants.CLASS_PART, partNumber + "-3");

  // Add these items to BOM table
  rowArray = tab.createRows(new Object[]{child1, child2, child3});

  // New values for child[0]
  mapx[0] = new HashMap();
  mapx[0].put(ItemConstants.ATT_BOM_FIND_NUM, new Integer(1));
  mapx[0].put(ItemConstants.ATT_BOM_QTY, new Integer(3));
  mapx[0].put(ItemConstants.ATT_BOM_REF_DES, "A1-A3");
  rows.put(rowArray[0], mapx[0]);

  // New values for child[1]
  mapx[1] = new HashMap();
  mapx[1].put(ItemConstants.ATT_BOM_FIND_NUM, new Integer(2));
  mapx[1].put(ItemConstants.ATT_BOM_QTY, new Integer(3));
  mapx[1].put(ItemConstants.ATT_BOM_REF_DES, "B1-B3");
  rows.put(rowArray[1], mapx[1]);

  // new values for child[2]
  mapx[2] = new HashMap();
  String strA = "BOM-Notes" + System.currentTimeMillis();
  mapx[2].put(ItemConstants.ATT_BOM_BOM_NOTES, strA);
  mapx[2].put(ItemConstants.ATT_BOM_FIND_NUM, new Integer(3));
  rows.put(rowArray[2], mapx[2]);

  // Update the BOM table rows
  tab.updateRows(rows);
}
```

テーブル行の繰り返し処理

Agile API を使用して [BOM] テーブルなどのテーブルを取得するとき、たいていの場合は、テーブルに含まれている行を参照する必要があります。個々の行にアクセスするには、最初に、テーブルに対する Iterator を取得する必要があります。次に、各行で処理を繰り返して、セルの値を設定できます。

Agile API では、テーブル内の行のランダム アクセスはサポートされていません。したがって、インデックス番号で特定の行を取得して更新することはできません。行を追加または削除すると、行全体が再度並べ替えられ、既存のテーブル Iterator は無効になります。

テーブル内のデータを参照するには、次のいずれかのメソッドを使用して、テーブルに対する Iterator を作成します。

- ITable.iterator() - Iterator オブジェクトを返します。このオブジェクトを使用すると、テーブルを最初の行から最後の行まで移動できます。
- ITable.getTableIterator() - ITwoWayIterator オブジェクトを返します。このオブジェクトを使用すると、テーブル行を前後に移動できます。また、ITwoWayIterator を使用して、任意の数の行をスキップすることもできます。ITwoWayIterator は、プログラムでテーブル行をユーザー インターフェースに表示する場合、Iterator より推奨されます。
- ITable.getTableIterator(ITable.ISortBy[]) - 並べ替えられた ITwoWayIterator オブジェクトを返します。
- ITable.getReferentIterator() - テーブルで参照されるオブジェクトに対する ITwoWayIterator オブジェクトを返します。

テーブルに対する Iterator を使用する場合、テーブル内の行の総数を認識している必要はありません。かわりに、一度に 1 行ずつ使用します。ITable インターフェースには、テーブル内の行の総数を計算する size() メソッドが用意されていますが、このメソッドは、パフォーマンスの点でリソース拡張操作と考えられるため、大規模なテーブルの場合 (特にテーブルの参照にすでに Iterator を使用している場合) はお薦めしません。

次の例は、テーブルに対する Iterator を取得し、ITwoWayIterator メソッドを使用してテーブル行を前後に移動する方法を示しています。

例: テーブル行の繰り返し処理

```
try {
  // Get an item
  IItem item = (IItem)m_session.getObject(ItemConstants.CLASS_PART,
"1000-02");
  // Get the BOM table
  ITable bom = item.getTable(ItemConstants.TABLE_BOM);
  ITwoWayIterator i = bom.getTableIterator();
  // Traverse forwards through the table
  while (i.hasNext()) {
    IRow row = (IRow)i.next();
    // Add code here to do something with the row
  }
  // Traverse backwards through the table
  while (i.hasPrevious()) {
    IRow row = (IRow)i.previous();
    // Add code here to do something with the row
  }
} catch (APIException ex) {
  System.out.println(ex);
}
```

ITwoWayIterator オブジェクトを使用すると、ユーザー インターフェースで複数のページにテーブル行を表示できます。この方法は、おそらく前述の例で示した ITwoWayIterator の使用方法より実用的です。たとえば、数百の BOM アイテムを含むスクロール ページを 1 つ表示するかわりに、1 ページに BOM アイテムを 20 個ずつ表示する複数のページにテーブルを分割できます。ページ間を移動するために、プログラムで、次の図に示されているようなナビゲーション コントロールを提供する必要があります。

図 4: Agile Web クライアントのナビゲーション コントロール

複数ページのテーブルを含む検索結果内のオブジェクトの更新

200 を超える結果を含む検索結果テーブル内のオブジェクトを getReferentIterator で更新すると、検索で返されたすべてのオブジェクトが更新されるとはかぎりません。たとえば、あるフィールド内の値と一致する検索を実行した後、getReferentIterator を使用して、結果で処理を繰り返している間に同じ値を編集するとします。検索は、最初のページは問題なく完了します。しかし、残りのページを検索したとき、一部のテーブル行は更新されません。この制限を解決する方法がいくつかあります。次にその例を示します。

多数の検索結果の繰り返し時にすべてのテーブル行を更新する手順は、次のとおりです。

1. この検索のテーブル ページ サイズを増やして、結果が 1 ページに収まるようにします。
2. 検索を複数回実行し、検索結果が空になるまで結果を更新し続けます。
3. 更新対象の同じフィールドで検索しないようにします。

テーブルの並べ替え

テーブル内の行を特定の属性で並べ替えるには、getTableIterator(ITable.ISortBy[]) を使用して、並べ替えられた Iterator を返します。getTableIterator() の ISortBy パラメータは、ITable.ISortBy オブジェクトの配列です。ISortBy オブジェクトを作成するには、createSortBy(IAttribute, ITable.ISortBy.Order) を使用します。createSortBy() の order パラメータは、ITable.ISortBy.Order 定数の ASCENDING または DESCENDING のいずれかです。

注意 Agile API では、1 つの属性でのみテーブルを並べ替えることができます。したがって、getTableIterator() の ISortBy パラメータに指定する ISortBy 配列に含まれるのは、1 つの ISortBy オブジェクトのみであることが必要です。

次の例では、[BOM] テーブルを [BOM | アイテム番号] 属性で並べ替えています。

例: テーブル Iterator の並べ替え

```
try {
  // Get an item
  IItem item = (IItem)m_session.getObject(ItemConstants.CLASS_PART,
"1000-02");

  // Get the BOM table
  ITable bom = item.getTable(ItemConstants.TABLE_BOM);

  // Get the BOM | Item Number attribute
  IAgileClass cls = item.getAgileClass();
  IAttribute attr =
cls.getAttribute(ItemConstants.ATT_BOM_ITEM_NUMBER);

  // Specify the sort attribute for the table iterator
  ITable.ISortBy sortByNumber = bom.createSortBy(attr,
ITable.ISortBy.Order.ASCENDING);

  // Create a sorted table iterator
  ITwoWayIterator i = bom.getTableIterator(new ITable.ISortBy[]
{sortByNumber});

  // Traverse forwards through the table
  while (i.hasNext()) {
    IRow row = (IRow)i.next();
```

```
        // Add code here to do something with the row
      }
    } catch (APIException ex) {
      System.out.println(ex);
    }
```

次の Product Sourcing および Program Execution の各オブジェクトは、多少異なる方法でテーブルにロードされるため、getTableIterator(ITable.ISortBy[]) メソッドを使用して並べ替えることができません。これらのオブジェクトのテーブルについては、iterator() または getTableIterator() メソッドを使用して、並べ替えられていない Iterator を作成します。

- IDiscussion
- IPrice
- IProgram
- IProject
- IRequestForQuote
- ISupplier
- ISupplierResponse

ITable.ISortBy インターフェースは、検索結果テーブルに対してはサポートされません。検索結果を並べ替えるには、SQL 構文を使用して、検索条件とともに ORDER BY ステートメントを指定します。詳細は、50 ページの「検索条件での SQL 構文の使用」を参照してください。

テーブル行の削除

テーブルから行を削除するには、ITable.removeRow() メソッドを使用します。このメソッドは、IRow オブジェクトという 1 つのパラメータを使用します。テーブル行で処理を繰り返して、行を取得できます。

テーブルが読み取り専用の場合、テーブルから行を削除することはできません。詳細は、71 ページの「読み取り専用テーブルの使用」を参照してください。アイテムのリリース済みリビジョンを使用している場合は、新しいリビジョンに対する設計変更を作成するまで、そのアイテムのテーブルから行を削除することはできません。

例: テーブル行の削除

```
    try {
      // get an item
      IItem item = (IItem)m_session.getObject(ItemConstants.CLASS_PART,
    "1000-02");
      // get the BOM table
      ITable bom = item.getTable(ItemConstants.TABLE_BOM);
      ITwoWayIterator i = bom.getTableIterator();

      // find the bom component 6642-01 and remove it
      while (i.hasNext()) {
        IRow row = (IRow)i.next();
        String bomitem =
    (String)row.getValue(ItemConstants.ATT_BOM_ITEM_NUMBER);
        if (bomitem.equals("6642-01")) {
          bom.removeRow(row);
          break;
        }
```

```
  }
} catch (APIException ex) {
  System.out.println(ex);
}
```

ITable には Collection インターフェースが実装されているため、Collection メソッドを使用してテーブル行を削除できます。テーブル内のすべての行を削除するには、Collection.clear() を使用します。

例: テーブルのクリア

```
public void clearAML(IItem item) throws APIException {
  // Get the Manufacturers table
  ITable aml = item.getTable(ItemConstants.TABLE_MANUFACTURERS);

  // Clear the table
  aml.clear();

}
```

行に対する参照オブジェクトの取得

いくつかの Agile PLM テーブルには、他の Agile PLM オブジェクトを参照する情報の行が格納されています。たとえば、[BOM] テーブルには、部品構成表 (BOM) に含まれているすべてのアイテムがリストされます。[BOM] テーブルの各行が各アイテムを表します。[BOM] テーブルの行を使用している間、プログラムによって、ユーザーが参照アイテムを開いて、そのデータを表示または変更できるようにします。

次の表に、他の Agile PLM オブジェクトを参照する Agile PLM テーブルを示します。すべての Agile PLM オブジェクトが番号 (例: アイテム番号、変更番号または製造元部品番号) で参照されます。

オブジェクト	テーブル	参照オブジェクト
IChange	対象アイテム 対象価格 添付ファイル 関係	IItem IPrice IAttachmentFile 複数のオブジェクト タイプ
ICommodity	添付ファイル 組成 部品 含有基準 サブスタンス サプライヤ	IAttachmentFile IDeclaration IItem ISpecification ISubstance ISupplier

オブジェクト	テーブル	参照オブジェクト
ICustomer	添付ファイル 関連 PSR	IAttachmentFile IServiceRequest
IDeclaration	添付ファイル アイテム組成 アイテム 製造元部品の組成 製造元部品 部品グループの組成 部品グループ 関係 含有基準	IAttachmentFile ISubstance IItem ISubstance IManufacturerPart ISubstance ICommodity 複数のオブジェクト タイプ ISpecification
IDiscussion	添付ファイル 使用箇所	IAttachmentFile サポートされていません。
IFileFolder	(ファイル) 関係 使用箇所	IAttachmentFile 複数のオブジェクト タイプ 複数のオブジェクト タイプ
IItem	添付ファイル BOM 変更履歴 組成 製造元 保留中変更使用箇所 保留中の変更 価格 品質 BOM のレッドライン 製造元のレッドライン 拠点 含有基準 サブスタンス 使用箇所	IAttachmentFile IItem IChange IDeclaration IManufacturerPart IItem IChange IPrice IServiceRequest または IQualityChangeRequest IItem IManufacturerPart IManufacturingSite ISpecification ISubstance IItem

オブジェクト	テーブル	参照オブジェクト
IManufacturerPart	添付ファイル 組成 価格 含有基準 サブスタンス サプライヤ 使用箇所	IAttachmentFile IDeclaration IPrice ISpecification ISubstance ISupplier IItem
IManufacturer	添付ファイル 使用箇所	IAttachmentFile IManufacturerPart
IManufacturingSite	添付ファイル	IAttachmentFile
IPackage	添付ファイル	IAttachmentFile
IPrice	添付ファイル 変更履歴 保留中の変更	IAttachmentFile IChange IChange
IProgram	添付ファイル 成果物 - 影響元 成果物 - 影響先 依存関係 - 依存対象 依存関係 - 必須対象 ディスカッション リンク スケジュール チーム	IAttachmentFile 複数のオブジェクト タイプ 複数のオブジェクト タイプ IProgram IProgram IDiscussion 複数のオブジェクト タイプ IProgram IUser および IUserGroup
IProject	添付ファイル BOM アイテムの変更 アイテム 製造元アイテム 保留中の変更 回答 RFQ	IAttachmentFile IItem IChange IItem IManufacturerPart IChange ISupplierResponse IRequestForQuote

オブジェクト	テーブル	参照オブジェクト
IQualityChangeRequest	対象アイテム 添付ファイル PSR アイテム 関係	IItem IAttachmentFile IItem 複数のオブジェクト タイプ
IRequestForQuote	添付ファイル	IAttachmentFile
IServiceRequest	対象アイテム 添付ファイル 関連 PSR 関係	IItem IAttachmentFile IServiceRequest 複数のオブジェクト タイプ
ISpecification	添付ファイル サブスタンス	IAttachmentFile ISubstance
ISubstance	添付ファイル 組成 使用箇所	IAttachmentFile ISubstance 複数のオブジェクト タイプ
ISupplierResponse	添付ファイル	IAttachmentFile
ISupplier	添付ファイル 製造元 PSR	IAttachmentFile IManufacturer IServiceRequest
ITransferOrder	添付ファイル 選択したオブジェクト	IAttachmentFile 複数のオブジェクト タイプ
IUser	添付ファイル 確認通知 ユーザー グループ	IAttachmentFile 複数のオブジェクト タイプ IUserGroup
IUserGroup	添付ファイル ユーザー	IAttachmentFile IUser

次の例は、アイテムに対する [保留中の変更] テーブルから参照 IChange オブジェクトを取得する方法を示しています。

例: 参照変更オブジェクトの取得

```
void getReferencedChangeObject(ITable changesTable) throws
APIException {
  Iterator i = changesTable.iterator();
  while (i.hasNext()) {
    IRow row = (IRow)i.next();
    IChange changeObj = (IChange)row.getReferent();
    if (changeObj != null) {
```

```
        //Add code here to do something with the IChange object
      }
    }
  }
```

次の表は、ITable.getReferentIterator() メソッドを使用してテーブルの参照オブジェクトで処理を繰り返すことによって、前述の例のコードを単純化する方法を示しています。

例: 参照オブジェクトでの繰り返し処理

```
void iterateReferencedChangeObjects(ITable changesTable) throws
APIException {
  Iterator i = changesTable.getReferentIterator();
  while (i.hasNext()) {
    IChange changeObj = (IChange)i.next();
    if (changeObj != null) {
      //Add code here to do something with the IChange object
    }
  }
}
```

行のステータス フラグの確認

特定のステータス条件と一致する場合にのみオブジェクトでアクションを実行する必要がある場合があります。たとえば、選択したオブジェクトがリリース済みの設計変更の場合は、ユーザーがそのオブジェクトを変更できないようにする場合があります。オブジェクトのステータスを確認するには、IRow.isFlagSet() メソッドを使用します。isFlagSet() メソッドは、ブール値 true または false を返します。

ステータス フラグ定数は、次のクラスで定義されます。

- CommonConstants - Agile PLM オブジェクトに共通のステータス フラグ定数が含まれます。
- ChangeConstants - IChange オブジェクトのステータス フラグ定数が含まれます。
- ItemConstants - IItem オブジェクトのステータス フラグ定数が含まれます。

次の例は、isFlagSet() メソッドを使用して、アイテムに添付ファイルがあるかどうかを判断する方法を示しています。

例: オブジェクトのステータス フラグの確認

```
private static void checkAttachments(IRow row) throws APIException {
  try {
    boolean b;
    b = row.isFlagSet(CommonConstants.FLAG_HAS_ATTACHMENTS);
    if (!b) {
      JOptionPane.showMessageDialog(null, "The specified row does not
      have attached files.", "Error", JOptionPane.ERROR_MESSAGE);
    }
  } catch (Exception ex) {}
}
```

[ページ 1]、[ユーザー定義 1] および [ユーザー定義 2] の使用

[ページ 1] ([タイトル ブロック]、[カバー ページ]、[一般情報] の各ページ)、[ユーザー定義 1] および [ユーザー定義 2] には、単一行のデータが含まれているため、形式はテーブルではありません。他のすべての表には、複数の行が含まれています。したがって、[ページ 1]、[ユーザー定義 1] および [ユーザー定義 2] のデータには直接アクセスできます。これらのページに対する値を取得および設定するために、テーブルを取得して行を選択する必要はありません。かわりに、指定したセルを取得し、`getValue()` および `setValue()` メソッドを使用して、データを表示または修正します。

プログラム全体で一貫した方法でデータ セルにアクセスする場合は、[ページ 1]、[ユーザー定義 1] および [ユーザー定義 2] の各テーブルを使用して値を取得および設定することも可能です。次の例は、アイテムに対する [ユーザー定義 1] の複数フィールドの値を編集する 2 つの方法を示しています。最初の方法では、[ユーザー定義 1] テーブルを取得し、次に複数のセルの値を設定します。2 番目の方法では、`IDataObject.getCell()` メソッドを呼び出し、[ユーザー定義 1] の各セルに直接アクセスします。どちらの方法も有効ですが、2 番目の方法のほうがコードの行数が少ないことがわかります。

例: [ユーザー定義 1] のセルの編集

```
// Edit Page Two cells by first getting the Page Two table
private static void editPageTwoCells(IItem item) throws Exception {
  ICell cell = null;
  DateFormat df = new SimpleDateFormat("MM/dd/yy");
  ITable table = item.getTable(ItemConstants.TABLE_PAGETWO);
  Iterator it = table.iterator();
  IRow row = (IRow)it.next();
  cell = row.getCell(ItemConstants.ATT_PAGE_TWO_TEXT01);
  cell.setValue("Aluminum clips");
  cell = row.getCell(ItemConstants.ATT_PAGE_TWO_MONEY01);
  cell.setValue(new Money(new Double(9.95), "USD"));
  cell = row.getCell(ItemConstants.ATT_PAGE_TWO_DATE01);
  cell.setValue(df.parse("12/01/03"));
}
// Edit Page Two cells by calling IDataObject.getCell()
private static void editPageTwoCells2(IItem item) throws Exception {
  ICell cell = null;
  DateFormat df = new SimpleDateFormat("MM/dd/yy");
  cell = item.getCell(ItemConstants.ATT_PAGE_TWO_TEXT01);
  cell.setValue("Aluminum clips");
  cell = item.getCell(ItemConstants.ATT_PAGE_TWO_MONEY01);
  cell.setValue(new Money(new Double(9.95), "USD"));
  cell = item.getCell(ItemConstants.ATT_PAGE_TWO_DATE01);
  cell.setValue(df.parse("12/01/03"));
}
```

レッドライン

リリース済みのアイテムまたは価格契約に対する変更を発行する場合、Agile API では、変更によって影響を受ける特定のテーブルをレッドラインできます。Agile PLM クライアントでは、レッドライン テーブルによって、以前のリビジョンから修正されている値が視覚的に識別されます。赤色の下線が引かれたテキスト (「レッドライン」という用語はこれに由来します) は、追加された値を示し、赤色の取り消し線のテキストは、削除された値を示します。変更を承認する担当者は、レッドライン データをレビューできます。

Agile PLM システムでは、次のレッドライン テーブルが提供されています。

- BOM のレッドライン
- 製造元のレッドライン (AML)
- 価格ラインのレッドライン

[BOM]、[製造元] または [価格ライン] テーブルをレッドラインする手順は、次のとおりです。

1. アイテムまたは価格オブジェクトのリリース済みリビジョンを取得します。
2. ECO、MCO、SCO または PCO など、新しい変更を作成します。
   - ECO では、アイテムの [BOM] テーブルまたは [製造元] テーブルを変更できます。
   - MCO では、アイテムの [製造元] テーブルを変更できます。
   - SCO では、アイテムの拠点別の [BOM] テーブルまたは [製造元] テーブルを変更できます。
   - PCO では、価格の [価格ライン] テーブルを変更できます。
3. アイテムまたは価格を変更の [対象アイテム] または [対象価格] テーブルに追加します。
4. ECO および PCO の場合は、変更に対する新しいリビジョンを指定します。SCO および MCO は、アイテムのリビジョンに影響を与えません。
5. レッドライン テーブル ([BOM のレッドライン]、[製造元のレッドライン] (AML) または [価格ラインのレッドライン] など) を変更します。

次の例は、アイテムの [製造元] テーブル (AML) をレッドラインするために必要な手順を示しています。

例: アイテムの [製造元] テーブルのレッドライン

```
private void redlineAML() throws APIException {
  IAttribute attrPrefStat = null;
  IAgileList listvalues = null;
  Map params = new HashMap();

  // Get a released item
  IItem item = (IItem)m_session.getObject("Part", "1000-02");

  // Get the Preferrred status value
  IAgileClass cls = item.getAgileClass();
  attrPrefStat =
cls.getAttribute(ItemConstants.ATT_MANUFACTURERS_PREFERRED_STATUS)
;
  listvalues = attrPrefStat.getAvailableValues();
  listvalues.setSelection(new Object[] { "Preferred" });

  // Create an MCO
  IChange change =
(IChange)m_session.createObject(ChangeConstants.CLASS_MCO,
"M000024");

  // Set the workflow ID of the MCO
  change.setWorkflow(change.getWorkflows()[0]);

  // Get the Affected Items table
  ITable affectedItems =
change.getTable(ChangeConstants.TABLE_AFFECTEDITEMS);
```

```
  // Add a new row to the Affected Items table
  IRow affectedItemRow = affectedItems.createRow(item);

  // Get the Redline Manufacturers table
  ITable redlineAML =
item.getTable(ItemConstants.TABLE_REDLINEMANUFACTURERS);

  // Add a manufacturer part to the table
  params.put(ItemConstants.ATT_MANUFACTURERS_MFR_NAME, "AMD");

params.put(ItemConstants.ATT_MANUFACTURERS_MFR_PART_NUMBER,
"1234-009");

params.put(ItemConstants.ATT_MANUFACTURERS_PREFERRED_STATUS,
listvalues);
  redlineAML.createRow(params);
  // Add another manufacturer part to the table
  params.clear();
  params.put(ItemConstants.ATT_MANUFACTURERS_MFR_NAME, "DIGITAL
POWER");

params.put(ItemConstants.ATT_MANUFACTURERS_MFR_PART_NUMBER,
"355355");

params.put(ItemConstants.ATT_MANUFACTURERS_PREFERRED_STATUS,
listvalues);
  redlineAML.createRow(params);
}
```

レッドラインの変更の削除

[BOM] などのテーブルにレッドラインの変更を加えた場合は、行に対する変更を取り消して、元の状態に復元する必要がある場合があります。IRedlinedRow.undoRedline() メソッドを使用すると、行に対するレッドラインの変更を取り消すことができます。

行に対するレッドラインを取り消すと、変更されたセルが元の値に復元されます。レッドライン付きの行は、追加された行または削除された行の場合もあります。追加された行に対するレッドラインを取り消すと、行全体がそのリビジョンから削除されます。削除された行に対するレッドラインを取り消すと、行全体が復元されます。

例: [BOM] テーブルからのレッドラインの変更の削除

```
private static undoBOMRedlines(IItem item, String rev) throws
APIException {
  item.setRevision(rev);
  ITable redlineBOM = item.getTable(ItemConstants.TABLE_REDLINEBOM);
  Iterator it = redlineBOM.iterator();
  while (it.hasNext()) {
    IRedlinedRow row = (IRedlinedRow)it.next();
    row.undoRedline();
  }
}
```

レッドライン付きの行およびレッドライン付きのセルの識別

IRedlined インターフェースは、レッドライン付きの行およびレッドライン付きのセルを識別するように設計されています。これは、レッドライン テーブルでのみサポートされます。このインターフェースは、isRedlineModified() メソッドとともに動作し、オブジェクトがレッドラインされているかどうかを示します。IRow および ICell オブジェクトを次のようにタイプキャストします。

- IRow は、行がレッドライン変更されたかどうかを示します。
- ICell は、セルがレッドライン変更されたかどうかを示します。

例: レッドライン付きの行およびセルの識別

```
public interface IRedlined {
public boolean isRedlineModified()
throws APIException;
}
```

IRedlined.isRedlineModified() は、ブール値を返します。返される値は、セルまたは行がレッドラインされている場合は TRUE です。

注意 IRedlined.isRedlineModified() は、レッドラインの追加された行またはレッドラインの削除された行のセルすべてについて、FALSE の値を返します。

ICell.getOldValue の使用

IRedlined インターフェースの実装により、ICell.getOldValue() メソッドは、レッドラインの追加された行またはレッドラインの削除された行に対して定義されなくなりました。ICell.getOldValue() メソッドの結果が有効であるのは、FLAG_IS_REDLINE_MODIFIED が行に対して true の場合のみです。

注意 このメソッドは、レッドラインの追加された行またはレッドラインの削除された行に対して呼び出さないでください。

第 5 章

データ セルの使用

扱うトピックは次のとおりです。

データ セルについて

ICell オブジェクトは、プログラムでロードまたは作成した Agile PLM オブジェクトに対するデータ フィールドです。セルは、Agile Web クライアント内のタブのフィールド、またはテーブルの単一のセルに対応します。ICell オブジェクトは、セルの現在の状態を説明する複数のプロパティで構成されます。Agile API プログラムで実行するデータ操作のほとんどに、セルの値またはプロパティに対する変更が含まれます。

データ タイプ

getValue() および setValue() メソッドに関連付けられるオブジェクトのタイプは、セルのデータ タイプによって異なります。次の表に、getValue() および setValue() メソッドに対するセル値のオブジェクト タイプを示します。

DataTypeConstants	getValue および setValue に関連付けられるオブジェクト タイプ
TYPE_DATE	Date
TYPE_DOUBLE	Double
TYPE_INTEGER	Integer
TYPE_MONEY	Money
TYPE_MULTILIST	IAgileList
TYPE_OBJECT	Object
TYPE_SINGLELIST	IAgileList
TYPE_STRING	String
TYPE_TABLE	Table

注意 TYPE_WORKFLOW などの他の Agile PLM データ タイプがありますが、これらはセル値には使用されません。

ディスカバリ権限の確認

ディスカバリ権限は、最も基本的な Agile PLM 権限です。この権限を使用すると、ユーザーはオブジェクトの存在を把握できます。オブジェクトに対するディスカバリ権限がない場合、そのオブジェクトは表示できません。

たとえば、ユーザーに製造元部品に対するディスカバリ権限がない場合は、プログラムで、ユーザーに対して [製造元] テーブルのいくつかのセルを表示することはできません。次の例に示すように、ICell.hasDiscoveryPrivilege() を使用すると、ユーザーに特定のセルに対するディスカバリ権限があるかどうかを確認できます。

注意 ディスカバリ権限のないセルの値を取得すると、Agile API よってヌル文字列 ("") が返されます。この動作は、他の Agile PLM クライアントと異なります。たとえば、Agile Web クライアントでは、参照する権限のないフィールドに対しては、値 [権限なし] が表示されます。

例: ディスカバリ権限の確認

```
Object v;
Integer attrID = ItemConstants.ATT_MANUFACTURERS_MFR_NAME;

try {
  // Get the Manufacturers table
  ITable aml = item.getTable(ItemConstants.TABLE_MANUFACTURERS);

  // Get the first row of the Manufacturers table
  IIterator iterator = aml.getTableIterator();
  if (iterator.hasNext()) {
    IRow amlRow = (IRow)iterator.next();
  }

  // Get the value for the Mfr. Name field.
  // If the user does not have Discovery privilege, the value is a null
String.
  v = amlRow.getValue(attrID);
  txtMfrName.setText(v.toString());

  // If the user does not have the Discovery privilege
  // for the cell, make its text color red.
  ICell cell = amlRow.getCell(attrID);
  if (cell.hasDiscoveryPrivilege()==false) {
    txtMfrName.setForeground(Color.red);
  }
} catch (APIException ex) {
  System.out.println(ex);
}
```

セルが読み取り専用かどうかの確認

役割と権限 (Agile PLM システムの管理者によりユーザーに割り当てられる) によって、Agile PLM オブジェクトとその基礎データに対するユーザーのアクセス範囲が決まります。たとえば、読み取り専用権限のみのユーザーは、Agile PLM オブジェクトを表示できますが、変更はできません。

プログラムでセルの値を表示する場合は必ず、現在のユーザーに対してセルが読み取り専用かどうかを確認する必要があります。読み取り専用の場合は、ユーザーが値を編集できないようにする必要があります。ユーザーが読み取り専用のセルに値を設定しようとすると、Agile API で例外が発生します。

例: フィールドが読み取り専用かどうかの確認

```
// ID for "Title Block.Description"
Integer attrID = ItemConstants.ATT_TITLE_BLOCK_DESCRIPTION;

// Set the value for the Description text field.
try {
  txtDescription.setText(item.getValue(attrID).toString());
  // Get the ICell object for "Title Block.Description"
  ICell cell = item.getCell(attrID);

  // If the cell is read-only, disable it
  if (cell.isReadOnly()) {
    txtDescription.setEnabled(false);
    txtDescription.setBackground(Color.lightGray);
  }
  else {
    txtDescription.setEnabled(true);
    txtDescription.setBackground(Color.white);
  }
} catch (APIException ex) {
  System.out.println(ex);
}
```

値の取得

次の表に、セルの値を取得するための Agile API メソッドを示します。

メソッド	説明
`ICell.getValue()`	セルの値を取得します。
`IRow.getValue()`	行内に含まれているセルの値を取得します。
`IRow.getValues()`	行内に含まれているすべてのセルの値を取得します。
`IDataObject.getValue()`	[ページ 1]、[ユーザー定義 1] または [ユーザー定義 2] のセルの値を取得します。

セルの値の使用を開始するには、セルを選択する必要があります。Agile PLM のセルは、単に属性のインスタンスです。セルの属性を指定するには、属性の ID 定数、その完全修飾名 (例: [タイトル ブロック.説明]) または `IAttribute` オブジェクトのいずれかを指定します。属性の参照に関する詳細は、284 ページの「属性の参照」を参照してください。

次の例は、属性 ID 定数でセルを参照する方法を示しています。

例: ID によるセルの指定

```
Object v;
Integer attrID = ItemConstants.ATT_TITLE_BLOCK_NUMBER;
try {
  v = item.getValue(attrID);
} catch (APIException ex) {
  System.out.println(ex);
}
```

次の例は、完全修飾属性名でセルを参照する方法を示しています。

例: 完全修飾名によるフィールドの指定

```
Object v;
String attrName = "Title Block.Number";
try {
  v = item.getValue(attrName);
} catch (APIException ex) {
  System.out.println(ex);
}
```

セルの値を取得するために使用するメソッドは、プログラムで使用中の現在のオブジェクトによって異なります。ICell オブジェクトをすでに取得していて、値を取得する場合は、ICell.getValue() メソッドを使用します。

例: ICell.getValue() を使用した値の取得

```
private static Object getCellVal(ICell cell) throws APIException {
  Object v;
  v = cell.getValue();
  return v;
}
```

たいていの場合、プログラムでは最初にアイテムなどのオブジェクトを取得し、次に、IDataObject.getValue(java.lang.Object cellId) メソッドを使用してその値を取得します。

例: IDataObject.getValue(Object cellID) を使用した値の取得

```
private static Object getDescVal(IItem item) throws APIException {
  Integer attrID = ItemConstants.ATT_TITLE_BLOCK_DESCRIPTION;
  Object v;
  v = item.getValue(attrID);
  return v;
}
```

getValue() メソッドで返されるオブジェクトは、Agile PLM 属性と同じデータ タイプです。データ タイプの詳細は、89 ページの「データ タイプ」を参照してください。

注意 検索で返されるテーブル内のすべてのセルには、そのセルに関連付けられているデータ タイプに関係なく、String 値が含まれています。検索結果テーブルの詳細は、61 ページの「検索結果の使用」を参照してください。

Agile PLM テーブル内の行で処理を繰り返している場合は、IRow.getValues() メソッドを使用して、テーブル内の特定の行に対するすべてのセルの値が含まれる Map オブジェクトを取得できます。返される Map オブジェクトでは、属性 ID キーがセルの値にマップされます。

SDK の日付フォーマットおよびユーザー プリファレンスの理解

SDK では、日付は Java 日付オブジェクトとして使用可能で、ユーザー プリファレンスに従ってフォーマットされません。ただし、エンド ユーザーは、GUI のユーザー プリファレンスで日付を必要なフォーマットに変換できます。

重要 エンド ユーザーは、PPM 日付に対して GMT 日付フォーマットを使用する必要があります。詳細は、『Agile PLM Product Portfolio Managementユーザー・ガイド』を参照してください。

値の設定

次の表に、セルの値を設定するための Agile API メソッドを示します。

メソッド	説明
`ICell.setValue()`	セルの値を設定します。
`IRow.setValue()`	行内に含まれているセルの値を設定します。
`IRow.setValues()`	行内に含まれている複数のセルの値を設定します。
`IDataObject.setValue()`	[ページ 1]、[ユーザー定義 1] または [ユーザー定義 2] のセルの値を設定します。
`IDataObject.setValues()`	[ページ 1]、[ユーザー定義 1] または [ユーザー定義 2] の複数のセルの値を設定します。

値を設定するために使用するメソッドは、プログラムで使用中の現在のオブジェクトによって異なります。

`ICell` オブジェクトをすでに取得していて、その値を設定する場合は、`ICell.setValue()` メソッドを使用します。

例: ICell.setValue() を使用した値の設定

```
private static void setDesc(ICell cell, String text) throws APIException
{
  cell.setValue(text);
}
```

プログラムで部品などのオブジェクトをすでに取得している場合は、`IDataObject.setValue()` メソッドを使用してその値を設定できます。

例: IDataObject.setValue() を使用した値の設定

```
private void setDesc(IItem item, String text) throws APIException {
  Integer attrID = ItemConstants.ATT_TITLE_BLOCK_DESCRIPTION;
  item.setValue(attrID, text);
}
```

Agile PLM テーブル内の行で処理を繰り返している場合は、`IRow.setValues()` メソッドを使用して、行全体に対してセルの値を設定できます。また、`IDataObject.setValues()` メソッドを使用すると、オブジェクトの [ページ 1]、[ユーザー定義 1] または [ユーザー定義 2] の複数のセルの値を設定できます。`setValues()` で指定する `Map` パラメータでは、属性がセルの値にマップされます。

例: IRow.setValues() を使用した行内の複数値の設定

```
private void setBOMRow(IRow row) throws APIException {
  Map map = new HashMap();
  map.put(ItemConstants.ATT_BOM_ITEM_NUMBER, "23-0753");
  map.put(ItemConstants.ATT_BOM_QTY, "1");
  map.put(ItemConstants.ATT_BOM_FIND_NUM, "0");

  row.setValues(map);
}
```

Agile PLM の値を設定する場合は、セルのデータ タイプを認識している必要があります。誤ったデータ タイプのオブジェクトを使用してセルの値を設定しようとすると、メソッドが失敗します。オブジェクトを使用して値を設定する前に、オブジェクトを別のクラスにキャストする必要がある場合があります。

注意 コードでトランザクション境界を明示的に定義しない場合は、プログラムで実行するすべての `setValue()` 操作が、個別のトランザクションとして処理されます。

ロックされたオブジェクトの例外の捕捉

他のユーザーがオブジェクトを変更している場合、オブジェクトはそのユーザーによって一時的にロックされます。別のユーザーがオブジェクトをロックしているときに、セルに値を設定しようとすると、プログラムで例外が発生します。したがって、セルの値を設定する場合は必ず、ロックされたオブジェクトに関連する次の Agile 例外を捕捉してください。

- `ExceptionConstants.APDM_ACQUIRE_DBLOCK_FAILED`
- `ExceptionConstants.APDM_RELEASE_DBLOCK_FAILED`
- `ExceptionConstants.APDM_OBJVERSION_MISMATCH`

ロックされたオブジェクトに関連する例外 813 も捕捉する必要があります。

ロックされたオブジェクトに対して Agile PLM が返す一般的な例外メッセージは、[このオブジェクトを使用しているユーザーがいます。後で再実行してください。] です。

例外の処理方法の詳細は、297 ページの第 18 章「例外の処理」を参照してください。

リスト値の取得および設定

リスト セルには 2 つの異なるデータ タイプがあります。1 つはシングルリスト セル用で、もう 1 つはマルチリスト セル用です。シングルリスト セルまたはマルチリスト セルの値を取得する場合、返されるオブジェクトは、`IAgileList` オブジェクトです。そのため、リスト セルの使用は、他のセルの使用より多少複雑です。`IAgileList` インターフェースには、現在のリスト選択項目を取得および設定するためのメソッドが用意されています。このセクションでは、カスケード リストなど、様々なタイプの Agile PLM リストに対する値を取得および設定する方法を示す例を提供します。

`ICell.getAvailableValues()` を使用してリスト セルの使用可能な値を取得する場合は、返される `IAgileList` オブジェクトに破棄されたリスト値が含まれている場合があります。プログラムでは、ユーザーがリスト セルの値に破棄された値を設定できないようにする必要があります。リスト値が破棄されているかどうかを確認する方法の詳細は、135 ページの「リスト値の破棄」を参照してください。

リストに String 値が含まれている場合、値は大文字と小文字が区別されます。したがって、リスト セルに値を設定する場合は必ず、値の大文字と小文字が正しいことを確認する必要があります。

シングルリスト セルの値の取得および設定

シングルリスト セルでは、リストから 1 つの値を選択できます。シングルリスト セルの値を取得する場合、返されるオブジェクトは IAgileList です。その IAgileList オブジェクトから、現在選択されている値の内容を判断できます。次の例は、アイテムに対する [タイトル ブロック.部品カテゴリ] セルの値を取得および設定する方法を示しています。

例: シングルリスト セルの値の取得および設定

```
private static String getPartCatValue(IItem item) throws APIException
{
  // Get the Part Category cell
  ICell cell =
item.getCell(ItemConstants.ATT_TITLE_BLOCK_PART_CATEGORY);

  // Get the current IAgileList object for Part Category
  IAgileList cl = (IAgileList)cell.getValue();

  // Get the current value from the list
  String value = null;
  IAgileList[] selected = cl.getSelection();
  if (selected != null && selected.length > 0) {
    value = (selected[0].getValue()).toString();
  }
  return value;
}
private static void setPartCatValue(IItem item) throws APIException
{
  // Get the Part Category cell
  ICell cell =
item.getCell(ItemConstants.ATT_TITLE_BLOCK_PART_CATEGORY);

  // Get available list values for Part Category
  IAgileList values = cell.getAvailableValues();

  // Set the value to Electrical
  values.setSelection(new Object[] { "Electrical" });
  cell.setValue(values);
}
```

マルチリスト セルの値の取得および設定

マルチリスト セルの動作は、シングルリスト セルとほぼ同じですが、複数の値を選択できる点が異なります。マルチリスト セルは、カスケード リストにすることはできません。次の例は、アイテムに対するマルチリスト セル [タイトル ブロック.製品ライン] の値を取得および設定する方法を示しています。

例: マルチリスト セルの値の取得および設定

```
private static String getProdLinesValue(IItem item) throws
APIException {
  String prodLines;
  // Get the Product Lines cell
  ICell cell =
item.getCell(ItemConstants.ATT_TITLE_BLOCK_PRODUCT_LINES);
```

```
  // Get the current IAgileList object for Product Lines
  IAgileList list = (IAgileList)cell.getValue();

  // Convert the current value from the list to a string
  prodLines = list.toString();

  return prodLines;
}

private static void setProdLinesValue(IItem item) throws APIException
{
  // Get the Product Lines cell
  ICell cell =
item.getCell(ItemConstants.ATT_TITLE_BLOCK_PRODUCT_LINES);

  // Get available list values for Product Lines
  IAgileList values = cell.getAvailableValues();

  // Set the Product Lines values
  values.setSelection(new Object[] {"Saturn","Titan","Neptune"});
  cell.setValue(values);
}
```

カスケード リストの値の取得および設定

シングルリスト セルは、カスケード リストになるように設定できます。カスケード リストでは複数の階層レベルでリストが表示されるため、リスト階層をドリル ダウンして特定の値を検索できます。カスケード リストの使用に関する詳細は、第 8 章「リストの使用」を参照してください。

カスケード リスト セルの値を取得する場合は、カスケード リスト内の各レベルが垂直バー (パイプ文字とも呼ばれます) で区切られます。カスケード リストの値を選択するには、IAgileList.setSelection() メソッドを使用します。IAgileList リーフ ノードの配列、または垂直バーで区切られた 1 つの文字列を含む String 配列のいずれかを指定できます。値を選択した後は、いずれかの setValue() メソッドを使用して値を保存します。

次の例は、カスケード リストの値を取得および設定する方法を示しています。

例: カスケード リストの値の取得および設定

```
private String getCascadeValue(IItem item) throws APIException {
  String value = null;
  // Get the Page Two.List01 value
  IAgileList clist =

(IAgileList)item.getValue(ItemConstants.ATT_PAGE_TWO_LIST01);
  // Convert the current value from the list to a string
  value = clist.toString();

  return value;
}
private void setCascadeValue(IItem item) throws APIException {
  String value = null;
```

```
  // Get the Page Two List01 cell
  ICell cell = item.getCell(ItemConstants.ATT_PAGE_TWO_LIST01);

  // Get available list values for Page Two List01
  IAgileList values = cell.getAvailableValues();

  // Set the value to "North America|United States|San Jose"
  values.setSelection(new Object[] {"North America|United States|San
Jose"});
  cell.setValue(values);
}
```

前述の例では、カスケード リストの値を設定する 1 つの方法が示されていますが、リストのツリー構造を示す別の長い形式も使用できます。カスケード リスト値を表す単一の String を指定するかわりに、リスト内の各レベルに対して選択項目を設定できます。次の例では、大陸、国および市町村の 3 つのレベルでカスケード リストの値を選択しています。

例: カスケード リストの値の設定 (長い形式)

```
private void setCascadeValue(IItem item) throws APIException{
String value = null;
// Get the Page Two List01 cell
ICell cell =
item.getCell(CommonConstants.ATT_PAGE_TWO_LIST01);

// Get available list values for Page Two List01
IAgileList values = cell.getAvailableValues();

// Set the continent to "North America"
IAgileList continent =
(IAgileList)values.getChildNode("North America");

// Set the country to "United States"
IAgileList country =
(IAgileList)continent.getChildNode("United States");

// Set the city to "San Jose"
IAgileList city = (IAgileList)country.getChildNode("San Jose");
values.setSelection(new Object[]{city});

// Set the cell value
cell.setValue(values);
}
```

参照指示セルの使用

Agile 9 SDK では、参照指示の使用方法を管理できます。参照指示セルは、システム設定によって縮小または展開できます。IReferenceDesignatorCell インターフェースには 3 つのパブリック API が含まれており、エンド ユーザーは、参照指示情報を 3 つの形式で取得できます。

- 縮小 - 例: A1–A3。getCollapsedValue() を使用します。
- 展開 - A1, A2, A3。getExpandedValue() を使用します。
- 個々の参照指示の配列 - [A1, A2, A3]。getReferenceDesignators[] を使用します。

次の表に、セルの参照指示の値を取得するための Agile API メソッドを示します。

メソッド	説明
IReferenceDesignatorCell.getCollapsedValue()	参照指示の縮小表示を取得します。たとえば、“A1,A2,A3” は “A1–A3” と表されます。範囲区切り文字 (–) は、システム プリファレンスの一部として定義されます。
IReferenceDesignatorCell.getExpandedValue()	参照指示の展開値を取得します。たとえば、“A1-A3” の場合は、文字列 “A1, A2, A3” が返されます。
IReferenceDesignatorCell.getReferenceDesignators()	個々の参照指示を文字列の配列として取得します。たとえば、“A1-A3” の場合は、これらの 3 つの文字列の配列 [“A1”, “A2”, “A3”] が返されます。

注意 以前のリリースの Agile SDK では、参照指示の値は、参照指示のカンマ区切りのリストでした。参照指示に対する cell.getValue() の機能は、参照指示の表現を制御するシステム設定によって異なるため、SDK ユーザーは、cell.getValue() または row.getValue() を使用しないでください。セルを取得して IReferenceDesignatorCell にキャストした後、処理するために必要なデータ構造に対応するメソッドを呼び出すか、または参照指示情報を表示することをお薦めします。

第 6 章

フォルダの使用

扱うトピックは次のとおりです。

フォルダについて

IFolder は、IQuery オブジェクトと IFolder オブジェクト、およびメイン Agile PLM オブジェクト (IChange、IItem、IManufacturer、IManufacturerPart、IPackage) を格納するために使用される汎用コンテナです。フォルダは、検索を整理するために使用されます。

注意 ファイル フォルダはフォルダとは異なるため、IFolder という独自のインターフェースがあります。ファイル フォルダには、他のオブジェクトの [添付ファイル] テーブルから参照できるファイルが 1 つ以上格納されます。ファイル フォルダの詳細は、145 ページの「添付ファイルとファイル フォルダの使用」を参照してください。

Agile PLM フォルダにはいくつかのタイプがあります。

- プライベート - 作成したユーザーのみがアクセスできるフォルダ。ユーザーは、自分のプライベート フォルダを作成または削除できます。
- パブリック - すべての Agile PLM ユーザーがアクセスできるフォルダ。パブリック フォルダは、グローバル検索権限のあるユーザーのみが作成、削除および変更できます。
- システム - Agile PLM システムに同梱されている事前定義済みのフォルダ。システム フォルダは、ほとんどのユーザーが変更または削除できません。
- 私のブックマーク (またはお気に入り) - Agile PLM オブジェクトに対する各ユーザーのブックマークが含まれる事前定義済みのフォルダ。[私のブックマーク] フォルダは削除できません。
- ホーム - 事前定義済みの Agile PLM ホーム フォルダ。[ホーム] フォルダは削除できません。
- パーソナル検索 - 各ユーザーのパーソナル検索に対する事前定義済みの親フォルダ。[パーソナル検索] フォルダは削除できません。
- 最近訪れたところ - 最近訪れたオブジェクトへのリンクが含まれる事前定義済みのフォルダ。このフォルダは、SDK では値は挿入されません。クライアント アプリケーションでのみ値が挿入されます。必要な場合は、アプリケーションでこのフォルダを指定します。

注意 [最近訪れたところ] フォルダは、データベースに定期的にのみフラッシュされます。したがって、ポータルを使用したプロセス拡張などの二次接続、またはスタンドアロン SDK アプリケーションでは、ユーザーの GUI に表示されるものと同一の情報は参照されません。

- レポート - レポートが含まれるフォルダ。Agile API を使用してレポート フォルダを作成、変更または削除することはできませんが、Agile PLM クライアントで作成、変更または削除できます。

注意 `FolderConstants` にも `TYPE_MODIFIABLE_CONTENTS` という定数がありますが、現在は使用されていません。

各ユーザーのフォルダの選択は異なる可能性があります。ただし、すべてのユーザーに [ホーム] フォルダがあります。各ユーザーの [ホーム] フォルダから、様々なサブフォルダを構築でき、パブリックおよびプライベート検索を参照できます。ユーザーに対する [ホーム] フォルダを取得するには、`IUser.getFolder(FolderConstants.TYPE_HOME)` メソッドを使用します。

フォルダは、他の Agile API オブジェクトと同じトランザクション モデルを使用します。フォルダのトランザクション境界を設定しない場合は、なんらかの項目がフォルダに追加されるか、またはフォルダから削除されるとすぐに、フォルダが自動的に更新されます。

`IFolder` は `java.util.Collection` を拡張し、`ITreeNode` は、これらのスーパーインターフェースで提供されるすべてのメソッドをサポートします。したがって、`IFolder` オブジェクトは Java の `Collection` と同様に使用できます。`ITreeNode` のメソッドを使用すると、子の追加と削除、子の取得、および親フォルダの取得によって、フォルダの階層構造を処理できます。

インターフェース	継承メソッド
`java.util.Collection`	`add()`、`addAll()`、`clear()`、`contains()`、`containsAll()`、`equals()`、`hashCode()`、`isEmpty()`、`iterator()`、`remove()`、`removeAll()`、`retainAll()`、`size()`、`toArray()`、`toArray()`
`ITreeNode`	`addChild()`、`getChildNode()`、`getChildNodes()`、`getParentNode()`、`removeChild()`

フォルダおよびオブジェクト名でのレベル区切り文字の使用

SDK では、`ITreeNode` オブジェクトの名前付けで、次のようにレベル区切り文字「|」および「/」をサポートしています。

- `IAgileList` オブジェクト名の「|」
- フォルダ名の「/」

この機能は主に、前述の表に示した継承 `ITreeNode` メソッドに影響を与えます。これらの文字を使用するには、文字の前に明示的に円記号 (¥) を付ける必要があります。

- ¥|
- ¥/

注意 SDK アプリケーションで定義されている Java リテラルで円記号を使用するには、円記号を 2 回 (¥¥) 指定する必要があります。

フォルダのロード

フォルダをロードする方法は、2 通りあります。

- IAgileSession.getObject() メソッドを使用して、フォルダの完全パスを指定します。
- IFolder.getChild() メソッドを使用して、サブフォルダの相対パスを指定します。

フォルダおよび検索名では、大文字と小文字は区別されません。したがって、大文字または小文字を使用してフォルダ パスを指定できます。たとえば、[Personal Searches] フォルダをロードするには、“/Personal Searches” または “/PERSONAL SEARCHES” と指定できます。

次の例は、フォルダの完全パスを指定してフォルダをロードする方法を示しています。

例: IAgileSession.getObject() を使用したフォルダのロード

```
try {
  //Load the Personal Searches folder
  IFolder folder = (IFolder)m_session.getObject(IFolder.OBJECT_TYPE,
"/Personal Searches");
} catch (APIException ex) {
  System.out.println(ex);
}
```

次の例は、別のフォルダからの相対パス、この場合はユーザーの [ホーム] フォルダからの相対パスを指定してフォルダをロードする方法を示しています。

例: IFolder.getChild() を使用したフォルダのロード

```
try {
  //Get the Home Folder
  IFolder homeFolder =
m_session.getCurrentUser().getFolder(FolderConstants.TYPE_HOME);
  //Load the Personal Searches subfolder
  IFolder folder = (IFolder)homeFolder.getChild("Personal Searches");
} catch (APIException ex) {
  System.out.println(ex);
}
```

フォルダの作成

フォルダを作成するには、**IAgileSession.createObject()** メソッドを使用します。フォルダを作成するとき、フォルダの名前とその親フォルダを指定する必要があります。次の例は、[Personal Searches] フォルダ内に “MyTemporaryQueries” というフォルダを作成する方法を示しています。

例:新規フォルダの作成

```
try {
  //Get an Admin instance
  IAdmin admin = m_session.getAdminInstance();
  //Load the Personal Searches folder
  IFolder parentFolder =
(IFolder)m_session.getObject(IFolder.OBJECT_TYPE, "/Personal
Searches");

  //Create parameters for a new folder
  Map params = new HashMap();
  params.put(FolderConstants.ATT_FOLDER_NAME,
"MyTemporaryQueries");
  params.put(FolderConstants.ATT_PARENT_FOLDER, parentFolder);

  //Create a new folder
  IFolder folder = (IFolder)session.createObject(IFolder.OBJECT_TYPE,
params);

} catch (APIException ex) {
  System.out.println(ex);
```

フォルダ タイプの設定

デフォルトでは、作成する新規フォルダはすべて、指定されないかぎりプライベート フォルダです。プライベート フォルダをパブリック フォルダに変更するには、IFolder.setType() メソッドを使用します。プライベート フォルダをパブリック フォルダに変更するには、グローバル検索権限が必要です。

フォルダのタイプの設定に使用できる 2 つのフォルダ タイプ定数は、FolderConstants.TYPE_PRIVATE および FolderConstants.TYPE_PUBLIC です。フォルダを他のフォルダ タイプに設定することはできません。

例:フォルダ タイプの設定

```
try {
  //Get an Admin instance
  IAdmin admin = m_session.getAdminInstance();
  //Load the My Cool Searches folder
  IFolder folder =
(IFolder)m_session.getObject(IFolder.OBJECT_TYPE,
                    "/Personal Searches/My Cool Searches");

  //Make the folder public
  folder.setFolderType(FolderConstants.TYPE_PUBLIC);

} catch (APIException ex) {
  System.out.println(ex);
}
```

フォルダ要素の追加および削除

Agile PLM フォルダには、IFolder オブジェクト (サブフォルダ)、IQuery オブジェクト、およびあらゆるデータオブジェクト (IChange、IItem、IManufacturer、IManufacturerPart オブジェクトなど) を格納できます。ITreeNode.addChild() メソッドを使用して、オブジェクトをフォルダに追加します。

フォルダ要素の追加

次の例は、オブジェクトをテーブルに追加する方法を示しています。

例: フォルダへのオブジェクトの追加

```
public void addFolderItem(IFolder folder, Object obj) {
  try {
    folder.addChild(obj);
  } catch (APIException ex) {
    System.out.println(ex);
  }
}
```

フォルダ要素の削除

単一のフォルダ要素を削除するには、ITreeNode.removeChild() メソッドを使用します。すべてのフォルダ要素をクリアするには、java.util.Collection.clear() メソッドを使用します。

例: フォルダからのオブジェクトの削除

```
void removeFolderElement(IFolder folder, Object obj) {
  try {
    folder.removeChild(obj);
  } catch (APIException ex) {
    System.out.println(ex);
  }
}
void clearFolder(IFolder folder) {
  try {
    folder.clear();
  } catch (APIException ex) {
    System.out.println(ex);
  }
}
```

フォルダ要素の取得

フォルダに含まれているすべてのオブジェクト (サブフォルダを含む) は、名前でロードできます。フォルダからオブジェクトを取得するには、IFolder.getChild() メソッドを使用します。フォルダ要素のオブジェクト タイプは様々であることに注意してください。オブジェクトに応じて、サブフォルダ、検索、または IItem などのデータオブジェクトを取得できます。

例: フォルダ要素の取得

```
public void getFolderChild(IFolder folder, String name) {
  try {
    IAgileObject object = folder.getChild(name);
    //If the object is a query, run it
    if (object.getType()==IQuery.OBJECT_TYPE) {
      IQuery query = (IQuery)object;
      ITable results = query.execute();

      //Add code here to do something with the query results
    }
  } catch (APIException ex) {
    System.out.println(ex);
  }
}
```

次の例は、IFolder.getChildren() メソッドを使用して IAgileObject 配列を返す方法を示しています。この場合、コードでは、配列内の各オブジェクトのオブジェクト タイプが確認された後、オブジェクトの名前が印刷されます。

例: フォルダの子の取得

```
private void browseFolder(int level, IFolder folder) throws
APIException {
  IAdmin admin = m_session.getAdminInstance();
  Collection subObjects = folder.getChildNodes();

  for (Iterator it = subObjects.iterator();it.hasNext();) {
    IAgileObject obj = (IAgileObject)it.next();
    System.out.println(indent(level * 4));

    switch (obj.getType()) {
      case IItem.OBJECT_TYPE:
        System.out.println("ITEM: " + obj.getName());
        break;

      case IFolder.OBJECT_TYPE:
        System.out.println("FOLDER: " + obj.getName());
        browseFolder(level + 1, (IFolder)obj);
        break;

      case IQuery.OBJECT_TYPE:
        System.out.println("QUERY: " + obj.getName());
        break;

      default:
        System.out.println(
```

```
        "UNKNOWN TYPE: " + obj.getType() + ":" + obj.getName());
    }
  }
}
private String indent(int level) {
  if (level <= 0) {
    return "";
  }
  char c[] = new char[level*2];
  Arrays.fill(c, ' ');
  return new String(c);
}
private String indent(int level) {
  if (level <= 0) {
    return "";
  }
  char c[] = new char[level*2];
  Arrays.fill(c, ' ');
  return new String(c);
}
```

フォルダの子を取得する別の方法は、フォルダの一方の端からもう一方の端に移動しながら、フォルダ要素で処理を繰り返すことです。IFolder オブジェクトの Iterator を作成するには、java.util.Collection.iterator() メソッドを使用します。

注意 フォルダのコンテンツを前後に移動する必要がある場合は、IFolder.getFolderIterator() メソッドを使用して ITwoWayIterator オブジェクトを返します。ITwoWayIterator では、previous()、next() および skip() メソッドなどが提供されます。

例: フォルダ要素での繰り返し処理

```
try {
  //Load the Project X folder
  IFolder folder = (IFolder)m_session.getObject(IFolder.OBJECT_TYPE,
                  "/Personal Searches/Project X");

  //Create a folder iterator
  Iterator it = folder.iterator();

  if (it.hasNext()) {
    //Get the next folder element
    Object obj = it.next();

    //Write code here to display each folder
    //element in your program's UI
  }
} catch (APIException ex) {
  System.out.println(ex);
}
```

フォルダの削除

フォルダを削除するには、IFolder.delete() メソッドを使用します。削除できるフォルダは、空のフォルダで、事前定義の Agile PLM システム フォルダ ([グローバル検索] および [私の受信トレイ] フォルダなど) 以外のフォルダです。

他のデータオブジェクトとは異なり、フォルダは最初の削除時にソフト削除されません。フォルダを削除すると、そのフォルダはシステムから完全に削除されます。

例: フォルダの削除

```
void deleteFolder(IFolder folder) throws APIException {
  folder.delete();
}
```

第 7 章

アイテム、BOM および AML の使用

扱うトピックは次のとおりです。

アイテムについて

アイテムは、製品の定義に使用するオブジェクトです。アイテムのタイプには、部品およびドキュメントなどがあります。部品は製品の一部として同梱され、コストが関連付けられています。部品はアセンブリの場合もあります。部品構成表 (BOM) には、アセンブリを構成する個々のコンポーネントがリストされます。ドキュメントは通常、部品に関係する社内の文書、図面、または手順です。

アイテムは、他の Agile PLM オブジェクトと次の点で異なります。

- リビジョン履歴があり、各リビジョンに一連のデータがあります。
- 確定済み、または将来の変更ができないようにロックされている場合があります。
- 拠点別の BOM または承認済み製造元リスト (AML) がある場合があります。

アイテムのリビジョンの取得および設定

アイテムのリビジョンは、Agile PLM 属性の特別なタイプです。リビジョンは常に、その関連変更オブジェクト (ECO など) の数である別の値と組み合わせて使用されます。アイテムをロードすると、常に最新のリリース済みリビジョンとともにロードされます。

他の属性とは異なり、アイテムの [タイトル ブロック.リビジョン] フィールド (その ID 定数が `ItemConstants.ATT_TITLE_BLOCK_REV`) に直接アクセスすることはできません。したがって、`getValue()` および `setValue()` メソッドを使用して、リビジョン値を取得したり、設定することはできません。たとえば、次のコード内の `revValue` 変数は常にヌル `String` です。

例: [タイトル ブロック.リビジョン] フィールドへのアクセスでリビジョンの取得に失敗する場合

```
IItem item = (IItem)m_session.getObject(IItem.OBJECT_TYPE,
"1000-02");
IAgileList listRevValue =

(IAgileList)item.getValue(ItemConstants.ATT_TITLE_BLOCK_REV);
   String revValue = listRevValue.toString();
if (revValue==null) {
  System.out.println("Failed to get the revision.");
}
```

アイテムのリビジョンを取得および設定する正しい方法は、次の例に示すように、IRevisioned インターフェースのメソッドを使用することです。この方法では、アイテムがロードされた後、アイテムのリビジョンで処理が繰り返されます。

例: アイテムのリビジョンの取得および設定

```
try {
  // Get an item
  IItem item = (IItem)m_session.getObject(IItem.OBJECT_TYPE,
"1000-02");

  // Print the item's current revision
  System.out.println("current rev : " + item.getRevision());

  // Get all revisions for the item
  Map revisions = item.getRevisions();

  // Get the set view of the map
  Set set = revisions.entrySet();

  // Get an iterator for the set
  Iterator it = set.iterator();

  // Iterate through the revisions and set each revision value
  while (it.hasNext()) {
    Map.Entry entry = (Map.Entry)it.next();
    String rev = (String)entry.getValue();
    System.out.println("Setting rev : " + rev + "....");
    item.setRevision(rev);
    System.out.println("current rev : " + item.getRevision());
  }
} catch (APIException ex) {
   System.out.println(ex);
}
```

IRevisioned.setRevision() メソッドは、リビジョンを指定するいくつかの異なる方法に対応しています。setRevision() メソッドの change パラメータには、次のタイプのオブジェクトを指定できます。

- 初版リビジョンを指定するためのヌル オブジェクト

  ```
  item.setRevision(null);
  ```

- 特定のリビジョンと関連付けられる IChange オブジェクト

  ```
  item.setRevision(changeObject);
  ```

- 特定のリビジョンと関連付けられる変更番号 (String)

  ```
  item.setRevision("C00450");
  ```

- リビジョン識別子 (“初版”、“A”、“B”、“C”などの String)

  ```
  item.setRevision("A");
  ```

- リビジョン識別子と変更番号が 8 つの空白で区切られた String(“A 23450”)

  ```
  item.setRevision("A C00450");
  ```

change パラメータに指定できる最後のタイプの String オブジェクトでは、Agile PLM テーブルの他のリビジョン セルで使用される同じ値を渡すことができます。たとえば、[BOM.アイテム リビジョン] セルは、[タイトル ブロック.リビジョン] とは異なり、直接アクセス可能です。セルの値を取得すると、リビジョン識別子と変更番号が 8 つの空白で区切られた String が返されます。

例: [BOM.アイテム リビジョン] を使用したリビジョンの設定

```
try {
  // Get an item
  IItem item = (IItem)m_session.getObject(IItem.OBJECT_TYPE,
"1000-02");

  // Get the BOM table
  ITable bomTable = item.getTable(ItemConstants.TABLE_BOM);

  // Get part 1543-01 in the BOM
  ITwoWayIterator it = bomTable.getTableIterator();
  while (it.hasNext()) {
    IRow row = (IRow)it.next();
    String num =
(String)row.getValue(ItemConstants.ATT_BOM_ITEM_NUMBER);
    if (num.equals("1543-01")) {
      // Get the revision for this BOM item
      // (bomRev = revID + 8 spaces + changeNumber)
      String bomRev =
(String)row.getValue(ItemConstants.ATT_BOM_ITEM_REV);

      // Load the referenced part
      IItem bomItem = (IItem)row.getReferent();

      // Set the revision
      System.out.println("Setting rev : " + bomRev + "....");
      bomItem.setRevision(bomRev);
      System.out.println("current rev : " + bomItem.getRevision());
      break;
    }
  }
} catch (APIException ex) {
   System.out.println(ex);
}
```

注意 アイテムにリリース済みリビジョンおよび保留中の変更がない場合、IRevisioned.getRevision() メソッドはヌル String を返し、IRevisioned.getRevisions() メソッドは空の Map オブジェクトを返します。

リビジョンの確定済みステータスの変更

アイテムの各リビジョンは確定済みにできます。アイテムのリビジョンを確定すると、そのリビジョンのすべての添付ファイルがロックされ、チェックアウトできなくなります。アイテムを確定した後でも、アイテムの添付ファイルは、Agile Web クライアントを使用して表示できますが、新しい変更を提出するまで修正はできません。

アイテムを確定または未確定にするには、IAttachmentContainer.setIncorporated() メソッドを使用します。アイテムを確定または未確定にするには、特別な Agile PLM 権限が必要です。適切な権限がない場合は、setIncorporated() メソッドで例外が発生します。

リビジョン番号があるアイテムのみを確定できます。したがって、リリースされていないプレリミナリ アイテムは確定できません。そのアイテムに対する ECO が提出され、保留中のリビジョン番号が指定されると、リビジョンを確定できるようになります。次の例では、アイテムの確定済みステータスを変更する方法を示しています。

例: アイテムの確定済みステータスの変更

```
try {
  // Get an item
  IItem item = (IItem)m_session.getObject(IItem.OBJECT_TYPE,
"1000-02");

  // Incorporate the item, or unincorporate it,
  // depending on its current state
  item.setIncorporated(!item.isIncorporated());
} catch (APIException ex) {
   System.out.println(ex);
}
```

BOM の使用

部品構成表 (BOM) には、製品を構成するコンポーネントがリストされます。BOM にリストされるアイテムは、単体のアイテムの場合も、複数のアイテムで構成されるアセンブリの場合もあります。

[BOM] テーブルは、他の Agile PLM テーブルとは異なり、データの列 (フィールド) で構成されます。各列が、[BOM.アイテム番号] などの Agile PLM 属性を表します。[BOM] テーブルの各行は、個別のアイテム (部品、ドキュメントまたはユーザー定義のサブクラスのいずれか) を表します。

[BOM] テーブルの他に、BOM に対するレッドライン変更が記録される [BOM のレッドライン] もあります。DataObject.getTable() メソッドを使用して [BOM] テーブルをロードする場合は、正しいテーブル ID 定数を指定していることを確認してください。

BOM テーブル	ID 定数
現在の [BOM] テーブル	ItemConstants.TABLE_BOM
[BOM のレッドライン] テーブル	ItemConstants.TABLE_REDLINEBOM

[BOM] テーブルの取得方法を示す例は、「テーブルの取得」を参照してください。

BOM へのアイテムの追加

[BOM] テーブルにアイテムを追加する前に、製造拠点を指定します。BOM のアイテムは、拠点別であるか、またはすべての拠点に共通です。IManufacturingSiteSelectable.setManufacturingSite() メソッドを使用して、拠点を指定します。共通の BOM にアイテムを追加するには、ManufacturingSiteConstants.COMMON_SITE を使用します。それ以外の場合は、ユーザーのデフォルトの拠点など、特定の拠点を指定します。

注意 親アイテムが現在すべての拠点を表示するように設定されている場合、その BOM には行を追加できません。BOM に行を追加する前に、アイテムの拠点が ManufacturingSiteConstants.ALL_SITES に設定されていないことを確認してください。ManufacturingSiteConstants.ALL_SITES に設定されていると、API で例外が発生します。

例: BOM へのアイテムの追加

```
//Add an item to the common BOM
public void addCommonBOMItem(IItem item, String bomnumber) throws
APIException {
  HashMap map = new HashMap();
  map.put(ItemConstants.ATT_BOM_ITEM_NUMBER, bomnumber);

item.setManufacturingSite(ManufacturingSiteConstants.COMMON_SITE);
  item.getTable(ItemConstants.TABLE_BOM).createRow(map);
}
//Add a site-specific item to the BOM using the user's default site
public void addSiteBOMItem(IItem item, String bomnumber) throws
APIException {
 HashMap map = new HashMap();
 map.put(ItemConstants.ATT_BOM_ITEM_NUMBER, bomnumber);

item.setManufacturingSite(((IAgileList)m_session.getCurrentUser().
getValue()

UserConstants.ATT_GENERAL_INFO_DEFAULT_SITE)).getSelection()[0].ge
tValue());
  );
  item.getTable(ItemConstants.TABLE_BOM).createRow(map);
}
```

製造拠点の詳細は、139 ページの「製造拠点の管理」を参照してください。

BOM の展開

[BOM] テーブルは、複数のレベルが含まれるテーブルとして表示できます。これは、API でこのように表示されない場合でも可能です。デフォルトでは、[BOM] テーブルにはトップレベルのアイテムのみが含まれています。BOM をその階層が表示されるように展開するには、各 BOM アイテムとそのサブアセンブリを再帰的にロードする必要があります。次の例は、複数レベルの BOM を印刷する方法を示しています。

例: 複数レベルの BOM の印刷

```
private void printBOM(IItem item, int level) throws APIException {
  ITable bom = item.getTable(ItemConstants.TABLE_BOM);
  Iterator i = bom.getReferentIterator();
  while (i.hasNext()) {
    IItem bomItem = (IItem)i.next();
    System.out.print(indent(level));
    System.out.println(bomItem.getName());
    printBOM(bomItem, level + 1);
  }
}
private String indent(int level) {
  if (level <= 0) {
    return "";
```

```
    }
    char c[] = new char[level*2];
    Arrays.fill(c, ' ');
    return new String(c);
  }
```

別の BOM への BOM のコピー

2 つのアイテムの BOM が非常に類似している場合がよくあります。BOM を最初から作成するかわりに、たいていの場合、あるアイテムの BOM を別のアイテムの BOM にコピーして、多少の変更を加える方が簡単です。Collection.addAll() メソッドを使用すると、あるテーブルのコンテンツをターゲット テーブルにコピーできます。addAll() メソッドでは、アイテムの新規リビジョンは設定されません。

注意 あるアイテムの BOM を別のアイテムの BOM にコピーする場合、ターゲット アイテムの関連製造拠点は、ソース アイテムと同じ必要があります。

例: Collection.addAll() を使用した BOM のコピー

```
private static void copyBOM(IItem source, IItem target) throws
APIException {
  // Get the source BOM
  ITable sourceBOM = source.getTable(ItemConstants.TABLE_BOM);

  // Get the target BOM
  ITable targetBOM = target.getTable(ItemConstants.TABLE_BOM);

  // Add all rows from the source BOM to the target BOM
  targetBOM.addAll(sourceBOM);
}
```

BOM をコピーする別の方法は、ソース BOM の行で処理を繰り返して、各行をターゲット BOM にコピーすることです。

例: 繰り返し処理による BOM のコピー

```
private static void copyBOM1(IItem source, IItem target) throws
APIException {
  // Get the source BOM
  ITable sourceBOM = source.getTable(ItemConstants.TABLE_BOM);

  // Get an iterator for the source BOM
  Iterator i = sourceBOM.iterator();

  // Get the target BOM
  ITable targetBOM = target.getTable(ItemConstants.TABLE_BOM);

  // Copy each source BOM row to the target BOM
  while (i.hasNext()) {
    targetBOM.createRow(i.next());
  }
}
```

BOM のレッドライン

[BOM] テーブルをレッドラインするには、次の手順に従います。

1. リリース済みのアセンブリ アイテムを取得します。
2. アイテムに対する新しい設計変更 (ECO など) を作成します。
3. ECO の [対象アイテム] テーブルにアイテムを追加します。また、変更の新規リビジョンを指定し、関連する変更にアイテムのリビジョンを設定します。
4. アイテムの [BOM のレッドライン] テーブルを変更します。

次のセクションに、これらの各手順のコード例があります。

注意 [BOM] テーブルの行からレッドラインを削除できます。87 ページの「レッドラインの変更の削除」を参照してください。

リリース済みのアセンブリ アイテムの取得

次の例は、部品サブクラスからアセンブリ アイテムをロードする方法を示しています。指定する部品がリリース済みであり、BOM があることを確認してください。

例: リリース済みのアセンブリの取得

```
// Load a released assembly item
private static IItem loadItem(IAgileSession myServer, Integer
ITEM_NUMBER) throws APIException {
  IItem item = (IItem)myServer.getObject("Part", ITEM_NUMBER);
  if (item != null) {
    //Check if the item is released and has a BOM
    if (item.getRevision().equals("Introductory") ||
      !item.isFlagSet(ItemConstants.FLAG_HAS_BOM)) {
      System.out.println("Item must be released and have a BOM.");
      item = null;
    }
  return item;
}
```

設計変更の作成

BOM をレッドラインするには、ECO などの設計変更を作成する必要があります。次の例は、ECO を作成し、そのワークフローを選択する方法を示しています。

例: ECO の作成

```
private static IChange createChange(IAgileSession myServer, Integer
ECO_NUMBER)
    throws APIException {
  IChange change =
(IChange)myServer.createObject(ChangeConstants.CLASS_ECO,
ECO_NUMBER);
  // Set the workflow ID
  change.setWorkflow(change.getWorkflows()[0]);
  return change;
}
```

設計変更の [対象アイテム] タブへのアイテムの追加

ECO の作成後は、ECO の [対象アイテム] テーブルに、ロードした部品を追加できます。すべての ECO がリビジョンに関連付けられます。次の例は、ECO の新規リビジョンを指定した後、部品のリビジョンに、ECO に関連付けられているリビジョンを設定する方法を示しています。

例: 設計変更の [対象アイテム] テーブルへのアイテムの追加

```
private static void addAffectedItems(IAgileSession myServer, IItem
item, IChange change)
  throws APIException {
  // Get the Affected Items table
  ITable affectedItems =
change.getTable(ChangeConstants.TABLE_AFFECTEDITEMS);

  // Create a Map object to store parameters
  Map params = new HashMap();

  // Set the value of the item number by specifying the item object

params.put(ChangeConstants.ATT_AFFECTED_ITEMS_ITEM_NUMBER, item);
  // Specify the revision for the change
  params.put(ChangeConstants.ATT_AFFECTED_ITEMS_NEW_REV, "B");

  // Add a new row to the Affected Items table
  IRow affectedItemRow = affectedItems.createRow(params);

  // Select the new revision for the part
  item.setRevision(change);
}
```

[BOM のレッドライン] テーブルの変更

ECO の [対象アイテム] テーブルに部品が追加され、リビジョンが指定されると、部品の [BOM のレッドライン] テーブルの変更を開始できます。次の例は、[BOM のレッドライン] テーブルを取得し、行を追加および削除し、特定のセル値を設定する方法を示しています。

例: [BOM のレッドライン] テーブルの変更

```
private static void modifyRedlineBOM(IAgileSession myServer, IItem
item) throws APIException {
  // Get the Redline BOM table
  ITable redlineBOM = item.getTable(ItemConstants.TABLE_REDLINEBOM);

  // Create two new items, 1000-002 and 1000-003
  IItem item1 = (IItem)
myServer.createObject(ItemConstants.CLASS_PART, "1000-002");
  IItem item2 = (IItem)
myServer.createObject(ItemConstants.CLASS_PART, "1000-003");

  // Add item 1000-002 to the table
```

```
  IRow redlineRow = redlineBOM.createRow(item1);
  redlineRow.setValue(ItemConstants.ATT_BOM_QTY, new Integer(50));
  redlineRow.setValue(ItemConstants.ATT_BOM_FIND_NUM, new
Integer(777));

  // Add item 1000-003 to the table
  redlineRow = redlineBOM.createRow(item2);
  redlineRow.setValue(ItemConstants.ATT_BOM_QTY, new Integer(50));
  redlineRow.setValue(ItemConstants.ATT_BOM_FIND_NUM, new
Integer(778));

  // Remove item 1000-003 from the table
  IRow    delRow;
  String   itemNumber;
  Iterator it = redlineBOM.iterator();
  while (it.hasNext()) {
    delRow    = (IRow)it.next();
    itemNumber =
(String)delRow.getValue(ItemConstants.ATT_BOM_ITEM_NUMBER);
    if (itemNumber.equals("1000-003")) {
      redlineBOM.removeRow(delRow);
      break;
    }
  }

  // Change the Qty value for item 1000-002
  IRow modRow;
  it = redlineBOM.iterator();
  while (it.hasNext()) {
    modRow    = (IRow)it.next();
    itemNumber =
(String)modRow.getValue(ItemConstants.ATT_BOM_ITEM_NUMBER);
    if (itemNumber.equals("1000-002")) {
      modRow.setValue(ItemConstants.ATT_BOM_QTY, new Integer(123));
    }
  }
}
```

AML の使用

アイテムの [製造元] テーブルは、承認済み製造元リスト (AML) とも呼ばれます。特定のアイテムについてその供給を承認された製造元がリストされます。このリストで、そのアイテムの製造元部品を特定できます。[製造元] テーブルは、データの列 (フィールド) で構成されます。各列は、[製造元.製造元名] などの Agile PLM 属性を表します。[製造元] テーブルの各行は、個別の製造元部品を表します。

[製造元] テーブルの他に、レッドライン変更が記録される [製造元のレッドライン] テーブルもあります。`DataObject.getTable()` メソッドを使用して [製造元] テーブルをロードする場合は、正しいテーブル ID 定数を指定していることを確認してください。

BOM テーブル	ID 定数
現在の [製造元] テーブル	`ItemConstants.TABLE_MANUFACTURERS`
[製造元のレッドライン] テーブル	`ItemConstants.TABLE_REDLINEMANUFACTURERS`

[製造元] テーブルへの承認済み製造元の追加

[BOM] テーブルとは異なり、[製造元] テーブルでは、テーブルに新規行を追加する前に、製造拠点を指定する必要があります。承認済み製造元は、拠点別であるか、またはすべての拠点に共通です。IManufacturingSiteSelectable.setManufacturingSite() メソッドを使用して、拠点を指定します。共通の [製造元] テーブルに承認済み製造元を追加するには、ManufacturingSiteConstants.COMMON_SITE を使用します。それ以外の場合は、ユーザーのデフォルトの拠点など、特定の拠点を選択します。

注意 親アイテムが現在すべての拠点を表示するように設定されている場合、その AML には行を追加できません。AML に行を追加する前に、アイテムの拠点が ManufacturingSiteConstants.ALL_SITES に設定されていないことを確認してください。ManufacturingSiteConstants.ALL_SITES に設定されていると、API で例外が発生します。

例: AML への承認済み製造元の追加

```
//Add a MfrPart to the common AML
public void addCommonApprMfr(IItem item, String mfrName, String
mfrPartNum) throws APIException {
  HashMap map = new HashMap();

map.put(ManufacturerPartConstants.ATT_GENERAL_INFO_MANUFACTURER_PA
RT_NUMBER, mfrPartNum);

map.put(ManufacturerPartConstants.ATT_GENERAL_INFO_MANUFACTURER_NA
ME, mfrName);
  IManufacturerPart mfrPart =
(IManufacturerPart)m_session.getObject(
   ManufacturerPartConstants.CLASS_MANUFACTURER_PART, map
  );

item.setManufacturingSite(ManufacturingSiteConstants.COMMON_SITE);

item.getTable(ItemConstants.TABLE_MANUFACTURERS).createRow(mfrPart
);
}
//Add a site-specific MfrPart to the AML using the user's default site
public void addSiteApprMfr(IItem item, String mfrName, String
mfrPartNum) throws APIException {
  HashMap map = new HashMap();

map.put(ManufacturerPartConstants.ATT_GENERAL_INFO_MANUFACTURER_PA
RT_NUMBER, mfrPartNum);

map.put(ManufacturerPartConstants.ATT_GENERAL_INFO_MANUFACTURER_NA
ME, mfrName);
  IManufacturerPart mfrPart =
(IManufacturerPart)m_session.getObject(
```

```
    ManufacturerPartConstants.CLASS_MANUFACTURER_PART, map
  );

item.setManufacturingSite(((IAgileList)m_session.getCurrentUser().
getValue(

UserConstants.ATT_GENERAL_INFO_DEFAULT_SITE)).getSelection()[0]
  );

item.getTable(ItemConstants.TABLE_MANUFACTURERS).createRow(mfrPart
);
}
```

製造拠点の詳細は、第 9 章「製造拠点の管理」を参照してください。

AML のレッドライン

アイテムがリリースされた後は、新規設計変更を発行することによってのみ、[製造元] テーブルを変更できます。設計変更を使用すると、[製造元] テーブルをレッドラインできます。

注意 [製造元] テーブルの行からレッドラインを削除できます。87 ページの「レッドラインの変更の削除」を参照してください。

[製造元] テーブルをレッドラインする手順は、次のとおりです。

1. アイテムのリリース済みリビジョンを取得します。
2. 新規 ECO、MCO または SCO を作成します。
   - ECO では、アイテムの [BOM] テーブルまたは [製造元] テーブルを変更できます。
   - MCO では、アイテムの [製造元] テーブルを変更できます。
   - SCO では、アイテムの拠点別の [BOM] テーブルまたは [製造元] テーブルを変更できます。
3. 変更の [対象アイテム] テーブルにアイテムを追加します。
4. ECO の場合は、変更に対する新しいリビジョンを指定します。SCO および MCO は、アイテムのリビジョンに影響を与えません。
5. [製造元のレッドライン] テーブルを変更します。

第 8 章

リストの使用

扱うトピックは次のとおりです。

リストについて

Agile PLM システムでは、多くの属性がリストとして設定されています。Agile には、リスト フィールドをサポートするために、次の 2 つのデータ タイプが用意されています。

- シングルリスト - 1 つの値のみを選択できるリスト
- マルチリスト - 複数の値を選択できるリスト

属性、プロパティおよびセルはすべてリストにできます。Agile API の `IAgileList` インターフェースには、リストを使用するためのメソッドが用意されています。このインターフェースは、すべての Agile リストで使用する一般化されたデータ構造です。`IAgileList` は、使用可能なリスト値のツリー構造を表すため、`ITreeNode` インターフェースを拡張しています。

`ITreeNode.addChild()` を使用すると、値をリストに追加できます。すべてのリスト値は一意である必要があります。リスト値を追加した後は、そのリスト値を破棄することによって、リスト値を選択できないようにできます。

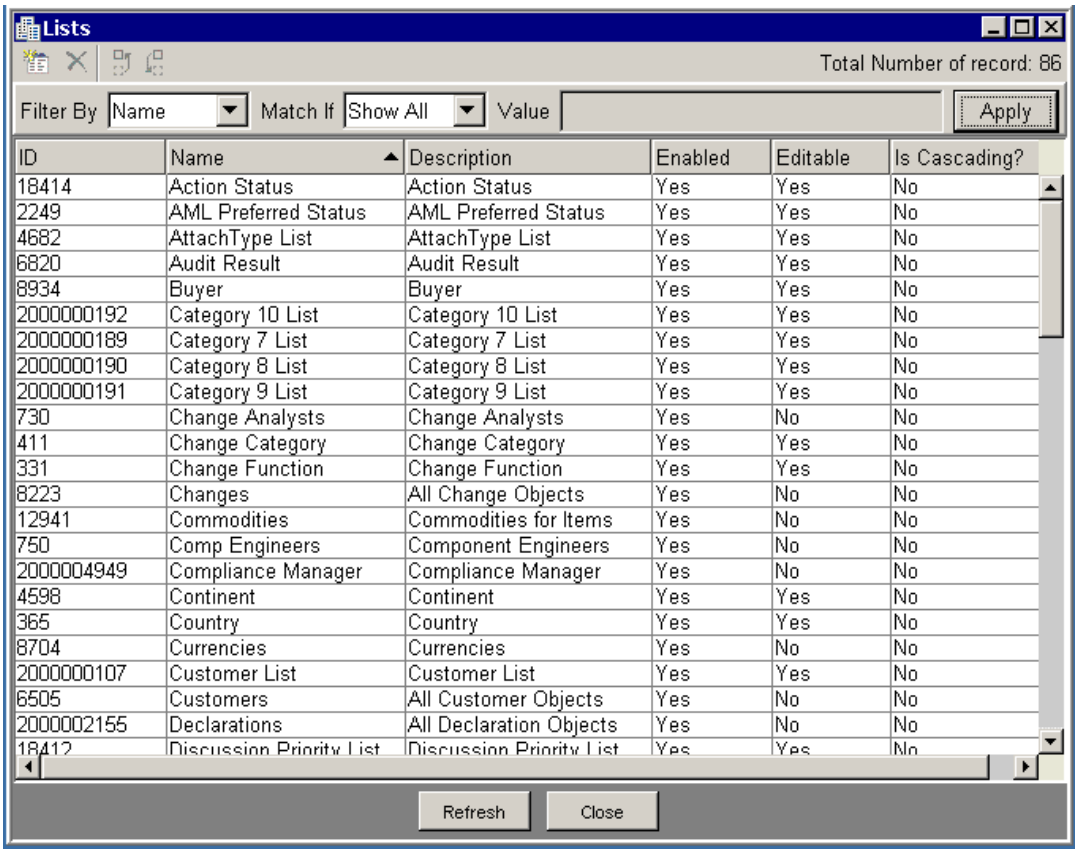
リスト ライブラリ

Agile Java クライアントで、管理者は、[ユーザー定義 1] および [ユーザー定義 2] リスト属性に使用できるカスタム リストを定義できます。カスタム リストは、Agile API を使用して定義することもできます。`IListLibrary` インターフェースには、Agile Java クライアントのリスト ライブラリと同等の機能が用意されています。`IAdminList` インターフェースを使用すると、リストの値またはプロパティを変更できます。

リスト ライブラリを取得するには、`IAdmin.getListLibrary()` メソッドを使用します。次に、`IListLibrary` インターフェースを使用して新規のカスタム リストを作成し、既存のリストを使用します。`AdminListConstants` には、リスト ライブラリ内の各リストに対する ID が用意されています。

注意 Agile API では、Agile Java クライアントのリスト ライブラリに表示されない内部の Agile リストをいくつかサポートしています。

図 5: リスト ライブラリ

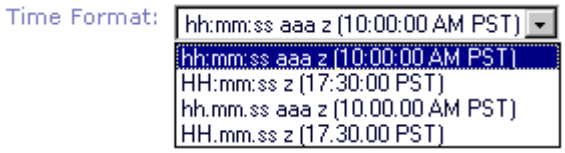


ID	Name	Description	Enabled	Editable	Is Cascading?
18414	Action Status	Action Status	Yes	Yes	No
2249	AML Preferred Status	AML Preferred Status	Yes	Yes	No
4682	AttachType List	AttachType List	Yes	Yes	No
6820	Audit Result	Audit Result	Yes	Yes	No
8934	Buyer	Buyer	Yes	Yes	No
2000000192	Category 10 List	Category 10 List	Yes	Yes	No
2000000189	Category 7 List	Category 7 List	Yes	Yes	No
2000000190	Category 8 List	Category 8 List	Yes	Yes	No
2000000191	Category 9 List	Category 9 List	Yes	Yes	No
730	Change Analysts	Change Analysts	Yes	No	No
411	Change Category	Change Category	Yes	Yes	No
331	Change Function	Change Function	Yes	Yes	No
8223	Changes	All Change Objects	Yes	No	No
12941	Commodities	Commodities for Items	Yes	No	No
750	Comp Engineers	Component Engineers	Yes	No	No
2000004949	Compliance Manager	Compliance Manager	Yes	No	No
4598	Continent	Continent	Yes	Yes	No
365	Country	Country	Yes	Yes	No
8704	Currencies	Currencies	Yes	No	No
2000000107	Customer List	Customer List	Yes	Yes	No
6505	Customers	All Customer Objects	Yes	No	No
2000002155	Declarations	All Declaration Objects	Yes	No	No
18412	Discussion Priority List	Discussion Priority List	Yes	Yes	No

シングルリストのリスト

シングルリストの属性がセルに表示され、そのリストから 1 つの値だけを選択することができます。次の図は、Agile Web クライアントの [時刻フォーマット] フィールドです。これはシングルリストのセルになっています。

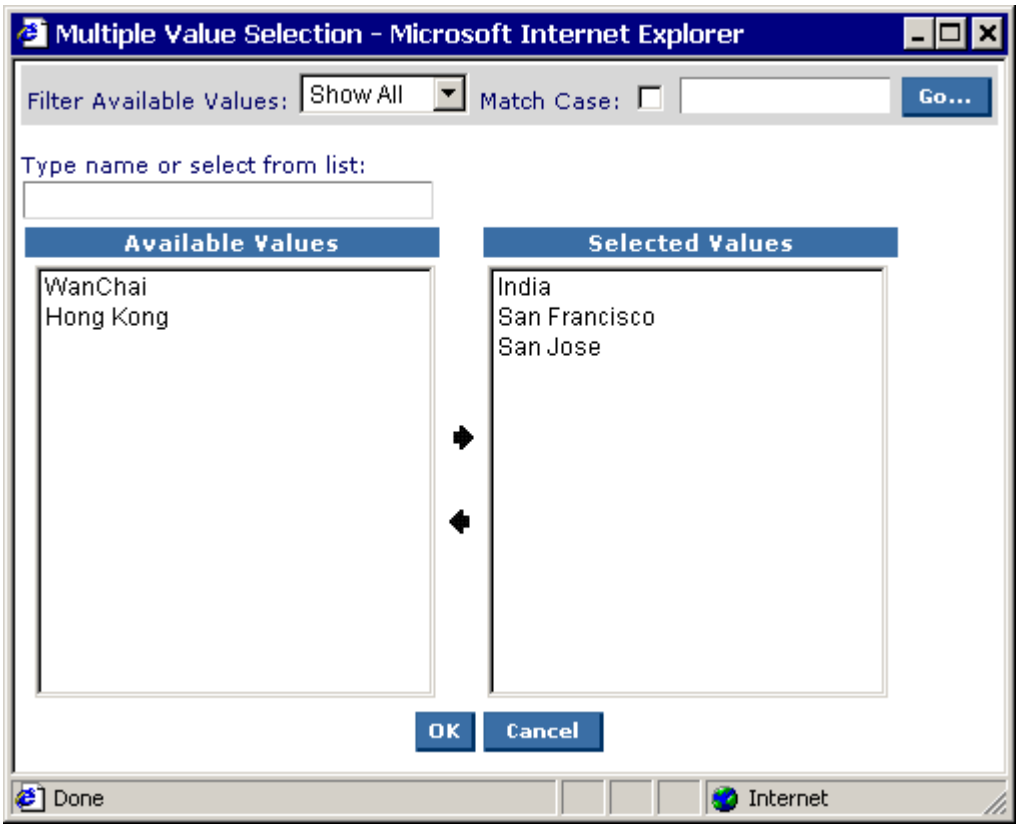
図 6: Agile Web クライアントのシングルリストのセル



マルチリストのリスト

マルチリストの属性がセルに表示され、そのリストから複数の値を選択することができます。Agile Web クライアントでは、次の図に示すように [複数の値の選択] ウィンドウを使用し、複数の値を選択してマルチリストのセルに追加することができます。

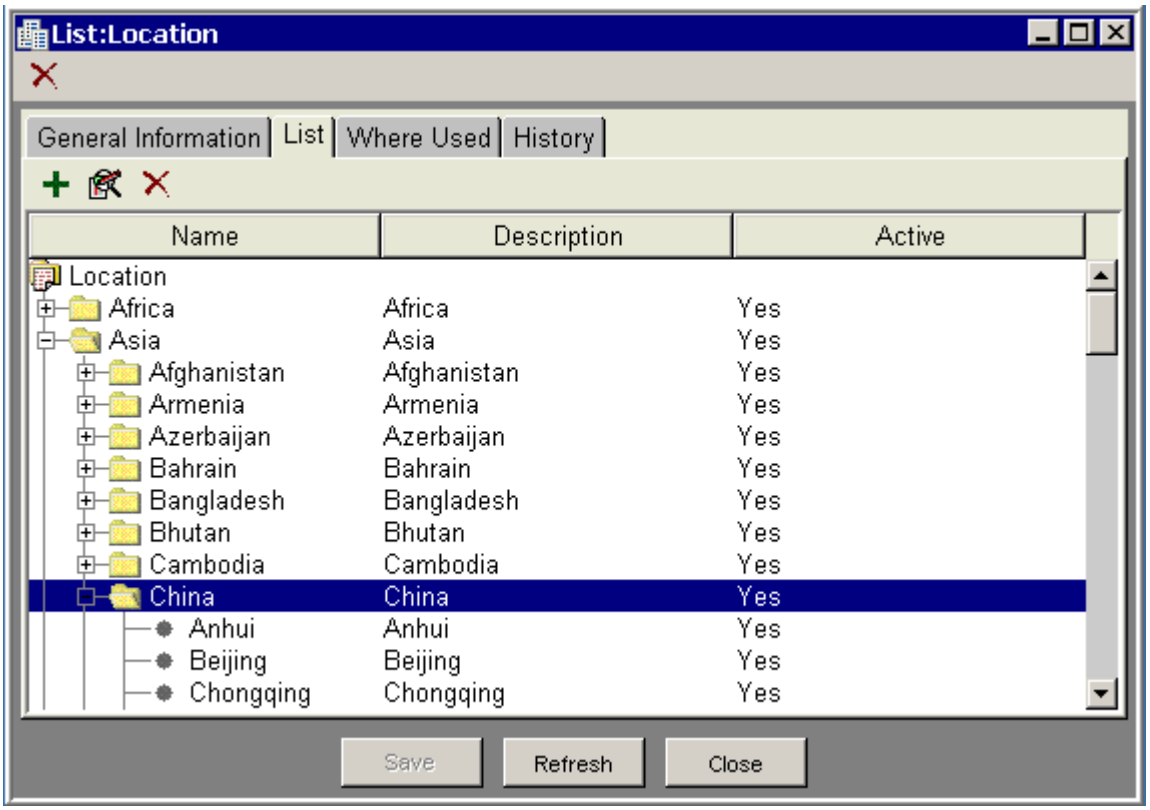
図 7: Agile Web クライアントの [複数の値の選択] ウィンドウ



カスケード リスト

Agile Java クライアントでは、シングルリストの属性を設定して、複数の階層レベルを指定できます。複数の階層レベルを持つリストは、カスケード リストと呼ばれます。次の図は、Agile Java クライアントに設定された [場所] リスト (カスケード リスト) を示しています。このリストは、大陸、国および市町村のレベルに分かれた構造になっています。

図 8: Agile Java クライアントでのカスケード リストの設定



注意 [場所] リストは、Agile PLM で出荷する唯一のカスケード リストです。ただし、独自のカスケード リストを定義することもできます。

IAgileList を使用するメソッド

`IAgileList` インターフェースには、リストから選択した値を取得して設定するのに必要なメソッドが用意されています。`IAgileList` インターフェースは、値オブジェクトをツリー構造で表すため、`ITreeNode` を拡張しています。

次の Agile API メソッドは、`IAgileList` オブジェクト (または `IAgileList` オブジェクトの配列) を返します。

- `IAdminList.getValues()`
- `IAdminList.setValues(IAgileList)`
- `IAttribute.getAvailableValues()`

- IAttribute.setAvailableValues(IAgileList)
- IAgileList.getSelection()
- ICell.getAvailableValues()
- IListLibrary.createAdminList(java.util.Map)
- IListLibrary.getAdminList(java.lang.Object)
- IListLibrary.getAdminLists()
- IProperty.getAvailableValues()

次の各メソッドは、関連する属性、セルまたはプロパティがリストの場合 (データ タイプが SingleList または MultiList の場合)、IAgileList を返すか、IAgileList パラメータを要求します。

- ICell.getValue() - シングルリストおよびマルチリストのセルの場合、返される Object は IAgileList です。
- ICell.setValue(java.lang.Object value) - シングルリストおよびマルチリストのセルの場合、value は IAgileList です。
- IProperty.getValue() - シングルリストおよびマルチリストのプロパティの場合、返される Object は IAgileList です。
- IProperty.setValue(java.lang.Object value) - シングルリストおよびマルチリストのプロパティの場合、value は IAgileList です。
- IRow.getValue(java.lang.Object cellId) - シングルリストおよびマルチリストのセルの場合、返される Object は IAgileList です。
- IRow.getValues() - 行のシングルリストまたはマルチリストの各セルの場合、返される Map オブジェクトには、IAgileList が含まれます。
- IRow.setValue(java.lang.Object cellId, java.lang.Object value) - cellID でシングルリストまたはマルチリストのセルが指定されている場合、value は IAgileList です。
- IRow.setValues(java.util.Map map) - 行のシングルリストまたはマルチリストの各セルの場合、map には、IAgileList が含まれます。

リスト値の選択

リスト値を選択するには、そのリストがシングルリストかマルチリストのリストかにかかわらず、最初にリストの利用可能な値を取得する必要があります。これによって、選択した値を設定できます。リスト値を選択した後は、セルまたはプロパティにその値を設定して選択内容を保存します。

次の例は、属性の [表示] プロパティの値を変更する方法を示しています。[表示] プロパティはシングルリスト プロパティで、有効値は No と Yes (つまり 0 と 1) です。

例: 属性の [表示] プロパティの変更

```
try {
  // Get the Admin instance
  IAdmin admin = m_session.getAdminInstance();
  // Get part  sub-class
   IAgileClass partClass =
admin.getAgileClass(ItemConstants.CLASS_PART);
```

```
  // Get the "Page Two.List03" attribute
  IAttribute attr =
partClass.getAttribute(ItemConstants.ATT_PAGE_TWO_LIST03);

  // Get the Visible property
  IProperty propVisible =
attr.getProperty(PropertyConstants.PROP_VISIBLE);

  // Get all available values for the Visible property
  IAgileList values = propVisible.getAvailableValues();

  // Set the selected list value to "Yes"
  values.setSelection(new Object[] { "Yes" });
  // Instead of setting the selection to "Yes", you could also
  // specify the corresponding list value ID, as in the following line:
  // values.setSelection(new Object[] { new Integer(1)});

  // Set the value of the property
  propVisible.setValue(values);

} catch (APIException ex) {
  System.out.println(ex);
}
```

IAgileList.setSelection() メソッドを使用するときは、childNodes パラメータに対して String[]、Integer[] または IAgileList[] 値を指定できます。値を IAgileList オブジェクトから選択するときは、そのオブジェクトの String 表現または Integer ID を使用できます。

現在リストに対して選択されている値を取得するには、IAgileList.getSelection() メソッドを使用します。シングルリストのセルまたはプロパティの場合、getSelection() は 1 つの IAgileList オブジェクトを含む配列を返します。マルチリストのセルまたはプロパティの場合、getSelection() は 1 つ以上の IAgileList オブジェクトを含む配列を返します。

次の例は、getSelection() も含めた複数の IAgileList メソッドを使用する方法を示しています。

例: [表示] プロパティの現在のリスト値の取得

```
try {
  // Get the Admin instance
  IAdmin admin = m_session.getAdminInstance();
  // Get the Parts class
  IAgileClass partClass =
admin.getAgileClass(ItemConstants.CLASS_PARTS_CLASS);

  // Get the "Page Two.List03" attribute
  IAttribute attr =
partClass.getAttribute(ItemConstants.ATT_PAGE_TWO_LIST03);

  // Get the Visible property
  IProperty propVisible =
attr.getProperty(PropertyConstants.PROP_VISIBLE);

  // Get the current value of the Visible property
  IAgileList value = (IAgileList)propVisible.getValue();
```

```
  // Print the current value
  System.out.println(value); // Prints "Yes"

  // Print the list value ID
  System.out.println(value.getSelection()[0].getId()); // Prints 1

  // Print the list value
  System.out.println(value.getSelection()[0].getValue()); // Prints
"Yes"

} catch (APIException ex) {
  System.out.println(ex);
}
```

リストは複数の属性 (異なるクラスの属性の場合でも) に対して再利用できます。次の例では、[ユーザー定義 1] 属性の利用可能な値のリストを再利用して、[ユーザー定義 2] リスト属性の利用可能な値のリストを設定しています。

例: 異なる属性のリスト値の再利用

```
try {
  // Get the Admin instance
  IAdmin admin = m_session.getAdminInstance();
  // Get the Parts class
  IAgileClass partClass =
admin.getAgileClass(ItemConstants.CLASS_PARTS_CLASS);

  // Get the "Page Two.List01" attribute
  IAttribute attr1 =
partClass.getAttribute(ItemConstants.ATT_PAGE_TWO_LIST01);

  // Get the "Page Three.List01" attribute
  IAttribute attr2 =
partClass.getAttribute(ItemConstants.ATT_PAGE_THREE_LIST01);

  // Set the available values for the list, using values from "Page
Two.List01"
  attr2.setAvailableValues(attr1.getAvailableValues());
} catch (APIException ex) { System.out.println(ex);
}
```

動的リストの使用

Agile サーバには、静的リストおよび動的リストの両方があります。静的リストには、実行時に変更されない値が含まれます。動的リストには、実行時に更新される値が含まれます。管理者権限のあるユーザーは、静的リストを変更し、新しい値を追加して現在値を破棄できます。動的リストは変更できません。したがって、動的リストの [編集可能] プロパティは、[いいえ] に設定されています。

一部の動的リストには、数千の値のオブジェクトを含めることができます。このようなリストの例には、アイテム、変更およびユーザーのリストがあります。これらのリストは、[ユーザー定義 1] と [ユーザー定義 2] フィールドに使用できますが、それらのリストに対して値を列挙することはできません。

同様に、Agile SDK オブジェクトのリストも値を列挙できる場合と列挙できない場合があります。特定のリストが列挙可能な場合は、そのリストの内容を読み取ることができます。リストが列挙不可能な場合、そのリストには直接アクセスできません。列挙不可能なリストの場合は、オブジェクトのリストで使用されている Agile クラスを検索して、リストが参照しているオブジェクトを取得します。オブジェクトの列挙プロパティはサーバでハードコードされているため、変更できません。

動的リストの値が列挙可能かどうかを判断するには、次の例に示すように IAgileList.getChildNodes() を使用します。getChildNodes() がヌルを返した場合、リストの値は列挙できません。ただし、リストの値を選択することはできます。

例: 動的リストの値が列挙可能かどうかの確認

```
private void setPageTwoListValue(IItem item) throws APIException {
  // Get the "Page Two.List01" cell
  ICell cell = item.getCell(CommonConstants.ATT_PAGE_TWO_LIST01);

  // Get available values for the list
  IAgileList values = cell.getAvailableValues();

  // If the list cannot be enumerated, set the selection to the current
user
  if (values.getChildNodes() == null) {
    values.setSelection(new
Object[]{m_session.getCurrentUser()});
    cell.setValue(values);
  }
}
private void setPageTwoMultilistValue(IItem item) throws
APIException {
  // Get the "Page Two.Multilist01" cell
  ICell cell =
item.getCell(CommonConstants.ATT_PAGE_TWO_MULTILIST01);

  // Get available values for the list
  IAgileList values = cell.getAvailableValues();

  // If the list cannot be enumerated, set the selection to
an array of users
  if (values.getChildNodes() == null) {
    IAgileClass cls =
cell.getAttribute().getListAgileClass();
    if (cls != null) {
      IUser user1 = (IUser)m_session.getObject(cls,
"hhawkes");
      IUser user2 = (IUser)m_session.getObject(cls,
"ahitchcock");
      IUser user3 = (IUser)m_session.getObject(cls,
"jhuston");
      Object[] users = new Object[] {user1, user2, user3};
      values.setSelection(users);
      cell.setValue(values);
    }
  }
}
```

ライフサイクル フェーズ セルの使用

[ライフサイクル フェーズ] 属性はシングルリスト データ タイプです。Agile PLM システム内の各サブクラスは、異なるライフサイクル フェーズを使用して定義できます。したがって、リストの利用可能な値を取得するには、その前に、サブクラスのライフサイクル フェーズ セルを取得する必要があります。IAttribute.getAvailableValues() を使用して、サブクラス固有のセルではなく [ライフサイクル フェーズ] 属性の利用可能な値を取得した場合は、空の IAgileList オブジェクトが返されます。次の例は、ライフサイクル フェーズ セルを使用する方法を示しています。

例: ライフサイクル フェーズ セルの使用

```
private static void setLifecyclePhase(IItem item) throws APIException
{
  // Get the Lifecycle Phase cell
  ICell cell =
item.getCell(ItemConstants.ATT_TITLE_BLOCK_LIFECYCLE_PHASE);

  // Get available list values for Lifecycle Phase
  IAgileList values = cell.getAvailableValues();

  // Set the value to the second phase
  values.setSelection(new Object[] { new Integer(1)});
  cell.setValue(values);
}
```

リスト ライブラリからのリストの選択

IListLibrary インターフェースを使用すると、Agile リストのライブラリを使用できます。既存のリストをロードしたり、新規リストを作成することができます。既存のリストをロードするには、IListLibrary.getAdminList() を使用します。リストの文字列名は「Disposition」のように指定できます。また、ID または AdminListConstants のいずれかでリストを指定することもできます (LIST_DISPOSITION_SELECTION など)。リスト ライブラリからリストを使用するには、その前に、そのリストが有効であることを確認してください。

カスケード リストは、シングルリストの属性でのみ使用し、マルチリストの属性では使用しません。リスト ライブラリからリストを選択するときは、IAdminList.isCascaded() を使用して、そのリストがカスケード リストかどうかを確認してください。

次の例は、[ユーザー定義 1] リスト属性を設定して「Users」というリストを使用する方法を示しています。

例: 属性の設定による Agile リストの使用

```
try {
  IAgileList values = null;
  // Get the Admin instance
  IAdmin admin = m_session.getAdminInstance();

  // Get the List Library
  IListLibrary listLib = admin.getListLibrary();

  // Get the Parts class
  IAgileClass partClass =
admin.getAgileClass(ItemConstants.CLASS_PARTS_CLASS);
```

```
  // Get the "Page Two.List01" attribute
  IAttribute attr =
partClass.getAttribute(ItemConstants.ATT_PAGE_TWO_LIST01);

  // Make the list visible
  IProperty propVisible =
attr.getProperty(PropertyConstants.PROP_VISIBLE);
  values = propVisible.getAvailableValues();
  values.setSelection(new Object[] { "Yes" });
  propVisible.setValue(values);

  // Change the name of the attribute to "Project Manager"
  IProperty propName =
attr.getProperty(PropertyConstants.PROP_NAME);
  propName.setValue("Project Manager");

  // Get the list property
  IProperty propList =
attr.getProperty(PropertyConstants.PROP_LIST);

  // Use the Users list from the list library.
  IAdminList users =
listLib.getAdminList(AdminListConstants.LIST_USER_OBJECTS);
  if (users != null ) {
    if (users.isEnabled()) {
      propList.setValue(users);
    } else {
      System.out.println("Users list is not enabled.");
    }
  }

  // Specify the Default Value to the current user
  IProperty propDefValue =
attr.getProperty(PropertyConstants.PROP_DEFAULTVALUE);
  values = propDefValue.getAvailableValues();
  values.setSelection(new
Object[]{m_session.getCurrentUser()});
  propDefValue.setValue(values);

} catch (APIException ex) {
  System.out.println(ex);
}
```

IListLibrary.getAdminList() を使用してユーザー定義リストを選択するときは、名前または ID でリストを指定できます。すべてのリスト名は一意である必要があります。次の例は、「Colors」という Agile リストを選択する方法を示しています。

例:「Colors」という名前のリストの選択

```
private void selectColorsList(IAttribute attr, IListLibrary
m_listLibrary) throws APIException {
  // Get the List property
  IProperty propList =
attr.getProperty(PropertyConstants.PROP_LIST);

  // Use the Colors list
  IAdminList listColors =
m_listLibrary.getAdminList("Colors");
  if (listColors != null ) {
```

```
        if (listColors.isEnabled()) {
          propList.setValue(listColors);
        } else {
          System.out.println("Colors list is not enabled.");
        }
      }
    }
```

カスタム リストの作成

Agile API を使用すると、異なるクラスのリスト属性を変更して、[ユーザー定義 1] および [ユーザー定義 2] のカスタム リスト属性を設定できます。また、これらのリスト属性をカスタマイズして、簡易リストまたはマルチリストを作成することができます。さらに、リストを重ねて表示し、複数のレベルを設定することもできます。

Agile Java クライアントで、管理者は [管理] > [データ設定] > [リスト] の順に選択して、カスタム リストのライブラリを設定できます。Agile API の IListLibrary インターフェースには、[管理] > [データ設定] > [リスト] を順に選択した場合と同等の機能が用意されています。IAdminList インターフェースには、各リストを設定およびカスタマイズするための機能が用意されています。

簡易リストの作成

新規リストを作成するには、IListLibrary.createAdminList() メソッドを使用します。このメソッドには map パラメータを指定します。createAdminList() とともに渡す map には、次の IAdminList フィールドの値が含まれている必要があります。

- ATT_NAME - リストの文字列名。これは必須フィールドです。リスト名は一意である必要があります。
- ATT_DESCRIPTION - リストの文字列の説明。これはオプションのフィールドです。デフォルト値は空の文字列です。
- ATT_ENABLED - リストが有効かどうかを指定するブール値。これはオプションのフィールドです。デフォルト値は false です。
- ATT_CASCADED - リストに複数のレベルを含めるかどうかを指定するブール値。これはオプションのフィールドです。デフォルト値は false です。リストを作成した後は、ATT_CASCADED の値は変更できません。

リストを作成した後は、IAdminList インターフェースを使用してそのリストを有効または無効にし、値を設定できます。

次の例は、「Colors」という新規リストを作成する方法を示しています。このリストは、1 レベルのみの簡易リストです。

例: 簡易リストの作成

```
try {
  // Get the Admin instance
  IAdmin admin = m_session.getAdminInstance();
  // Get the List Library
  IListLibrary listLib = admin.getListLibrary();

  // Create a new Admin list
  HashMap map = new HashMap();
  String name = "Colors";
```

```
  map.put(IAdminList.ATT_NAME, name);
  map.put(IAdminList.ATT_DESCRIPTION, name);
  map.put(IAdminList.ATT_ENABLED, new Boolean(true));
  map.put(IAdminList.ATT_CASCADED, new Boolean(false));
  IAdminList listColors = listLib.createAdminList(map);

  // Add values to the list
  IAgileList list = listColors.getValues(); //The list is empty at this
point.
  list.addChild("Black");
  list.addChild("Blue");
  list.addChild("Green");
  list.addChild("Purple");
  list.addChild("Red");
  list.addChild("White");
  listColors.setValues(list);
} catch (APIException ex) {
  System.out.println(ex);
}
```

文字列値を含むリストでは、大文字と小文字が区別されます。つまり、リストには、同じ値を大文字、小文字およびそれらの混合で含めることができますが、これは適切でない場合があります。たとえば、次のコード例では、1 つの色について 3 種類の値を「Colors」リストに追加しています。

例: リストへの大文字と小文字を区別した値の追加

```
  IAgileList list = listColors.getValues(); //The list is empty at this
point.
  list.addChild("Black");
  list.addChild("BLACK");
  list.addChild("black");
  list.addChild("Blue");
  list.addChild("BLUE");
  list.addChild("blue");
  list.addChild("Green");
  list.addChild("GREEN");
  list.addChild("green");
  list.addChild("Purple");
  list.addChild("PURPLE");
  list.addChild("purple");
  list.addChild("Red");
  list.addChild("RED");
  list.addChild("red");
  list.addChild("White");
  list.addChild("WHITE");
  list.addChild("white");
```

既存リストの変更による新規リストの自動作成

各リスト属性は、その値について Agile リストを参照する必要があります。Agile リストを取得してその値を変更し、リストを保存せずにリスト属性に対してその値を使用すると、Agile API では新規リストが自動的に作成されます。次の例では、「Colors」リストを取得した後、このリストを使用してリスト フィールドに値が挿入される前に、新しい値 “Violet” がリストに追加されます。IAttribute.setAvailableValues() が呼び出されると、新規リストが作成されます。

注意 Agile API で自動的に作成されたリストの名前は、接頭辞「SDK」の後にランダム番号が続きます。このリスト名は、必要に応じて変更できます。

例: 既存リストの変更による新規リストの自動作成

```
try {
  // Get the Colors list
  IAdminList listColors = m_listLibrary.getAdminList("Colors");

  // Get the Parts class
  IAgileClass partsClass =
admin.getAgileClass(ItemConstants.CLASS_PARTS_CLASS);

  // Get the "Page Two.List01" attribute
  IAttribute attr =
partsClass.getAttribute(ItemConstants.ATT_PAGE_TWO_LIST01);

  // Get the color values
  IAgileList values = listColors.getValues();

  // Add a new color
  values.addChild("Violet");

  // Set the available list values for "Page Two.List01". Because the
list
  // was modified, a new AdminList is created automatically.
  attr.setAvailableValues(values);
} catch (APIException ex) { System.out.println(ex);
}
```

カスケード リストの作成

カスケード リストは、複数のレベルがあるリストです。シングルリストの属性とセルは、簡易リストのかわりにカスケード リストを使用して設定できます。

注意 リストをカスケード リストとして設定した後は、そのリストをフラット リストに変更することはできません。リストを作成した後は、IAdminList.ATT_CASCADED の値は変更できません。

次の例は、「Field Office」という新規のカスケード リストを作成する方法を示しています。このリストには、2 つのレベルがあります。

重要 カスケード リストにレベル名を設定するときは、次の 2 つの例に示すように、最初のレベルは常にインデックス 0 から開始し、後続のレベルのインデックスを 1 ずつ増分します。

例: カスケード リストの作成

```
try {
  // Get the Admin instance
  IAdmin admin = m_session.getAdminInstance();
  // Get the List Library
  IListLibrary listLib = admin.getListLibrary();

  // Create a new Admin list
  HashMap map = new HashMap();
  String name = "Field Office";
  map.put(IAdminList.ATT_NAME, name);
  map.put(IAdminList.ATT_DESCRIPTION, name);
```

```
        map.put(IAdminList.ATT_ENABLED, new Boolean(true));
        map.put(IAdminList.ATT_CASCADED, new Boolean(true));
        IAdminList listFO = listLib.createAdminList(map);

        // Get the empty list
        IAgileList list = listFO.getValues();

        // Add the list of countries
        IAgileList india = (IAgileList)list.addChild("India");
        IAgileList china = (IAgileList)list.addChild("China");
        IAgileList usa = (IAgileList)list.addChild("USA");
        IAgileList australia = (IAgileList)list.addChild("Australia");

        // Add the list of cities
        india.addChild("Bangalore");
        china.addChild("Hong Kong");
        china.addChild("Shanghai");
        china.addChild("Suzhou");
        usa.addChild("San Jose");
        usa.addChild("Milpitas");
        usa.addChild("Seattle");
        usa.addChild("Jersey City");
        australia.addChild("Sidney");

        // Save the list values
        listFO.setValues(list);

      // Set level names starting with index 0 for level 1.
        list.setLevelName(0, "Field Office Country");
        list.setLevelName(1, "Field Office City");

      } catch (APIException ex) {
        System.out.println(ex);
      }
```

カスケード リストでは、リストで使用するレベル名は一意である必要があり、リスト間でレベル名を共有することはできません。レベル名は内部的に保存されますが、現在は、Agile Java クライアントおよび Web クライアントでは表示されません。レベル名が必要なのは、作成したカスケード リストの UI にレベル名を表示する場合のみです。

`IAdminList.setValues()` メソッドを呼び出すと、有効な ID が各リスト値に割り当てられます。有効な ID があるのは、リーフ ノード (カスケード リストの最低レベルのノード) のみです。前述の例では、city ノードがリーフ ノードです。他のすべてのノードの ID はヌルです。`IAgileList` オブジェクトを選択するには、この ID を使用できます。

リスト値とその親ノードは、親ノードを追加してからそのサブノードを追加するのではなく、1 つのステートメントで追加できます。ノードを区切るには | 文字を使用します。これらのノードは、文字列でレベルを表します。次の例では、前述の例のコードが一部置き換えられています。どちらの例も同じリスト値を追加する方法を示していますが、次の例ではコードの行数が少なくなっています。

例: カスケード リストへの親ノードとサブノードの追加

```
        // Get the list values
        IAgileList list = listFO.getValues(); // The list is empty
        at this point.
```

```
  // Add nodes
  list.addChild("India|Bangalore");
  list.addChild("Hong Kong|Hong Kong");
  list.addChild("China|Suzhou");
  list.addChild("USA|San Jose");
  list.addChild("USA|Milpitas");
  list.addChild("USA|Jersey City");
  list.addChild("Australia|Sidney");

// Save the list values
listFO.setValues(list);

  // Set level names
  list.setLevelName(0, "Field Office Country");
  list.setLevelName(1, "Field Office City");
```

リストのデータ タイプの確認

リストには、任意の Agile データ タイプのオブジェクトを含めることができます。したがって、リスト値を取得したり設定する前には、リスト内のオブジェクトのデータ タイプを確認する必要があります。カスケードリストを使用する場合、データ タイプが各レベルで異なっている場合があります。リストのデータ タイプを確認するにはいくつかの方法があります。

- リスト ライブラリの事前定義済みのリストの場合は、IAdminList.getListDataType() を使用してデータ タイプを取得します。
- リスト レベルが 1 つのみのシングルリスト属性およびマルチリスト属性の場合は、IAttribute.getListDataType() メソッドを使用してリスト全体のデータ タイプを取得します。
- カスケード リスト内のレベルの場合は、IAgileList.getLevelType() メソッドを使用して特定レベルのデータ タイプを取得します。

例: リストのデータ タイプの確認

```
public void setDefaultValue() throws APIException {
  // Get the Parts class
  IAgileClass partClass =
m_admin.getAgileClass(ItemConstants.CLASS_PARTS_CLASS);

  // Get the "Page Two.List01" attribute
  IAttribute attr =
partClass.getAttribute(ItemConstants.ATT_PAGE_TWO_LIST01);

  switch (attr.getListDataType()) {
    case DataTypeConstants.TYPE_OBJECT:
      //Add code here to handle Object values
      break;

    case DataTypeConstants.TYPE_STRING:
      //Add code here to handle String values
      break;
    default:
      //Add code here to handle other datatypes
  }
}
```

リストの変更

作成したリストは、次の方法で変更できます。

- リストへの値の追加
- リスト値の破棄
- リスト名と説明の設定
- カスケード リストのレベル名の設定
- リストの有効化または無効化
- リストの削除
- リストに追加した値の変更または削除

リストへの値の追加

次の例は、リストにいくつかの値を追加する方法を示しています。リストに値を追加する前に、ITreeNode.getChildNode() メソッドを使用して、値がすでに存在していないことを確認します。

例: リストへの値の追加

```
private static void updateProductLinesList() throws APIException {
  // Get the Admin instance
  IAdmin admin = m_session.getAdminInstance();

  // Get the List Library
  IListLibrary listLib = admin.getListLibrary();

  // Get the Product Lines list
  IAdminList listProdLine = listLib.getAdminList("Product Line");

  // Add values to the list
  IAgileList listValues = listProdLine.getValues();
  addToList(listValues, "Saturn");
  addToList(listValues, "Titan");
  addToList(listValues, "Neptune");
  listProdLine.setValues(listValues);
}
private static void addToList(IAgileList list, String value) throws
APIException {
  if (list.getChildNode(value) == null) {
    list.addChild(value);
  }
}
```

リスト値の破棄

リスト エントリを破棄することによって、リスト値を選択できないようにできます。ただし、IProperty.getAvailableValues() メソッドを呼び出すと、返された IAgileList オブジェクトに破棄されたリスト値が含まれる場合があります。これは、リスト値が破棄としてマークされると、サーバでは、破棄されたリスト値を使用する既存のオブジェクト用にその値を保持し続けるためです。

次の例は、リスト値が破棄済みかどうかを確認し、リスト値を破棄する方法を示しています。

例: リスト値の破棄

```
public void checkIfObsolete(IAgileList list) throws APIException {
  if (list != null ) {
    if (list.isObsolete() == false) {
      System.out.println(list.getValue());
    }
  }
}
public void setObsolete(IAgileList list, String value) throws
APIException {
  if (list != null ) {
    list.setObsolete(true);
      System.out.println(list.getValue() + " is now obsolete.");
  }
}
```

リスト名と説明の設定

リストを作成するには、そのリストに一意の名前を指定する必要があります。したがって、IListLibrary.createAdminList() を使用する場合は、IAdminList.ATT_NAME フィールドの値を渡す必要があります。他の IAdminList フィールド (ATT_DESCRIPTION など) はオプションです。リストを作成した後は、名前と説明を変更できます。次の例は、リストの名前と説明を設定する方法を示しています。

例: リスト名と説明の設定

```
try {
  IAdminList list = m_listLibrary.getAdminList("Packaging Styles");
  list.setName("Packaging Color Codes");
  list.setDescription("Color codes for product packaging");
} catch (APIException ex) {
  System.out.println(ex);
}
```

カスケード リストのレベル名の設定

リスト名と同様に、リストのレベル名は一意である必要があります。別のカスケード リストで使用されているレベル名を再利用することはできません。特定の名前のリストがすでに存在するかどうかを確認するには、IListLibrary.getAdminList() を使用します。次のいずれかの方法でカスケード リストのレベル名を設定します。

- IAgileList.setLevelName(int, String) - 指定したレベルにレベル名を設定する
- IAgileList.setLevelName(String) - 現在のレベルのレベル名を設定する

カスケード リストのレベル名を設定する方法の例については、131 ページの「カスケード リストの作成」を参照してください。

注意 カスケード リストのレベル名は、Agile Java クライアントまたは Web クライアントに表示されません。ただし、Agile SDK を使用して作成したクライアントにはレベル名を表示できます。

リストの有効化または無効化

カスタム リストを作成するときは、IAdminList.ATT_ENABLED フィールドを使用して、そのリストを有効にするかどうかを指定することができます。このフィールドを省略すると、リストはデフォルトで無効になります。次の例は、作成したリストを有効または無効にする方法を示しています。

例: リストの有効化または無効化

```
public void enableList(IAdminList list) throws APIException {
    list.enable(true);
    System.out.println("List " + list.getName() + " enabled.");
}
public void disableList(IAdminList list) throws APIException {
    list.enable(false);
    System.out.println("List " + list.getName() + " disabled.");
}
```

リストの削除

読み取り専用でなく、Agile データオブジェクトで現在使用されていないリストは、削除できます。それ以外の場合は、IAdminList.delete() メソッドで例外が発生します。リストを削除すると、そのリストは完全に削除されます。削除を取り消すことはできません。

次の例は、リストを削除する方法を示しています。

例: リストの削除

```
public void deleteList(IAdminList list) throws APIException {
  // Make sure the list is not read-only
  if (!list.isReadOnly()) {
    // Delete the list
    list.delete();
    System.out.println("List " + list.getName() + " deleted.");
  } else {
    System.out.println("List " + list.getName() + " is read-only.");
  }
}
```

リスト値の変更および削除

SDK には、文字列要素エントリを変更したり Agile リストのエントリを削除するために、次のメソッドが用意されています。

- `IAgileList.setValue(Object)` メソッドは、Agile 管理リストの文字列リスト要素エントリを変更する場合に使用します。

注意 このメソッドは、文字列値にのみ適用されます。このメソッドを使用できるのは、文字列エントリを変更する場合のみで、オブジェクト エントリには使用できません。

- `IAgileList.clear()` および `ITree.removeChild(Object)` メソッドは、適用されるビジネス ルールで制限されていない Agile リスト エントリを削除する場合に使用します。

次の例では、これらのメソッドを使用して、Agile リストの値を変更およびクリアしています。

例: 管理リスト エントリの名前変更および削除

```
public void exampleClearList() throws Exception {
  IAdmin admin = m_session.getAdminInstance();
  IListLibrary listLibrary = admin.getListLibrary();
    HashMap map = new HashMap();
    String name = "Color";
    String desc = "Example";
    map.put(IAdminList.ATT_NAME, name);
    map.put(IAdminList.ATT_DESCRIPTION, desc);
    map.put(IAdminList.ATT_ENABLED, new Boolean(true));
    map.put(IAdminList.ATT_CASCADED, new Boolean(false));
    IAdminList newList = listLibrary.createAdminList(map);

IAgileList list = newList.getValues();
list.addChild("RED");
list.addChild("GREEN");
list.addChild("BLUE");
newList.setValues(list);
list = newList.getValues();

    // Removing the selection
    IAgileList agList = (IAgileList)list.getChild("BLUE");
    Object errorCode = null;
    try {
    list.removeChild(agList);
    }catch(APIException e){
    errorCode = e.getErrorCode();
    }

    //  Clear the list
    list = newList.getValues();
    list.clear();
    newList.setValues(list);

    // Clean up
    newList.delete();
  }
```

IAgileList オブジェクトのコンテンツの印刷

IAgileList オブジェクトを使用するとき、特にそのオブジェクトに複数のレベルがある場合は、リストの階層全体を印刷すると便利です。次のコードでは、IAgileList オブジェクト内に含まれるリスト ノードを印刷します。

例: IAgileList オブジェクトのリスト ノードの印刷

```
private void printList(IAgileList list, int level) throws APIException
{
  if (list != null ) {
    System.out.println(indent(level*4) + list.getLevelName() + ":" +
    list.getValue() + ":" + list.getId());
    Object[] children = list.getChildren();
    if (children != null) {
      for (int i = 0; i < children.length; ++i) {
        printList((IAgileList)children[i], level + 1);
      }
    }
  }
}
private String indent(int level) {
  if (level <= 0) {
    return "";
  }
  char c[] = new char[level*2];
  Arrays.fill(c, ' ');
  return new String(c);
}
```

第 9 章

製造拠点の管理

扱うトピックは次のとおりです。

製造拠点について

分散型製造を実践している企業は、自社の製品を複数箇所の製造拠点で製造しています。企業では、Agile PLM 拠点オブジェクトを使用して、製品の部品に関する拠点別の情報を維持することができます。たとえば、製造場所が異なると新規リビジョンの有効日が異なる場合があります。また、製造場所に応じて製造手順書が異なり、コンポーネントを購入する製造元も異なる場合があります。

アイテムのすべての製造拠点または特定の拠点を対象として、変更を反映することができます。変更の [対象アイテム] テーブルを使用すると、変更を適用する製造拠点を選択できます。アイテムの有効日と対応策は、拠点ごとに異なるものとすることができます。有効日および対応策は、ECO や SCO の [対象アイテム] タブで指定します。新しい有効日または対応策を割り当てる際に新規リビジョンを作成するには、ECO を使用します。リビジョンを進めずに拠点別の有効日および対応策を割り当てるには、SCO を使用します。

Agile PLM の製造拠点機能の詳細は、『Product Collaboration Guide』を参照してください。

拠点へのアクセスの管理

拠点の使用は、組織のライセンスに加えて、ユーザーのライセンス、役割、権限およびデフォルトの拠点プロパティによって管理されます。必要な数の製造拠点を作成できますが、有効にできる拠点の数は、組織のライセンスによって異なります。組織によっては、ユーザーが特定の拠点に関する情報のみにアクセスできるように、Agile PLM システムを設定している場合があります。

アイテムに対して拠点別の BOM を作成するには、そのアイテムのサブクラスで [拠点別の BOM] を [可] に設定する必要があります。設定しない場合、そのサブクラスのアイテムには、すべての拠点に共通の BOM が設定されます。

製造拠点の作成

製造拠点は、名前によって一意に識別されます。製造拠点を作成するには、IAgileSession.createObject メソッドを使用してクラスと拠点名の両方を指定します。

すべてのユーザーが製造拠点を作成できるわけではありません。製造拠点を作成できるのは、製造拠点オブジェクトの作成権限があるユーザーのみです。

注意 製造拠点を作成すると、そのライフサイクル フェーズがデフォルトで Disabled に設定されます。拠点を使用するには、その拠点を有効にしてください。

例: 製造拠点の作成および有効化

```
try {
  // Create a manufacturing site
  HashMap params = new HashMap();

params.put(ManufacturingSiteConstants.ATT_GENERAL_INFO_NAME,
"Taipei");
  IManufacturingSite mfrSite =
(IManufacturingSite)m_session.createObject(

ManufacturingSiteConstants.CLASS_SITE, params);
  // Enable the manufacturing site
  ICell cell = mfrSite.getCell(

ManufacturingSiteConstants.ATT_GENERAL_INFO_LIFECYCLE_PHASE);
  IAgileList values = cell.getAvailableValues();
  values.setSelection(new Object[] { "Enabled" });
  cell.setValue(values);
} catch (APIException ex) {
  System.out.println(ex);
}
```

製造拠点のロード

IManufacturingSite オブジェクトをロードするには、IAgileSession.getObject() メソッドの 1 つを使用します。次の例は、製造拠点のオブジェクト タイプを指定する 3 種類の方法を示しています。

例: 製造拠点のロード

```
try {
  // Load the Hong Kong site
  IManufacturingSite siteHK =

(IManufacturingSite)m_session.getObject(ManufacturingSiteConstants
.CLASS_SITE, "Hong Kong");
  // Load the Taipei site
  IManufacturingSite siteTaipei =

(IManufacturingSite)m_session.getObject(IManufacturingSite.OBJECT_
TYPE, "Taipei");
```

```
  // Load the San Francisco site
  IManufacturingSite siteSF =
(IManufacturingSite)m_session.getObject("Site", "San Francisco");
} catch (APIException ex) {
  System.out.println(ex);
}
```

アイテムの [拠点] テーブルの取得

各アイテムには、そのアイテムを使用できる製造拠点をリストした [拠点] テーブルがあります。アイテムの [拠点] テーブルを取得するには、DataObject.getTable() メソッドを使用します。

例: [拠点] テーブルの取得

```
//Get the Sites table
private static void getSites(IItem item) throws APIException {
  IRow row;
  ITable table = item.getTable(ItemConstants.TABLE_SITES);
  ITwoWayIterator it = table.getTableIterator();
  while (it.hasNext()) {
    row = (IRow)it.next();
    //Add code here to do something with the Sites table
  }
}
```

アイテムに関連付けられた製造拠点を確認するには、IManufacturingSiteSelectable.getManufacturingSites() メソッドを使用します。[拠点] テーブルを繰り返し処理して同じ情報を取得することも可能ですが、getManufacturingSites() メソッドを使用すると、より簡単で迅速に取得できます。getManufacturingSites() の使用例は、142 ページの「アイテムの現在の製造拠点の選択」を参照してください。

[拠点] テーブルへの製造拠点の追加

[拠点] テーブルの各行は、異なる IManufacturingSite オブジェクトを参照しています。製造拠点を [拠点] テーブルに追加するには、ITable.createRow() メソッドを使用します。

アイテムの [拠点] テーブルに製造拠点がリストされていない場合、そのアイテムはその製造拠点固有の親アイテムの BOM 内に表示されません。たとえば、アイテム P1001 を別アイテムの Taipei 固有の BOM に追加するには、P1001 の [拠点] テーブルに Taipei 拠点がリストされている必要があります。

例: [拠点] テーブルへの行の追加

```
private static void addSite(String itemNumber, IManufacturingSite
site)
  throws APIException {
    //Load the item
    IItem item = (IItem)session.getObject(IItem.OBJECT_TYPE,
itemNumber);

    //Get the Sites table
    ITable table = item.getTable(ItemConstants.TABLE_SITES);

    //Add the manufacturing site to the table
    IRow row = table.createRow(site);
}
```

アイテムの現在の製造拠点の選択

[BOM] テーブルと [製造元] テーブル (または AML) は、アセンブリで使用する製造拠点ごとに異なる場合があります。アイテムの [BOM] テーブルまたは [製造元] テーブルを取得するときは、すべての拠点の情報または特定の拠点の情報を表示できます。特定の拠点を選択すると、その拠点の情報のみがテーブルに表示されます。

IManufacturingSiteSelectable インターフェースには、アイテムの製造拠点を取得したり、設定するためのメソッドが用意されています。アイテムに対して選択した現在の製造拠点を取得するには、IManufacturingSiteSelectable.getManufacturingSite() メソッドを使用します。

例: アイテムに対して現在選択されている製造拠点の取得

```
private static IManufacturingSite getCurrentSite(IItem
item)
  throws APIException {
    IManufacturingSite site = item.getManufacturingSite();
    return site;
}
```

IManufacturingSiteSelectable.getManufacturingSites() メソッドは、アイテムの [拠点] テーブルに追加された利用可能なすべての製造拠点を取得します。

例: アイテムに関連付けられたすべての製造拠点の取得

```
private static void getItemSites(IItem item)
  throws APIException {
    IManufacturingSite[] sites =
item.getManufacturingSites();
    //Print the name of each site
    for (int i = 0; i < sites.length; ++i) {
      String siteName = (String)sites[i].getValue(
        ManufacturingSiteConstants.ATT_GENERAL_INFO_NAME
        );
      System.out.println(siteName);
    }
}
```

IManufacturingSiteSelectable.setManufacturingSite() メソッドは、アイテムに対して現在の製造拠点を設定します。アイテムには、特定の製造拠点を設定すること、拠点別でないこと、すべての拠点を使用することを指定できます。アイテムが拠点別でないことを指定するには、ManufacturingSiteConstants.COMMON_SITE を使用します。すべての拠点の使用を指定するには、ManufacturingSiteConstants.ALL_SITES 値を渡します。

アイテムに対して製造拠点を設定すると、そのアイテムは更新され、拠点別の情報が反映されます。したがって、プログラムでは、行に対しても処理を繰り返して、[BOM] テーブルと [製造元] テーブルを更新する必要があります。

例: アイテムに対する現在の製造拠点の設定

```
try {
  // Load sites
  IManufacturingSite siteSF =
(IManufacturingSite)m_session.getObject("Site", "San Francisco");
  IManufacturingSite siteHK =
(IManufacturingSite)m_session.getObject("Site", "Hong Kong");
  // Load an item
  IItem item  = (IItem)m_session.getObject("Part", "1000-02");
  // Set the Hong Kong site
  item.setManufacturingSite(siteHK);
  String desc =
(String)item.getValue(ItemConstants.ATT_TITLE_BLOCK_DESCRIPTION);
  System.out.println("Hong Kong description = " + desc);
  // Set the San Francisco site
  item.setManufacturingSite(siteSF);
  desc =
(String)item.getValue(ItemConstants.ATT_TITLE_BLOCK_DESCRIPTION);
  System.out.println("San Francisco description = " + desc);
  // Set the item to use all sites

item.setManufacturingSite(ManufacturingSiteConstants.ALL_SITES);
  desc =
(String)item.getValue(ItemConstants.ATT_TITLE_BLOCK_DESCRIPTION);
  System.out.println("All Sites description = " + desc);
  // Set the item to be common site (the item is not site-specific)

item.setManufacturingSite(ManufacturingSiteConstants.COMMON_SITE);
  desc =
(String)item.getValue(ItemConstants.ATT_TITLE_BLOCK_DESCRIPTION);
  System.out.println("Global description = " + desc);
  // Set the item to use the user's default site

item.setManufacturingSite(((IAgileList)m_session.getCurrentUser().
getValue(

UserConstants.ATT_GENERAL_INFO_DEFAULT_SITE)).getSelection()[0].ge
tValue());
  desc =
(String)item.getValue(ItemConstants.ATT_TITLE_BLOCK_DESCRIPTION);
  System.out.println("User's Default Site description = " + desc);
} catch (APIException ex) {
  System.out.println(ex);
}
```

拠点の無効化

製造拠点には 2 つのライフサイクル フェーズがあり、有効または無効のいずれかになります。拠点が無効の場合、その拠点は、拠点別の BOM、AML および変更の作成には使用できません。

製造拠点を無効にするには、[ライフサイクル フェーズ] 属性の値を `Disabled` に設定します。

例: 製造拠点の無効化

```
private static void disableSite(IManufacturingSite site)
  throws APIException {
  // Get the Lifecycle Phase cell
  ICell cell = site.getCell(
    ManufacturingSiteConstants.ATT_GENERAL_INFO_LIFECYCLE_PHASE
  );

  // Get available list values for Lifecycle Phase
  IAgileList values = cell.getAvailableValues();

  // Set the value to Disabled
  values.setSelection(new Object[] { "Disabled" });
  cell.setValue(values);
}
```

第 10 章

添付ファイルとファイル フォルダの使用

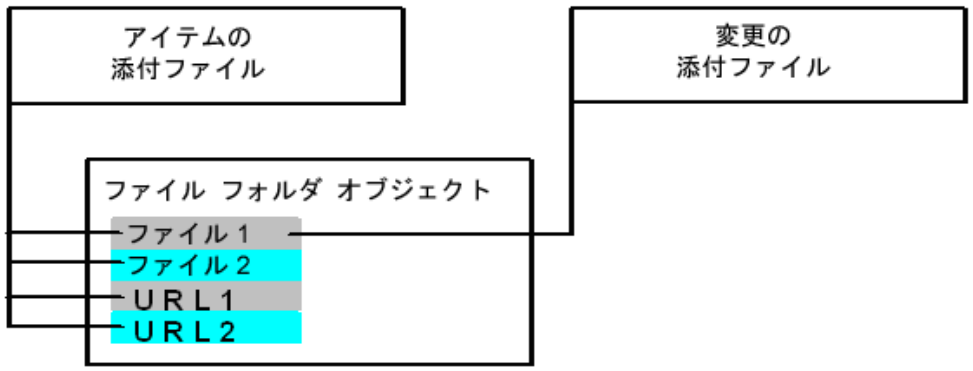
扱うトピックは次のとおりです。

添付ファイルについて

ほとんどの主要な Agile API オブジェクト (`IItem`、`IChange`、`IManufacturerPart`、`IManufacturer`、`IPackage`、`ITransferOrder`、`IUser`、`IUserGroup` など) には、[添付ファイル] テーブルがあります。このテーブルには、個別のファイル フォルダ内にあるファイルまたは URL への間接参照がリストされています。[添付ファイル] テーブル内の各行は、参照先ファイル フォルダから 1 ファイルまたはすべてのファイルを参照できます。ファイル フォルダは、ファイル サーバ格納庫に保存されている 1 つ以上の URL (インターネット アドレス) またはファイルを指定するビジネス オブジェクトです。

次の図は、複数のビジネス オブジェクト (この例ではアイテムと変更) の [添付ファイル] テーブルから、ファイル フォルダに含まれているファイルまたは URL を間接的に参照する例を示しています。

図 9: [添付ファイル] テーブル行からファイル フォルダのファイルまたは URL を間接的に参照する方法



Agile API は、添付ファイルの表示または印刷をサポートしていません。ただし、ファイルは、ダウンロードしてから、別のアプリケーションを使用して表示、編集または印刷できます。

重要 Agile PLM 添付ファイルを追加してファイル フォルダを使用するには、その前に、Agile Java クライアントで [ファイル マネージャ内部ロケーター] プロパティを設定してください。[管理] > [設定] > [サーバ設定] > [場所] > [ファイル マネージャ] > [詳細] > [ファイル マネージャ内部ロケーター] の順に選択します。値の形式は、<protocol>://<*machinename*>:<*port*>/<*virtualPath*>/services/FileServer です。たとえば、http://agileserver.agile.agilesoft.com:8080/Filemgr/services/FileServer は有効な値です。Agile PLM サーバ設定の詳細は、『Agile PLM管理者ガイド』を参照してください。

ファイル フォルダの使用

ファイル フォルダは、独自のテーブルのセットを備えた Agile PLM ビジネス オブジェクトです。これは、独立したファイル フォルダを作成およびロードして、1 つ以上のファイルをその [ファイル] テーブルに追加できることを意味します。また、ファイル フォルダは、アイテムや変更を検索するのと同様に検索することもできます。現在、デフォルトのファイル フォルダ クラスには、ファイル フォルダと履歴レポート ファイル フォルダがあり、それぞれのクラスにはサブクラスがあります。Agile PLM 管理者は、新規ファイル フォルダのサブクラスを定義できます。

`IFileFolder` は、ファイル フォルダ ビジネス オブジェクトに対応するインターフェースです。次の例は、ファイル フォルダの作成方法を示しています。

例: ファイル フォルダの作成

```
public void createFileFolder() throws Exception {
  IAgileClass attClass =
m_admin.getAgileClass(FileFolderConstants.CLASS_FILE_FOLDER);
  IAutoNumber an = cls.getAutoNumberSources()[0];
  String attNumber = an.getNextNumber();
  IFileFolder ff = (IFileFolder)m_session.createObject(attClass,
attNumber);
}
```

ファイルまたは URL をビジネス オブジェクトの [添付ファイル] テーブルの行に追加すると、関連するファイルまたは URL が含まれた新規のファイル フォルダが自動的に作成されます。参照先ファイル フォルダは、次の例に示すように、`IRow.getReferent()` メソッドを使用してロードできます。

例: [添付ファイル] テーブルへの行の追加によるファイル フォルダの作成

```
public IFileFolder addRowToItemAttachments(IItem item, File file)
throws Exception {
  ITable attTable = item.getTable(ItemConstants.TABLE_ATTACHMENTS);
  IRow row = attTable.createRow(file);
  IFileFolder ff = (IFileFolder)row.getReferent();
  return ff;
}
```

ファイル フォルダのテーブル

テーブル	定数	読み取り/書き込みモード
タイトル ブロック	TABLE_TITLEBLOCK	読み取り/書き込み
ユーザー定義 1	TABLE_PAGETWO	読み取り/書き込み
ユーザー定義 2	TABLE_PAGETHREE	読み取り/書き込み
ファイル	TABLE_FILES	読み取り/書き込み
ワークフロー	TABLE_WORKFLOW	読み取り/書き込み
関係 - 影響先	TABLE_RELATIONSHIPSAFFECTS	読み取り専用
関係 - 影響元	TABLE_RELATIONSHIPSAFFECTEDBY	読み取り専用
関係 - 参照	TABLE_RELATIONSHIPSREFERENCES	読み取り/書き込み
使用箇所	TABLE_HISTORY	読み取り専用
履歴	TABLE_ATTACHMENTS	読み取り専用

ファイル フォルダの [ファイル] テーブルの使用

ファイル フォルダの [ファイル] テーブルには、オブジェクトに関連付けられているファイルと URL がリストされています。テーブルを編集するには、最初にファイル フォルダをチェックアウトする必要があります。[ファイル] テーブルに対するファイルや URL の追加または削除は、ファイル フォルダをチェックアウトしないかぎり実行できません。

次の例は、ファイル フォルダをチェックアウトしてから、ファイルと URL を [ファイル] テーブルに追加する方法を示しています。

例: ファイル フォルダの [ファイル] テーブルへのファイルおよび URL の追加

```
public void addFiles(IFileFolder ff, File[] files, URL[] urls) throws
Exception {
  // Check out the file folder
  ff.checkOutEx();

  // Get the Files table
  ITable filesTable = ff.getTable(FileFolderConstants.TABLE_FILES);

  // Add files to the Files table
  for (int i = 0; i < files.length; ++i) {
    filesTable.createRow(files[i]);
  }
  // Add URLs to the Files table
  for (int i = 0; i < urls.length; ++i) {
    filesTable.createRow(urls[i]);
  }
  // Check in the file folder
  ff.checkIn();
}
```

IAttachmentFile インターフェースの使用

IAttachmentFile は、Agile PLM ファイル格納庫に保存されているファイルへの一般化されたアクセスを提供するインターフェースです。このインターフェースは、次の Agile API オブジェクトでサポートされています。

- ファイル フォルダ - IFileFolder を IAttachmentFile にクラス キャストできます。
- ファイル フォルダの [ファイル] テーブルの行 - [ファイル] テーブルから IAttachmentFile に IRow をクラス キャストできます。
- ビジネス オブジェクトの [添付ファイル] テーブルの行 - [添付ファイル] テーブルから IAttachmentFile に IRow をクラス キャストできます。

IAttachmentFile には、添付ファイルを使用するために次のメソッドが用意されています。

- getFile()
- isSecure()

注意 IAttachmentFile には、添付ファイルのファイルを変更できる setFile() メソッドもありますが、これは [添付ファイル] テーブルの行に対してのみサポートされています。

IAttachmentFile メソッドから返される結果は、次の表に示すように、使用するオブジェクトによって異なります。

呼び出し側オブジェクト	getFile() 戻り値	isSecure() 戻り値
ビジネス オブジェクトの [添付ファイル] テーブルの行	単一ファイル InputStream (行がファイル フォルダから特定のファイルを参照する場合)、またはファイル フォルダのすべてのファイルを含む ZIP された InputStream が返されます。	参照先ファイルが URL でない場合、またはすべてのファイルが URL でない場合は true。
FileFolder オブジェクト	ファイル フォルダのすべてのファイルを含む ZIP された InputStream が返されます。	ファイル フォルダに含まれるすべてのファイルが URL でない場合は true。
ファイル フォルダの [ファイル] テーブルの行	ファイル フォルダから特定の行を参照する単一ファイル InputStream が返されます。	参照先ファイルが URL でない場合は true。

注意 ZIP された InputStream 内のファイルを読み取るには、java.util.zip.ZipInputStream クラスのメソッドを使用します。

次の例は、アイテムの [添付ファイル] テーブルの行から IAttachmentFile.isSecure() および IAttachmentFile.getFile() を使用する方法を示しています。

例: isSecure() および getFile() の使用

```
public InputStream getItemAttachment(IItem item) throws Exception {
  InputStream content = null;
  ITable attachments =
item.getTable(ItemConstants.TABLE_ATTACHMENTS);
  IRow row = (IRow)attachments.iterator().next();
  if (((IAttachmentFile)row).isSecure())
    content = ((IAttachmentFile)row).getFile();
  return content;
}
```

[添付ファイル] テーブルの使用

オブジェクトの [添付ファイル] テーブルを使用するには、次の手順に従います。

1. 対象の添付ファイルがあるオブジェクトを取得します。

   たとえば、IAgileSession.getObject() メソッドを使用して特定のオブジェクトを取得したり、検索条件を作成してオブジェクトを返すことができます。

2. [添付ファイル] テーブルを取得します。テーブルを取得するには、IDataObject.getTable() または IAttachmentContainer.getAttachments() メソッドを使用します。

3. [添付ファイル] テーブルの行を選択します。

   テーブルに対して Iterator を作成し、特定の行を選択します。テーブルの双方向の Iterator を取得するには、ITable.getTableIterator() メソッドを使用します。

次の例は、アイテムを取得して、そのアイテムの [添付ファイル] テーブルを取得してから、最初の添付ファイルを選択する方法を示しています。

例: アイテムの添付ファイルの取得

```
try {
  // Get Item P1000
  Map params = new HashMap();
  params.put(ItemConstants.ATT_TITLE_BLOCK_NUMBER,
"P1000");
  IItem item =
(IItem)m_session.getObject(IItem.OBJECT_TYPE, params);

  // Get the attachment table for file attachments
  ITable attTable = item.getAttachments();

  // Get a table iterator
  ITwoWayIterator it = attTable.getTableIterator();

  // Get the first attachment in the table
  if (it.hasNext()) {
    IRow row = (IRow)it.next();
    // Read the contents of the stream
    InputSteam stream = ((IAttachmentFile)row).getFile();
  }
  else {
    JOptionPane.showMessageDialog(null, "There are no files
listed.",
```

```
    "Error", JOptionPane.ERROR_MESSAGE);
  }
} catch (APIException ex) {
  System.out.println(ex);
}
```

ICheckoutable インターフェースの使用

ICheckoutable は、オブジェクトに関連付けられているファイルをチェックインおよびチェックアウトできるインターフェースです。このインターフェースは、[添付ファイル] テーブルの行に対してのみ適用されます。IRow は、[添付ファイル] テーブルから ICheckoutable にクラス キャストできます。

ICheckoutable には、添付ファイルを使用するために次のメソッドが用意されています。

- cancelCheckout()
- checkIn()
- checkOutEx()
- isCheckedOut()

次の例は、ICheckoutable インターフェースを使用して、[添付ファイル] テーブルの行からファイルをチェックアウトおよびチェックインする方法を示しています。

例: ICheckoutable メソッドを使用した添付ファイルのチェックアウトおよびチェックイン

```
public InputStream checkOutRow(IRow row) throws
APIException {
  // Check out the attachment
  ((ICheckoutable)row).checkOutEx();

  // Read the contents of the stream
  InputStream stream = ((IAttachmentFile)row).getFile();
  return stream;
}
public checkInRow(IRow row, String filePath) throws
APIException {
  if (row.isCheckedOut()) {
    // Set the new file
    ((IAttachmentFile)row).setFile(new File(filePath));

    // Check in the file
    ((ICheckoutable)row).checkIn();
  }
  else {
    JOptionPane.showMessageDialog(null, "The attachment is
not checked out.",
    "Error", JOptionPane.ERROR_MESSAGE);
  }
}
```

アイテムのリビジョンの指定

アイテムを使用するときは、リビジョンごとに添付ファイルが異なる場合があります。1 つのアイテムに複数のリビジョンがある場合は、プログラムでユーザーがリビジョンを選択できるようにする必要があります。リビジョンの指定については、107 ページの「アイテムのリビジョンの取得および設定」を参照してください。

リビジョンが確定済みかどうかの確認

通常は、アイテムのリビジョンがリリースされると、そのリビジョンも確定されます。確定アイテムの添付ファイルはロックされるため、チェックアウトすることはできません。添付ファイルは表示できますが、新規の変更を提出 (つまり、新規のリビジョンを作成) しないかぎり、その添付ファイルは変更できません。リビジョンが確定済みかどうかを確認する方法の詳細は、109 ページの「リビジョンの確定済みステータスの変更」を参照してください。

ファイル フォルダのチェックアウト

ファイル フォルダに含まれているファイルを追加、削除または変更するには、その前に、そのファイル フォルダをチェックアウトする必要があります。ファイル フォルダは、別のユーザーがすでにチェックアウトしていないかぎり、適切な権限を使用してチェックアウトできます。ファイル フォルダをチェックアウトすると、他のユーザーはそのファイル フォルダをチェックアウトしたり変更することはできません。

変更分析者やコンポーネント エンジニアと同様に、ファイル フォルダをチェックアウトしたユーザーは、そのファイル フォルダをチェックインできます。ファイル フォルダがネットワーク上のある場所、または共有のドライブやディレクトリにチェックアウトされた場合、そのネットワークの場所または共有ディレクトリにアクセスできるユーザーは、そのファイル フォルダをチェックインできます。

次の例は、ファイル フォルダをチェックアウトする方法を示しています。

例: ファイル フォルダのチェックアウト

```
void checkOutFileFolder(IFileFolder ff) throws Exception {
  ff.checkOutEx();
}
```

注意 [添付ファイル] テーブルの行は、`ICheckoutable.checkOutEx()` を使用してチェックアウトすることもできます。150 ページの「ICheckoutable インターフェースの使用」を参照してください。

ファイル フォルダのチェックアウトのキャンセル

ファイル フォルダをチェックアウトして変更を行わない場合、または変更を破棄して元のファイル フォルダに戻す場合は、そのチェックアウトをキャンセルできます。チェックアウトをキャンセルすると、他のユーザーがそのファイル フォルダをチェックアウトできるようになります。

注意 チェックアウトをキャンセルできるのは、ファイル フォルダをチェックアウトしたユーザーのみです。

次の例は、ファイル フォルダのチェックアウトをキャンセルする方法を示しています。

例: ファイル フォルダのチェックアウトのキャンセル

```
void cancelCheckOut(IFileFolder ff) {
  // Show a confirmation dialog box
  int i = JOptionPane.showConfirmDialog(null,
  "Are you sure you want to cancel checkout?",
  "Cancel Checkout", JOptionPane.YES_NO_OPTION);

  // If the user clicks Yes, cancel checkout
  try {
    if (i == 0) {
      ff.cancelCheckout();
    }
  } catch (APIException ex) {
    System.out.println(ex);
  }
}
```

注意 [添付ファイル] テーブルの行のチェックアウトは、ICheckoutable.cancelCheckout() を使用してキャンセルすることもできます。150 ページの「ICheckoutable インターフェースの使用」を参照してください。

[添付ファイル] テーブルへのファイルおよび URL の追加

Agile API を使用すると、多くのタイプのオブジェクト (IItem、IChange、IManufacturerPart、IManufacturer など) の [添付ファイル] テーブルにファイルと URL を追加できます。添付ファイルは、1 つ以上の物理ファイルまたはインターネット アドレス (URL) です。ファイルは Agile PLM ファイル格納庫に保存されているため、セキュアな添付ファイルとみなされます。これに対して、URL は保護されていない添付ファイルです。

ファイルまたは URL をビジネス オブジェクトの [添付ファイル] テーブルに追加すると、サーバでは、関連するファイルまたは URL が含まれた新規のファイル フォルダが自動的に作成されます。[添付ファイル] テーブルの新規の行は、新規のファイル フォルダを参照します。

URL 添付ファイルを追加すると、サーバではインターネットの場所への参照が保存されますが、ファイルはアップロードされません。したがって、URL 添付ファイルはダウンロードできません。Agile API では、添付ファイルとしてチェックインする URL 文字列が検証されます。URL が無効な場合、Agile API は、その文字列を URL ではなくファイル名とみなします。

次の場合は、ファイルまたは URL をアイテムの [添付ファイル] テーブルに追加できません。

- 現在のリビジョンに保留中またはリリース済みの MCO がある場合。
- 現在のリビジョンが確定済みの場合。

ITable.createRow(java.lang.Object) メソッドを使用して行を [添付ファイル] テーブルに追加すると、param メソッドは次のいずれかのオブジェクト タイプになります。

- String - ローカル パスで指定した 1 つの添付ファイルを追加します。
- String[] - ローカル パスの配列で指定した複数の添付ファイルを追加します。
- File - 1 つの添付ファイルを追加します。
- File[] - 複数の添付ファイルを追加します。
- InputStream - 1 つの添付ファイルを追加します。

- InputStream[] - 複数の添付ファイルを追加します。
- URL - 1 つの URL 添付ファイルを追加します。
- URL[] - 複数の URL 添付ファイルを追加します。
- ([添付ファイル] または [ファイル] テーブルの) IRow - ファイルまたは URL 添付ファイルを追加します。
- IFileFolder - 指定したファイル フォルダのすべてのファイルと URL を追加します。
- Map - 添付ファイル パラメータを含むハッシュ テーブルで指定した 1 つ以上のファイルを追加します。

注意 添付ファイルを追加するとき、パフォーマンスが最も高いのは File オブジェクト タイプです。

次の例は、addAttachment() メソッドを使用して行を [添付ファイル] テーブルに追加する方法をいくつか示しています。

例: [添付ファイル] テーブルへのファイルの追加

```
// Add a single file to the Attachments table row by specifying a file
path
public static IRow addAttachment(ITable attTable, String path) throws
APIException {
  IRow row = attTable.createRow(path);
  return row;
}
// Add a single file to the Attachments table
public static IRow addAttachment(ITable attTable, File file) throws
APIException {
  IRow row = attTable.createRow(file);
  return row;
}
// Add multiple files to the Attachments table
public static IRow addAttachment(ITable attTable, File[] files) throws
APIException {
  IRow row = attTable.createRow(files);
  return row;
}
// Add a URL attachment to the Attachments table
public static IRow addAttachment(ITable attTable, URL url) throws
APIException {
  IRow row = attTable.createRow(url);
  return row;
}
// Add a file folder to the Attachments table
public static IRow addAttachment(ITable attTable, IFileFolder ff)
throws APIException {
  IRow row = attTable.createRow(ff);
  return row;
}
// Add an FileFolder.Files row object or a [BusinessObject].Attachments
row object
// to the Attachments table. The Agile API validates the row object
at run time to
// determine if it is from a valid table (Files or Attachments).
public static IRow addAttachment(ITable attTable, IRow filesRow) throws
APIException {
  IRow row = attTable.createRow(filesRow);
```

```
  return row;
}
// Add a file folder to the Attachments table and specify the version
for all files
public static IRow addAttachmentWithVersion(ITable attTable,
IFileFolder ff) throws APIException {
  ff.setCurrentVersion(new Integer(1));
  IRow row = attTable.createRow(ff);
  return row;
}
```

オブジェクト間での添付ファイルおよびファイルのディープ クローンの作成

オブジェクト間で添付ファイルを簡単にコピーするには、CommonConstants.MAKE_DEEP_COPY 仮想属性を ITable.createRow(Object) のブール パラメータとして使用します。このパラメータを使用すると、元のファイルを参照するかわりに、プログラムで Agile ファイル マネージャの格納庫にファイルの新規コピーを作成できます。

例: [添付ファイル] テーブル行のディープ クローンの作成

```
// Clone an attachment table row and its file from one item to another
public static cloneAttachment(IItem item1, IItem item2, File file)
throws APIException {
  // Get the attachments tables of item1 and item2
  ITable tblAttach1 = item1.getAttachments();
  ITable tblAttach2 = item2.getAttachments();

  // Prepare params for the first row
  HashMap params = new HashMap();
  params.put(CommonConstants.ATT_ATTACHMENTS_CONTENT, file);

  // Add the file to the attachments table of item1
  IRow row1 = tblAttach1.createRow(params);

  // Prepare params for the second row
  params.clear();
  params.put(CommonConstants.ATT_ATTACHMENTS_CONTENT, row1);
  params.put(CommonConstants.MAKE_DEEP_COPY, Boolean.TRUE);
  // Add the same file to the attachments table of item2
  IRow row2 = tblAttach2.createRow(params);
}
```

例: ファイル フォルダの [ファイル] テーブル行のディープ クローンの作成

```
// Clone a Files table row and its file from one File Folder to another
public static cloneFilesRow(IFileFolder folder1, IFileFolder folder2,
File file) throws APIException {
  // Check out folder1 and folder2
  folder1.checkOutEx();
  folder2.checkOutEx();

  // Get the Files tables of folder1 and folder2
  ITable tblFiles1 =
folder1.getTable(FileFolderConstants.TABLE_FILES);
  ITable tblFiles2 =
folder2.getTable(FileFolderConstants.TABLE_FILES);
```

```
        // Prepare params for the first row
        HashMap params = new HashMap();
        params.put(CommonConstants.ATT_ATTACHMENTS_CONTENT, file);

        // Add the file to the attachments table of folder1
        IRow row1 = tblFiles1.createRow(params);

        // Prepare params for the second row
        params.clear();
        params.put(CommonConstants.ATT_ATTACHMENTS_CONTENT, row1);
        params.put(CommonConstants.MAKE_DEEP_COPY, Boolean.TRUE);
        // Add the same file to the Files table of folder2
        IRow row2 = tblFiles2.createRow(params);

        // Check in folder1 and folder2
        folder1.checkIn();
        folder2.checkIn();
    }
```

添付ファイル追加時のファイル フォルダ サブクラスの指定

Agile PLM システムは、複数のファイル フォルダ サブクラスを使用して設定できます。その場合は、ファイル フォルダをビジネス オブジェクトの [添付ファイル] テーブルに追加するときに、使用するファイル フォルダ サブクラスを指定できます。サブクラスを指定しない場合、Agile API ではデフォルトのファイル フォルダ サブクラスが使用されます。仮想属性 CommonConstants.ATT_ATTACHMENTS_FOLDERCLASS を使用すると、必要なファイル フォルダ サブクラスを簡単に指定できます。これによって、属性を任意のファイル フォルダ サブクラスに設定できます。

次の例は、ファイル フォルダを [添付ファイル] テーブルに追加するときに、ATT_ATTACHMENTS_FOLDERCLASS 属性を使用してサブクラスを指定する方法を示しています。

例: 添付ファイル追加時のファイル フォルダ サブクラスの指定

```
IAgileClass ffclass = m_admin.getAgileClass("MyFileFolder");
// init item
IItem item = (IItem)session.createObject(ItemConstants.CLASS_PART,
"P0001");

// get attachments table
ITable tab_attachment = item.getAttachments();

// prepare map
HashMap map = new HashMap();
map.put(CommonConstants.ATT_ATTACHMENTS_CONTENT, new
File("files/file.txt"));
map.put(CommonConstants.ATT_ATTACHMENTS_FOLDERCLASS, ffclass);
// add file
IRow row = tab_attachment.createRow(map);
```

ファイル フォルダのファイルのバージョンの設定

ファイル フォルダには、複数のバージョンを指定できます。ファイル フォルダを別のビジネス オブジェクトの [添付ファイル] テーブルに追加するときは、使用するファイル バージョンを指定できます。ファイル バージョンを指定しない場合、Agile API ではデフォルトまたは最新のバージョンが使用されます。ファイル バージョンを指定すると、[添付ファイル] テーブルの行はそのバージョンにリンクされます。

[添付ファイル] テーブルが含まれる親オブジェクトがアイテムの場合は、そのアイテムを確定して、添付ファイルの指定したバージョンをロックできます。アイテムを確定する方法の詳細は、109 ページの「リビジョンの確定済みステータスの変更」を参照してください。

例: [添付ファイル] テーブルへの行追加時のバージョンの設定

```
// Add a file folder to the Attachments table and use version 1 of all
files
public static IRow addAttachment(ITable attTable, IFileFolder ff)
throws APIException {
  ff.setCurrentVersion(new Integer(1));
  IRow row = attTable.createRow(ff);
  return row;
}
// Add a file folder to the Attachments table and use version 1 of all
files.
// This method passes a hash table for the params parameter of createRow().
public static IRow addAttachment(ITable attTable, IFileFolder ff)
throws APIException {
  HashMap map = new HashMap();
  map.put(CommonConstants.ATT_ATTACHMENTS_CONTENT, ff);
  map.put(CommonConstants.ATT_ATTACHMENTS_FOLDER_VERSION, new
Integer(1));
  IRow row = attTable.createRow(map);
  return row;
}
// Add a row from the Files table of a file folder to the Attachments
table and use version 2 of the file
public static IRow addAttachment(ITable attTable, IFileFolder ff)
throws APIException {
  ff.setCurrentVersion(new Integer(2));
  IRow filesRow =
(IRow)ff.getTable(FileFolderConstants.TABLE_FILES).iterator().next
();
  IRow row = attTable.createRow(filesRow);
  return row;
}
```

ファイル フォルダのチェックイン

チェックアウトしたファイル フォルダの編集が完了した後は、そのファイル フォルダを再度チェックインできます。チェックインすると、他のユーザーがそのファイル フォルダをチェックアウトできるようになります。ファイル フォルダは、チェックアウトに使用したコンピュータだけでなく、すべてのコンピュータからチェックインできます。

ファイル フォルダには、複数のファイルが含まれている場合があります。1 つのファイル フォルダをチェックインすると、そのファイル フォルダに含まれるすべてのファイルが自動的にチェックインされます。したがって、ファイル フォルダに含まれるファイルをリストする必要は特にありません。

例: ファイル フォルダのチェックイン

```
void checkInFiles(IFileFolder ff) throws Exception {
  // Set the local file path
  String path = "d:¥files¥file1.doc";

  // Get the Files table
  ITable files = ff.getTable(FileFolderConstants.TABLE_FILES);

  // Get the first row
  IRow row = (IRow)files.iterator().next();

  // Replace the file
  row.setValue(FileFolderConstants.ATT_FILES_FILE_NAME, new
File(path));

  ff.checkIn();
}
```

注意 [添付ファイル] テーブルの行は、ICheckoutable.checkIn() を使用してチェックインすることもできます。150 ページの「ICheckoutable インターフェースの使用」を参照してください。

ファイルの置換

ファイル フォルダの [ファイル] テーブルにリストされているファイルを置換するには、IRow.setValue() メソッドを使用します。IRow.setValue() の cellID パラメータには、[ファイル名] フィールドの属性 ID 定数を指定します。また、IRow.setValue() の value パラメータには、String (ファイルのローカル パス)、File オブジェクトまたは InputStream オブジェクトのいずれかを指定します。

例: ファイルの置換

```
// Replacing a file by specifying the path to a file
String path = "d:¥files¥file1.doc";
row.setValue(FileFolderConstants.ATT_FILES_FILE_NAME, path);

// Replacing a file by specifying a File object
String path = "d:¥files¥file1.doc";
row.setValue(FileFolderConstants.ATT_FILES_FILE_NAME, new
File(path));
```

IRow.setValue() のかわりに IRow.setValues() を使用する場合は、Map オブジェクトを渡してコンテンツとファイル名を同時に変更できます。この Map には、FileFolderConstants.ATT_FILES_FILE_NAME と FileFolderConstants.ATT_FILES_CONTENT の 2 つのキー パラメータを含める必要があります。これらのキーに対する可能な値は、String、File または InputStream です。

例: ファイル コンテンツとファイル名の同時更新

```
void changeContentAndFilename(IFileFolder ff, String newFilename,
File newFile) throws Exception {
  // Check out the file folder
  ff.checkOutEx();

  // Get the Files table
  ITable files = ff.getTable(FileFolderConstants.TABLE_FILES);
```

```
        // Get the first row
        IRow row = (IRow)files.iterator().next();

        // Create a Map containing the new file and filename
        Map map = new HashMap();
        map.put(FileFolderConstants.ATT_FILES_CONTENT, newFile);
        map.put(FileFolderConstants.ATT_FILES_FILE_NAME, newFilename);

        // Set values for content and file name
        row.setValues(map);

        // Check in
        ff.checkIn();
    }
```

ファイル名を変更しない場合は、次の例に示すように、ファイル コンテンツのみを更新できます。

例: ファイル コンテンツのみの更新

```
        ...
        // Create a Map containing the new file
        Map map = new HashMap();
        map.put(FileFolderConstants.ATT_FILES_CONTENT, newFile);

        // Set values
       row.setValues(map);
        ...
```

ビジネス オブジェクトの [添付ファイル] テーブルにリストされているファイルを置換するには、IFileFolder.setFile() メソッドを使用することもできます。setFile() の param パラメータには、File、InputStream または Map オブジェクトを指定します。Map オブジェクトを指定する場合は、次の Agile API 定数で表される 2 つのキー パラメータを含める必要があります。

```
CommonConstants.ATT_ATTACHMENTS_FILE_NAME
CommonConstants.ATT_ATTACHMENTS_CONTENT
```

- ファイル コンテンツ属性に対する可能な値は、String、File または InputStream です。

例: [添付ファイル] テーブルの行に対するファイルの置換

```
public replaceFileInRow(IRow row, String filename, String filePath)
throws Exception {
  ((ICheckoutable)row).checkOutEx();
  Map map = new HashMap();
  map.put(CommonConstants.ATT_ATTACHMENTS_FILE_NAME, filename);
  map.put(CommonConstants.ATT_ATTACHMENTS_CONTENT, new
File(filePath));
  ((IAttachmentFile)row).setFile(map);
  ((ICheckoutable)row).checkIn();
}
```

添付ファイルの取得

別のユーザーがファイル フォルダをチェックアウトした場合は、そのファイル フォルダのファイルのコピーを取得してローカル マシンに保存できます。IAttachmentFile.getFile() メソッドは、[添付ファイル] テーブルの行に関連付けられたファイル ストリームを返します。このファイル ストリームは、関連するファイル フォルダに含まれるファイルの数に応じて、1 ファイルのファイル ストリーム、または ZIP されたファイル ストリーム (ファイルが複数の場合) になります。また、IAttachmentFile.getFile() を使用すると、別のビジネス オブジェクトの [添付ファイル] テーブルにアクセスせずに、ファイル フォルダから 1 つ以上のファイルを直接取得できます。ファイル フォルダ オブジェクトから getFile() を呼び出した場合は、[ファイル] テーブルにリストされているすべてのファイルの ZIP されたファイル ストリームを返します。ファイル フォルダの [ファイル] テーブルの行から getFile() を呼び出した場合は、その行に関連付けられた特定のファイルのファイル ストリームを返します。

注意 IAttachmentFile.getFile() を使用した場合、返されるファイル ストリームに含まれるのは添付ファイルのみです。URL 添付ファイルには、関連付けられたファイルがありません。

次の例は、添付ファイルのコピーを取得する方法を示しています。

例: 添付ファイルの取得

```
// Get one or more files associated with the row of an Attachments table
or a Files table
public InputStream getAttachmentFile(IRow row) throws APIException {
  InputStream content = ((IAttachmentFile)row).getFile();
  return content;
}
// Get all files associated with a file folder
public InputStream getAttachmentFiles(IFileFolder ff) throws
APIException {
  InputStream content = ((IAttachmentFile)ff).getFile();
  return content;
}
```

IFileFolder.getFile() を使用して、ファイル フォルダに含まれるすべてのファイルの ZIP されたファイル ストリームを返す場合は、次の例に示すように、java.util.zip.ZipInputStream クラスのメソッドを使用して、ZIP された InputStream からファイルを抽出できます。

例: ZIP されたファイル ストリームからのファイルの抽出

```
static void unpack(InputStream zippedStream) throws IOException {
  ZipInputStream izs = new ZipInputStream(zippedStream);
  ZipEntry e = null;
  while ((e = izs.getNextEntry()) != null) {
    if (!e.isDirectory()) {
      FileOutputStream ofs = new FileOutputStream(e.getName());
      byte[] buf = new byte[1024];
      int amt;
      while ((amt = izs.read(buf)) != -1) {
        ofs.write(buf, 0, amt);
      }
      ofs.close();
    }
  }
  zippedStream.close();
}
```

Agile API では、添付ファイルを直接開くためのメソッドは提供されていません。ただし、ファイルを取得してから、プログラムよって別のアプリケーションでそのファイルを開いたり、ブラウザ ウィンドウに表示することはできます。

ファイル フォルダと添付ファイルの削除

ファイル フォルダ (複数のファイルが含まれている場合があります) を削除するには、`IDataObject.delete()` メソッドを使用します。ファイル フォルダを削除するには、ファイル フォルダの削除権限が必要です。オブジェクトの削除の詳細は、2-21 ページの「オブジェクトの削除および削除取消」を参照してください。

注意 ファイル フォルダを削除しても、関連付けられたファイルはファイル サーバから自動的に削除されません。Agile PLM 管理者には、削除されたファイルをパージする責任があります。

ビジネス オブジェクトの [添付ファイル] テーブルから行を削除するには、`ITable.removeRow()` メソッドを使用します。詳細は、4-11 ページの「テーブル行の削除」を参照してください。[添付ファイル] テーブルから行を削除しても、関連付けられたファイル フォルダは削除されません。

次の場合は、[添付ファイル] テーブルから行を削除できません。

- 親オブジェクトが、リビジョンが確定しているアイテムである場合。
- 選択した添付ファイルが、現在チェックアウトされている場合。

第 11 章

ワークフローの管理

扱うトピックは次のとおりです。

ワークフローについて

Agile には、電子送信、通知およびサインオフの機能があるため、変更管理プロセスが自動化され、簡潔で強力なワークフロー メカニズムが提供されます。これらのワークフロー機能を使用すると、次の処理を実行できます。

- 変更を承認または監視する必要のあるユーザーに、該当する変更を自動的に送信します。
- 承認者およびオブザーバに、変更が送信されたことを通知する電子メールの警告を自動的に送信します。
- オンラインで変更を承認または却下します。
- 変更にコメントを添付します。

変更管理プロセス

変更管理プロセスは、送信可能なオブジェクトに対して定義されたワークフローごとに異なる場合があります。次の表に、送信可能なオブジェクトの各タイプに対するデフォルトのワークフローの順序を示します。変更の場合、順序の最初の 4 つのステップは同じで、最後のステップのみ異なります。

ワークフロー	デフォルトの順序
デフォルトのアクティビティ	未開始 > 進行中 > 完了
デフォルトの添付ファイル	レビュー
デフォルトの検証	準備完了 > 開始済み > 検証済み > 発行済み > 改善済み > 検証済み > 終了
デフォルトの是正・予防処置	確認 > 認定 > 調査 > 実施 > 検証済み > 終了
デフォルトの設計変更	保留中 > 提出済み > CCB > リリース済み > 実施
デフォルトの設計変更依頼	保留中 > 提出済み > CCB > リリース済み > 終了

ワークフロー	デフォルトの順序
デフォルトのコンテンツ転送	保留中 > レビュー > リリース済み > 完了
デフォルトのデクラレーション	保留中 > サプライヤへ開示 > マネージャに送信 > レビュー > リリース済み > 実施
デフォルトの期限付き設計変更	保留中 > 提出済み > CCB > リリース済み > 期限切れ
デフォルトのゲート	終了 > レビュー中 > オープン
デフォルトの製造元依頼	保留中 > 提出済み > CCB > リリース済み > 検収済み
デフォルトの不具合レポート	保留中 > 提出済み > レビュー > リリース済み > 終了
デフォルトのパッケージ	保留中 > 提出済み > レビュー > 承認済み > 終了
デフォルトの価格変更	保留中 > 提出済み > 価格のレビュー > リリース済み > 実施
デフォルトの問題レポート	保留中 > 提出済み > レビュー > リリース済み > 終了
デフォルトの拠点毎変更	保留中 > 提出済み > CCB > リリース済み > 実施
デフォルトの出荷停止	保留中 > 提出済み > CCB > リリース済み > 再開

動的なワークフロー機能

特定の送信可能なオブジェクトに対して各ユーザーが使用できるワークフロー機能は、その送信可能なオブジェクトのステータスとユーザーの権限によって異なります。Agile API プログラムでは、ワークフローのこれらの動的機能を考慮する必要があるため、可能な場合はプログラムを適切に調整してください。

変更のステータスがワークフロー機能に与える影響

保留中の変更に使用できるワークフローのアクションは、リリース済みの変更に対するアクション処理とは異なります。変更が保留中またはリリース済みであるかどうかを判断するためにそのステータスを確認するには、`IRoutable.getStatus()` メソッドを使用します。`getStatus()` メソッドは、ワークフロー ステータスに対応する `IStatus` オブジェクトを返します。`IStatus` は `INode` インターフェースを拡張し、ステータス ノードで使用するための便利なメソッドを提供します。次の例は、`getStatus()` を使用して変更がリリース済みかどうかを判断する方法を示しています。

例: 変更オブジェクトのステータスの取得

```
private static boolean isReleased(IChange change) throws APIException
{
    return
(change.getStatus().getStatusType().equals(StatusConstants.TYPE_RE
LEASED);
}
```

ユーザー権限がワークフロー機能に与える影響

Agile 権限によって、ユーザーが変更に対して実行できるワークフロー アクションのタイプが決まります。Agile システム管理者は、各ユーザーに役割と権限を割り当てます。次の表に、ワークフロー アクションの実行に必要な権限を示します。

権限	関連 API
ステータスの変更	IRoutable.changeStatus()
コメント	IRoutable.comment()
送信	DataObject.send()

ユーザーにアクションを実行するための適切な権限があるかどうかを実行時に判断するには、IUser.hasPrivilege() メソッドを使用します。プログラムの UI は、ユーザーの権限に基づいて調整できます。次の例は、IRoutable.changeStatus() メソッドを呼び出す前に、ユーザーに変更のステータスを変更する権限があるかどうかを確認する方法を示しています。

例: 変更のステータス変更前のユーザーの権限の確認

```
private void goToNextStatus(IChange change, IUser user) throws
APIException {
  // Check if the user can change status
  if(user.hasPrivilege(UserConstants.PRIV_CHANGESTATUS, change)) {
    IUser[] approvers = new IUser[] { user };
    IStatus nextStatus = change.getDefaultNextStatus();
    change.changeStatus(nextStatus, true, "", true, true, null,
approvers, null, false);
  } else {
     System.out.println("Insufficient privileges to change status.");
  }
}
```

ワークフローの選択

新規の変更、パッケージ、製品サービス依頼または品質変更依頼を作成する場合は、ワークフローを選択する必要があります。選択しない場合は、オブジェクトが未割り当ての状態になり、ワークフロー プロセスを進行できません。Agile システムでは、送信可能なオブジェクトの各タイプに対して複数のワークフローを定義できます。オブジェクトに対して有効なワークフローを取得するには、IRoutable.getWorkflows() メソッドを使用します。送信可能なオブジェクトにワークフローが割り当てられていない場合は、IRoutable.getWorkflows() メソッドを使用してワークフローを選択できます。

変更のステータスが [保留中] の間は、別のワークフローを選択できます。変更が [保留中] ステータスから移動した後は、ワークフローを変更できません。

例: ワークフローの選択

```
private IChange createECO(IAgileSession session) throws APIException
{
  // Get an Admin instance
  IAdmin admin = session.getAdminInstance();
```

```
  // Create a change
  IAgileClass ecoClass =
admin.getAgileClass(ChangeConstants.CLASS_ECO);
  IAutoNumber[] autoNumbersPart = ecoClass.getAutoNumberSources();
  IChange change = (IChange)m_session.createObject(ecoClass,
autoNumbersPart[0]);

  // Get the current workflow (a null object,
  // since the workflow has not been set yet)
  IWorkflow wf = change.getWorkflow();

  // Get all available workflows
  IWorkflow[] wfs = change.getWorkflows();

   // Set the change to use the first workflow
   change.setWorkflow(wfs[0]);

   // Set the change to use the second workflow
   change.setWorkflow(wfs[1]);

   return change;
}
```

変更が [保留中] ステータス タイプの場合は、ワークフローの選択を解除して変更を未割り当てにできます。変更を未割り当てにするには、IRoutable.setWorkflow() メソッドを使用してワークフロー パラメータにヌルを指定します。

例: 変更の未割り当て

```
private void unassign(IChange change) throws APIException {
  change.setWorkflow(null);
}
```

承認者の追加および削除

変更が送信され、オンライン承認プロセスが開始された後で、担当者を承認者またはオブザーバのリストに追加したり、リストから削除する必要が生じる場合があります。承認者やオブザーバを追加または削除するには、ユーザーに、Agile Product Change Server ライセンスと送信権限の両方が必要です。

承認者のリストを変更するために [ワークフロー] テーブルをロードする必要はありません。ECO (設計変更) などの送信可能なオブジェクトがある場合は、IRoutable.addApprovers() および IRoutable.removeApprovers() メソッドを使用して承認者のリストを変更できます。addApprovers() または removeApprovers() を使用する場合は、承認者とオブザーバのリスト、通知が緊急かどうか、およびオプションのコメントを指定します。Agile API には、ユーザーまたはユーザー グループを承認者リストに追加したり、承認者リストから削除するために、オーバーロードされた addApprovers() および removeApprovers() のメソッドが用意されています。詳細は、API リファレンスを参照してください。

承認者またはオブザーバとして選択するユーザーに変更を表示する適切な権限がない場合は、プログラムで APIException が発生します。例外の発生を回避するために、ユーザーを承認者またはオブザーバのリストに追加する前に、それぞれのユーザーの権限を確認してください。

次の例は、変更に対する承認者を追加および削除する方法を示しています。

例: 承認者とオブザーバの追加および削除

```
public void modifyApprovers(IChange change) {
 try {
   // Get current approvers for the change
   IDataObject[] currApprovers =
change.getApproversEx(change.getStatus());

   // Get current observers for the change
   IDataObject[] currObservers =
change.getObserversEx(change.getStatus());

   // Add hhawkes to approvers
   IUser user = (IUser)m_session.getObject(IUser.OBJECT_TYPE,
"hhawkes");
   IUser[] approvers = new IUser[]{user};

   // Add flang to observers
   user = (IUser)m_session.getObject(IUser.OBJECT_TYPE, "flang");
   IUser[] observers = new IUser[]{user};

   // Add approvers and observers
   change.addApprovers(change.getStatus(), approvers,
observers, true,
   "Adding jsmith to approvers and jdoe to observers");
   // Add skubrick to approvers
   user = (IUser)m_session.getObject(IUser.OBJECT_TYPE, "skubrick");
   approvers[0] = user;

   // Add kwong to observers
   user = (IUser)m_session.getObject(IUser.OBJECT_TYPE, "kwong");
   observers[0] = user;

   // Remove skubrick from approvers and kwong from observers
   change.removeApprovers(change.getStatus(), approvers, observers,
   "Removing skubrick from approvers and kwong from observers");
  } catch (APIException ex) {
    System.out.println(ex);
  }
}
```

変更に対する承認者のリストまたはオブザーバのリストのみを変更する場合、変更する必要のないパラメータにはヌル値を渡すことができます。次の例は、オブザーバのリストを変更せずに、承認者のリストに現在のユーザーを追加する方法を示しています。

例: オブザーバを変更せずに承認者を追加する場合

```
public void addMeToApprovers(IChange change) {
  try {
    // Get the current user
    IUser user = m_session.getCurrentUser();

    // Add the current user to the approvers list for the change
    IUser[] approvers = new IUser[]{user};
    change.addApprovers(change.getStatus(), approvers, null, true,
    "Adding current user to approvers list");
  } catch (APIException ex) {
    System.out.println(ex);
  }
}
```

[サインオフ ユーザー二重識別タイプ] プリファレンスの設定

サインオフ ユーザー二重識別機能は、承認または却下のサインオフに二重識別が必要か、または 2 番目のサインオフが必要かどうかを制御するシステム全体のプリファレンスです。この機能は、FDA 規制会社で必要となり、設計変更の承認または却下時にユーザー識別の二重認証を必要とする企業ポリシーがある会社で活用できます。詳細は、『Agile PLM 管理者ガイド』を参照してください。

次に、サインオフ ユーザー二重識別機能をサポートする API の説明と、これらのメソッドを使用するコードサンプルを示しています。

送信可能なオブジェクトの承認

このメソッドは、オブジェクトが承認者によって承認されたり、1 つ以上のユーザー グループの代理として承認者がオブジェクトを承認したときにユーザーに通知します。このメソッドを使用して、サーバの [プリファレンス] 設定に設定されている secondSignature、escalations、transfers または signoffForSelf パラメータを指定することもできます。

```
// Approving a user
/*
Parameters:
password:User's approval password
secondSignature:User's second signature for approval
comment:A character string for user comments (4000
characters maximum)
notifyList:List of users and user groups to notify
approveForGroupList:List of user groups to approve for
escalations:Escalated from other users and user groups
to approve for
transfers:From other users and user groups to approve for
signoffForSelf:True to signoff for self and False
otherwise
APIException:
Thrown if the method fails to approve the routable object
*/
public void approve (String password, String
secondSignature,
            String comment, Collection notifyList,
            Collection approveForGroupList, Collection
escalations,
          Collection transfers, boolean signoffForSelf)
      throws APIException;
```

次のコードの例では、二重識別が必要な送信オブジェクトを承認します。その他の条件は、次のとおりです。

- [サーバ設定] > [プリファレンス] > [サインオフ ユーザー二重識別タイプ] が選択されている場合は、[ユーザー ID] を表示します。
- [ワークフロー ステータス] が [CCB] のワークフロー設定: [デフォルトの設計変更] の [二重識別が必要] を [はい] に設定します。
- 変更オブジェクトを作成し、**admin** などのユーザーを承認者として [CCB] ステータスと [リリース済み] ステータスに追加します。

例: 送信可能なオブジェクトの承認

```
// Admin approves the change by supplying approval password and "User
// ID" as the second signature
IWorkflow [] wfs = chg.getWorkflows();
IWorkflow wf = wfs[0];
chg.setWorkflow(wf);

// Advance ECO to submitted state
IStatus [] sts;
sts = wf.getStates(StatusConstants.TYPE_SUBMIT);
IStatus submit = sts[0];

// Change status to submit
m_session.disableWarning(new Integer(553));
m_session.disableWarning(new Integer(574));

Object [] nullObjectList = null;
chg.changeStatus(submit, false, null, false, false, nullObjectList,
nullObjectList, nullObjectList, false);

// Add approvers to CCB status and to Released status
IUser usr = (IUser) m_session.getObject(IUser.OBJECT_TYPE, "admin");
Object [] apprList = new IUser [] {usr};
sts = wf.getStates(StatusConstants.TYPE_REVIEW);
IStatus ccb = sts[0];
chg.addApprovers(ccb, apprList, nullObjectList, false,
"ADDAPPROVER_OBSERVER");

//   Change it to CCB status
chg.changeStatus(ccb, false, null, false, false, nullObjectList,
nullObjectList, nullObjectList, false);
m_session.enableWarning(new Integer(553));
m_session.enableWarning(new Integer(574));

// Admin approves the change by supplying approval password and "User
// ID" as the second signature
String userName = session.getCurrentUser().getName();
chg.approve ("agile", userName, "OK, Approved", null, null, null, null,
true);
```

送信可能なオブジェクトの却下

このメソッドは、送信可能なオブジェクトが承認者によって却下されたり、1 つ以上のユーザー グループの代理として承認者がオブジェクトを却下したときにユーザーに通知します。このメソッドを使用して、サーバの [プリファレンス] 設定に設定されている secondSignature、escalations、transfers または signoffForSelf パラメータを指定することもできます。

```
// Rejecting a user
/*
Parameters
   password:User's approval password
   secondSignature:User's second signature for approval
   comment:A character string for user comments (4000 characters
maximum)
   notifyList:List of users and user groups to notify
   approveForGroupList:List of user groups to approve for
   escalations:Escalated from other users and user groups to approve
for
   transfers:From other users and user groups to approve for
   signoffForSelf:True to signoff for self and False otherwise
APIException
   Thrown if the method fails to approve the routable object
*/
public void reject(String password, String secondSignature,
   String comment, Collection notifyList,
   Collection approveForGroupList, Collection escalations,
   Collection transfers, boolean signoffForSelf)
   throws APIException;
```

次のコード サンプルでは、送信オブジェクトを却下するために二重識別が必要です。その他の条件は、次のとおりです。

- [サーバ設定] > [プリファレンス] > [サインオフ ユーザー二重識別タイプ] が選択されている場合は、[ユーザー ID] を表示します。
- [ワークフロー ステータス] が [CCB] のワークフロー設定: [デフォルトの設計変更] の [二重識別が必要] を [はい] に設定します。
- 変更オブジェクトを作成し、**admin** などのユーザーを [CCB] ステータスと [リリース済み] ステータスの承認者として追加します。

例: 送信可能なオブジェクトの却下

```
// Admin rejects the change by supplying approval password and "User
// ID" as the second signature
IChange chg;
String chgNo;

IUser curUser = session.getCurrentUser();
String userName = curUser.getName();

chg = (IChange) session.getObject(ChangeConstants.CLASS_ECO, chgNo);
chg.reject("agile", userName, "Rejected", null, null, null, null,
true);
```

送信可能なオブジェクトを承認する承認者およびユーザーのユーザー グループの追加

このメソッドは、特定のワークフロー ステータスの承認者として追加され、現在のユーザーまたは承認者もそのグループ メンバーであるユーザー グループの配列を取得するように設計されています。

```
/*
Parameters
     status:A node corresponding to the desired workflow
     status.You can
     retrieve the current status using getStatus().To retrieve
     the default next status, use getDefaultNextStatus().
APIExceptions and Returns
```

- throws APIException if the method fails

- returns an array of IUserGroup objects

```
*/
public IDataObject[] getPossibleUserGroupsForSignoff(IStatus status)
       throws APIException;
```

次のコード サンプルでは、[CCB] ステータスで ECO を承認し、現在のユーザーがそのメンバーとして含まれるユーザー グループを追加します。二重識別に加えて、次の条件も必須です。

- [サーバ設定] > [プリファレンス] > [サインオフ ユーザー二重識別タイプ] が選択されている場合は、[ユーザー ID] を表示します。
- [ワークフロー ステータス] が [CCB] のワークフロー設定: [デフォルトの設計変更] の [二重識別が必要] を [はい] に設定します。

例: 現在のユーザーがユーザー グループのメンバーとして含まれる、承認者のユーザー グループの追加

```
// IChange change;
//Set Workflow
IWorkflow[] wfs=change.getWorkflows();
IWorkflow workflow = wfs[0];
change.setWorkflow(workflow);

//Add the User Group as approver for CCB
Object[] appr=new Object[] {user_group};
IStatus current=change.getStatus();
StatusConstants type=current.getStatusType();

m_session.disableWarning(new Integer(574));
while(!(type == (StatusConstants.TYPE_REVIEW))){ IStatus
nextstatus=change.getDefaultNextStatus();
      change.changeStatus(nextstatus,true,"",true,true,(Object[])n
ull,(Object[])null,             (Object[])null,false);
      current=change.getStatus();
      type=change.getStatus().getStatusType();
}
m_session.enableWarning(new Integer(574));
change.addApprovers(current, appr, (Object [])null, false, "");
IDataObject[] u = change.getPossibleUserGroupsForSignoff(status);
```

```
/* APPROVE */
Collection gl = new ArrayList();

//Group list
gl.add(u[0]);
change.approve("agile", session.getCurrentUser().getName(),
     "ESIGN-FIRST", null, gl, null, null, false);
```

転送元ユーザーを代行するユーザーによる送信可能なオブジェクトの承認

このメソッドは、特定のワークフロー ステータスについて、現在のユーザーの権限委譲として追加されたユーザーの配列を取得するように設計されています。

```
/*
Parameters
     status:A node corresponding to the desired workflow
     status.You can
     retrieve the current status using getStatus().To retrieve
     the default next status, use getDefaultNextStatus().
APIExceptions and Returns
     - throws APIException if the method fails
     - returns an array of IUserGroup objects
*/
public IDataObject[] getPossibleTransferredFromUsers(IStatus status)
       throws APIException;
```

次のコード サンプルでは、ユーザー A から別のユーザー B に権限委譲を設定し、ECO を作成し、[CCB] ステータスの承認者としてユーザー A を追加します。

注意 ユーザー B の [CCB] ステータスに対する `getPossibleTransferredFromUsers(IStatus status)` の戻り値は、サインオフ権限がユーザー B に委譲されるユーザーの配列です。

二重識別に加えて、次の条件も必須です。

- [サーバ設定] > [プリファレンス] > [サインオフ ユーザー二重識別タイプ] が選択されている場合は、[ユーザー ID] を表示します。
- [ワークフロー ステータス] が [CCB] のワークフロー設定: [デフォルトの設計変更] の [二重識別が必要] を [はい] に設定します。

例: ユーザー間の権限委譲の設定

```
Log in and execute the following code as User B
IDataObject [] usrs = chg.getPossibleTransferredFromUsers(status);

// Prepare the collection
Collection col = new ArrayList ();

for (int i=0; i < usrs.length; i++){
col.add(usrs[i]);
}

// approve the change
chg.approve("agile", userName, "OK, Approved", null, null, null, col,
false);
```

送信可能なオブジェクトを承認する現在のユーザーの有効なエスカレーションの追加

このメソッドは、特定のワークフロー ステータスについて、現在のユーザーの有効なエスカレーションとして機能するユーザーの配列を取得するように設計されています。このメソッドは、ユーザーのカバー ページにある [エスカレーションの指定承認を許可] 属性の設定を上書きします。

```
/*
Parameters
     status:A node corresponding to the desired workflow
     status.You can
     retrieve the current status using getStatus().To retrieve
     the default next status, use getDefaultNextStatus().
APIExceptions and Returns
     - throws APIException if the method fails
     - returns an array of IUser and IUserGroup objects
*/
public IDataObject[] getPossibleEscalatedFromUsers(IStatus status)
       throws APIException;
```

次のコード サンプルでは、ユーザー A からユーザー B にエスカレーションを設定し、ECO を作成し、[CCB] ステータスの承認者としてユーザー A を追加します。

注意 [CCB] ステータスに対するユーザー B の `getPossibleEscalatedFromUsers(IStatus status)` は、エスカレーションがユーザー B に設定されるユーザーの配列を返します。

二重識別に加えて、次の条件も必須です。

- [サーバ設定] > [プリファレンス] > [サインオフ ユーザー二重識別タイプ] が選択されている場合は、[ユーザー ID] を表示します。
- [ワークフロー ステータス] が [CCB] のワークフロー設定: [デフォルトの設計変更] の [二重識別が必要] を [はい] に設定します。

例: ユーザーへのエスカレーションの設定

```
// Log in and execute the following code as "User B"
IDataObject [] usrs = chg. getPossibleEscalatedFromUsers(IStatus
status);

// Prepare the collection
Collection col = new ArrayList ();

for (int i=0; i < usrs.length; i++){
     col.add(usrs[i]);
}

// approve the change
chg.approve("agile", userName, "OK, Approved", null, null, null, col,
false);
```

送信可能なオブジェクトを承認する 2 番目の署名の指定

このメソッドは、送信可能なオブジェクトの承認に 2 番目の署名が必要かどうかを検証するように設計されています。このメソッドは、secondSignature パラメータも設定する 173 ページの「送信可能なオブジェクトを承認する 2 番目の署名としてユーザー ID を追加」、166 ページの「送信可能なオブジェクトの承認」および 168 ページの「送信可能なオブジェクトの却下」で説明されているメソッドを組み合わせたメソッドとともに使用します。

```
/*
Parameters
     Status:The status (IStatus) of the object checked for
     the next workflow status.
Returns
     true if a second signature is required, false otherwise
*/
public boolean isSecondSignatureRequired(IStatus status)
       throws APIException;
```

次のコード サンプルでは、ECO (設計変更) の chg を作成します。[CCB] ステータスの chg.isSecondSignatureRequired (IStatus status) は、true を返します。

二重識別に加えて、次の条件も必須です。

- [サーバ設定] > [プリファレンス] > [サインオフ ユーザー二重識別タイプ] が選択されている場合は、[ユーザー ID] を表示します。
- [ワークフロー ステータス] が [CCB] のワークフロー設定: [デフォルトの設計変更] の [二重識別が必要] を [はい] に設定します。

例: 送信可能なオブジェクトを承認する 2 番目の署名の指定

```
// set the "Signoff User Dual Identification Type" preferences Node
IAdmin admin = session.getAdminInstance();
INode node = admin.getNode(NodeConstants.NODE_PREFERENCES);

// Node Properties
IProperty propSecondSignature = node.getProperty("Signoff User Dual
Identification Type");

IAgileList lst = propSecondSignature.getAvailableValues();
lst.setSelection(new Object [] {"User ID"});
propSecondSignature.setValue(lst);

// set the "Dual Identification Required" property for "Workflow Status
CCB: Default Change Orders" to "Yes"
IAdmin admin = session.getAdminInstance();
INode root=admin.getNode(NodeConstants.NODE_AGILE_WORKFLOWS);
INode CCBStatus=(INode)root.getChildNode("Default Change
Orders/Status List/CCB");
IProperty propDualIdentification = CCBStatus.getProperty("Dual
Identification Required");

IAgileList lst = propDualIdentification.getAvailableValues();
lst.setSelection(new Object [] {"Yes"});
propDualIdentification.setValue(lst);

// Get and print the "Second Signature Required Property" for the
// various states of a workflow
IWorkflow [] wfs = chg.getWorkflows();
```

```
IWorkflow wf = wfs[0];
chg.setWorkflow(wf);
IStatus [] sts = wf.getStates();
boolean secondSigReqd;

for (int i=0; i< sts.length; i++) {
     IStatus st = sts[i];
     System.out.println("Status Name =" + st.getName());
     System.out.println("IS Second Signature Reqd =
     "+       chg.isSecondSignatureRequired(st));
     System.out.println("IS Second Signature UserId = "
     +       chg.isSecondSignatureUserId(st));

     secondSigReqd = chg.isSecondSignatureRequired(st);
}
```

送信可能なオブジェクトを承認する 2 番目の署名としてユーザー ID を追加

このメソッドは、送信可能なオブジェクトを承認する 2 番目の署名としてユーザー ID を設定するように設計されています。このメソッドは、secondSignature パラメータも設定する 172 ページの「送信可能なオブジェクトを承認する 2 番目の署名の指定」、166 ページの「送信可能なオブジェクトの承認」および 168 ページの「送信可能なオブジェクトの却下」で説明されているメソッドを組み合わせたメソッドとともに使用します。

```
/*
Parameters
     Status:The status (IStatus) of the object checked for
     the next workflow status.
Returns
     true if a second signature required is User ID, false
     otherwise
*/
public boolean isSecondSignatureUserId(IStatus status)
       throws APIException;
```

次のコード サンプルでは、ECO (設計変更) の chg を作成します。[CCB] ステータスの chg.isSecondSignatureRequired (IStatus status) は、true を返します。

二重識別に加えて、次の条件も必須です。

- [サーバ設定] > [プリファレンス] > [サインオフ ユーザー二重識別タイプ] が選択されている場合は、[ユーザー ID] を表示します。
- [ワークフロー ステータス] が [CCB] のワークフロー設定: [デフォルトの設計変更] の [二重識別が必要] を [はい] に設定します。

例: 2 番目の署名としてユーザー ID を指定

```
boolean secondSigUserID;
secondSigUserID = chg.isSecondSignatureUserId (status);
```

変更の承認または却下

変更がグループ承認者に送信されると、オンライン承認プロセスが開始します。変更の [ワークフロー] テーブルにリストされたユーザーは、変更を承認または却下できます。

変更を承認すると、Agile システムによって承認が [ワークフロー] テーブルに記録されます。すべての承認者が変更を承認すると、変更をリリースする準備が整ったことを示す電子メール通知が変更分析者またはコンポーネント エンジニアに送信されます。

注意 変更を承認または却下するには、ユーザーに、作成および管理ユーザー ライセンスまたは依頼および承認ユーザー ライセンスが必要です。

`IRoutable.approve()` メソッドを使用する場合は、ユーザーの承認用パスワードとオプションのコメントを指定します。オーバーロードされた `approve()` メソッドを使用すると、通知リストおよび承認するユーザー グループのコレクションを指定できます。詳細は、API リファレンスを参照してください。

次に、特定の送信可能なオブジェクトの承認または却下について説明します。2 番目の署名が必須の場合は、PC 変更オブジェクトの承認または却下をサポートする API の詳細について、166 ページの「[サインオフ ユーザー二重識別タイプ] プリファレンスの設定」を参照してください。

次の例は、変更を承認する方法を示しています。

例: 変更の承認

```
public void approveChange(IChange change) {
  try {
    change.approve("agile", "Looks good to me");
  } catch (APIException ex) {
    System.out.println(ex);
  }
}
```

変更に根本的な欠陥がある場合、[ワークフロー] テーブルにリストされたユーザーはその変更を却下できます。変更を却下すると、その変更の [ワークフロー] タブに却下が記録され、電子メール通知が変更分析者またはコンポーネント エンジニアに送信されます。変更分析者またはコンポーネント エンジニアは、却下された変更を作成者に差し戻すことを決定し、そのステータスを [保留中] に戻す場合があります。

`IRoutable.reject()` メソッドを使用する場合は、ユーザーの承認用パスワードとオプションのコメントを指定する必要があります。オーバーロードされた `reject()` メソッドを使用すると、通知リストおよび承認するユーザー グループのコレクションを指定できます。詳細は、API リファレンスを参照してください。

次の例は、変更を却下する方法を示しています。

例: 変更の却下

```
public void rejectChange(IChange change) {
  try {
    change.reject("agile", "Incorrect replacement part!");
  } catch (APIException ex) {
    System.out.println(ex);
  }
}
```

変更のコメント

変更をコメントする場合は、オンライン承認プロセス時に他の CCB レビューアにコメントを送信します。コメントに加えて、作成者、変更分析者および変更管理委員会に通知するかどうかを指定できます。オーバーロードされた comment() メソッドを使用すると、通知リストを指定できます。詳細は、API リファレンスを参照してください。

次の例は、変更をコメントする方法を示しています。

例: 変更のコメント

```
public void commentChange(IChange change) {
  try {
    change.comment(true, true, true, "Change flagged for transfer to
ERP.");
  } catch (APIException ex) {
    System.out.println(ex);
  }
}
```

変更の検証

変更のライフサイクルの任意の時点で、必須の入力セルが未完了かどうか、または変更が Agile スマートルールに違反するかどうかを判断するために変更を検証できます。IRoutable.audit() メソッドを使用する場合、このメソッドは、キーとして ICell オブジェクトを、値として APIException オブジェクトのリストを含む Map オブジェクトを返します。変更に問題がない場合、ICell キーはヌルになります。APIException オブジェクトは、関連する入力セルの問題を記述します。

audit() メソッドが返す Map オブジェクトにも、ヌル オブジェクトがキーとして含まれる場合があります。ヌル オブジェクトに関連付けられている APIException オブジェクトは、データ セルに関連しない問題を記述します。

次の例は、変更を検証する方法を示しています。

例: 変更の検証

```
public void auditChange(IChange change) {
  try {
    // Audit the release
    Map results = change.audit();

    // Get the set view of the map
    Set set = results.entrySet();

    // Get an iterator for the set
    Iterator it = set.iterator();

    // Iterate through the cells and print each cell name and exception
    while (it.hasNext()) {
      Map.Entry entry = (Map.Entry)it.next();
      ICell cell = (ICell)entry.getKey();
      if(cell != null) {
        System.out.println("Cell : " + cell.getName());
      } else {
```

```
            System.out.println("Cell : No associated data cell");
          }
          //Iterate through exceptions for each map entry.
          //(There can be multiple exceptions for each data cell.)
          Iterator jt = ((Collection)entry.getValue()).iterator();
          while (jt.hasNext()) {
            APIException e = (APIException)jt.next();
            System.out.println("Exception : " + e.getMessage());
          }
        }
      } catch (APIException ex) {
        System.out.println(ex);
      }
    }
```

オブジェクトのワークフロー ステータスの変更

IRouteable.changeStatus() メソッドは、Agile オブジェクトのステータスを変更するための汎用メソッドです。たとえば、changeStatus() を使用すると変更を提出、リリースまたはキャンセルできます。検証の失敗などのインスタンスでは、複合例外 ExceptionConstants.API_SEE_MULTIPLE_ROOT_CAUSES が発生します。この例外を無効にするには、例外が検出されたコードを変更します。次の例を参照してください。

例: 複合例外の発生

```
    while (true) {
        try {
            change.changeStatus(
          wf.getStates(expectStatus)[0],
          false,
          "comment",
          false,
          false,
          null,
          null,
          null,
          false
             );
        } catch (APIException ae) {
          try {
             if
    (ae.getErrorCode().equals(ExceptionConstants.API_SEE_MULTIPLE_ROOT
    _CAUSES)){
               Throwable[] causes = ae.getRootCauses();
                for (int i = 0; i < causes.length; i++) {
                 m_session.disableWarning(
                   (Integer)((APIException)causes[i]).getErrorCode()
                       );
                 }
                } else {
                  m_session.disableWarning((Integer)ae.getErrorCode()
    );
```

```
                }
              } catch (Exception e) {
              throw ae;
            }
            continue;
        }
        break;
    }
```

変更は通常、CCB メンバーによってサインオフされた後にリリースします。changeStatus() を使用すると、変更のステータスの変更に加えて、通知リスト、オプションのコメント、および作成者と変更管理委員会に通知するかどうかも指定できます。

使用するオーバーロードされた changeStatus() メソッドに応じて、notifyList パラメータは、変更のステータスについて通知する IUser または IUserGroup オブジェクトの配列になります。詳細は、API リファレンスを参照してください。ワークフロー ステータスについてデフォルトの通知リストを使用するには、ヌル値を指定します。ユーザーに通知しないことを示すには、空の配列を指定します。

changeStatus() メソッドの approvers と observers の両方のパラメータには、ユーザーまたはユーザー グループの配列を明示的に渡す必要があります。ヌルを渡すと、承認者またはオブザーバが使用されません。特定のワークフロー ステータスについてデフォルトの承認者およびオブザーバを取得するには、getApproversEx() と getObserversEx() をそれぞれ使用します。

次の例は、変更のワークフロー ステータスを確認する方法を示しています。

例: 変更のステータスの確認

```
    void checkStatus(IChange change) {
      try {
        // Get current workflow status (an IStatus object)
        IStatus status = change.getStatus();
        System.out.println("Status name = " + status.getName());

        // Get next available workflow statuses
        IStatus[] nextStatuses = change.getNextStatuses();
        for (int i = 0; i < nextStatuses.length; i++) {
          System.out.println("nextStatuses[" + i +"] = " +
          nextStatuses[i].getName());
        }
        // Get next default workflow status
        IStatus nextDefStatus = change.getDefaultNextStatus();
        System.out.println("Next default status = " +
    nextDefStatus.getName());

      } catch (APIException ex) {
        System.out.println(ex);
      }
    }
```

次の例は、変更のステータスを変更する方法を示しています。

例: 変更のステータスの変更

```
public void nextStatus(IChange change, IUser[] notifyList,
  IUser[] approvers, IUser[] observers) {
  try {
    // Check if the user has privileges to change to the next status
    IStatus nextStatus = change.getDefaultNextStatus();
    if (nextStatus == null) {
      System.out.println("Insufficient privileges to change status.");
      return;
    }
    // Change to the next status
    else {
      change.changeStatus(nextStatus, true, "", true, true,
notifyList,
     approvers, observers, false);
    }
  } catch (APIException ex) {
   System.out.println(ex);
  }
}
```

次の例は、送信可能なオブジェクトのステータスを変更するときに、デフォルトの承認者とオブザーバを使用する方法を示しています。

例: ステータスの変更およびデフォルトの承認者とオブザーバへの送信

```
public void changeToDefaultNextStatus(IChange change) throws
APIException {
  // Get the next status of the change
  IStatus nextStatus = change.getDefaultNextStatus();

  // Get default approvers for the next status
  IDataObject[] defaultApprovers =
change.getApproversEx(nextStatus);

  // Get default observers for the next status
  IDataObject[] defaultObservers =
change.getObserversEx(nextStatus);

  // Change to the next status
  change.changeStatus(nextStatus, false, "", false, false, null,
defaultApprovers,
    defaultObservers, false);

}
```

選択したユーザーへの Agile オブジェクトの送信

Agile オブジェクトは、選択したユーザーのグループに送信できます。ECO などのオブジェクトを送信する場合、サインオフは必要ありません。選択した受信者は、オブジェクトへのリンクが添付された電子メール メッセージを受信します。IDataObject.send() メソッドを使用すると、Agile ユーザーの配列とオプションのコメントを指定できます。他のワークフロー コマンドとは異なり、send() メソッドは送信可能なオブジェクトに限定されません。このメソッドを使用すると、アイテムを含む任意のタイプの Agile データオブジェクトを送信できます。

次の例は、オブジェクトをすべてのユーザーに送信する方法を示しています。

例: 選択したユーザーへの Agile オブジェクトの送信

```
public void sendToAll(IDataObject object) {
  try {
    // Get all users
    IQuery q = (IQuery)m_session.createObject(IQuery.OBJECT_TYPE,
"select * from [Users]");
    ArrayList userList = new ArrayList();
    Iterator i = q.execute().getReferentIterator();
    while (i.hasNext()) {
      userList.add(i.next());
    }
    IUser[] users = (IUser[])(userList.toArray());
    // Send the object to all users
    object.send(users, "Please read this important document.");
  } catch (APIException ex) {
    System.out.println(ex);
  }
}
```

ユーザー グループへの Agile オブジェクトの送信

Agile 変更オブジェクトまたはアイテム オブジェクトは、ユーザー グループに送信できます。ECO などのオブジェクトを送信する場合、サインオフは必要ありません。選択した受信者は、オブジェクトへのリンクが添付された電子メール メッセージを受信します。IDataObject.send(IDataObject[] to, String Comment) メソッドを使用すると、Agile ユーザー グループの配列とオプションのコメントを指定できます。IDataObject 親インターフェースは、IUserGroup Agile オブジェクトを表します。他のワークフロー コマンドとは異なり、send() メソッドは送信可能なオブジェクトに限定されません。このメソッドを使用すると、アイテムを含む任意のタイプの Agile データオブジェクトを送信できます。

次の例は、オブジェクトをすべてのユーザー グループに送信する方法を示しています。

例: 選択したユーザー グループへの Agile オブジェクトの送信

```
public void sendToAll(IDataObject[] object) {
  try {
    // Get all user groups
    IQuery q = (IQuery)m_session.createObject(IQuery.OBJECT_TYPE,
"select * from [UserGroup]");
    ArrayList userList = new ArrayList();
    Iterator i = q.execute().getReferentIterator();
    while (i.hasNext()) {
      usergroupList.add(i.next());
    }
    IUserGroup[] group = (IUserGroup[])(usergroupList.toArray());
    // Send the object to all user groups
    object.send(usergroups, "Please read this important document.");
  } catch (APIException ex) {
    System.out.println(ex);
  }
}
```

第 12 章

品質の管理および追跡

扱うトピックは次のとおりです。

品質管理について

Agile PLM システムには、企業が次の品質関連アイテムを追跡および管理できるツールが用意されています。

- 顧客の不満
- 製品および製造の品質問題
- 拡張と是正処置の依頼

Agile PLM システムの是正処置プロセスは柔軟性があり、様々な方法で実装できます。たとえば、Agile PLM システムをカスタマイズする 1 つの方法は、Agile API を使用してこのシステムと顧客関係管理 (CRM) システムを統合することです。

品質関連の API オブジェクト

Agile API には、次の新規インターフェースが含まれています。

- `ICustomer` - `Customer` クラスのインターフェース。顧客は、企業の製品を使用するユーザーです。Agile PLM の実装によっては、顧客と問題レポートが顧客関係管理 (CRM) システムから直接インポートされる場合があります。
- `IServiceRequest` - `ServiceRequest` クラスのインターフェース。`IServiceRequest` は `IRoutable` のサブインターフェースです。`IServiceRequest` を使用すると、問題レポートと不具合レポート (NCR) の 2 つのタイプのサービス依頼を作成できます。
- `IQualityChangeRequest` - ECR や他のタイプの変更依頼に類似した `QualityChangeRequest` クラスのインターフェース。このインターフェースは、品質問題に対応する閉ループ方式ワークフロー プロセスを表します。検証および CAPA (是正処置/予防処置) は、`QualityChangeRequest` のサブクラスです。

品質関連の役割と権限

問題レポート、問題点、NCR、CAPA および QCR を作成、表示および変更するには、適切な権限が必要です。Agile PLM システムには 2 つのデフォルトのユーザー役割があり、これによって、ユーザーに次の品質関連オブジェクトを使用する権限が提供されます。

- 品質分析者 - 問題レポート、問題点および NCR を管理するユーザーの役割です。
- 品質管理者 - 検証および CAPA を管理するユーザーの役割です。

役割と権限の詳細は、『Agile PLM 管理者ガイド』を参照してください。

顧客の使用

このセクションでは、ICustomer オブジェクトを作成、ロードおよび保存する方法について説明します。

顧客について

ICustomer オブジェクトには、顧客の連絡先情報が格納されます。Agile PLM システムでは、顧客に役割があります。顧客は、企業の製品についてフィードバックを提供し、品質の問題点または検出した問題を警告します。

ICustomer オブジェクトは、CRM システムなど別のシステムで作成できます。Agile API を使用すると、顧客データと問題レポートを CRM システムから Agile PLM システムにインポートできます。

顧客の作成

顧客を作成するには、IAgileSession.createObject() メソッドを使用します。少なくとも、General Info.Customer Name 属性および General Info.Customer Number 属性に値を指定する必要があります。

例: 顧客の作成

```
try {
  //Create a Map object to store parameters
  Map params = new HashMap();
  //Initialize the params object
  params.put(CustomerConstants.ATT_GENERAL_INFO_CUSTOMER_NUMBER,
"CUST00006");
  params.put(CustomerConstants.ATT_GENERAL_INFO_CUSTOMER_NAME,
"Western Widgets");
  //Create a new customer
  ICustomer cust1 =
(ICustomer)m_session.createObject(CustomerConstants.CLASS_CUSTOMER,
params);

  } catch (APIException ex) {
    System.out.println(ex);
  }
```

顧客のロード

顧客をロードするには、IAgileSession.getObject() メソッドを使用します。顧客を一意に識別するには、[一般情報 | 顧客番号] 属性に値を指定します。

例: 顧客のロード

```
try {
  // Load a customer by specifying a CustomerNumber
  ICustomer cust =
(ICustomer)m_session.getObject(ICustomer.OBJECT_TYPE,
"CUST00006");
} catch (APIException ex) {
  System.out.println(ex);
}
```

顧客を別の顧客として保存

顧客を別の顧客として保存するには、IDataObject.saveAs() メソッドを次の構文で使用します。

```
public IAgileObject saveAs(java.lang.Object type, java.lang.Object
params)
```

params パラメータには、[一般情報 | 顧客名] 属性と [一般情報 | 顧客番号] 属性を指定します。

例: 顧客を別の顧客として保存

```
try {
  // Load an existing customer
  ICustomer cust1 =
(ICustomer)m_session.getObject(ICustomer.OBJECT_TYPE,
"CUST00006");

  //Create a Map object to store parameters
  Map params = new HashMap();

  //Initialize the params object
  params.put(CustomerConstants.ATT_GENERAL_INFO_CUSTOMER_NUMBER,
"CUST00007");
  params.put(CustomerConstants.ATT_GENERAL_INFO_CUSTOMER_NAME,
"Wang Widgets");
  // Save the customer
  ICustomer cust2 =
(ICustomer)cust1.saveAs(CustomerConstants.CLASS_CUSTOMER, params);

} catch (APIException ex) {
  System.out.println(ex);
}
```

製品サービス依頼の使用

このセクションでは、問題レポートと不具合レポートの 2 つのクラスの製品サービス依頼を使用する方法について説明します。

問題レポートについて

問題レポートには、製品に発生した問題が顧客の観点から記載されます。問題レポートは、顧客、販売代理店または顧客サービス担当者が提出できます。

通常、問題レポートは顧客によって作成されるため、問題の実際の原因が正確に説明されていない場合があります。問題の根本原因を把握するには、品質分析者が問題を調査する必要があります。

問題レポートは調査のために送信できます。品質分析者で構成された調査チームは、問題の根本原因を確認し、その問題を問題点にエスカレートするかどうかを判断します。

不具合レポートについて

不具合レポート (NCR) は、顧客またはサプライヤが受け取る製品のマテリアルの破損、不良モードまたは欠陥をレポートするために使用します。NCR は通常、サプライヤから製品を受理した後、製品出荷を検査するときに識別されます。顧客の要件または含有基準を満たさない製品は、不適合製品です。そのような製品は通常却下されるか、分離されて処分待ちになります。不具合レポートでは、品質分析者が問題を調査し、是正措置が必要かどうかを判断することが必要になる場合があります。

NCR はレビューのために送信できます。レビューは通常、承認および却下以外に、追加情報を収集するために使用されます。

製品サービス依頼の作成

問題レポートまたは不具合レポートを作成するには、`IAgileSession.createObject()` メソッドを使用します。指定する必要のある必須属性値はオブジェクトの番号のみです。次の例では、問題レポートおよび NCR を作成する方法を示しています。

例: 問題レポートまたは NCR の作成

```
public IServiceRequest createPR(String strNum) throws APIException {
  IServiceRequest pr = (IServiceRequest)m_session.createObject(
    ServiceRequestConstants.CLASS_PROBLEM_REPORT, strNum);
  return pr;
}
public IServiceRequest createNCR(String strNum) throws APIException
{
  IServiceRequest ncr = (IServiceRequest)m_session.createObject(
    ServiceRequestConstants.CLASS_NCR, strNum);
  return ncr;
}
```

品質分析者への製品サービス依頼の割り当て

問題レポートまたは NCR を品質分析者に割り当てるには、リスト フィールドである [カバー ページ | 品質分析者] フィールドに値を設定します。リスト フィールドで使用可能な値は、Agile PLM ユーザーで構成されています。次の例は、問題レポートまたは NCR の Cover Page.Quality Analyst フィールドに値を設定する方法を示しています。

例: 問題レポートまたは不具合レポートの割り当て

```
void assignServiceRequest(IServiceRequest sr) throws APIException {
  Integer attrID;
  //Set attrID equal to the Quality Analyst attribute ID
  attrID = ServiceRequestConstants.ATT_COVER_PAGE_QUALITY_ANALYST;

  //Get the Cover Page.Quality Analyst cell
  ICell cell = sr.getCell(attrID);

  //Get available list values for the list
  IAgileList values = cell.getAvailableValues();

  //Set the value to the current user
  IUser user = m_session.getCurrentUser();
  values.setSelection(new Object[] { user });
  cell.setValue(values);
}
```

製品サービス依頼への対象アイテムの追加

問題レポートまたは不具合レポートを 1 つ以上のアイテムに関連付けるには、アイテムを [対象アイテム] テーブルに追加します。各製品サービス依頼には、複数のアイテムを関連付けることができます。

注意 製品サービス依頼 (PSR) が [関連 PSR] テーブルに追加されている場合、[対象アイテム] テーブルは変更できません。

例: 製品サービス依頼への対象アイテムの追加

```
void addAffectedItem(IServiceRequest sr, String strItemNum) throws
APIException {
  //Get the class
  IAgileClass cls = sr.getAgileClass();

  //Attribute variable
  IAttribute attr = null;

  //Get the Affected Items table
  ITable affItems =
sr.getTable(ServiceRequestConstants.TABLE_AFFECTEDITEMS);

  //Create a HashMap to store parameters
  HashMap params = new HashMap();

  //Set the Item Number value
```

```
params.put(ServiceRequestConstants.ATT_AFFECTED_ITEMS_ITEM_NUMBER,
strItemNum);
  //Set the Latest Change value
  attr =
cls.getAttribute(ServiceRequestConstants.ATT_AFFECTED_ITEMS_LATEST
_CHANGE);
  IAgileList listvalues = attr.getAvailableValues();
  listvalues.setSelection(new Object[] { new Integer(0)});

params.put(ServiceRequestConstants.ATT_AFFECTED_ITEMS_LATEST_CHANG
E, listvalues);
  //Set the Affected Site value
  attr =
cls.getAttribute(ServiceRequestConstants.ATT_AFFECTED_ITEMS_AFFECT
ED_SITE);
  IAgileList listvalues = attr.getAvailableValues();
  listvalues.setSelection((new Object[] { "Hong Kong" });

params.put(ServiceRequestConstants.ATT_AFFECTED_ITEMS_AFFECTED_SIT
E, listvalues);
  //Create a new row in the Affected Items table
  IRow row = affItems.createRow(params);
}
```

製品サービス依頼への関連 PSR の追加

製品サービス依頼は、複数の問題レポートまたは NCR を 1 つのマスターに集約するために使用できます。そのためには、新規の製品サービス依頼を作成し、アイテムは [対象アイテム] テーブルに追加しません。かわりに、[関連 PSR] テーブルを選択し、関連製品サービス依頼ごとに 1 行追加します。

注意 アイテムが [対象アイテム] テーブルに追加されている場合、[関連 PSR] テーブルは変更できません。

例: 製品サービス依頼への関連 PSR の追加

```
void addRelatedPSRs(IServiceRequest sr, String[] psrNum) throws
APIException {
  //Get the Related PSR table
  ITable relPSR =
sr.getTable(ServiceRequestConstants.TABLE_RELATEDPSR);

  //Create a HashMap to store parameters
  HashMap params = new HashMap();

  //Add PSRs to the Related PSR table
  for (int i = 0; i < psrNum.length; i++)
  {
    //Set the PSR Number value
    params.put(ServiceRequestConstants.ATT_RELATED_PSR_PSR_NUMBER,
psrNum[i]);
    //Create a new row in the Related PSR table
    IRow row = relPSR.createRow(params);

    //Reset parameters
    params = null;
  }
}
```

品質変更依頼の使用

品質分析者は、品質変更依頼 (QCR) を使用して、製品、ドキュメント、サプライヤおよび顧客に関連する問題が集約された品質記録を管理できます。QCR はレビューおよび承認のために送信でき、問題点は是正または予防処置を使用して解決できます。その結果、ECO や MCO が開始され、製品、プロセスまたはサプライヤが変更される場合があります。QCR では、問題、是正処置、予防処置および設計変更間の検証記録も提供されます。

Agile PLM には、次の 2 つのクラスの品質変更依頼があります。

- CAPA - (通常) 問題レポートから表面化した欠陥に対応する是正処置または予防処置を表します。問題が CAPA 段階に到達するまでに、チームは特定のアイテムに修正が必要であることを把握します。その結果、CAPA の対象アイテムは、関連する問題レポートの対象アイテムとは異なる場合があります。たとえば、顧客が DVD-ROM ドライブに関する問題をレポートしたとします。CAPA が開始され、根本原因が IDE コントローラの欠陥であることが識別されます。その結果、CAPA と関連する問題レポートの対象アイテムは異なるアイテムになります。
- 検証 - 根拠を取得して客観的に評価することで、条件を満たす範囲を判断するための一貫した独立性のあるドキュメント化されたプロセス。検証は、問題がレポートされていないアイテムに対して実行できます。

品質変更依頼の作成

QCR を作成するには、`IAgileSession.createObject()` メソッドを使用します。指定する必要のある必須属性値はオブジェクトの番号のみです。次の例は、CAPA と検証の両方の QCR を作成する方法を示しています。

例: QCR の作成

```
public IQualityChangeRequest createCAPA(String strNum) throws
APIException {
  IQualityChangeRequest capa =
(IQualityChangeRequest)m_session.createObject(
    QualityChangeRequestConstants.CLASS_CAPA, strNum);
  return capa;

}
public IQualityChangeRequest createAudit(String strNum) throws
APIException {
  IQualityChangeRequest audit =
(IQualityChangeRequest)m_session.createObject(
    QualityChangeRequestConstants.CLASS_AUDIT, strNum);
  return audit;
}
```

品質管理者への品質変更依頼の割り当て

QCR を品質管理者に割り当てるには、[カバー ページ | 品質管理者] フィールドに値を設定します。このプロセスは、製品サービス依頼を品質分析者に割り当てる方法に類似しています。

例: QCR の割り当て

```
void assignQCR(IQualityChangeRequest qcr) throws APIException {
  Integer attrID;
  //Set attrID equal to the Quality Administrator attribute ID
  attrID =
QualityChangeRequestConstants.ATT_COVER_PAGE_QUALITY_ADMINISTRATOR
;
  //Get the Cover Page.Quality Administrator cell
  ICell cell = qcr.getCell(attrID);

  //Get available list values for the list
  IAgileList values = cell.getAvailableValues();

  //Set the value to the current user
  IUser user = m_session.getCurrentUser();
  values.setSelection(new Object[] { user });
  cell.setValue(values);
}
```

品質変更依頼を変更として保存

`IDataObject.saveAs()` メソッドを使用すると、QCR を別の QCR または ECO (つまり、変更依頼の別のタイプ) として保存できます。QCR を ECO として保存する場合、QCR の対象となるアイテムは ECO の [対象アイテム] タブに自動的に転送されません。対象アイテムを QCR から ECO に転送する場合は、その機能を提供するようにプログラムにコードを記述する必要があります。

注意 QCR を品質変更依頼または変更のスーパークラスのサブクラスではないオブジェクトとして保存しようとすると、Agile API で例外が発生します。

例: QCR を ECO として保存

```
public IChange saveQCRasECO(IQualityChangeRequest qcr) throws
APIException {
  // Get the ECO class
  IAgileClass cls =
m_admin.getAgileClass(ChangeConstants.CLASS_ECO);

  // Get autonumber sources for the ECO class
  IAutoNumber[] numbers = cls.getAutoNumberSources();

  // Save the QCR as an ECO
  IChange eco = (IChange)qcr.saveAs(ChangeConstants.CLASS_ECO,
numbers[0]);

  // Add code here to copy affected items from the QCR to the ECO
  return eco;
}
```

PSR および QCR でのワークフロー機能の使用

PSR と QCR では、すべてのワークフロー機能を IRoutable インターフェースから導出します。次の表に、製品品質オブジェクトの管理に使用できるワークフロー コマンドを示します。

機能	対応する API
PSR または QCR の検証	IRoutable.audit()
PSR または QCR のステータスの変更	IRoutable.changeStatus()
別のユーザーへの PSR または QCR の送信	IDataObject.send()
PSR または QCR の承認	IRoutable.approve()
PSR または QCR の却下	IRoutable.reject()
PSR または QCR に関するコメント	IRoutable.comment()
PSR または QCR の承認者の追加または削除	IRoutable.addApprovers() IRoutable.removeApprovers()

ワークフローの選択

新規の製品サービス依頼または品質変更依頼を作成する場合は、ワークフローを選択する必要があります。Agile PLM システムでは、製品サービス依頼および品質変更依頼の各タイプに対して複数のワークフローを定義できます。オブジェクトに対して有効なワークフローを取得するには、IRoutable.getWorkflows() を使用します。ワークフローが未割り当ての場合は、次の例に示すように IRoutable.getWorkflows() を使用してワークフローを選択できます。

例: ワークフローの選択

```
public static IServiceRequest createPSR() throws APIException {
    // Create a problem report
    IAgileClass prClass =
admin.getAgileClass(ServiceRequestConstants.CLASS_PROBLEM_REPORT);
    IAutoNumber[] numbers = prClass.getAutoNumberSources();
    IServiceRequest pr =
(IServiceRequest)m_session.createObject(prClass, numbers[0]);

    // Get the current workflow (a null object, since the workflow has
not been set yet)
    IWorkflow wf = pr.getWorkflow();
    // Get all available workflows
    IWorkflow[] wfs = pr.getWorkflows();

    // Set the problem report to use the first workflow
    pr.setWorkflow(wfs[0]);

    return pr;
}
```

次の例に示すように、Cover Page.Workflow フィールドの値を選択して製品サービス依頼または品質変更依頼にワークフローを設定することもできます。

例 12-12: [カバー ページ.ワークフロー] 属性の値の設定によるワークフローの選択

```
void selectWorkflow(IServiceRequest psr) throws APIException {
  int nAttrID;
  //Set nAttrID equal to the Workflow attribute ID
  nAttrID = ServiceRequestConstants.ATT_COVER_PAGE_WORKFLOW;

  //Get the Workflow cell
  ICell cell = psr.getCell(nAttrID);

  //Get available list values for the list
  IAgileList values = cell.getAvailableValues();

  //Select the first workflow
  values.setSelection(new Object[] {new Integer(0));
  cell.setValue(values);
}
```

第 13 章

プログラムの作成および管理

扱うトピックは次のとおりです。

プログラムについて

Agile Program Execution (PE) のプログラム管理機能を使用すると、プログラムとすべての関連要素 (アクティビティ スケジュール、成果物、ディスカッションなど) を定義できます。これらの機能では、必要なリソースの可用性の判別、タスクへのリソースの割り当て、ボトルネックの識別、割り当て超過および割り当て不足のリソース状況への対応が可能です。また、プログラム テンプレートを作成し、再利用することもできます。

プログラムをスケジュールし、実行するには、プログラム オブジェクトを使用します。各プログラムには、スケジュール情報のみでなく、添付ファイル、ディスカッションとアクション アイテム、リソースと役割、プログラムに関連するアクティビティすべての履歴が含まれています。データはルールと親子関係に基づいて上位レベルにロールアップされるため、管理状況の識別が容易になります。

Agile API では、プログラムの作成、ロードおよび使用がサポートされています。`IProgram` インターフェースは、プログラム、フェーズ、タスクおよびゲートなど、すべてのプログラム オブジェクトを表します。

他の Agile PLM ビジネス オブジェクトと同様に、`IProgram` インターフェースには `IRoutable` が実装されています。したがって、プログラムのワークフロー ステータスの変更、および他のユーザーへの送信には、同じ `IRouteable.changeStatus()` メソッドが使用されます。詳細は、176 ページの「オブジェクトのワークフロー ステータスの変更」を参照してください。

プログラム オブジェクトの動作の相違点

IProgram インターフェースには、他の Agile PLM オブジェクトで一般的に使用されるインターフェースがいくつか実装されています。ただし、他のオブジェクトとは区別される次の固有の機能も提供されます。

- プログラム オブジェクトは、フェーズ、タスクおよびゲートなど、他の基礎となるプログラム オブジェクトのコンテナです。基礎となるプログラム オブジェクトは、[スケジュール] テーブルを介して、親オブジェクト (通常はプログラム) と関連付けられます。
- プログラムには、スケジュールの変更を追跡できる基準があります。したがって、IProgram インターフェースには、基準を作成、取得または削除できるメソッドが用意されています。
- プログラムはアーカイブ可能です。ルート プログラムをアーカイブすると、プログラム ツリー全体がシステムからソフト削除されます。
- プログラムは、ロックまたはロック解除可能です。

プログラムの作成

プログラムを作成するには、IAgileSession.createObject() メソッドを使用します。プログラム パラメータを指定するときは、プログラムのサブクラス (例: プログラム、フェーズ、タスクまたはゲート) を指定する必要があります。プログラム、フェーズおよびタスクについては、次の必須プログラム属性を指定する必要があります。

- General Info.Name
- General Info.Schedule Start Date
- General Info.Schedule End Date
- General Info.Duration Type

ゲートについては、General Info.Name および General Info.Schedule End Date の 2 つの属性のみが必須です。

次の例は、新規プログラムを作成し、必須属性を指定する方法を示しています。

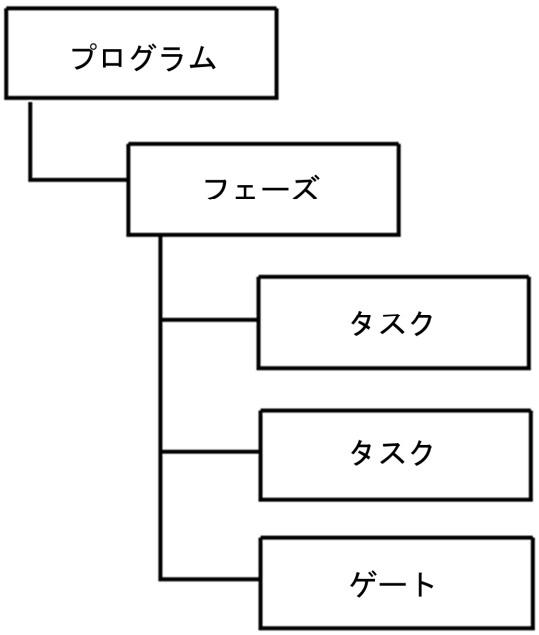
例: プログラムの作成

```
try {
  // Create a Map object to store parameters
  Map params = new HashMap();
  // Set program name
  String name = "APOLLO PROGRAM";

  // Set program start date
  Date start = new Date();
  start.setTime(1);

  // Set program end date
  Date end = new Date();
  end.setTime(1 + 2*24*60*60*1000);
```

```
  // Set program duration type
  IAttribute attr =
m_admin.getAgileClass(ProgramConstants.CLASS_PROGRAM).

getAttribute(ProgramConstants.ATT_GENERAL_INFO_DURATION_TYPE);
  IAgileList avail = attr.getAvailableValues();
  avail.setSelection(new Object[] {"Fixed"});

  // Initialize the params object
  params.put(ProgramConstants.ATT_GENERAL_INFO_NAME, name);
  params.put(ProgramConstants.ATT_GENERAL_INFO_SCHEDULE_START_DATE,
start);
  params.put(ProgramConstants.ATT_GENERAL_INFO_SCHEDULE_END_DATE,
end);
  params.put(ProgramConstants.ATT_GENERAL_INFO_DURATION_TYPE,
avail);
  // Create the program
  IProgram program =
(IProgram)m_session.createObject(ProgramConstants.CLASS_PROGRAM,
params);

} catch (APIException ex) {
   System.out.println(ex);
}
```

プログラムには、フェーズ、タスク、ゲートなど、他のタイプのアクティビティが含まれます。ゲートとは、一連の関連フェーズ、タスクまたはプログラムの完了を示す特別なマイルストーンで、終了日はあるが期間のないタスクです。次の図は、プログラム オブジェクトの階層を示しています。

図 10: プログラム階層



フェーズ、タスクおよびゲートは、IAgileSession.createObject() メソッドを使用して、他のプログラム オブジェクトと同じ方法で作成できます。これらの様々なタイプのアクティビティを作成した後は、プログラムの [スケジュール] テーブルに追加できます。詳細は、198 ページの「プログラムのスケジュール」を参照してください。

プログラムのロード

プログラムをロードするには、IAgileSession.getObject() メソッドを使用します。プログラムを一意に識別するために、General Info.Number 属性に値を指定します。また、プログラムを名前で検索して、その検索結果から選択する方法でプログラムをロードすることもできます。

注意 IProgram.getName() メソッドは、実際には General Info.Name ではなく General Info.Number 属性の値を返します。

例: プログラムのロード

```
public IProgram loadProgram(String number) throws APIException {
  IProgram program =
(IProgram)m_session.getObject(IProgram.OBJECT_TYPE, number);
  return program;
}
```

注意 プログラムに対する [ニュース] テーブルは、デフォルトで無効になっています。有効にするには、管理者として Java クライアントにログインし、[ニュース] タブを表示状態にします。

プログラム テンプレートの使用

プログラム テンプレートを使用すると、新しいプログラム、アクティビティまたはタスクの定義が簡単になります。テンプレートは、General Info.Template 属性が「Template」に設定されているプログラムです。テンプレートを使用すると、テンプレートをロードして IProgram.saveAs() メソッドを使用することによって、新規プログラムを作成できます。

この特別なバージョンの saveAs() メソッドでは、SDK を使用して次のことができます。

- テンプレートから新規プログラムを作成し、コピーするテーブルを指定できます。
- プログラムの所有者および子の所有者を変更できます。
- プログラムをテンプレートとして保存して、新規プログラム テンプレートを作成できます。

テンプレートを使用した新規プログラムの作成

この特別なバージョンの saveAs() メソッドを使用すると、元のプログラムから新規プログラムにコピーするプログラム テーブルを指定できます。すべてのテーブルを指定する必要はありません。[一般情報]、[スケジュール]、[依存関係の依存対象]、[依存関係の必須対象] および [ワークフロー] テーブルは、自動的にコピーされます。[ディスカッション]、[ニュース] および [履歴] テーブルはコピーできません。通常、次の例に示すように、[ユーザー定義 1]、[ユーザー定義 2] (使用される場合) および [チーム] テーブルはコピーする必要があります。

例: テンプレートからの新規プログラムの作成

```
try {
  // Get the program template whose number is PGM00004
  IProgram template =
(IProgram)m_session.getObject(IProgram.OBJECT_TYPE, "PGM00004");
  if (template != null) {

    // Create a hash map of the program attributes to use for the new
program
    HashMap map = new HashMap();
    String name = "Scorpio Program";
    IAttribute att =
m_admin.getAgileClass(ProgramConstants.CLASS_PROGRAM).getAttribute
(

ProgramConstants.ATT_GENERAL_INFO_TEMPLATE);
    IAgileList templateList = att.getAvailableValues();
    // Note: Available values for the Template attribute are Active,
Proposed, and Template
    templateList.setSelection(new Object[] {"Active"});
    map.put(ProgramConstants.ATT_GENERAL_INFO_NAME, name);

map.put(ProgramConstants.ATT_GENERAL_INFO_SCHEDULE_START_DATE, new
Date());
    map.put(ProgramConstants.ATT_GENERAL_INFO_TEMPLATE,
templateList);
    // Define the tables to copy to the new program from the template
    Integer pagetwo = ProgramConstants.TABLE_PAGETWO;
    Integer pagethree = ProgramConstants.TABLE_PAGETHREE;
    Integer team = ProgramConstants.TABLE_TEAM;
    Object[] tables = new Object[]{pagetwo, pagethree, team};

    // Save the template as a new program
    IProgram program =
(IProgram)template.saveAs(ProgramConstants.CLASS_PROGRAM,
                  tables, map);
  }
} catch (APIException ex) {
    System.out.println(ex);
}
```

プログラムの作成および所有権の変更

saveAs() API 呼び出しを使用してテンプレートからプログラムを作成する場合は、プログラムの所有権を変更し、その変更をプログラムの子に継承できます。SDK では、公開されている API は、次のとおりです。

```
public IAgileObject saveAs(Object type, Object[] tablesToCopy,Object
params, boolean applyToChildren)
throws APIException;
```

これを実行するには、ProgramConstants と OWNER 属性の両方の値を指定します。OWNER 属性の値は、プログラム所有者を変更するためには必須です。所有者の値をすべての子に適用する場合は、ブールの applyToChildren を true に設定します。

UI では、テンプレートからプログラムを作成するとき、プログラムの所有権を変更し、その変更を子に適用できます。この場合、SDK では、UI の動作がミラー化されます。ただし、SDK の saveAs() API を介してテンプレートからプログラムを作成するには、元のプログラムがテンプレートである必要があります。

注意 SDK では、元のプログラムの General Info.Template 属性の値が「Template」に設定されている場合、プログラムはテンプレートです。

例: テンプレートからのプログラムの作成、所有者の変更および変更の継承

```
 public  IProgram saveTemplateAndSetOwner (IProgram  template, String
userID, boolean applyToChildren) throws APIException {
    //  "template" is a program template
    //   userID --  The "userID" of the user that is specified as the
owner of the Saved program object
    //   applyToChildren --  true or false. If "true" the "specified
owner" will be the owner of the entire program tree
    //    If "false", the specified owner will be the owner of the Root
Parent object only

    HashMap map = new HashMap () ;
    String newPgmName = "PROG" + System.currentTimeMillis() ;  //
Generate a random name for the Saved Program object
    IUser user = session.getObject(UserConstants.CLASS_USER, userID) ;
    map.put(ProgramConstants.ATT_GENERAL_INFO_NAME, newPgmName);
    map.put(ProgramConstants.ATT_GENERAL_INFO_OWNER, User);

map.put(ProgramConstants.ATT_GENERAL_INFO_SCHEDULE_START_DATE, new
Date());
    //     Define the tables to copy  from the template
    //   If you do not want any tables to be copied, specify "null" for
the "tables" param

    Integer pageTwo = ProgramConstants.TABLE_PAGETWO ;
    Integer pageThree = ProgramConstants.TABLE_PAGETHREE ;
    Integer team = ProgramConstants.TABLE_TEAM ;

    Object[] tables = { pageTwo, pageThree, team } ;
    IProgram pgm = (IProgram)
root.saveAs(ProgramConstants.CLASS_PROGRAM, tables, map,
applyToChildren);
    System.out.println("New Program Number = " + pgm.getName()) ;
    System.out.println("Owner Value = " +
pgm.getValue(ProgramConstants.ATT_GENERAL_INFO_OWNER).toString())
return pgm ;
}
```

プログラムをテンプレートとして保存

プログラムを作成するとき、テンプレート属性 (ProgramConstants.ATT_GENERAL_INFO_TEMPLATE) の値を「Template」に設定すると、プログラムがテンプレートであることを指定できます。テンプレートとして指定できるのは、プログラムを作成するとき、または新規プログラムとして保存するときのみです。既存のプログラムを、「Active」または「Proposed」の状態から「Template」に変更することはできません。次の例は、プログラムを開いて、テンプレートとして保存する方法を示しています。

例: プログラムをテンプレートとして保存

```
try {
  // Get the program whose number is PGM00005
  IProgram program =
(IProgram)m_session.getObject(IProgram.OBJECT_TYPE, "PGM00005");
  if (program != null) {

    // Create a hash map of the program attributes to use for the new
program
    HashMap map = new HashMap();
    String name = "Rapid Development");
    IAttribute att =
m_admin.getAgileClass(ProgramConstants.CLASS_PROGRAM).getAttribute
(

ProgramConstants.ATT_GENERAL_INFO_TEMPLATE);
    IAgileList templateList = att.getAvailableValues();
    // Note: Available values for the Template attribute are Active,
Proposed, and Template
    templateList.setSelection(new Object[] {"Template"});
    map.put(ProgramConstants.ATT_GENERAL_INFO_NAME, name);

map.put(ProgramConstants.ATT_GENERAL_INFO_SCHEDULE_START_DATE, new
Date());
    map.put(ProgramConstants.ATT_GENERAL_INFO_TEMPLATE,
templateList);
    //Define the tables to copy to the template
    Integer pagetwo = ProgramConstants.TABLE_PAGETWO;
    Integer pagethree = ProgramConstants.TABLE_PAGETHREE;
    Integer team = ProgramConstants.TABLE_TEAM;
    Object[] tables = new Object[]{pagetwo, pagethree, team};

    // Save the program as a template
    IProgram program =
(IProgram)template.saveAs(ProgramConstants.CLASS_PROGRAM,
                  tables, map);
  }
} catch (APIException ex) {
    System.out.println(ex);
}
```

プログラムのスケジュール

プログラムをスケジュールするには、[スケジュール] テーブルを編集します。[スケジュール] テーブルでは、スケジュール アイテムを追加、編集および削除できます。[スケジュール] テーブルに新規行を追加するには、ITable.createRow() メソッドを使用し、パラメータに IProgram オブジェクトを指定します。

例: [スケジュール] テーブルの変更

```
try {
  // Define a row variable
  IRow row = null;

  // Set the date format
  DateFormat df = new SimpleDateFormat("MM/dd/yy");

  // Get a program
  IProgram program =
(IProgram)m_session.getObject(ProgramConstants.CLASS_PROGRAM,
"PGM00012");
  if (program != null) {
    // Get the Schedule table
    ITable schedule =
program.getTable(ProgramConstants.TABLE_SCHEDULE);
    Iterator i = schedule.iterator();

    // Find task T000452 and remove it
    while (i.hasNext()) {
      row = (IRow)i.next();
      String num =
(String)row.getValue(ProgramConstants.ATT_GENERAL_INFO_NUMBER);
     if (num.equals("T000452")) {
       schedule.removeRow(row);
       break;
     }
    }
    // Add a phase
    HashMap info = new HashMap();
    info.put(ProgramConstants.ATT_GENERAL_INFO_NAME,
"Specifications phase");

info.put(ProgramConstants.ATT_GENERAL_INFO_SCHEDULE_START_DATE,
df.parse("06/01/05"));

info.put(ProgramConstants.ATT_GENERAL_INFO_SCHEDULE_END_DATE,
df.parse("06/10/05"));
    IAttribute attr =
m_admin.getAgileClass(ProgramConstants.CLASS_PHASE).

getAttribute(ProgramConstants.ATT_GENERAL_INFO_DURATION_TYPE);
    IAgileList list = attr.getAvailableValues();
    list.setSelection(new Object[] {"Fixed"});
```

```
info.put(ProgramConstants.ATT_GENERAL_INFO_DURATION_TYPE, list);
    IProgram phase =
(IProgram)m_session.createObject(ProgramConstants.CLASS_PHASE,
info);
    row = schedule.createRow(phase);
    // Add a task
    info = null;
    list = null;
    info.put(ProgramConstants.ATT_GENERAL_INFO_NAME, "Write
specifications");
    info.put(ProgramConstants.ATT_GENERAL_INFO_NUMBER, "T000533");

info.put(ProgramConstants.ATT_GENERAL_INFO_SCHEDULE_START_DATE,
df.parse("06/01/05"));

info.put(ProgramConstants.ATT_GENERAL_INFO_SCHEDULE_END_DATE,
df.parse("06/05/05"));
    attr = m_admin.getAgileClass(ProgramConstants.CLASS_TASK).

getAttribute(ProgramConstants.ATT_GENERAL_INFO_DURATION_TYPE);
    list = attr.getAvailableValues();
    list.setSelection(new Object[] {"Fixed"});

info.put(ProgramConstants.ATT_GENERAL_INFO_DURATION_TYPE, list);
    IProgram task =
(IProgram)m_session.createObject(ProgramConstants.CLASS_TASK,
info);
    row = schedule.createRow(task);
    // Add a gate
    info = null;
    info.put(ProgramConstants.ATT_GENERAL_INFO_NAME,
"Specifications complete");

info.put(ProgramConstants.ATT_GENERAL_INFO_SCHEDULE_END_DATE,
df.parse("06/10/05"));
    IProgram gate =
(IProgram)m_session.createObject(ProgramConstants.CLASS_GATE,
info);
    row = schedule.createRow(gate);
  }
} catch (APIException ex) {
  System.out.println(ex);
}
```

プログラムのスケジュールが定義されていると、IProgram.reschedule() メソッドを使用してプログラムを簡単に再スケジュールできます。reschedule() メソッドで使用されるのは、いくつかのパラメータ、IProgram.RESCHEDULE 定数およびそのスケジュール オプションに対する新しい値です。次は、使用可能な IProgram.RESCHEDULE 定数のリストです。

- STARTDATE - スケジュールの開始日を指定した日付に移動します。
- ENDDATE - スケジュールの終了日を指定した日付に移動します。
- BACKWARDDAYS - スケジュールを指定した日数だけ前の日付に移動します。
- FORWARDDAYS - スケジュールを指定した日数だけ先の日付に移動します。

例: プログラムの再スケジュール

```
try {
  // Get a program
  IProgram program =
(IProgram)m_session.getObject(IProgram.OBJECT_TYPE, "PGM00012");
  if (program != null) {

    // Define new start and end dates
    String startDate = "02/01/2005 GMT";
    String endDate = "06/01/2005 GMT";
    SimpleDateFormat df = new SimpleDateFormat("MM/dd/yyyy z");
    Date start = df.parse(startDate);
    Date end = df.parse(endDate);

    // Change the schedule start date
    program.reschedule(IProgram.RESCHEDULE.STARTDATE, start);

    // Change the schedule end date
    program.reschedule(IProgram.RESCHEDULE.ENDDATE, end);

    // Move the schedule backward three days
    program.reschedule(IProgram.RESCHEDULE.BACKWARDDAYS, new
Integer(3));

    // Move the schedule forward two days
    program.reschedule(IProgram.RESCHEDULE.FORWARDDAYS, new
Integer(2));
  }
} catch (Exception ex) {
  System.out.println(ex);
}
```

プログラムの基準の使用

プログラムの基準を使用すると、実際の進行状況を元の計画と比較できます。基準を作成すると、プログラムのスケジュールのスナップショットが保持されます。基準に含まれる予測データは、更新されたタスク構造、スケジュール、実際の日付などを比較するための恒久的な基準です。

基準は、ルート プログラム オブジェクトに対してのみ作成できます。複数の基準を保存でき、比較のために後で取得できます。IProgram インターフェースには、基準を作成、取得および削除するための次のメソッドが用意されています。

- createBaseline(java.lang.Object)
- getBaseline()
- getBaselines()
- removeBaseline(java.lang.Object)
- selectBaseline(java.lang.Object)

例: 基準の作成および取得

```
try {
  // Get a program
  IProgram program =
(IProgram)m_session.getObject(IProgram.OBJECT_TYPE, "PGM00012");
  if (program != null) {

    // Create a baseline
    Object baseline = program.createBaseline("august 8 baseline");

    // Get all baselines
    Map map = program.getBaselines();

    // Get the first baseline
    Set keys = map.keySet();
    Object[] objs = keys.toArray();
    baseline = map.get(objs[0]);

    // Remove the first baseline
    program.removeBaseline(baseline);

    // Get all baselines again
    map = program.getBaselines();

    // Select the first baseline
    If (map.size() > 0) {
      keys = map.keySet();
      objs = keys.toArray();
      baseline = map.get(objs[0]);
      program.selectBaseline(baseline);
    }
  }
} catch (APIException ex) {
  System.out.println(ex);
}
```

別のユーザーへのプログラム所有権の委譲

プログラム オブジェクトの所有者またはプログラム マネージャは、プログラムの所有権を委譲することによって、それを他のユーザーに割り当てることができます。委譲されたユーザーは、承認または拒否できる依頼を受け取ります。承認すると、委譲されたユーザーがそのタスクの所有者になります。委譲された所有者には、委譲されたプログラム オブジェクトに対する [プログラム マネージャ] 役割が自動的に付与されます。

プログラムの所有権を委譲するには、IProgram.delegateOwnership() メソッドを使用します。プログラムの所有権を委譲すると、[委任された所有者] フィールドが自動的に更新されます。このフィールドは読み取り専用です。delegateOwnership() メソッドを使用すると、委譲された所有権をプログラムの子にも適用するかどうかを指定できます。

例: プログラム オブジェクトの所有権の委譲

```
try {
  // Get the task whose number is T00012
  IProgram task = (IProgram)m_session.getObject(IProgram.OBJECT_TYPE,
"T00012");
  if (task != null) {
    // Get a user
    IUser user1 = (IUser)m_session.getObject(UserConstants.CLASS_USER,
"kkieslowski");
    if (user1 != null) {
      // Delegate the task to the user
      task.delegateOwnership(user1, false);
    }
  }
} catch (APIException ex) {
  System.out.println(ex);
}
```

プログラムのチームへのリソースの追加

[チーム] テーブルを使用すると、プログラム オブジェクトに対するチーム メンバー リストを管理できます。チーム メンバーの追加または削除、チーム メンバーの役割の変更、チーム メンバーの割り当ての変更を行うことができます。プログラムの [チーム] テーブルを変更するには、適切な権限が必要です。

[チーム] テーブルにリソースを追加する場合は、そのプログラム オブジェクトに対してユーザーまたはユーザー グループに割り当てる役割を指定します。使用可能な役割は、Agile PLM の役割の完全なセットではありません。これらは、Program Execution 機能に特に関連している役割です。次は、チーム メンバーに割り当てることができる役割のリストです。

- エグゼクティブ
- 変更分析者
- プログラム チーム メンバー
- プログラム マネージャ
- リソース プール所有者
- プログラム管理者

これらの各役割の説明は、『Agile PLM 管理者ガイド』を参照してください。

[チーム] テーブルには、特別な注意が必要な 2 つの属性 `ProgramConstants.ATT_TEAM_NAME` および `ProgramConstants.ATT_TEAM_ROLES` があります。これらはそれぞれ、シングルリスト属性とマルチリスト属性です。これらの属性に対して使用可能な値を取得するには、`IAttribute.getAvailableValues()` ではなく、`ITable.getAvailableValues()` メソッドを使用します。このメソッドを使用しない場合は、メソッドから返される `IAgileList` オブジェクトに無効なリスト値が含まれる場合があります。

例: プログラムのチームへのリソースの追加

```
try {
      // Get users
     IUser user1 = (IUser)session.getObject(UserConstants.CLASS_USER,
"daveo");
     IUser user2 = (IUser)session.getObject(UserConstants.CLASS_USER,
"yvonnec");
     IUser user3 = (IUser)session.getObject(UserConstants.CLASS_USER,
"albertl");
     IUser user4 = (IUser)session.getObject(UserConstants.CLASS_USER,
"brians");

     //  Get a resource pool (user group)
     IUserGroup pool =
(IUserGroup)session.getObject(IUserGroup.OBJECT_TYPE,
"Development");
     // Add all four users to the resource pool
     ITable usersTable =
pool.getTable(UserGroupConstants.TABLE_USERS);
     usersTable.createRow(user1);
     usersTable.createRow(user2);
     usersTable.createRow(user3);
     usersTable.createRow(user4);

     // Get a program
     IProgram program =
(IProgram)session.getObject(IProgram.OBJECT_TYPE, "PGM02423");
     if (program != null) {

     // Get the Team table of the program
     ITable teamTable =
program.getTable(ProgramConstants.TABLE_TEAM);

    // Get Roles attribute values (use ITable.getAvailableValues)
     IAgileList attrRolesValues =
teamTable.getAvailableValues(ProgramConstants.ATT_TEAM_ROLES);
    // Create a hash map to hold values for row attributes
     Map map = new HashMap();
      // Add the first user to the team
     attrRolesValues.setSelection(new Object[]{"Change
Analyst","Program Manager"});
     map.put(ProgramConstants.ATT_TEAM_NAME, user1);
     map.put(ProgramConstants.ATT_TEAM_ROLES, attrRolesValues);

     IRow row1 = teamTable.createRow(map);
     // Add the second user to the team
     attrRolesValues.setSelection(new Object[]{"Program
Administrator"});
     map.put(ProgramConstants.ATT_TEAM_NAME, user2);
     IRow row2 = teamTable.createRow(map);

     // Add the resource pool to the team
     attrRolesValues.setSelection(new Object[]{"Program Team
Member"});
     map.put(ProgramConstants.ATT_TEAM_NAME, pool);
     IRow row3 = teamTable.createRow(map);

     }
```

Agile Web クライアントでは、[チーム] テーブルにリソース プールを追加する場合は、プールをその中に含まれている 1 つ以上のリソースで置き換えることができます。つまり、リソース プール全体を割り当てるかわりに、プールから選択したユーザーを割り当てることができます。この機能を使用するには、IProgram.assignUsersFromPool() メソッドを使用します。assignUsersFromPool() を使用するには、プログラムの [チーム] テーブルにすでに追加されているユーザー グループを指定する必要があります。

例: リソース プールからのユーザーの割り当て

```
public void replaceUserGroupWithUser(IProgram program) throws
Exception {
  // Get the Team table
  ITable teamTable = program.getTable(ProgramConstants.TABLE_TEAM);

  // Get a table iterator
  Iterator it = teamTable.iterator();

  // Find a user group and replace it with one of its members, kwong
  while(it.hasNext()){
    IRow row = (IRow)it.next();
    IDataObject object = row.getReferent();
    if(object instanceof IUserGroup){
      IUserGroup ug = (IUserGroup)object;
      ITable users = ug.getTable(UserGroupConstants.TABLE_USERS);
      Iterator ref_it = users.getReferentIterator();
      while(ref_it.hasNext()){
        IUser user = (IUser)ref_it.next();
        if(user.getName().equals("kwong")) {
          program.assignUsersFromPool(new IUser[]{user}, ug, true);
          break;
        }
      }
    }
  }
}
```

プログラム リソースの入れ替え

リソースの可用性は、過負荷、再割り当て、休暇および病気によって頻繁に変わる可能性があります。既存のリソースを別のリソースに入れ替えることができます。現在のリソースの役割は、入れ替えられたリソースに割り当てられます。ただし、割り当てられるのはそのプログラムに対してのみです。プログラム リソースを入れ替えるには、IProgram.substituteResource() メソッドを使用します。

リソースを入れ替える場合は、ユーザーおよびユーザー グループを指定できます。また、リソース割り当てをプログラムの子に適用するかどうかも指定できます。

例: プログラム リソースの入れ替え

```
try {
  // Get a program
  IProgram program =
(IProgram)m_session.getObject(IProgram.OBJECT_TYPE, "PGM00012");
  if (program != null) {

    // Get users
    IUser u1 = (IUser)m_session.getObject(UserConstants.CLASS_USER,
"akurosawa");
    IUser u2 = (IUser)m_session.getObject(UserConstants.CLASS_USER,
"creed");
    IUser u3 = (IUser)m_session.getObject(UserConstants.CLASS_USER,
"dlean");
    IUser u4 = (IUser)m_session.getObject(UserConstants.CLASS_USER,
"jford");

    // Get a user group
    IUserGroup ug =
(IUserGroup)m_session.getObject(IUserGroup.OBJECT_TYPE,
                "Directors");
    // Substitute u1 with u3 and do not apply to children
    program.substituteResource(u1, u3, false);

    // Substitute u2 with u4 and apply to children
    program.substituteResource(u2, u4, true);

    // Substituete u4 with a user group, and apply to children
    program.substituteResource(u4, ug, true);
  }
} catch (APIException ex) {
  System.out.println(ex);
}
```

プログラムのロックまたはロック解除

プログラムの所有者は、プログラムをロックまたはロック解除できます。プログラムがロックされていると、そのスケジュールは変更できません。プログラムをロックまたはロック解除するには、IProgram.setLock() メソッドを使用します。

注意 Agile Web クライアントでガント チャートまたは Microsoft Project 統合機能を使用すると、プログラムは自動的にロックされます。

例: プログラムのロック

```
try {
          // Get a program
          IProgram program =
        (IProgram)m_session.getObject(IProgram.OBJECT_TYPE, "PGM00012");
          if (program != null) {
            // Lock it
            program.setLock(true);
          }
        } catch (APIException ex) {
          System.out.println(ex);
        }
```

ディスカッションの使用

プロジェクトの進行中に、ユーザーが協力して、情報を交換する必要がある問題が発生します。Agile PLM には、チーム メンバーがそのフィードバックに返信し、自分の意見や考えの記録を提供できるスレッド ディスカッション機能が用意されています。ディスカッションは非同期です。つまり、すべてのディスカッション参加者が同時に接続する必要はありません。参加者は、ディスカッションのどのスレッドにも自由に返信できます。問題を終了するために、アクション アイテムをチーム リソースに割り当てることができます。ディスカッション オブジェクトは、スレッド ディスカッションとそれに関連するアクション アイテムの両方を管理するために使用されます。

ディスカッション オブジェクトは、プログラムとは異なり送信可能なオブジェクトではありません。したがって、ディスカッションにワークフローは関連付けられていません。

注意 [アクション アイテム]、[カバー ページ] および [返信] テーブルは、Agile PLM クライアントの [ディスカッション] タブに表示されます。[ユーザー定義 1] テーブルは、Agile PLM クライアントの [詳細] タブに表示されます。[使用箇所] テーブルはサポートされていません。その機能は、General Info.Related To フィールドで置き換えられています。

ディスカッションの作成

ディスカッションを作成するには、IAgileSession.createObject() メソッドを使用します。ディスカッション パラメータを指定するとき、ディスカッションのサブクラスと、次の必須ディスカッション属性を指定する必要があります。

- カバー ページ.番号
- カバー ページ.件名

[カバー ページ.通知リスト] および [カバー ページ.メッセージ] 属性のデータも指定する必要があります。指定しない場合は、ユーザーが返信できる通知リストまたはメッセージがディスカッションに対して設定されません。

次の例は、新規ディスカッションを作成して、プログラムの [ディスカッション] テーブルに追加する方法を示しています。

例: ディスカッションの作成

```
try {
  // Create a hash map variable
  Map map = new HashMap();

  // Set the Number field
  IAgileClass discussionClass =
m_session.getAdminInstance().getAgileClass(

DiscussionConstants.CLASS_DISCUSSION);
  String number =
discussionClass.getAutoNumberSources()[0].getNextNumber();
  // Set the Subject field
  String subject = "Packaging issues";

  // Make the Message field visible
  IAttribute attr =
discussionClass.getAttribute(DiscussionConstants.ATT_COVER_PAGE_ME
SSAGE);
  IProperty propVisible =
attr.getProperty(PropertyConstants.PROP_VISIBLE);
  IAgileList list = propVisible.getAvailableValues();
  list.setSelection(new Object[] { "Yes" });

  // Set the Message field
  String message = "We still have problems with the sleeves and inserts."
+
                "Let's resolve these things at the team meeting on
Friday.";
  // Set the Notify List field
  IUser user1 = m_session.getCurrentUser();
  IUser user2 = (IUser)m_session.getObject(UserConstants.CLASS_USER,
"jdassin");
  attr =
discussionClass.getAttribute(DiscussionConstants.ATT_COVER_PAGE_NO
TIFY_LIST);
  list = attr.getAvailableValues();
  list.setSelection(new Object[] {user1, user2});

  // Put the values into the hash map
  map.put(DiscussionConstants.ATT_COVER_PAGE_NUMBER, number);
  map.put(DiscussionConstants.ATT_COVER_PAGE_SUBJECT, subject);
  map.put(DiscussionConstants.ATT_COVER_PAGE_MESSAGE, message);
  map.put(DiscussionConstants.ATT_COVER_PAGE_NOTIFY_LIST, list);
```

```
  // Create a Discussion object
  IDiscussion discussion = (IDiscussion)m_session.createObject(

DiscussionConstants.CLASS_DISCUSSION, map);
  // Get a program
  IProgram program =
(IProgram)m_session.getObject(IProgram.OBJECT_TYPE, "PGM00012");

  if (program != null) {
    // Get the Discussion table
    ITable discTable =
program.getTable(ProgramConstants.TABLE_DISCUSSION);

    // Add the new discussion to the table
    discTable.createRow(discussion);
  }

} catch (APIException ex) {
  System.out.println(ex);
}
```

ディスカッションへの返信

チーム メンバーまたは通知ユーザー、つまり、ディスカッションの [カバー ページ.通知リスト] フィールドにリストされているユーザーは、ディスカッションに返信できます。ディスカッションに返信する場合は、[返信] テーブルに別のネスト テーブルを作成します。

例: ディスカッションへの返信

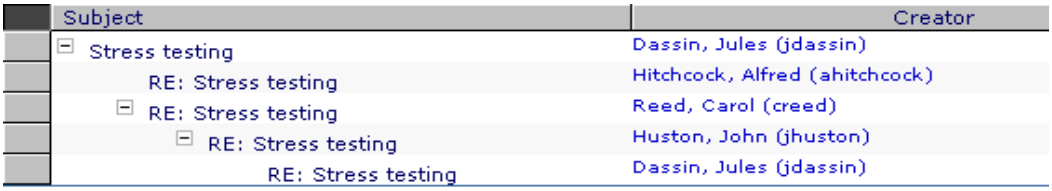
```
private void replyToDiscussion() throws Exception {
  Iterator it;
  IDiscussion discussion;

  // Get a program
  IProgram program =
(IProgram)m_session.getObject(IProgram.OBJECT_TYPE, "PGM00012");

  // Get the Discussion table
  ITable discTable =
program.getTable(ProgramConstants.TABLE_DISCUSSION);

  // Get the first Discussion listed
  if (discTable.size()!=0) {
    it = discTable.iterator();
    if (it.hasNext()) {
      IRow row = (IRow)it.next();
      discussion = (IDiscussion)row.getReferent();
    }
    // Get the Replies table
    ITable repliesTable =
discussion.getTable(DiscussionConstants.TABLE_REPLIES);
```

```
    // Iterate to the only row of the Replies table and send a reply
    it = repliesTable.iterator();
    if (it.hasNext()) {
      IRow row = (IRow)it.next();
      IMessage message = (IMessage)row;
      HashMap response = new HashMap();

      // Set the Subject field (use the same Subject as the parent)
      response.put(MessageConstants.ATT_COVERPAGE_SUBJECT,

row.getValue(DiscussionConstants.ATT_REPLIES_SUBJECT));
      // Make the Message field visible
      IAgileClass discussionClass =
m_session.getAdminInstance().getAgileClass(

DiscussionConstants.CLASS_DISCUSSION);
      IAttribute attr =
discussionClass.getAttribute(DiscussionConstants.ATT_COVER_PAGE_ME
SSAGE);
      IProperty propVisible =
attr.getProperty(PropertyConstants.PROP_VISIBLE);
      IAgileList list = propVisible.getAvailableValues();
      list.setSelection(new Object[] { "Yes" });

      // Set the Message field
      response.put(MessageConstants.ATT_COVERPAGE_MESSAGE,
                "The spec needs to be updated to reflect the latest
decisions.");
      // Send a reply
      message.reply(response);
    }
  }
}
```

この例は、ルート ディスカッションに返信する方法を示しています。次に、ディスカッションに複数の返信があり、最新の返信に返信する場合はどのように処理するかを考えてみます。この場合はもう少し複雑で、[返信] テーブルをさらに理解する必要があります。

ディスカッションの [返信] テーブルは、他の Agile PLM テーブルとは異なります。このテーブルには、複数の返信がある場合でも、行が 1 つしか含まれません。ディスカッションに複数の返信がある場合、返信は一連のネスト テーブル内に含まれます。最新の返信を選択するには、最後のネスト テーブルまで [返信] テーブルを展開します。次の図は、Agile Web クライアントの展開された [返信] テーブルを示しています。

図 1: 展開された [返信] テーブル



次の例に示すように、再帰メソッド (それ自身を呼び出すメソッド) を使用すると、[返信] テーブルのすべてのレベルを展開できます。[返信] テーブルの次に続く各レベルは、子テーブル属性 (DiscussionConstants.ATT_REPLIES_CHILD_TABLE) の値を取得することによって取得されます。

例: [返信] テーブルの展開方法

```
// Read the Replies table
public void readRepliesTable(IDiscussion discussion) throws Exception
{
  ITable replies =
discussion.getTable(DiscussionConstants.TABLE_REPLIES);
  browseReplies(0, replies);
}
// Recursively browse through all levels of the Replies table
void browseReplies(int indent, ITable replies) throws Exception {
  Iterator i = replies.iterator();
  while (i.hasNext()) {
    IRow row = (IRow) i.next();
    System.out.print(indent(indent*4));
    readRow(row);
    System.out.println();
    ITable followup =
(ITable)row.getValue(DiscussionConstants.ATT_REPLIES_CHILD_TABLE);
    browseReplies(indent + 1, followup);
  }
}
// Read each cell in the row and print the attribute name and value
static protected void readRow(IRow row) throws Exception {
  ICell[] cells = row.getCells();
  for (int j = 0; j < cells.length; ++j) {
    Object value = cells[j].getValue();
    System.out.print( "\t" + cells[j].getAttribute().getName() + "="+
value);
  }
}
// Indent text
private String indent(int level) {
  if (level <= 0) {
    return "";
  }
  char c[] = new char[level*2];
  Arrays.fill(c, ' ');
  return new String(c);
}
```

ディスカッションへの参加

Agile Web クライアントでは、プログラムの [ディスカッション] タブをクリックし、[参加] ボタンをクリックすると、ディスカッションに参加できます。ディスカッションに参加すると、ディスカッション オブジェクトの [通知リスト] フィールドにユーザー名が追加されます。Agile API を使用してディスカッションに参加するには、自分自身を [通知リスト] フィールドに追加します。ディスカッションに参加できるのは、そのプログラムのチーム メンバーである場合のみです。

注意 ディスカッション オブジェクトの通知リストに含まれていない場合、返信を表示することはできません。ただし、プログラムの [チーム] テーブルにリストされているユーザーは、そのプログラムに関連付けられているディスカッションに参加できます。

例: ディスカッションへの参加

```
try {
  // Get a program
  IProgram program =
(IProgram)m_session.getObject(ProgramConstants.CLASS_PROGRAM,
"PGM00012");
  if (program != null) {
    // Get the Discussion table
    ITable discTable =
program.getTable(ProgramConstants.TABLE_DISCUSSION);

    // Get the first discussion
    IRow row = (IRow)discTable.iterator().next();
    IDiscussion discussion = (IDiscussion)row.getReferent();

    // Add yourself and another user to the Notify List field
    IUser user1 = m_session.getCurrentUser();
    IUser user2 = (IUser)m_session.getObject(UserConstants.CLASS_USER,
"owelles");
    ICell cell =
discussion.getCell(DiscussionConstants.ATT_COVER_PAGE_NOTIFY_LIST)
;
    IAgileList list = (IAgileList)cell.getAvailableValues();
    list.setSelection(new Object[] {user1, user2});
  }
} catch (APIException ex) {
  System.out.println(ex);
}
```

アクション アイテムの作成

アクション アイテムは、ディスカッション オブジェクトの一部として作成できます。ディスカッションで、他のユーザーがアクションを実行する必要がある問題が発生した場合は、そのアクションを別のユーザーに割り当てることができます。アクション アイテムには、件名、ステータス、締切日および割り当てられたユーザーが設定されます。アクション アイテムを作成すると、割り当てられたユーザーの [通知と依頼] 受信トレイに表示されます。

アクション アイテムを作成するには、ITable.createRow() メソッドを使用して、プログラム オブジェクトの [アクション アイテム] テーブルに行を追加します。行の初期化に使用されるマップ オブジェクトに、[件名]、[割り当て先] および [締切日] フィールドに対するパラメータが含まれていることを確認してください。

例: アクション アイテムの作成

```
private void replyToDiscussion() throws Exception {
  // Get a program
  IProgram program =
(IProgram)m_session.getObject(IProgram.OBJECT_TYPE, "PGM00012");
  if (program != null) {
    // Create a hash map for Action Item parameters
    HashMap map = new HashMap();

    // Set the Subject field
    String subj = "Update packaging requirements";
    map.put(ProgramConstants.ATT_ACTION_ITEMS_SUBJECT, subj);
```

```
    // Set the Assigned To field
    IUser user1 = (IUser) m_session.getObject(UserConstants.CLASS_USER,
"akurosawa");
    IAttribute attr = m_session.getAdminInstance().getAgileClass(

ProgramConstants.CLASS_PROGRAM).getAttribute(

ProgramConstants.ATT_ACTION_ITEMS_ASSIGNED_TO);
    IAgileList list = attr.getAvailableValues();
    list.setSelection(new Object[] {user1});
    map.put(ProgramConstants.ATT_ACTION_ITEMS_ASSIGNED_TO, list);

    // Set the Due Date field
    DateFormat df = new SimpleDateFormat("MM/dd/yy");
    map.put(ProgramConstants.ATT_ACTION_ITEMS_DUE_DATE,
df.parse("03/30/05"));

    // Get the Action Items table
    ITable table =
program.getTable(ProgramConstants.TABLE_ACTIONITEMS);

    // Add the new Action Item to table
    table.createRow(map);
  }
} catch (APIException ex) {
  System.out.println(ex);
}
```

第 14 章

Product Cost Management の使用

扱うトピックは次のとおりです。

Product Sourcing について

Agile PLM の Product Sourcing モジュールでは、製品のライフサイクル全体を通して、製品コスト関連の全データの処理がサポート、強化および簡略化されます。この結果、ソーシング コンテンツの効率的な管理と操作、サプライヤとの協力による新しいソーシング コンテンツの設定、およびデータの分析が可能になります。Product Sourcing では、次の機能がサポートされています。

- ソーシング プロジェクトの作成
- 製品コンテンツの収集および準備
- 価格契約および履歴の利用
- 見積依頼の作成
- サプライヤ見積依頼回答の管理および価格の交渉 (PCM SDK では未サポート)
- プロジェクト分析の実施

Agile API では、次の Product Sourcing オブジェクトがサポートされています。

- `IChange - Change` クラスのインターフェース。価格変更 (PCO) が含まれます。
- `IPrice - Price` クラスのインターフェース。公表価格と履歴価格の両方が処理されます。
- `IProject - Project` クラスのインターフェース。プロダクト ソーシング データ用に使用されるコンテナです。
- `IRequestForQuote - RequestForQuote` クラスのインターフェース。ソーシング プロジェクトの見積依頼を表します。
- `ISupplier - Supplier` クラスのインターフェース。

`ISupplierResponse` オブジェクトを除いて、Agile API では、すべての Product Sourcing オブジェクトを読み取りおよび変更できます。次の表に、Product Sourcing オブジェクトに対する作成、読み取りおよび変更権限を示します。

オブジェクト	作成	読み取り	変更
IChange (PCO を含む)	可能	可能	可能
IPrice	可能	可能	可能
IProject	可能	可能	可能
IRequestForQuote	可能	可能	可能
ISupplier	可能	可能	可能

価格の管理

非効率的な手動システムは、Agile PLM の価格管理ソリューションによって置き換えられます。手動システムでは、たいていの場合、価格は異なる場所にあるファイル、スプレッドシートまたはデータベースに格納されます。Agile PLM システムを使用すると、アイテムと製造元部品の価格と条件を作成でき、中央で管理できます。

次の 2 つのデフォルトの価格クラスがシステムに付属しています。

- 見積履歴 - 見積履歴オブジェクトには、以前のプロジェクトまたはレガシー データからの価格見積が含まれています。
- 公表価格 - 公表価格には、対象のアイテムおよび製造元部品に関する公表価格または契約価格が含まれています。

次は、アイテムまたは製造元部品の価格の定義に使用される基本ステップです。

1. 適切な役割があるユーザーが、番号、説明、アイテムまたは製造元部品、サプライヤ、拠点および顧客を指定して、新しい価格オブジェクトを作成できます。
2. 価格オブジェクトの作成後、ユーザーは、各関連アイテムまたは製造元部品に対する価格/条件のマトリックスを作成できます。価格と条件のマトリックスには、有効日、数量、価格およびキャンセル期間が含まれます。
3. 価格オブジェクトが提出され、ワークフロー承認プロセスに従って処理されます。他のユーザーは、オブジェクトを承認または却下できます。
4. 適切な役割があるユーザーが、価格変更 (PCO) を作成してリリース済みの価格オブジェクトを変更できます。更新された価格オブジェクトは、承認のために再度提出されます。

価格オブジェクトの作成

価格オブジェクトを作成するには、いくつかの手順があります。最初に、オブジェクト クラスと一意の識別属性を指定し、次に、新しい価格オブジェクトを返す `IAgileSession.createObject()` を使用します。

価格オブジェクトは、指定する必要があるキー属性が複数あるため、他の Agile API オブジェクトより複雑です。他のほとんどの Agile API オブジェクトでは、キー オブジェクトは、オブジェクトの番号などの 1 つのみです。価格オブジェクトでは、番号、顧客、アイテムまたは製造元部品、リビジョン (アイテムの場合)、プログラム、拠点およびサプライヤを指定する必要があります。これらの属性のいずれかが指定されない場合は、例外が発生し、価格オブジェクトは作成されません。

注意 拠点別の情報を処理していない場合は、製造拠点属性にグローバル拠点を指定します。

価格オブジェクトの作成後は、[カバー ページ]、[ユーザー定義 1]、[ユーザー定義 2] のフィールドに値を設定して、価格オブジェクトを詳細に定義できます。アイテムおよび製造元部品の価格と条件を定義するには、[価格ライン] テーブルに行を追加します。添付するファイルまたはドキュメントがある場合は、[添付ファイル] テーブルに追加します。

デフォルト

価格を `Program==All and Customer==All` で作成する場合は、価格作成時に `PriceConstants.ATT_GENERAL_INFORMATION_CUSTOMER` および `PriceConstants.ATT_GENERAL_INFORMATION_Program` に値を渡す必要はありません。デフォルトで、価格は `Progam==All and Customer==All` で作成されます。

アイテム リビジョンの指定

価格作成時にアイテム リビジョンを指定する場合は、リビジョン番号のかわりに変更番号を渡す必要があります。

例: 変更番号を渡してアイテム リビジョンを指定する場合

```
//Pass the change number
params.put(Priceconstants.ATT_GENERAL_INFORMATION_ITEM_REV,
"CO-35884");

//Instead of the revision number
params.put(PriceConstants.ATT_GENERAL_INFORMATION_ITEM_REV, "B")
```

公表価格の作成

次の例は、公表価格を作成する方法を示しています。

例: 公表価格の作成

```
public void createPublishedPrice(ICustomer customer, ISupplier
supplier) throws Exception {
  HashMap params = new HashMap();
  IAgileClass cls =
m_admin.getAgileClass(PriceConstants.CLASS_PUBLISHED_PRICE);
  IAutoNumber an = cls.getAutoNumberSources()[0];
  params.put(PriceConstants.ATT_GENERAL_INFORMATION_NUMBER, an);

params.put(PriceConstants.ATT_GENERAL_INFORMATION_CUSTOMER,
customer);

params.put(PriceConstants.ATT_GENERAL_INFORMATION_ITEM_NUMBER,
"1000-02");

params.put(PriceConstants.ATT_GENERAL_INFORMATION_ITEM_REV,
"CO-35884");

params.put(PriceConstants.ATT_GENERAL_INFORMATION_PROGRAM,
"PROGRAM0023");

params.put(PriceConstants.ATT_GENERAL_INFORMATION_MANUFACTURING_SI
TE, "San Jose");
```

```
params.put(PriceConstants.ATT_GENERAL_INFORMATION_SUPPLIER,
supplier);
  IPrice price = (IPrice)m_session.createObject(cls, params);
}
```

価格オブジェクトのロード

価格オブジェクトをロードするには、IAgileSession.getObject() メソッドを使用します。価格オブジェクトを一意に識別するために、[タイトル ブロック | 番号] 属性に値を指定します。

例: 価格オブジェクトのロード

```
public IPrice getPrice() throws Exception {
  IPrice price = (IPrice)m_session.getObject(IPrice.OBJECT_TYPE,
"PRICE10008");
  return price;
}
```

価格オブジェクトのテーブルのリストについては、SDK コードが記述されている、Javadoc で生成された HTML ファイルを参照してください。このファイルは、Agile ドキュメント Web サイト (http://docs.agile.com) の Agile 9.2.2.3 SDK Sample (Zip ファイル) フォルダにあります。パスは samples¥html¥pages.html です。

価格ラインの追加

価格オブジェクトの [価格ライン] テーブルでは、関連アイテムまたは製造元部品の価格と条件を定義できます。[価格ライン] テーブルに行を追加する場合は、行を値で初期化する必要があります。少なくとも、次の属性に値を指定する必要があります。

- ATT_PRICE_LINES_SHIP_FROM
- ATT_PRICE_LINES_SHIP_TO
- ATT_PRICE_LINES_PRICE_EFFECTIVE_FROM_DATE
- ATT_PRICE_LINES_PRICE_EFFECTIVE_TO_DATE
- ATT_PRICE_LINES_QTY

これらの属性のいずれかに値を指定しない場合、価格ライン行は作成されません。

例: 価格ラインの追加

```
public void addPriceLines(IPrice price) throws Exception {
DateFormat df = new SimpleDateFormat("MM/dd/yy");
  IAgileClass cls = price.getAgileClass();
  ITable table = price.getTable(PriceConstants.TABLE_PRICELINES);
  IAttribute attr = null;
  IAgileList listvalues = null;
  HashMap params = new HashMap();
```

```
  //Set Ship-To Location (List field)
  attr = cls.getAttribute(PriceConstants.ATT_PRICE_LINES_SHIP_TO);
  listvalues = attr.getAvailableValues();
  listvalues.setSelection(new Object[] { "San Jose" });
  params.put(PriceConstants.ATT_PRICE_LINES_SHIP_TO, listvalues);

  //Set Ship-From Location (List field)
  attr =
cls.getAttribute(PriceConstants.ATT_PRICE_LINES_SHIP_FROM);
  listvalues = attr.getAvailableValues();
  listvalues.setSelection(new Object[] { "Hong Kong" });
  params.put(PriceConstants.ATT_PRICE_LINES_SHIP_FROM, listvalues);

  //Set Effective From (Date field)

params.put(PriceConstants.ATT_PRICE_LINES_PRICE_EFFECTIVE_FROM_DAT
E, df.parse("10/01/03"));
  //Set Effective To (Date field)

params.put(PriceConstants.ATT_PRICE_LINES_PRICE_EFFECTIVE_TO_DATE,
df.parse("10/31/03"));
  //Set Quantity (Number field)
  params.put(PriceConstants.ATT_PRICE_LINES_QTY, new
Integer(1000));

  //Set Currency Code (List field)
  attr =
cls.getAttribute(PriceConstants.ATT_PRICE_LINES_CURRENCY_CODE);
  listvalues = attr.getAvailableValues();
  listvalues.setSelection(new Object[] { "USD" });
  params.put(PriceConstants.ATT_PRICE_LINES_CURRENCY_CODE,
listvalues);

  //Set Total Price (Money field)
  params.put(PriceConstants.ATT_PRICE_LINES_TOTAL_PRICE, new
Money(new Double(52.95), "USD"));

  //Set Total Material Price (Money field)

params.put(PriceConstants.ATT_PRICE_LINES_TOTAL_MATERIAL_PRICE,
new Money(new Double(45.90), "USD"));
  //Set Total Non-Materials Price (Money field)

params.put(PriceConstants.ATT_PRICE_LINES_TOTAL_NON_MATERIAL_PRICE,
new Money(new Double(7.05),
   "USD"));
  //Set Lead Time (Number field)
  params.put(PriceConstants.ATT_PRICE_LINES_LEAD_TIME, new
Integer(5));
```

```
  //Set Transportation Time (List field)
  attr =
cls.getAttribute(PriceConstants.ATT_PRICE_LINES_TRANSPORTATION_TIM
E);
  listvalues = attr.getAvailableValues();
  listvalues.setSelection(new Object[] { "FOB" });

params.put(PriceConstants.ATT_PRICE_LINES_TRANSPORTATION_TIME,
listvalues);
  //Set Country of Origin (List field)
  attr =
cls.getAttribute(PriceConstants.ATT_PRICE_LINES_COUNTRY_OF_ORIGIN)
;
  listvalues = attr.getAvailableValues();
  listvalues.setSelection(new Object[] { "United States" });

params.put(PriceConstants.ATT_PRICE_LINES_COUNTRY_OF_ORIGIN,
listvalues);
  //Create a new Price Lines row and initialize it with data
  IRow row = table.createRow(params);
}
```

価格変更の作成

公表価格や契約などの価格オブジェクトには、リビジョン履歴があります。価格オブジェクトがリリースされている場合、変更するには、最初に価格変更 (PCO) を作成し、価格オブジェクトを [対象価格] テーブルに追加する必要があります。次に、PCO が承認のために提出されます。価格オブジェクトへの変更は、PCO のワークフロー承認プロセスが完了すると有効になります。

PCO は、ECO や ECR などの他の変更オブジェクトと同じです。`IAgileSession.createObject()` メソッドを使用して PCO を作成できます。

例 14-5: PCO の作成

```
public void createPCO(IPrice price) throws Exception {
  //Get the PCO class
  IAgileClass cls =
m_admin.getAgileClass(ChangeConstants.CLASS_PCO);

  //Get autonumber sources for the PCO class
  IAutoNumber[] numbers = cls.getAutoNumberSources();

 //Create the PCO
 IChange pco =
(IChange)m_session.createObject(ChangeConstants.CLASS_PCO,
numbers[0]);

  //Get the Affected Prices table
  ITable affectedPrices =
pco.getTable(ChangeConstants.TABLE_AFFECTEDPRICES);

  //Add the Price object to the Affected Prices table
  IRow row = affectedPrices.createRow(price);
}
```

サプライヤの使用

Agile PLM システムには、次の 5 種類のサプライヤ クラスが用意されています。

- ブローカー
- 部品メーカー
- 受託製造業者
- ディストリビュータ
- メーカー代表者

各サプライヤを一意に識別する 2 つのプライマリ キー属性 GENERAL_INFO_NUMBER および GENERAL_INFO_NAME があります。

サプライヤのロード

サプライヤをロードするには、IAgileSession.getObject() メソッドを使用します。サプライヤを一意に識別するために、[一般情報 | 番号] 属性に値を指定します。

例 14-6: サプライヤのロード

```
public ISupplier getSupplier() throws APIException {
  ISupplier supplier =
(ISupplier)m_session.getObject(ISupplier.OBJECT_TYPE, "SUP20013");
  return supplier;
}
```

注意 Agile API では、[サプライヤ] テーブルへの新規行の追加はサポートされていません。

サプライヤ データの変更

Agile API を使用すると、[サプライヤ] のすべての読み取り/書き込みフィールドを読み取りおよび更新できます。[一般情報]、[ページ 1] および [ユーザー定義 2] のフィールドについては、セルに直接アクセスできます。[コンタクト ユーザー] テーブルなど、複数行のテーブルのセルにアクセスするには、最初にテーブルをロードし、特定の行を選択する必要があります。

例: サプライヤ データの変更

```
public void updateSupplierGenInfo(ISupplier supplier) throws
Exception {
  ICell cell = null;
  IAgileList listvalues = null;

  //Update Name (Text field)
  cell = supplier.getCell(SupplierConstants.ATT_GENERAL_INFO_NAME);
  cell.setValue("Global Parts");

  //Update URL (Text field)
  cell = supplier.getCell(SupplierConstants.ATT_GENERAL_INFO_URL);
  cell.setValue("http://wwww.globalpartscorp.com");
```

```
  //Update Corporate Currency (List field)
  cell =
supplier.getCell(SupplierConstants.ATT_GENERAL_INFO_CORPORATE_CURR
ENCY);
  listvalues = cell.getAvailableValues();
  listvalues.setSelection(new Object[] { "EUR" });
  cell.setValue(listvalues);
}

public void updateSupplierContactUsers(ISupplier supplier) throws
Exception {
  ICell cell = null;
  IAgileList listvalues = null;

  //Load the Contact Users table
  ITable contactusers =
supplier.getTable(SupplierConstants.TABLE_CONTACTUSERS);

  //Get the first row
  ITwoWayIterator i = contactusers.getTableIterator();
  IRow row = (IRow)i.next();

  //Update Email (Text field)
  cell = row.getCell(SupplierConstants.ATT_CONTACT_USERS_EMAIL);
  cell.setValue("wangsh@globalpartscorp.com");
}
```

ソーシング プロジェクトの使用

ソーシング プロジェクトでは、見積依頼 (RFQ) やソーシング分析など、ソーシング タスクのコンテンツを準備できます。プロジェクトは、中央で管理される協力性の高いソリューションです。複数のユーザーがプロジェクトにデータを追加でき、ソーシング結果の分析を実行できます。ソーシング プロジェクトは、すべてのソーシング活動のホームの役割を果たすため、`Supplier`、`RequestForQuote` (RFQ) および `SupplierResponse` など、多数のオブジェクト クラスにリンクされます。

Agile API を使用すると、次のアクションを実行できます。

- 既存のソーシング プロジェクトのロード
- 数量割引の指定によるプロジェクトの作成
- 価格期間の指定によるプロジェクトの作成
- プロジェクトを開くおよび閉じる
- アイテムの追加 (プロジェクト アイテムへの AML の追加を含む)
- プロジェクト内のオブジェクト、テーブルおよび属性へのアクセスと変更
- プロジェクト ステータスへのアクセスおよび変更
- プロジェクト AML の更新
- プロジェクト内の [ページ 1]、[ユーザー定義 1] および [ユーザー定義 2] の更新
- プロジェクト内のネストされた [価格] テーブルの読み取りおよび更新

- ソーシング プロジェクトのアイテムの数量の設定
- ソーシング プロジェクトのアイテムの目標価格の更新
- ソーシング プロジェクトのアイテムに対するパートナーの設定
- ソーシング プロジェクトでの数量ロールアップの実行
- ソーシング プロジェクト内の「最良」に指定された回答の設定

ソーシング プロジェクトに追加機能を提供する Web クライアントとは異なり、Agile API では、簡易データ抽出および更新の目的でプロジェクトが公開されます。したがって、Agile API では、次の機能はサポートされていません。

- アイテム、部品分類または製造元部品の検証
- プロジェクト テーブルのフィルタリング
- プロジェクトの価格算出ケースの変更 (数量割引および有効期間の変更)

サポートされている API メソッド

ソーシング プロジェクトに対して、次の API 9.2.1.5 メソッドがサポートされています。これらのインターフェースの詳細は、SDK コードが記述されている HTML ファイルを参照してください。このファイルは、Agile ドキュメント Web サイト (http://docs.agile.com) の 9.2.1.5 SDK Sample (Zip ファイル) フォルダにあります。パスは samples¥html¥pages.html です。

- `IAgileSession.createObject(Object, Object)`
- `IAgileSession.createObject(int, Object)`
- `IAgileSession.getObject(Object, Object)`
- `IAgileSession.getObject(int, Object)`
- `IProject`
  - `assinSupplier (Object partnerParams)`
  - `rollupQuantity ()`
- `IProject.getName()`
- `IProject.changeStatusToOpen()`
- `IProject.changeStatusToClose()`
- `IProject.getTable(Object)`
- `IRow.getValue(Object)`
- `IRow.setValue(Object, Object)`
- `ITable.iterator()`
- `ITable.getName()`
- `ITable.getTableDescriptor()`
- `ITable.size()`
- `ITable.createRow(Object)`

注意 PCM SDK では IRow.getReferent() メソッドはサポートされていません。

既存のプロジェクトのロード

既存のプロジェクトをロードするには、IAgileSession.getObject() メソッドを使用します。ソーシングプロジェクトを一意に識別するために、[カバー ページ | 番号] 属性に値を指定します。

例: プロジェクトのロード

```
public IProject getProject() throws APIException {
  String prjnum = "PRJACME_110";
  IProject prj = (IProject)m_session.getObject(IProject.OBJECT_TYPE,
prjnum);

  return prj;
}
```

数量割引の指定によるプロジェクトの作成

プロジェクトの定義には、汎用 IAgileSession メソッドを使用します。

例: プロジェクトの作成

```
IAgileObject createObject (Object objectType, Object params)
  throws APIException;
```

プロジェクトの作成では、次のパラメータ セットのいずれかを指定する必要があります。

- プロジェクト番号と数量割引

または

- プロジェクト番号、数量割引および価格期間情報

注意 数量割引は必須パラメータであり、必ず指定されます。次の例では、数量割引パラメータを使用してプロジェクトを作成しています。

例: 数量割引の指定によるプロジェクトの作成

```
IAgileClass agClass =
m_admin.getAgileClass(ProjectConstants.CLASS_SOURCING_PROJECT);
IAutoNumber number = agClass.getAutoNumberSources()[0];
HashMap map = new HashMap();
map.put(ProjectConstants.ATT_GENERAL_INFORMATION_NUMBER, number);
map.put(ProjectConstants.ATT_GENERAL_INFORMATION_NUMBER_OF_QTY_BRE
AKS, new Integer(4));
IProject prj = (IProject) m_session.createObject(agClass, map);
```

重要 QUANTITY_BREAK 属性に 2 桁を超える数値を渡さないでください。

数量割引および価格期間の指定によるプロジェクトの作成

別の方法として、数量割引と価格期間情報 (期間数、期間タイプおよび開始日) を指定してプロジェクトを作成できます。次の例では、これらのパラメータを使用してプロジェクトを作成しています。

重要 価格期間情報を期間タイプに設定してソーシング プロジェクトを作成する場合は、Period Type 属性を指定する必要があります。サポートされている値は、[毎月]、[四半期ごと]、[半年ごと] および [毎年] です。ただし、後で、getValue(ProjectConstants.ATT_GENERAL_INFORMATION_PERIOD_TYPE) を起動するなどの方法で期間タイプの値を確認すると、PeriodType は正しく返されません。つまり、プロジェクトの作成時に設定した値ではなく、将来返される値は常に “Weekly” です。これはエラーではありません。正常な SDK の動作であり、指定した期間タイプの値は、内部のみで使用されるため変更されません。

例: 数量割引および価格期間の指定によるプロジェクトの作成

```
/*
Descriptions
     ATT_GENERAL_INFORMATION_PERIOD_TYPE is described in
     ProjectConstants
     Name: Period Type
     Description: Period Type indicates the recurrence of
     price periods in a sourcing project.
     Type: List
     List: Period Type List
     List Id: 4565
     List Valid Values: {Monthly, Quarterly, Semi-Annually,
     Yearly}
     Restrictions: Required, Read Only. Used only when
     creating sourcing project. Internal use only. Not
     available through Agile UI clients.
     ATT_GENERAL_INFORMATION_PERIOD_START_DATE is described
     in ProjectConstants
     Name: Period Start Date
     Description: Period Start Date indicates the start date
     for price periods in a sourcing project
     Type: Date
     Valid Values: any Date object.
     Restrictions: Required, Read only, Used only when
     creating sourcing project, Internal use only /Not
     available through Agile UI clients
*/

IAgileClass agClass =
m_admin.getAgileClass(ProjectConstants.CLASS_SOURCING_PROJECT);
       IAutoNumber number = agClass.getAutoNumberSources()[0];
       HashMap map = new HashMap();

map.put(ProjectConstants.ATT_GENERAL_INFORMATION_NUMBER, number);
```

```
map.put(ProjectConstants.ATT_GENERAL_INFORMATION_NUMBER_OF_QTY_BRE
AKS, new Integer(4));

map.put(ProjectConstants.ATT_GENERAL_INFORMATION_NUMBER_OF_PERIODS,
new Integer(4));
IAgileList list =
agClass.getAttribute(PERIODTYPE).getAvailableValues();
String TYPE = "Monthly";
list.setSelection(new Object[]{TYPE});

map.put(ProjectConstants.ATT_GENERAL_INFORMATION_PERIOD_TYPE,
list);

map.put(ProjectConstants.ATT_GENERAL_INFORMATION_PERIOD_START_DATE,
(new GregorianCalendar()).getTime());
       IProject prj = (IProject) m_session.createObject(agClass, map);
```

オブジェクト、テーブルおよび属性へのアクセスと変更

汎用の IDataObject メソッドを、getObject、getTable、getValue、setValue などの標準呼び出しで使用すると、次のように、オブジェクト、テーブルおよび属性にアクセスし、続いてこれらを変更できます。

- [ページ 1] ([カバー ページ])、[ユーザー定義 1]、[ユーザー定義 2]、[アイテム]、[AML]、[分析] およびネストされた [価格] テーブルの読み取り
- [ページ 1] ([カバー ページ])、[ユーザー定義 1]、[ユーザー定義 2] および [AML] テーブルの更新
- アイテムの追加 ([アイテム] テーブルへの AML の追加を含む)
- 見積依頼テーブルの読み取り
- 見積依頼テーブルのロード

com.agile.api.ProjectConstants.java ファイルには、クラス、テーブルおよび属性に関する情報が含まれています。

PCM では、次のテーブル操作はサポートされていません。

- PCM 固有のテーブルの並べ替え。これらのテーブルにはデフォルトの並べ替え順がありません。該当するテーブルは、[プロジェクト アイテム]、[AML]、[分析]、[見積依頼]、[見積依頼回答]、[回答ステータス]、[サプライヤ回答] および [変更] です。
- [見積依頼回答] および [ソーシング プロジェクト] テーブルからのアイテムの削除。PCM SDK では、ITable.clear() または ITable.removeRow() がサポートされていないためです。

プロジェクトの [カバー ページ] の値の設定

プロジェクトのすべての読み取り/書き込みセルを読み取りおよび更新できます。次の例では、プロジェクトの [カバー ページ] ([ページ 1]) のセルを更新しています。

例: プロジェクトの [カバー ページ] の値の設定

```
public void updateProjectGenInfo (IProject project) throws Exception
{
  ICell cell = null;
  IAgileList listvalues = null;

  //Update Customer (List field)
  cell =
project.getCell(ProjectConstants.ATT_GENERAL_INFORMATION_CUSTOMER)
;
  listvalues = cell.getAvailableValues();
  listvalues.setSelection(new Object[] { "CUST00010" });
  cell.setValue(listvalues);

  //Update Description (Text field)
  cell =
project.getCell(ProjectConstants.ATT_GENERAL_INFORMATION_DESCRIPTI
ON);
  cell.setValue("Sourcing project for Odyssey III");

  //Update Manufacturing Site (List field)
  cell =
project.getCell(ProjectConstants.ATT_GENERAL_INFORMATION_MANUFACTU
RING_SITE);
  listvalues = cell.getAvailableValues();
  listvalues.setSelection(new Object[] { "Global" });
  cell.setValue(listvalues);

  //Update Ship To Location (List field)
  cell =
project.getCell(ProjectConstants.ATT_GENERAL_INFORMATION_SHIP_TO_L
OCATION);
  listvalues = cell.getAvailableValues();
  listvalues.setSelection(new Object[] { "San Jose" });
  cell.setValue(listvalues);
}
```

PCM のネスト テーブルの理解

ネスト テーブルは、テーブル内のテーブルです。これらは、AML を含む BOM やアイテムなど、多段階オブジェクト内のデータにアクセスし、変更するために使用されます。SDK でこの機能を実現する方法は、ネスト テーブル内のセル値をテーブルとして処理することです。たとえば、SDK で [BOM] テーブル内のセルに次のレベルがあることが検出された場合、セルはテーブルとして処理されます。ネスト テーブルは、PCM SDK に固有です。

プロジェクトの親テーブルとネスト子テーブルの定数

親プロジェクト テーブルと対応するネスト子テーブルの定数のリストを、「親プロジェクト テーブルと対応するネスト プロジェクト テーブル」に示します。

親テーブルの定数	ネスト子テーブルの定数	読み取り/書き込みモード
TABLE_ITEMS	ATT_ITEMS_AML	読み取り/書き込み
TABLE_ITEMS	ATT_ITEMS_PRICING	読み取り/書き込み

親テーブルの定数	ネスト子テーブルの定数	読み取り/書き込みモード
TABLE_AML	ATT_AML_PRICETABLE	読み取り/書き込み
TABLE_ITEM	ATT_ITEM_PRICE_TABLE	読み取り/書き込み
TABLE_ITEM	ATT_ITEM_BOM_TABLE	読み取り
TABLE_ANALYSIS	ATT_ANALYSIS_AML	読み取り
TABLE_ANALYSIS	ATT_ANALYSIS_PRICING	読み取り

プロジェクトまたは見積依頼のネスト テーブルへのアクセスおよび変更

次の例は、ネスト テーブルにアクセスする読み取りの例です。ネスト テーブルを変更/更新する場合は、14-19 ページの「ネストされた見積依頼テーブルの更新例」を参照してください。

注意 ネストされた PCM 価格テーブル内の通貨タイプ属性では、デフォルトの通貨単位として常に USD が使用されます。これは、バイヤーが別の通貨単位を指定した場合でも適用されます。この場合、United State Dollar (米国ドル) がデフォルトであり、サポートされている唯一の通貨です。

例: ネスト テーブルへのアクセス

```
Row row = (IRow) table.iterator.next();
ITable nested_table =
(ITable)row.getValue(ProjectConstants.ATT_ITEMS_AML);
```

ネスト テーブル変更後の更新の表示

ネスト テーブルを変更した後、変更を有効にするには、後述の例のようにテーブルを再ロードする必要があります。次の例のように、テーブルで処理を繰り返しているのみの場合は、古いデータが再表示され、新しい値は表示されません。

例: ネスト テーブルでの繰り返し処理

```
/*
 * In nested AML table, make modifications.
 * For example, insert a row, assign suppliers
 *
 */
row.getValue(attribute);
```

プロジェクト ステータスへのアクセスおよび変更

プロジェクトにはワークフローが結び付けられていないため、そのステータス変更は内部的に制御されます。ステータスは一連のメソッドで制御されます。これは、プロジェクトや見積依頼など、一部の PCM オブジェクトに対する特別なケースです。このリリースでは、[ドラフト] から [オープン] および [オープン] から [クローズ] へのプロジェクトのステータス変更がサポートされています。

プロジェクトのステータスにアクセスするには、[カバー ページ] ([ページ 1]) の [ライフサイクル フェーズ] フィールドに対して標準の IDataObject メソッドを使用します。プロジェクトのステータスを変更するには、IProject メソッドを使用します。このメソッドでは、プロジェクトを開く、変更する、および閉じることができます。プロジェクトを開くには、次の例に示すように、出荷先パラメータを設定する必要があります。

例: プロジェクトの [カバー ページ] の値の設定

```
// add Ship To //
String sj = "San Jose";
IAgileList ship2List = (IAgileList)
prj.getValue(ProjectConstants.ATT_GENERAL_INFORMATION_SHIP_TO_LOCA
TION);
ship2List.setSelection(new Object[]{sj});
prj.setValue(ProjectConstants.ATT_GENERAL_INFORMATION_SHIP_TO_LOCA
TION, ship2List);

// open project //
prj.changeStatusToOpen();

// close project //
prj.changeStatusToClose();
```

ソーシング プロジェクトでの追加データの設定

次に、ソーシング プロジェクトで見積依頼を発行するための準備について説明し、例を示します。これによって、SDK を使用して見積依頼関連のタスクを完了できるようになります。

ソーシング プロジェクトのアイテムの数量の設定

SDK を使用して、ソーシング プロジェクト内のアイテム オブジェクトに対する必要な数量を設定できます。次のコード サンプルでは、単一の価格目標に対する [アイテム] タブで、[アイテム] テーブルのこの値を設定しています。エンド ユーザーは、表示された名前 (例: 次の例の QuantityBreak2) を使用して目標価格を指定できます。

例: アイテムの数量の設定

```
// Setting Quantity for an Item //
ITable tab_item = dObj.getTable(ProjectConstants.TABLE_ITEM);
IRow row = (IRow) tab_item.iterator().next()
ITable priceTable =
row.getValue(ProjectConstants.ATT_ITEM_PRICE_TABLE);

for (Iterator iterator = priceTable.iterator(); iterator.hasNext();)
{
    IRow row = (IRow) iterator.next();
    String name = row.getName();
    if(name.equals("QuantityBreak2")){
         row.setValue(ProjectConstants.ATT_PRICEDETAILS_QUANTITY,
new Double(123));
    }
}
```

注意 アイテムの場合、数量はルート レベルでのみ設定されます。したがって、アイテムがルートでない場合は、例外 ExceptionConstants.PCM_PROJECT_ITEM_IS_NOT_ROOT が発生します。

さらに、priceTable はネスト テーブルであるため、数量の更新された値を取得するには、テーブルを再ロードする必要があります。これは、次の例で示されています。

例: ネスト テーブルの再ロードによる、更新された値の取得

```
// Getting the updated value //
priceTable = row.getValue(ProjectConstants.ATT_ITEM_PRICE_TABLE);
for (Iterator iterator = priceTable.iterator(); iterator.hasNext();)
{
     IRow iRow = (IRow) iterator.next();
     String name = iRow.getName();
     if(name.equals("QuantityBreak2")){
        Object qty =
row.getValue(ProjectConstants.ATT_PRICEDETAILS_QUANTITY));
     }
}
```

ソーシング プロジェクトでの数量ロールアップの実行

数量ロールアップでは、ソーシング プロジェクト内の選択したアイテムに対する数量値に関連するデータが生成されます。SDK では、次の API を使用して、ソーシング プロジェクトで数量ロールアップを起動できます。

```
public void rollupQuantity() throws APIException, RemoteException,
Exception;
```

次のコード サンプルでは、rollupQuantity() を使用して数量ロールアップを実行しています。

例: 数量ロールアップ

```
IProject prj =
(IProject)m_session.getObject(ProjectConstants.CLASS_SOURCING_PROJ
ECT,"PRJ00001");
prj.rollupQuantity();
```

注意 数量の更新された値を取得するには、前述の例と同様に、プロジェクトで rollupQuantity() を起動する必要があります。これは、getValue() では、数量設定後の対象アイテムの更新された値が返されないために必要です。

ソーシング プロジェクトでのパートナーの設定

パートナーは、見積依頼の完了プロジェクトの BOM を表示できます。アイテムをパートナーに送信する見積依頼に追加するとき、プロジェクトのアイテムにパートナーを割り当てることもできます。複数のパートナーが選択されている場合は、各サプライヤから受信するアイテムの割合を指定して、パートナーの間で数量を分割できます。たとえば、2 つのパートナーが同じアイテムを供給する場合は、両方のパートナーをリストに追加し、それぞれに特定の割合 (例: 50%-50% または 60%-40% など) を割り当てることができます。

SDK では、次の API を使用して、ソーシング プロジェクト内のアイテムに対するパートナーを設定し、パートナーの間で割合を分割できます。

```
public void assignSupplier(Object partnerParams) throws APIException,
RemoteException, Exception;
```

この API の動作と使用ケースは、IRequestForQuote.assignSupplier() と同じです。ただし、この API でアイテムに対する新しいパートナーを追加すると、既存のパートナーが上書きされます。したがって、既存のパートナーを削除しないようにするには、既存のパートナーを再度追加して、各パートナーに対する分割 (それぞれの割合) レベルを設定する必要があります。これは SDK でのみ発生する問題であり、GUI では、新しいパートナーを追加するときに既存のパートナーを追加する必要はありません。アイテムに対するパートナーは削除できませんが、split = 0 (所有権/参加の割合) を割り当てるとそのパートナーは削除されます。GUI の動作の詳細は、『Agile Product Lifecycle Management - Product Cost Management Supplier Guide』を参照してください。

次のコード サンプルでは、パートナーを設定し、割り当てられたパートナーの間で割合を分割しています。

例: パートナーの設定およびパートナー間での割合の分割

```
HashMap map = new HashMap();
HashMap supplierSplit = new HashMap();
HashMap partnerMap = new HashMap();

map.put(ProjectConstants.ATT_ITEM_NUMBER, item);
Double split1 = new Double(55);
Double spl1it2 = new Double(75);
supplierSplit.put(supplier1, split1);
supplierSplit.put(supplier2, split2);
partnerMap.put(ProjectConstants.ATT_PARTNERS_PARTNER,
supplierSplit);
map.put(ProjectConstants.ATT_ITEM_PARTNER_TABLE, partnerMap);
prj.assignSupplier(map);
```

item または supplier には、IItem オブジェクト、ISupplier オブジェクトまたは String オブジェクトを指定できます。パートナーは、アイテム部品番号 (IPN) に割り当てることはできますが、製造元部品番号 (MPN) に割り当てることはできません。アイテムがプロジェクトに存在しない場合は、ExceptionConstants.PCM_ERROR_INVALID_PROJECT_ITEM が発生します。

分割割合には、数値を表す任意のオブジェクトを指定できます。数値以外の場合は、ExceptionConstants.API_INVALID_PARAM 例外が発生します。

特定のパートナーのデータを取得するには、次の例に示すように、[アイテム] または [AML] タブを使用できます。

例: アイテムまたは AML の使用によるパートナー データの取得

```
ITable tab_item = prj.getTable(ProjectConstants.TABLE_ITEMS);
IRow row = (IRow) tab_item.iterator().next();
ITable partnerTable = (ITable)
row.getValue(ProjectConstants.ATT_ITEMS_PARTNERS);
```

または

```
ITable tab_item = prj.getTable(ProjectConstants.TABLE_ITEM);
IRow row = (IRow) tab_item.iterator().next();
ITable partnerTable = (ITable)
row.getValue(ProjectConstants.ATT_ITEM_PARTNER_TABLE);

for (Iterator iterator = partnerTable.iterator();
iterator.hasNext();) {
IRow iRow = (IRow) iterator.next();
String partner =
iRow.getValue(ProjectConstants.ATT_PARTNERS_PARTNER).toString();
String split =
iRow.getValue(ProjectConstants.ATT_PARTNERS_PARTNER_SPLIT).toStrin
g();
}
```

ソーシング プロジェクトでのアイテムの目標価格の変更

目標価格とは、アイテムまたは製造元部品のユニット当たりの市場コストです。目標価格は、アイテムの注文時に指定されます。各アイテムごと、各価格ポイントごとに、[AML] タブで、目標価格が [アイテム] テーブルに設定されます。価格ポイントは、アイテムの特定数量に対して見積られる目標価格です。たとえば、タイヤ X 個に対して見積られる価格などです。これは、同じタイヤ Y 個に対して見積られる価格と異なる場合があります。

注意 目標価格は常に正数です。目標価格に負の値を設定すると、`ExceptionConstants.PCM_NEGATIVE_TARGET_PRICE` 例外が発生します。

目標価格は、アイテム レベルでのみ設定されます。AML レベルで設定することはできません。エンド ユーザーは、価格ポイントに表示される名前を使用して価格ポイントを指定します。たとえば、次の例では `QuantityBreak2` です。

例: ソーシング プロジェクトでの目標価格の設定

```
ITable tab_item = dObj.getTable(ProjectConstants.TABLE_ITEMS);
IRow row = (IRow) tab_item.iterator().next();
ITable priceTable =
row.getValue(ProjectConstants.ATT_ITEMS_PRICING);

for (Iterator iterator = priceTable.iterator(); iterator.hasNext();)
{
    IRow row = (IRow) iterator.next();
   String name = row.getName();
   if(name.equals("QuantityBreak2")){
            row.setValue(ProjectConstants.ATT_PRICEDETAILS_TARGET_
COST,
                  new Money(new Double(1.23), "USD") );
    }
}
```

`priceTable` はネスト テーブルであるため、目標価格の更新された値を取得するには、次の例に示すように、このテーブルを再ロードする必要があります。これは、227 ページの「ソーシング プロジェクトのアイテムの数量の設定」と同じです。

例 14-22: priceTable の再ロードによる、更新された目標価格の値の取得

```
priceTable = row.getValue(ProjectConstants. ATT_ITEMS_PRICING);
for (Iterator iterator = priceTable.iterator(); iterator.hasNext();)
{
     IRow iRow = (IRow) iterator.next();
     String name = iRow.getName();
     if(name.equals("QuantityBreak2")){
        Object qty =
row.getValue(ProjectConstants.ATT_PRICEDETAILS_TARGET_COST));
     }
}
```

ソーシング プロジェクトでのアイテムの最良回答の設定

最良回答は、アイテムおよび製造元部品番号オブジェクトの両方に対して、[分析] タブで [分析] テーブルに設定されます。エンド ユーザーは、最低コスト、リード タイム制約中の最低コスト、最短リード タイム、サプライヤ格付および AML 推奨ステータスのパラメータの中から 3 つを指定します。詳細は、『Agile Product Lifecycle Management - Product Cost Management Supplier Guide』を参照してください。

SDK を使用すると、次のコード サンプルに示すように、アイテム部品番号 (IPN)、製造元部品番号 (MPN)、および IPN と MPN に対する最良回答を検索できます。

例: IPN に対する最良回答の設定

```
//set best response for ipn //
ITable table_analysis =
prj.getTable(ProjectConstants.TABLE_ANALYSIS);
Iterator it = table_analysis.iterator();
while(it.hasNext()) {
      IRow row = (IRow) it.next();
      String itemName =
row.getValue(ProjectConstants.ATT_ANALYSIS_NUMBER);
      String suppName =
row.getValue(ProjectConstants.ATT_ANALYSIS_SUPPLIER);
      if (itemName.equals("IPN1") && suppName.equals("suppName1
(suppNumber1)")) {
                row.setValue(ProjectConstants.ATT_ANALYSIS_BEST_RES
PONSE, “Yes”);
      }
}
```

例: MPN に対する最良回答の設定

```
ITable table_aml = (ITable)
row.getValue(ProjectConstants.ATT_ANALYSIS_AML);
Iterator _it = table_aml.iterator();
while(it.hasNext()){
      String itemName =
row.getValue(ProjectConstants.ATT_ANALYSIS_NUMBER);
      String suppName =
row.getValue(ProjectConstants.ATT_ANALYSIS_SUPPLIER);
      String mfrName = row.getValue(ProjectConstants.
ATT_ANALYSIS_MANUFACTURER);
      if (itemName.equals("MPN1") && suppName.equals("suppName1
(suppNumber1)"
            && mfrName.equals("MFR1"))) {
            row.setValue(ProjectConstants.ATT_ANALYSIS_BEST_RESPONS
E, "Yes");
      }
}
```

注意 最良回答に設定できるのは `Yes` のみであるため、`Yes` 以外の値を渡すと、`ExceptionConstants.API_INVALID_PARAM` 例外が発生します。

例: IPN と MPN に対する最良回答の取得

```
String bResp =
row.getValue(ProjectConstants.ATT_ANALYSIS_BEST_RESPONSE).toString
();
```

見積依頼の使用

見積依頼 (RFQ) を使用すると、サプライヤからの価格情報を依頼できます。見積依頼は、アイテムまたは製造元部品の価格と条件を交渉するための手段の役割を果たします。見積依頼はプロジェクトに対して定義されます。したがって、見積依頼を定義するには、最初にプロジェクトを作成し、次にそのプロジェクトに対する必要な見積依頼を作成する必要があります。

単一のプロジェクトで複数の見積依頼を生成できます。見積依頼では、サプライヤとの一対多の関係がサポートされます。つまり、1 つの見積依頼で、サプライヤからの複数の回答が生成される場合があります。

Agile API では、次の見積依頼関連タスクがサポートされています。

- プロジェクトに対する見積依頼の作成
- 見積依頼オブジェクト、テーブルおよび属性のロードと変更
- [ページ 1]、[ユーザー定義 1] および [見積依頼回答] テーブルへのアクセスと変更
- 見積依頼のプロジェクトから [見積依頼回答] テーブルへのアイテムの追加
- [ページ 1]、[ユーザー定義 1] および [見積依頼回答] テーブル内のネスト テーブルの読み取りと更新
- [見積依頼回答] テーブル内のアイテムまたは製造元部品へのサプライヤの割り当て

これらの見積依頼機能をサポートしている API メソッドのリストは、14-8 ページの「サポートされている API メソッド」を参照してください。

注意 PCM SDK 見積依頼オブジェクトには [ユーザー定義 2] がなく、[ユーザー定義 2] の見積依頼定数はサポートされていません。見積依頼で予測した結果が生成されなくなるため、これらの定数は呼び出さないでください。

サポートされている API メソッド

次の API メソッドが見積依頼に対してサポートされています。これらのインターフェースの詳細は、SDK コードが記述されている HTML ファイルを参照してください。このファイルは、Agile ドキュメント Web サイト (http://docs.agile.com) の SDK Sample (Zip ファイル) フォルダにあります。

- `IAgileSession.createObject(Object, Object)`
- `IAgileSession.createObject(int, Object)`
- `IAgileSession.getObject(Object, Object)`
- `IAgileSession.getObject(int, Object)`
- `IRequestForQuote.getName()`
- `IRequestForQuote.assignSupplier(Object)`
- `IRequestForQuote.getTable(Object)`
- `ITable.iterator()`
- `ITable.getTableDescriptor()`
- `ITable.size()`

- ITable.createRow(Object)
- IRow.getValue(Object)
- IRow.setValue(Object, Object)

ソーシング プロジェクトに対する見積依頼の作成

見積依頼は、特定のプロジェクトに対して定義されます。見積依頼の作成では、汎用 IAgileSession メソッドを使用します。

ソーシング プロジェクトと同様に、14-10 ページの「オブジェクト、テーブルおよび属性へのアクセスと変更」を参照してください。IDataObject メソッドを、getObject、getTable、getValue、setValue などの標準呼び出しで使用すると、次のように、オブジェクト、テーブルおよび属性にアクセスし、続いてこれらを変更できます。

- [ページ 1] ([カバー ページ])、[ユーザー定義 1] テーブルの読み取り
- [ページ 1] ([カバー ページ])、[ユーザー定義 1] テーブルの更新

例: オブジェクトの作成

```
IAgileObject createObject(Object objectType, Object params)
throws APIException;
```

見積依頼を作成するには、プロジェクトを開く必要があります。ただし、プロジェクトを開くには、最初に出荷先を設定する必要があります。226 ページの「プロジェクト ステータスへのアクセスおよび変更」のコード例を参照してください。

プロジェクト番号のみを指定して見積依頼を作成することはできません。関連プロジェクトも必須パラメータであるため、指定する必要があります。これは、次の例で示されています。

例: プロジェクトに対する見積依頼の作成

```
IAgileClass rfqClass =
m_admin.getAgileClass(RequestForQuoteConstants.CLASS_RFQ);
      IAutoNumber rfqNumber =
      rfqClass.getAutoNumberSources()[0];
      HashMap map = new HashMap();
map.put(RequestForQuoteConstants.ATT_COVERPAGE_RFQ_NUMBER,
rfqNumber);
map.put(RequestForQuoteConstants.ATT_COVERPAGE_PROJECT_NUMBER,
pnumber);
      IRequestForQuote rfq = (IRequestForQuote)
      m_session.createObject(rfqClass, map);
```

既存の見積依頼のロード

既存の見積依頼をロードするには、IAgileSession.getObject() メソッドを使用するか、またはプロジェクト オブジェクトの [見積依頼] テーブルから選択します。

見積依頼をロードするには、IAgileSession.getObject() メソッドを使用します。見積依頼を一意に識別するために、[カバー ページ | 見積依頼番号] 属性に値を指定します。

例: 見積依頼のロード

```
public IRequestForQuote getRFQ() throws APIException {
  IRequestForQuote rfq =
(IRequestForQuote)m_session.getObject(IRequestForQuote.OBJECT_TYPE
,
   "RFQ01004");
  return rfq;
}
```

プロジェクトの [見積依頼] テーブルからの見積依頼のロード

IAgileSession.getObject() メソッドを使用して見積依頼をロードする方法の他に、プロジェクト オブジェクトの [見積依頼] テーブルから見積依頼を選択することもできます。

例: プロジェクトの [見積依頼] テーブルからの見積依頼のロード

```
ITable table = prj.getTable(ProjectConstants.TABLE_RFQS);
Iterator it = table.iterator();
IRow row1 = (IRow) it.next();
IDataObject obj1 = (IDataObject)
m_session.getObject(IRequestForQuote.OBJECT_TYPE,

row1.getValue(ProjectConstants.ATT_RFQS_RFQ_NUMBER));
```

注意 getReferent() メソッドでは、見積依頼テーブルなど、PCM SDK はサポートされていません。サポートされている見積依頼テーブルのリストを、次の表に示します。

サポートされている見積依頼テーブル

サポートされている見積依頼テーブルとそれぞれの定数は、次の表のとおりです。

テーブル	定数	読み取り/書き込みモード
カバー ページ	TABLE_COVERPAGE	読み取り/書き込み
ユーザー定義 1	TABLE_PAGETWO	読み取り/書き込み
回答	TABLE_RESPONSES	読み取り/書き込み

注意 Agile API では、見積依頼テーブルへの新規行の追加はサポートされていません。ただし、[見積依頼回答] テーブルに新規行を追加することはできます。

見積依頼のオブジェクト、テーブル、ネスト テーブルおよび属性へのアクセスと変更

汎用の IAgileSession メソッドと IDataObject メソッド、および getObject、getValue、setValue などの標準呼び出しを使用して、見積依頼のオブジェクト、テーブルおよび属性にアクセスできます。これらのクラス、テーブルおよび属性に関する情報は、com.agile.api.RequestForQuoteConstants.java ファイルに記載されています。

見積依頼の親テーブルとネスト子テーブルの定数

親の見積依頼テーブルと対応するネスト子テーブルの定数のリストを、次の表に示します。

親テーブルの定数	ネスト子テーブルの定数	読み取り/書き込みモード
TABLE_RESPONSES	ATT_RESPONSES_AML	読み取り/書き込み
TABLE_RESPONSES	ATT_RESPONSES_PRICING	読み取り/書き込み

ソーシング プロジェクトと同様に、ネストされた見積依頼テーブルには、そのセル値をテーブルとして処理することによってアクセスできます。226 ページの「プロジェクトまたは見積依頼のネスト テーブルへのアクセスおよび変更」を参照してください。次の例では、ネスト テーブルを更新しています。

注意 見積依頼のステータスの取得に Project.ATT_RFQ_RFQ_STATE を使用しないでください。これは、SDK には表示されず、見積依頼の行の正しい値がレンダリングされないためです。見積依頼のステータスを取得するには、最初に見積依頼をロードし、次に見積依頼自体からステータスを取得する必要があります。

例: ネストされた見積依頼テーブルの更新例

```
ITable subtab1 =
(ITable)row.getValue(RequestForQuoteConstants.ATT_RESPONSES_PRICIN
G);
IRow pricing1 = (IRow)subtab1.iterator().next();
Integer nest = ProjectConstants.ATT_PRICEDETAILS_MATERIAL_PRICE;
Object nre = pricing1.getValue(nest);
Money tc = new Money(new Integer(100), "USD");
pricing1.setValue(nest, (Object)tc);
```

注意 [見積依頼回答] テーブルのエントリを更新する前に、サプライヤを割り当てる必要があります。

見積依頼回答の処理

PCM SDK では、見積依頼回答、アイテム回答のネスト テーブルおよび子 AML 回答に対して、次の操作がサポートされています。

- 様々なビュー (価格算出ケース、通貨モード) での見積依頼テーブルの読み取り

  これは、汎用 SDK API によってサポートされます。

- 見積依頼へのアイテムの追加
- 回答ラインの追加 (サプライヤの割り当て)

  PCM の見積依頼には、アイテムまたは製造元部品にサプライヤを割り当てるための新規メソッドが用意されています。

  ```
  public void assignSupplier(Object supplierParams)
    throws APIException, RemoteException, Exception;
  ```

  次に示すように、サプライヤを割り当てることができます。

  ```
  IRequestForQuote dObj =
  (IRequestForQuote)m_session.getObject(RequestForQuot
  eConstants.CLASS_RFQ, number);
  ITable tab =
  dObj.getTable(RequestForQuoteConstants.TABLE_RESPONS
  ES);
  ```

```
Map mp = new HashMap();
mp.put(ProjectConstants.ATT_RESPONSES_NUMBER,
"P00007");
mp.put(ProjectConstants.ATT_RESPONSES_SUPPLIER,
"SDKSUP");
dObj.assignSupplier(mp);
```

注意 RequestForQuote.TABLE_RESPONSE を呼び出してサプライヤをアイテム コンポーネントに割り当てると、そのアイテム コンポーネントに対して複数のサプライヤが存在する場合は、テーブルサイズが変更される可能性があります。つまり、アイテムのサプライヤが 1 つの場合は、各アイテムとその対応するサプライヤで、TABLE_RESPONSE テーブル内のそれぞれに固有の独立した行が占有されます。ただし、アイテムのサプライヤが複数の場合、このアイテム コンポーネントに対する行はサプライヤ数に分割されるため、テーブル内の行数が増えて TABLE_RESPONSE が変更されます。したがって、ITERATOR をすぐに再ロードして、TABLE_RESPONSE テーブルの変更を反映する必要があります。これは SDK の欠陥ではなく、SUN J2SE ITERATOR の動作が原因です。

- 回答ラインの更新

  PCM SDK では、汎用 SDK API によって [見積依頼回答] テーブルがサポートされます。見積依頼回答クラスまたはサプライヤ回答はサポートされていません。

  注意 見積依頼回答ラインの回答通貨は、回答通貨属性によって決まります。このため、サーバでは、マテリアル価格の通貨パラメータは無視されます。バイヤーは回答ラインの回答通貨を変更でき、その変更は、回答ライン内のすべての価格属性に適用されます。サプライヤの見積依頼回答通貨は、見積依頼回答プリファレンスに設定され、サプライヤ回答では変更できません。回答ラインがサプライヤに対して公開された場合、回答ラインをロックしないと、バイヤーは回答ラインを変更できません。

第 15 章

Agile PLM オブジェクトの確認通知

扱うトピックは次のとおりです。

ユーザー確認通知について

Agile PLM ビジネス オブジェクト (アイテムや変更など) をロードするとき、そのオブジェクトの確認通知を有効にできます。オブジェクトの確認通知を有効にすると、そのオブジェクトでトリガーとなるイベントが発生した場合に通知を受信します。どのイベントを通知のトリガーとするかを指定できます。確認通知イベントには、ライフサイクルの変更、添付ファイルへの変更、または確認通知に使用可能にしているセルの値の変更などがあります。

送信可能なオブジェクトと送信不可能なオブジェクトの両方の確認通知を有効にできます。Agile API には、`ISubscribable` というインターフェースが用意されており、オブジェクトに対するすべての確認通知を取得して変更できます。ユーザーが確認通知を有効にしているすべてのオブジェクトが、ユーザーの [確認通知] テーブルにリストされます。

確認通知イベント

確認通知イベントは、オブジェクト クラスによって異なります。確認通知を有効にできる完全なイベント セットは、次の表のとおりです。

確認通知イベント	SubscriptionConstants
ステータスの変更 (送信可能なオブジェクトの場合)	`EVENT_STATUS_CHANGE`
ライフサイクル フェーズの変更 (送信不可能なオブジェクトの場合)	`EVENT_LIFECYCLE_CHANGE`
フィールドの変更	`EVENT_FIELD_CHANGE`
ファイルの追加	`EVENT_ADD_FILE`
ファイルの削除	`EVENT_DELETE_FILE`
ファイルのチェックイン	`EVENT_CHECKIN_FILE`
ファイルのチェックアウト	`EVENT_CHECKOUT_FILE`
ファイルのチェックアウトのキャンセル	`EVENT_CANCELCHECKOUT_FILE`

注意 Program Execution オブジェクトに対する追加の確認通知イベントがありますが、Agile API ではサポートされていません。

ほとんどの送信可能なオブジェクトおよび送信不可能なオブジェクトで、前述の表にリストされている 7 つの確認通知がサポートされていますが、例外がいくつかあります。

- ユーザー オブジェクトでは、ライフサイクルの変更確認通知イベントはサポートされていません。
- ファイル フォルダ オブジェクトでは、ファイルの追加およびファイルのチェックアウトのキャンセル確認通知イベントはサポートされていません。

フィールドの変更確認通知イベントは、[確認通知に使用可能] プロパティが [はい] に設定されている属性に関連付けられます。したがって、各クラスとサブクラスに、確認通知可能な属性の異なるセットを設定できます。

確認通知権限

オブジェクトの確認通知を有効にするには、そのクラスに対する確認通知権限が必要です。作成者など、事前定義の多数の Agile PLM 役割には、いくつかのオブジェクト クラスに対する確認通知権限がすでに含まれています。役割と権限を変更するには、Agile PLM システムの管理者にお問い合わせください。

確認通知

確認通知イベントは、電子メールと受信トレイという 2 種類の通知のトリガーとなります。Agile PLM の受信トレイ通知は、ユーザー プリファレンスに関係なく自動的に発生します。電子メールによる通知は、ユーザーの [電子メール通知を受信] プリファレンスが [はい] に設定されている場合にのみ送信されます。

注意 Agile API では、現在通知オブジェクトは公開されていません。ただし、Agile API を使用すると、電子メール通知プリファレンスを設定できます。

確認通知の対象とするオブジェクトの削除

Agile PLM ビジネス オブジェクトは、`IDataObject.delete()` メソッドを使用して削除できます。ただし、オブジェクトは、その確認通知が削除されるまで削除できません。ユーザーは自分自身の確認通知を削除できますが、他のユーザーの確認通知は削除できません。

オブジェクトに対する確認通知の取得

オブジェクトに対する現在の確認通知を取得するには、`ISubscribable.getSubscriptions()` を使用します。このメソッドは、有効と無効の両方の `ISubscription` オブジェクトすべての配列を返します。次の例は、オブジェクトに対する確認通知を取得する方法を示しています。

例: オブジェクトに対する確認通知の取得

```
public void getSubscriptionStatus(IAgileObject obj) throws
APIException {
  ISubscription[] subs = ((ISubscribable)obj).getSubscriptions();
  for (int i = 0; i < subs.length; ++i) {
    if
(subs[i].getId().equals(SubscriptionConstants.EVENT_ADD_FILE)) {
     if (subs[i].isEnabled()) {
       System.out.println("Add File subscription is enabled");
```

```
      }
    }
    else if
(subs[i].getId().equals(SubscriptionConstants.EVENT_CANCELCHECKOUT
_FILE)) {
      if (subs[i].isEnabled()) {
        System.out.println("Cancel Checkout File subscription is
enabled");
      }
    }
    else if
(subs[i].getId().equals(SubscriptionConstants.EVENT_CHECKIN_FILE))
{
      if (subs[i].isEnabled()) {
        System.out.println("Checkin File subscription is enabled");
      }
    }
    else if
(subs[i].getId().equals(SubscriptionConstants.EVENT_CHECKOUT_FILE)
) {
      if (subs[i].isEnabled()) {
        System.out.println("Checkout File subscription is enabled");
      }
    }
    else if
(subs[i].getId().equals(SubscriptionConstants.EVENT_DELETE_FILE))
{
      if (subs[i].isEnabled()) {
        System.out.println("Delete File subscription is enabled");
      }
    }
    else if
(subs[i].getId().equals(SubscriptionConstants.EVENT_FIELD_CHANGE))
{
      if (subs[i].isEnabled()) {
        IAttribute attr = subs[i].getAttribute();
        if (attr != null) {
          String attrName = attr.getFullName();
          System.out.println("Field Change subscription is enabled for
" + attrName);
        }
      }
    }
    else if
(subs[i].getId().equals(SubscriptionConstants.EVENT_LIFECYCLE_CHAN
GE)) {
      if (subs[i].isEnabled())
        System.out.println("Lifecycle Change subscription is
enabled");
    }
    else if
```

```
(subs[i].getId().equals(SubscriptionConstants.EVENT_STATUS_CHANGE)
) {
     if (subs[i].isEnabled())
       System.out.println("Status Change subscription is enabled");
   }
   else
     System.out.println("Unrecognized subscription event: " +
subs[i].getId());
  }
}
```

オブジェクトに対する確認通知の変更

Agile API を使用して、現在のユーザーに対する確認通知のみを変更できます。特定のビジネス オブジェクトに対する確認通知を変更する場合、そのオブジェクトに対する他のユーザーの確認通知は影響を受けません。

オブジェクトに対する確認通知イベントのリストは、サーバで設定され、Agile API では変更できません。ただし、確認通知を有効にするフィールド (属性) は選択できます。管理者権限がある場合は、確認通知に使用可能にするフィールドを定義するためにクラスを変更することもできます。詳細は、次のセクションを参照してください。

確認通知を処理するには、次の ISubscription メソッドを使用します。

- enable(boolean) - 確認通知を有効または無効にします。
- getAttribute() - 確認通知に関連付けられている IAttribute オブジェクトを返します。フィールドの変更確認通知にのみ関連属性があります。
- isEnabled() - 確認通知が有効な場合は true、無効な場合は false を返します。
- getId() - 確認通知 ID を返します。この ID は、SubscriptionConstants のいずれかと同じです。

ISubscriptionは、値オブジェクト インターフェースです。したがって、確認通知に変更を加えた場合 (例: 確認通知を有効にする)、Agile PLM システムでは、ISubscribable.modifySubscriptions() を呼び出すまで変更されません。

次の例は、ライフサイクルの変更およびフィールドの変更確認通知イベントを有効にし、[ユーザー定義 1] の 2 つのフィールドの確認通知を有効にする方法を示しています。他の確認通知イベントはすべて無効です。

例: オブジェクトに対する確認通知の有効化および無効化

```
public void setSubscriptions(IAgileObject obj) throws APIException {
  ISubscription[] subs = ((ISubscribable)obj).getSubscriptions();
  for (int i = 0; i < subs.length; ++i) {
    // Enable the Status Change subscription event
    if
(subs[i].getId().equals(SubscriptionConstants.EVENT_STATUS_CHANGE)
) {
     subs[i].enable(true);
   }
```

```
    // Enable the Field Change subscription event for Page Two.Text01
and Page Two.List01
    else if
(subs[i].getId().equals(SubscriptionConstants.EVENT_FIELD_CHANGE))
{
     if (subs[i].getAttribute() != null)
       System.out.println(subs[i].getAttribute().getFullName() + ":
" +
       subs[i].getAttribute().getId());
     if ((subs[i].getAttribute() != null) &&

((subs[i].getAttribute().getId().equals(CommonConstants.ATT_PAGE_T
WO_LIST01)) ||

(subs[i].getAttribute().getId().equals(CommonConstants.ATT_PAGE_TW
O_TEXT01))) )
        subs[i].enable(true);
     else
        subs[i].enable(false);
    }
   // Disable all other subscription events
   else
     subs[i].enable(false);
  }
  ((ISubscribable)obj).modifySubscriptions(subs);
}
```

確認通知での属性の使用可能化

確認通知可能な属性は、Agile PLM クラスによって異なります。通常、ほとんどの [ページ 1] ([タイトル ページ]、[カバー ページ] および [一般情報]) の属性が確認通知可能であるため、確認通知に使用可能にできます。ATT_PAGE_TWO_CREATE_USER を除く [ユーザー定義 1] のすべての属性、および [ユーザー定義 2] のすべての属性も確認通知可能です。

属性の [確認通知に使用可能] プロパティが [はい] に設定されている場合、ユーザーは属性の確認通知を有効にできます。オブジェクトに対して ISubscribable.getSubscriptions() を呼び出した場合、返される ISubscription[] 配列には、各確認通知イベントに対する ISubscription オブジェクトが含まれます。1 つのフィールドの変更イベント (定数は SubscriptionConstants.EVENT_FIELD_CHANGE) のみの場合でも、確認通知の対象とする各属性は、確認通知のトリガーとなる別々のイベントとして処理されます。Agile PLM システムの設定方法によっては、特定のオブジェクトに対して、確認通知に使用可能な属性が多数存在する場合があります。

属性の表示が有効になっていない場合、その [確認通知に使用可能] プロパティが [はい] に設定されている場合でも、その属性は確認通知可能ではありません。したがって、[確認通知に使用可能] プロパティを [はい] に設定する前に、[表示] プロパティも [はい] に設定されていることを確認してください。次の例は、ECO に対する [ユーザー定義 1] のすべての属性を確認通知で使用可能にする方法を示しています。

例: 確認通知での [ユーザー定義 1] 属性の使用可能化

```
try {
  // Get the ECO subclass
  IAgileClass classECO = m_admin.getAgileClass("ECO");
  // Get Page Two attributes
  IAttribute[] attr =
classECO.getTableAttributes(ChangeConstants.TABLE_PAGETWO);

  // Make all visible Page Two attributes subscribable
  for (int i = 0; i < attr.length; ++i) {
    IProperty prop = null;
    IAgileList listValues = null;
    String strVal = "";

    // Check if the attribute is visible
    prop = attr[i].getProperty(PropertyConstants.PROP_VISIBLE);
    listValues = (IAgileList)prop.getValue();
    strVal = listValues.toString();

    // If the attribute is visible, make it subscribable
    if (strVal.equals("Yes")) {
      prop =
attr[i].getProperty(PropertyConstants.PROP_AVAILABLE_FOR_SUBSCRIBE
);
      if (prop != null) {
        listValues = prop.getAvailableValues();
        listValues.setSelection(new Object[] { "Yes" });
        prop.setValue(listValues);
      }
    }
  }
} catch (APIException ex) {
  System.out.println(ex);
}
```

親属性と子属性

いくつかの読み取り専用属性には、親属性と子の関係にあるものがあります。子属性には、親属性の値が反映されます。したがって、親属性は確認通知に使用可能ですが、子属性は使用可能ではありません。子属性の例には、[BOM.アイテム リスト02] や [BOM.アイテム テキスト01] などの [BOM] テーブルの属性があります。

[確認通知] テーブルの使用

ユーザーの [確認通知] テーブルには、ユーザーが有効にしているすべての確認通知がリストされます。[確認通知] テーブルには、限定された編集機能があります。たとえば、テーブルに新しい行を追加することはできません。Agile API を使用して確認通知を追加する唯一の方法は、データオブジェクトに対して `ISubscribable.modifySubscriptions()` を呼び出すことです。ただし、テーブルから確認通知を削除することはできます。

次の例は、現在のユーザーの [確認通知] テーブルを取得する方法を示しています。また、番号が 1000-02 の部品に対する確認通知を削除する方法も示しています。

例: 確認通知の削除

```
try {
  // Get the current user
  IUser user = m_session.getCurrentUser();
  // Get the Subscription table
  ITable tblSubscriptions =
user.getTable(UserConstants.TABLE_SUBSCRIPTION);
  Iterator i = tblSubscriptions.iterator();

  // Stop subscribing to part 1000-02
  while (i.hasNext()) {
    IRow row = (IRow)i.next();
    String n =
(String)row.getValue(UserConstants.ATT_SUBSCRIPTION_NUMBER);
    if (n.equals("1000-02")) {
      tblSubscriptions.removeRow(row);
      break;
    }
  }
} catch (APIException ex) {
  System.out.println(ex);
}
```

[確認通知] テーブルから個々の行を削除する以外に、`Collection.clear()` メソッドを使用してテーブルをクリアすることもできます。

例: [確認通知] テーブルのクリア

```
public void clearSubscriptionTable(IUser user) throws APIException {
  // Get the Subscription table
  ITable tblSubscriptions =
user.getTable(UserConstants.TABLE_SUBSCRIPTION);

  // Clear the table
  tblSubscriptions.clear();
}
```

[確認通知] テーブルには、確認通知を有効にしているイベントがオブジェクトごとにはリストされません。この情報を検索するには、各参照オブジェクトを開く必要があります。次の例は、**`ITable.getReferentIterator()`** を使用して、テーブル内の参照オブジェクトで処理を繰り返す方法を示しています。

例: [確認通知] テーブルで参照されるオブジェクトの取得

```
try {
  // Get the current user
  IUser user = m_session.getCurrentUser();
  // Get the Subscription table
  ITable tblSubscriptions =
user.getTable(UserConstants.TABLE_SUBSCRIPTION);
  Iterator i = tblSubscriptions.getReferentIterator();

  // Get each object referenced in the table
  while (i.hasNext()) {
    IAgileObject obj = (IAgileObject)i.next();
    if (obj instanceof ISubscribable) {
      ISubscription[] subscriptions =
((ISubscribable)obj).getSubscriptions();
     for (int j = 0; j < subscriptions.length; j++) {
       ISubscription subscription = subscriptions[j];
       System.out.println(subscription.getName());
       // Add code here to handle each subscription
     }
     System.out.println(obj.getName());
    }
  }
} catch (APIException ex) {
  System.out.println(ex);
}
```

第 16 章

製品の規制および適合性の管理

扱うトピックは次のとおりです。

Agile Product Governance & Compliance について

Agile Product Governance & Compliance (PG&C) は、製品の定義や、規制されたサブスタンスのインポート、エクスポートおよび廃棄に影響を与える多くの環境規制や企業の環境ポリシーに対処します。Agile PG&C は、OEM メーカーが自社の製品で使用する規制サブスタンスの量を検証し、それらのサブスタンスが含まれる電子機器を責任を持って廃棄、リサイクルまたは再利用できるように設計されています。

Agile PG&C によって、企業は環境規制にコスト効率よく準拠できます。また、Agile PG&C を使用すると、部品の適合性データをサプライヤから取得できます。このデータによって、企業は次のことを実現できます。

- サブスタンス制限の適合
- 規制のレポート要件の達成
- リサイクル可能製品の設計
- 適合性コストの最小化
- 将来の製品における不適合の除去

Agile PG&C は適合性管理者とサプライヤとの間のコミュニケーションを支援するツールです。適合性管理者は、企業の製品が政府規制と企業のポリシーに従っていることを確認します。サプライヤ側では、マテリアル プロバイダがマテリアル デクラレーションを完成してサインオフし、提供するコンポーネントやサブアセンブリに含まれる有害化学物質の種類を公表します。

Agile PG&C 機能の詳細は、別途『Product Governance & Compliance ユーザー・ガイド』を参照してください。

Agile PG&C のインターフェースとクラス

次の表に、Agile PG&C に関連するインターフェースとクラスを示します。

オブジェクト	インターフェース	定数クラス
デクラレーション	IDeclaration	DeclarationConstants
含有基準	ISpecification	SpecficationConstants
サブスタンス	ISubstance	SubstanceConstants
部品グループ	ICommodity	PartGroupConstants

アイテム、製造元部品および部品グループは、Agile PG&C にも関連するオブジェクトです。これらのオブジェクトには [含有基準]、[組成] および [サブスタンス] テーブルがあり、デクラレーションをリリースするとデータが挿入されます。製造元部品の場合は、デクラレーションを提出せずに [組成] および [サブスタンス] テーブルを直接編集できます。

注意 このマニュアルで使用される「部品グループ」と「部品分類」という用語は同じ意味で、ICommodity オブジェクトを指します。ICommodity は、[部品分類] および [部品ファミリ] サブクラスを含む [部品グループ] 基本クラスを表します。

Agile PG&C オブジェクトを使用するために、ITable、IDataObject および ICell など、他の共通の Agile API インターフェースも使用されます。

Agile PG&C の役割

Agile PLM には、Agile PG&C ユーザー用に設計されたデフォルトの役割が 2 つあります。

- 適合性管理者 - デクラレーション、サブスタンス、含有基準などの Agile PG&C オブジェクトの作成および管理に必要な権限を提供し、Agile PG&C レポートを実行します。適合性管理者はマテリアル デクラレーションをサプライヤに送信する役割を果たします。
- (限定) マテリアル プロバイダ - デクラレーションの作成と変更、およびその他すべてのタイプの Agile PG&C オブジェクトの読み取りに必要な権限を提供します。この役割は、通常、Agile PLM システムに制限付きのアクセスを持つサプライヤ ユーザーに割り当てられます。マテリアル プロバイダは、マテリアル デクラレーションを完成してサインオフする役割を果たします。

この章で説明する Agile PG&C API を使用するには、[適合性管理者] または [(限定) マテリアル プロバイダ] の役割が割り当てられたユーザーでログインしてください。Agile PLM 役割の詳細は、『Agile PLM 管理者ガイド』を参照してください。

デクラレーション、含有基準およびサブスタンスの作成

このセクションでは、Agile PG&C の各クラスを作成する方法を示します。

デクラレーションの作成

デクラレーション オブジェクトは Agile PG&C のメイン レコードです。アイテム、製造元部品および部品グループで使用されるサブスタンスおよびマテリアルを追跡します。デクラレーションをリリースすると、収集した情報はプロダクト レコードに公表され、デクラレーションによってリストされたアイテム、製造元部品、部品グループ内に含まれる組成データが更新されます。

Agile PLM に付属しているデクラレーション サブクラスは次の 7 つです。

- 均質材のデクラレーション - マテリアル レベルの含有基準を使用する均質材組成デクラレーション。
- IPC 1752-1 デクラレーション - IPC 標準に適合し、1 部品レベルの含有基準のみを使用する電子製品のマテリアル組成デクラレーション。
- IPC 1752-2 デクラレーション - IPC 標準に適合し、1 マテリアル レベルの含有基準のみを使用する電子製品の均質材組成デクラレーション。
- JGPSSI デクラレーション - 日本グリーン調達 (JGP) 標準に準拠し、部品レベルの含有基準を使用するマテリアル組成デクラレーション。
- 部品のデクラレーション - 部品レベルまたはマテリアル レベルの含有基準を使用するマテリアル組成デクラレーション。
- サブスタンスのデクラレーション - 部品レベルの含有基準内にある各サブスタンスのマテリアル組成デクラレーション。
- 適合のサプライヤ デクラレーション - サプライヤの適合性を顧客と政府機関の含有基準で評価するアンケート。調査は、一般の会社レベルで適合性に対処します。CSR タイプのデクラレーションで使用できます。

デクラレーションを作成する手順は、すべてのデクラレーション サブクラスについて同じです。デクラレーション サブクラスを指定し、[カバー ページ.名前] および [カバー ページ.サプライヤ] 属性に値を指定する必要があります。他のデクラレーション属性はオプションです。

デフォルトで、[カバー ページ.名前] フィールドは (「Material Declaration」の) 接頭辞「MD」を持つ [自動採番] フォーマットを使用します。[自動採番] フォーマットは必須ではありませんが、検索を簡単にするためにすべてのデクラレーションに対して同じ接頭辞を使用すると合理的です。

注意 [カバー ページ.名前] フィールドに大文字と小文字のいずれで入力する必要があるかは、フィールドに対して選択された文字セットに基づきます。

[(限定) マテリアル プロバイダ] の役割を持つサプライヤ ユーザーは、デクラレーションを作成することもできます。ただし、オブジェクトの作成に必要な属性は [カバー ページ.名前] のみです。[カバー ページ.サプライヤ] 属性にはユーザーのサプライヤ組織が自動的に入力されます。

次の例は、JGPSSI デクラレーションを作成する方法を示しています。

例: JGPSSI デクラレーションの作成

```
public void CreateJGPSSIDeclaration(String num, ISupplier supplier)
throws Exception {
   // Create a Map object to store parameters
   Map params = new HashMap();

   // Initialize the params object
   params.put(DeclarationConstants.ATT_COVER_PAGE_NAME, num);
   params.put(DeclarationConstants.ATT_COVER_PAGE_SUPPLIER,
supplier);

   // Get the JGPSSI Declaration subclass
   IAgileClass declClass =
m_session.getAdminInstance().getAgileClass(

DeclarationConstants.CLASS_JGPSSI_DECLARATION);
   // Create a new JGPSSI declaration
   IDeclaration object =
(IDeclaration)m_session.createObject(declClass, params);
}
```

含有基準の作成

含有基準は、製品が適合するまたは超える条件を示すために使用されます。通常は、製品に含まれる規制サブスタンスの量を制限するために使用されます。含有基準は、企業または業界が発行する内部文書の場合がありますが、一般的には管理機関が発行する規制です。次に、政府規制の例を示します。

- 欧州連合 (EU) によって発行された、電子電気機器の指針における特定有害化学物質の使用に関する規制 (RoHS)
- EU によって発行された、電子電気機器廃棄 (WEEE) の指針
- 米食品医薬品局 (FDA) によって発行された、アレルゲン表示および消費者保護法 (FALCPA)

含有基準は、サブスタンスのリスト、各サブスタンスの PPM (100 万分の 1 単位) しきい値、および特定のサブスタンスが制限されるかどうかを定義します。適合性管理者は、含有基準を使用して適切なサブスタンスが含まれたマテリアル デクラレーションを事前に作成し、適合性を確認できます。

含有基準の作成時に指定する必要のある必須属性は [一般情報.名前] のみです。この名前は一意である必要があります。名前は大文字と小文字を区別しません。これは、「ROHS」が「Rohs」と同じように扱われることを意味します。

[一般情報.検証タイプ] 属性は、含有基準が [部品レベル] (デフォルト) か [均質材レベル] かを判断し、含有基準で使用可能なデクラレーションのタイプに影響を与えるため、重要な属性です。別のオプションの属性は [一般情報.ライフサイクル フェーズ] です。含有基準を作成する場合、デフォルトのライフサイクル フェーズは [アクティブ] です。含有基準を破棄するには、そのライフサイクル フェーズの属性の値を [破棄] に変更します。

例: 含有基準の作成

```
public void createSpecification(String name) throws Exception {
  ISpecification spec = (ISpecification)

m_session.createObject(SpecificationConstants.CLASS_SPECIFICATION,
name);

}
```

サブスタンスの作成

Agile PLM に付属しているサブスタンス サブクラスは次の 4 つです。

- サブパート - コンポーネントの製造元部品のサブユニット。サブパートの [組成] テーブルには、その他のサブパート、マテリアル、サブスタンス グループ、サブスタンスを含めることができます。
- マテリアル - 複数のサブスタンスで構成される複合物。マテリアルの [組成] テーブルには、サブスタンス グループまたはサブスタンスを含めることができます。
- サブスタンス グループ - サブスタンスのグループ。サブスタンス グループの [組成] テーブルには、サブスタンスのみを含めることができます。
- サブスタンス - 鉛、クロミウム、カドミウムなどの単一の要素。サブスタンスに [組成] テーブルはありません。

これらのサブスタンス サブクラスは、[組成] テーブルに表示可能なオブジェクトの階層 (サブスタンス構成表とも呼ばれます) で構成されます。

サブパートの作成

サブパート オブジェクトは Agile PLM で追跡されるコンポーネントのサブユニットです。サブパートは部品番号のない部品で、製造元部品または組成内の部品の部品構成表 (BOM) を作成するために使用されます。

例: サブパートの作成

```
public void createSubpart(String num) throws Exception {
   // Create a Map object to store parameters
   Map params = new HashMap();

   // Initialize the map object
   params.put(SubstanceConstants.ATT_GENERAL_INFO_NAME, num);

   // Get the Subpart subclass
   IAgileClass subsClass = m_session.getAdminInstance().

getAgileClass(SubstanceConstants.CLASS_SUBPART);
   // Create a new Subpart
   ISubstance sub = (ISubstance)m_session.createObject(class,
params);
}
```

サブスタンス グループの作成

サブスタンス グループ オブジェクトは、共通のベース サブスタンスを含む複数のサブスタンスのグループで、Agile PLM で追跡されます。グループ内のすべてのサブスタンスには、そのグループのベース サブスタンスの重量を換算するために使用する換算係数があります。

例: サブスタンス グループの作成

```
public void createSubstanceGroup(String num, ISubstance sub) throws
Exception {
   // Create a Map object to store parameters
   Map params = new HashMap();

   // Initialize the map object
   params.put(SubstanceConstants.ATT_GENERAL_INFO_NAME, num);
   params.put(SubstanceConstants.ATT_GENERAL_INFO_BASE_SUBSTANCE,
sub);
   // Get the Substance Group subclass
   IAgileClass subsClass = m_session.getAdminInstance().

getAgileClass(SubstanceConstants.CLASS_SUBSTANCE_GROUP);
   // Create a new Substance Group
   ISubstance sub = (ISubstance)m_session.createObject(class,
params);
}
```

マテリアルの作成

マテリアル オブジェクトを作成する場合、指定する必要のある属性は、サブスタンス番号に相当する [一般情報.名前] 属性のみです。マテリアル オブジェクトの作成後は、その [組成] テーブルにサブスタンスを追加できます。

例: マテリアル オブジェクトの作成

```
public void createMaterial(String num, ISubstance[] substances) throws
Exception {
  // Create a Map object to store parameters
  Map params = new HashMap();

  // Initialize the params object
  params.put(SubstanceConstants.ATT_GENERAL_INFO_NAME, num);

  // Create a new material
  ISubstance material = (ISubstance)m_session.createObject(
     SubstanceConstants.CLASS_MATERIAL, params );
  // Get the Composition table
  ITable composition =
material.getTable(SubstanceConstants.TABLE_COMPOSITION);

  // Add substances to the Composition table
  for (int i = 0; i < substances.length; ++i) {
    IRow row = composition.createRow(substances[i]);
  }
}
```

サブスタンスの作成

マテリアル オブジェクトと同様に、サブスタンスを作成する際に指定する必要のある属性は、サブスタンス番号に相当する [一般情報.名前] 属性のみです。[一般情報.CAS 番号] などの他のオプション属性を指定することもできます。

例: サブスタンスの作成

```
public void createSubstance(String num, String casNumber) throws
Exception {
   // Create a Map object to store parameters
   Map params = new HashMap();

   // Initialize the params object
   params.put(SubstanceConstants.ATT_GENERAL_INFO_NAME, num);
   params.put(SubstanceConstants.ATT_GENERAL_INFO_CAS_NUMBER,
casNumber);
   // Get the Substance subclass
   IAgileClass subsClass = m_session.getAdminInstance().

getAgileClass(SubstanceConstants.CLASS_SUBSTANCE);
   // Create a new substance
   ISubstance substance =
(ISubstance)m_session.createObject(subsClass, params);
}
```

デクラレーションへのアイテム、製造元部品および部品グループの追加

各デクラレーションには、アイテム、製造元部品および部品グループごとに個別のテーブルがあります。また、各デクラレーションには、関連する組成テーブルである [アイテム組成]、[製造元部品の組成] および [部品グループの組成] があります。

アイテムをデクラレーションの [アイテム] テーブルに追加する際は、そのアイテムの最新のリリース済みリビジョンが使用されます。アイテムにリリース済みリビジョンがない場合は、初版リビジョンが使用されます。

次の例は、アイテム、製造元部品および部品グループをデクラレーションに追加する方法を示しています。

例: デクラレーションへのアイテム、製造元部品および部品グループの追加

```
public void addDecObjects(IDeclaration dec) throws APIException {
  try {
    HashMap params = new HashMap();
     //Add an Item to the Items table
     ITable tblItems =
dec.getTable(DeclarationConstants.TABLE_ITEMS);
     params.clear();
     params.put(DeclarationConstants.ATT_ITEMS_ITEM_NUMBER,
"1000-02");
     IRow rowItems = tblItems.createRow(params);
```

```
    //Add a Manufacturer Part to the Manufacturer Parts table
    ITable tblMfrParts =
dec.getTable(DeclarationConstants.TABLE_MANUFACTURERPARTS);
    params.clear();

params.put(DeclarationConstants.ATT_MANUFACTURER_PARTS_MFR_PART_NU
MBER, "Widget103");

params.put(DeclarationConstants.ATT_MANUFACTURER_PARTS_MFR_NAME,
"ACME");
    IRow rowMfrParts = tblMfrParts.createRow(params);
    //Add a Commodity to the Part Groups table
    ITable tblPartGroups =
dec.getTable(DeclarationConstants.TABLE_PARTGROUPS);
    params.clear();
    params.put(DeclarationConstants.ATT_PART_GROUPS_NAME, "RES");
    IRow rowPartGroups = tblPartGroups.createRow(params);

  } catch (APIException ex) {
     System.out.println(ex);
  }
}
```

デクラレーションへのサブスタンスの追加

サブスタンスは、デクラレーション内に含まれる [アイテム組成]、[製造元部品の組成] および [部品グループの組成] の各テーブルに追加できます。サブスタンスをアイテム、製造元部品および部品グループに公表するには、デクラレーションをリリースします。デクラレーションがリリースされると、サブスタンスが対応するアイテム、製造元部品および部品グループの [サブスタンス] テーブルに自動的に追加されます。

デクラレーションの組成テーブルはマッピング テーブルであり、部品をそれぞれのサブスタンスにマップします。親オブジェクトにサブスタンスがない場合、組成テーブルには行が設定されません。

デクラレーションの組成テーブルに行を追加するには、`ITable.createRow()` メソッドを使用します。組成テーブルはマッピング テーブルであるため、`ISubstance` オブジェクトを渡して行を作成することはできません。かわりに、属性と値のペアを含む `Map` オブジェクトを指定します。

重要 アイテムと部品グループの [サブスタンス] および [組成] テーブルは読み取り専用です。これらのテーブルにデータが挿入されるのは、デクラレーションがリリースされた場合のみです。

サブスタンスをデクラレーションの [組成] テーブルのいずれかに追加する手順は、次のとおりです。

1. アイテム、製造元部品または部品グループをデクラレーションの [アイテム]、[製造元部品] または [部品グループ] の各テーブルに追加します。
2. サブスタンス行を、[アイテム]、[製造元部品] または [部品グループ] テーブルの親行を参照する [組成] テーブルに追加します。仮想属性 `DeclarationConstants.ATT_PARENT_ROW` を使用して親行を指定します。サブスタンスを追加するときに、サブスタンス名とサブスタンス タイプを指定します。

> **重要** Agile SDK の場合、デクラレーションの [組成] テーブルには、[アイテム]、[製造元部品] および [部品グループ] テーブルに含まれるすべての親オブジェクトがリストされます。Agile Web クライアントでは、異なる方法で [組成] テーブルが表示されます。親オブジェクトごとに個別の [組成] テーブルが表示されます。

[組成] テーブルに行を作成するときに、属性と値のペアを含む `Map` オブジェクトを渡します。次の表に、`Map` オブジェクトに含める必要がある属性を示します。

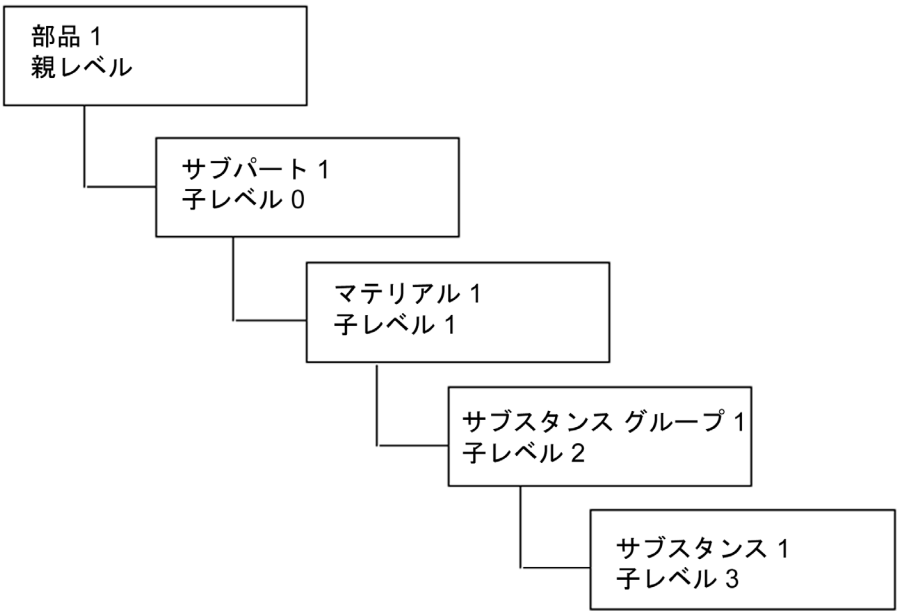
[組成] テーブル	必須属性	DeclarationConstants
アイテム組成	アイテム行 サブスタンス名	`ATT_PARENT_ROW` `ATT_ITEM_COMPOSITION_SUBSTANCE_NAME`
製造元部品の組成	製造元部品行 サブスタンス名	`ATT_PARENT_ROW` ATT_MANUFACTURER_PART_COMPOSITION_SUBSTANCE_NAME
部品グループの組成	部品グループ行 サブスタンス名	`ATT_PARENT_ROW` `ATT_PART_GROUP_COMPOSITION_SUBSTANCE_NAME`

BOS (サブスタンス構成表) の構造

サブスタンスをデクラレーションの [組成] テーブルに追加するときに、それらを複数レベルで構成できます。使用できるレベルの数はデクラレーションのタイプによって異なります。

- 均質材のデクラレーション - サブパート、マテリアル、サブスタンス グループおよびサブスタンスが含まれた複数レベルのサブスタンス構成表を作成できます。組成には、サブパートまたはマテリアルを直接の子として含める必要があります。また、サブスタンスとサブスタンス グループを含めることもできますが、それらはサブパートまたはマテリアルに関連付ける必要があります。
- サブスタンス デクラレーション/JGPSSI デクラレーション - ユーザーは、サブスタンスまたはサブスタンス グループを [組成] テーブルに追加できます。
- 部品のデクラレーション/適合のサプライヤ デクラレーション - これらのデクラレーションには [組成] テーブルはありません。

次の図は、4 つの子レベルが含まれるサブスタンス構成表 (組成) の階層を示しています。



サブスタンスの追加に関するルール

サブスタンスを [組成] テーブルに追加する場合は、次のルールに従います。

- 親オブジェクトを追加してからそれぞれの子を追加する必要があります。
- サブパートには、他のサブパート、マテリアル、サブスタンス グループまたはサブスタンスを子として追加できます。
  - サブパートには、サブパート、マテリアル、サブスタンス グループおよびサブスタンスをすべて同じレベルで含めることはできません。
  - サブパートには、他のサブパートとマテリアルを同じレベルで含めることができます。
  - サブパートには、サブスタンス グループとサブスタンスを同じレベルで含めることができます。
- マテリアルには、サブスタンス グループまたはサブスタンスを子として追加できます。
- サブスタンス グループには、サブスタンスのみを子として追加できます。

存在しないサブパートとマテリアルの追加

サブスタンスをデクラレーションの [組成] テーブルに追加するときに、Agile PLM システムに存在しないダミーのサブパートとマテリアルを指定できます。このようなサブパートとマテリアルは [組成] テーブル内でのみ表示されます。ダミーのサブパートとマテリアルを [組成] テーブルに追加する場合は、次の [サブスタンス タイプ] 属性を指定する必要があります。

- ATT_ITEM_COMPOSITION_SUBSTANCE_TYPE
- ATT_MANUFACTURER_PART_COMPOSITION_SUBSTANCE_TYPE
- ATT_PART_GROUP_COMPOSITION_SUBSTANCE_TYPE

次の例は、ダミーのサブパートまたはマテリアルを [製造元部品の組成] テーブルに追加する方法を示しています。Substance Type フィールドはリスト フィールドであるため、渡される値は IAgileList です。

例: [製造元部品の組成] テーブルへのダミーのサブパートまたはマテリアルの追加

```
public IRow addDummy(IDeclaration dec, IRow parentRow, String dummyName,
IAgileList subtype)
  throws APIException {
  try {

    HashMap params = new HashMap();
    ITable tblMfrPartComp = dec.getTable(

DeclarationConstants.TABLE_MANUFACTURERPARTCOMPOSITION);
    params.put(DeclarationConstants.ATT_PARENT_ROW, parentRow);

params.put(DeclarationConstants.ATT_MANUFACTURER_PART_COMPOSITION_
SUBSTANCE_NAME,
     dummyName);

params.put(DeclarationConstants.ATT_MANUFACTURER_PART_COMPOSITION_
SUBSTANCE_TYPE,
     subtype);
    IRow dummyRow = tblMfrPartComp.createRow(params);
    return dummyRow;

  } catch (APIException ex) {
     System.out.println(ex);
  }
}
```

サブスタンスを追加する例

次の例は、サブスタンスを追加する方法を示しています。

- 均質材のデクラレーションの [製造元部品の組成] テーブルへのサブスタンスの追加
- サブスタンス デクラレーションの [製造元部品の組成] テーブルへのサブスタンスの追加

均質材のデクラレーションの [製造元部品の組成] テーブルへのサブスタンスの追加

次の例は、サブスタンスを均質材のデクラレーションの [製造元部品の組成] テーブルに追加する方法を示しています。このテーブルには、サブパート、マテリアル、サブスタンス グループおよびサブスタンスの 4 つのレベルがあります。サブスタンス行をテーブルに追加するときに、入力パラメータとしてサブスタンス名 (String) のかわりにサブスタンス オブジェクト (ISubstance) を渡すことをお薦めします。

例: [製造元部品の組成] テーブルへの均質材レベルのサブスタンスの追加

```
public void addHomogeneousMaterialComp(IAgileSession m_session)
throws APIException {
  try {
    HashMap params = new HashMap();
    // Create a Declaration
    String num = "MDTEST001";
    ISupplier supplier =
(ISupplier)m_session.getObject(ISupplier.OBJECT_TYPE,
                    "DISTRIBUTOR00007");
    params.put(DeclarationConstants.ATT_COVER_PAGE_NAME, num);
    params.put(DeclarationConstants.ATT_COVER_PAGE_SUPPLIER,
supplier);
    IAgileClass declClass =
m_session.getAdminInstance().getAgileClass(

DeclarationConstants.CLASS_HOMOGENEOUS_MATERIAL_DECLARATION);
    IDeclaration dec = (IDeclaration)m_session.createObject(declClass,
params);
    // Add a Homogeneous Material Level spec to the Specifications table
    ITable tblSpec =
dec.getTable(DeclarationConstants.TABLE_SPECIFICATION);
    params.clear();
    ISpecification spec =
(ISpecification)m_session.getObject(ISpecification.OBJECT_TYPE,
                     "Lead Homogeneneous Material Level");
    IRow rowSpec = tblSpec.createRow(spec);
    // Add a Manufacturer Part to the Manufacturer Parts table
    ITable tblMfrParts =
dec.getTable(DeclarationConstants.TABLE_MANUFACTURERPARTS);
    params.clear();

params.put(DeclarationConstants.ATT_MANUFACTURER_PARTS_MFR_PART_NU
MBER, "Widget103");

params.put(DeclarationConstants.ATT_MANUFACTURER_PARTS_MFR_NAME,
"ACME");
    IManufacturerPart mfrPart = (IManufacturerPart) m_session.

getObject(IManufacturerPart.OBJECT_TYPE, params);
    IRow rowMfrParts = tblMfrParts.createRow(mfrPart);
    // Add a subpart to the Composition table
    ITable tblMfrPartComp = dec.getTable(

DeclarationConstants.TABLE_MANUFACTURERPARTCOMPOSITION);
    ISubstance subpart = (ISubstance)m_session.
```

```
getObject(SubstanceConstants.CLASS_SUBPART, "Steel Casing");
    params.clear();
    params.put(DeclarationConstants.ATT_PARENT_ROW, rowMfrParts);

params.put(DeclarationConstants.ATT_MANUFACTURER_PART_COMPOSITION_
SUBSTANCE_NAME,
      subpart);
    IRow rowSubpart = tblMfrPartComp.createRow(params);
    // Add a material
    ISubstance material =
(ISubstance)m_session.getObject(SubstanceConstants.CLASS_MATERIAL,
                      "Steel");
    params.clear();
    params.put(DeclarationConstants.ATT_PARENT_ROW, rowSubpart);

params.put(DeclarationConstants.ATT_MANUFACTURER_PART_COMPOSITION_
SUBSTANCE_NAME, material);
    IRow rowMaterial = tblMfrPartComp.createRow(params);
    // Add a substance group
    ISubstance sg =
(ISubstance)m_session.getObject(SubstanceConstants.CLASS_SUBSTANCE
_GROUP,
      "Lead Componds");
    params.clear();
    params.put(DeclarationConstants.ATT_PARENT_ROW, rowMaterial);

params.put(DeclarationConstants.ATT_MANUFACTURER_PART_COMPOSITION_
SUBSTANCE_NAME, sg);
    IRow rowSubGroup = tblMfrPartComp.createRow(params);
    // Add a substance
    ISubstance sub =
(ISubstance)m_session.getObject(SubstanceConstants.CLASS_SUBSTANCE
,
      "Lead");
    params.clear();
    params.put(DeclarationConstants.ATT_PARENT_ROW, rowSubGroup);

params.put(DeclarationConstants.ATT_MANUFACTURER_PART_COMPOSITION_
SUBSTANCE_NAME, sub);
    IRow rowSubs = tblMfrPartComp.createRow(params);
  } catch (APIException ex) {
      System.out.println(ex);
  }

}
```

サブスタンス デクラレーションの [製造元部品の組成] テーブルへのサブスタンスの追加

次の例は、サブスタンスをサブスタンス デクラレーションの [製造元部品の組成] テーブルに追加する方法を示しています。このテーブルには、サブスタンス グループとサブスタンスの 2 つのレベルがあります。

例: [製造元部品の組成] テーブルへの部品レベルのサブスタンスの追加

```
public void addSubstanceComp(IAgileSession m_session) throws
APIException {
  try {
    HashMap params = new HashMap();
    //Create a Declaration
    String num = "MDTEST001";
    ISupplier supplier =
(ISupplier)m_session.getObject(ISupplier.OBJECT_TYPE,
                     "DISTRIBUTOR00007");
    params.put(DeclarationConstants.ATT_COVER_PAGE_NAME, num);

params.put(DeclarationConstants.ATT_COVER_PAGE_SUPPLIER,
supplier);
    IAgileClass declClass =
m_session.getAdminInstance().getAgileClass(

DeclarationConstants.CLASS_SUBSTANCE_DECLARATION);
    IDeclaration dec = (IDeclaration)m_session.createObject(declClass,
params);
    //Add a Specification to the Specifications table
    ITable tblSpec =
dec.getTable(DeclarationConstants.TABLE_SPECIFICATION);
    params.clear();
    // Part Level
    ISpecification spec =
(ISpecification)m_session.getObject(ISpecification.OBJECT_TYPE,
                      "Lead Part Level");
    IRow rowSpec = tblSpec.createRow(spec);
    //Add a Manufacturer Part to the Manufacturer Parts table
    ITable tblMfrParts =
dec.getTable(DeclarationConstants.TABLE_MANUFACTURERPARTS);
    params.clear();

params.put(DeclarationConstants.ATT_MANUFACTURER_PARTS_MFR_PART_NU
MBER, "Widget103");

params.put(DeclarationConstants.ATT_MANUFACTURER_PARTS_MFR_NAME,
"ACME");
    IManufacturerPart mfrPart = (IManufacturerPart)
m_session.getObject(

IManufacturerPart.OBJECT_TYPE, params);
    IRow rowMfrParts = tblMfrParts.createRow(mfrPart);
    //Add a substance group
    ITable tblMfrPartComp = dec.getTable(

DeclarationConstants.TABLE_MANUFACTURERPARTCOMPOSITION);
    ISubstance sg =
(ISubstance)m_session.getObject(SubstanceConstants.CLASS_SUBSTANCE
_GROUP,
                  "Lead Componds");
    params.clear();
    params.put(DeclarationConstants.ATT_PARENT_ROW, rowMfrParts);
```

```
params.put(DeclarationConstants.ATT_MANUFACTURER_PART_COMPOSITION_
SUBSTANCE_NAME, sg);
    IRow rowSubGroup = tblMfrPartComp.createRow(params);
    //Add a substance
    ISubstance sub =
(ISubstance)m_session.getObject(SubstanceConstants.CLASS_SUBSTANCE
,
                  "Lead");
    params.clear();
    params.put(DeclarationConstants.ATT_PARENT_ROW, rowSubGroup);

params.put(DeclarationConstants.ATT_MANUFACTURER_PART_COMPOSITION_
SUBSTANCE_NAME, sub);
    IRow rowSubs = tblMfrPartComp.createRow(params);
  } catch (APIException ex) {
      System.out.println(ex);
  }
}
```

含有基準へのサブスタンスの追加

含有基準の [サブスタンス] テーブルは、制限されるサブスタンスとそのしきい値 PPM (100 万分の 1 単位) を識別するため、Agile PG&C では重要です。含有基準の [サブスタンス] テーブルに追加できるのは、サブスタンスとサブスタンス グループのみです。サブスタンスを [サブスタンス] テーブルに追加するには、`ITable.createRow()` メソッドを使用します。`ISubstance` または `Map` オブジェクトを渡すと、新規行を作成できます。

例: 含有基準へのサブスタンスの追加

```
public void addSubstanceToSpec(ISpecification spec, ISubstance
substance)
  throws Exception {
  IRow row = null;
  //Add a substance to the Substances table
  ITable tableSub =
spec.getTable(SpecificationConstants.TABLE_SUBSTANCES);
  row = tableSub.createRow(substance);

  if (row!=null){
    //Set value of Restricted
    ICell cell =
row.getCell(SpecificationConstants.ATT_SUBSTANCES_RESTRICTED);
    IAgileList list = (IAgileList)cell.getAvailableValues();
    list.setSelection(new Object[] {"Yes"});
    cell.setValue(list);

    //Set value of Threshold Mass PPM

row.setValue(SpecificationConstants.ATT_SUBSTANCES_THRESHOLD_MASS_
PPM, new Integer(10));
  }
}
```

デクラレーションへの含有基準の追加

デクラレーションの [含有基準] テーブルは、デクラレーションに含まれたアイテム、製造元部品および部品グループに関連する含有基準をリストします。デクラレーションの目的は、サプライヤが含有基準に記載されたすべての規制に準拠できるようにすることです。

含有基準の追加に関するルール

デクラレーションでは含有基準はオプションです。デクラレーションを含有基準なしで提出する場合は、生データ (質量または PPM) をサブスタンス レベルで収集することを意味します。サプライヤは、すべてのマテリアルとサブスタンスに関する情報を提供する必要があります。

含有基準をデクラレーションに追加する場合は、デクラレーション クラスが様々なタイプの含有基準をサポートすることに注意してください。次の表に、各タイプのデクラレーションに対する含有基準の要件を示します。

デクラレーション タイプ	サポートされる含有基準の検証タイプ
均質材のデクラレーション	均質材レベル
IPC 1752-1 デクラレーション	部品レベル
IPC 1752-2 デクラレーション	均質材レベル
JGPSSI デクラレーション	部品レベル
部品のデクラレーション	部品レベルおよび均質材レベル
サブスタンスのデクラレーション	部品レベル
適合のサプライヤ デクラレーション	部品レベルおよび均質材レベル

含有基準は、デクラレーションに含まれる部品で使用されていないサブスタンスも含め、多くのサブスタンスに影響を与えます。デクラレーションがサプライヤに開示されると、含有基準の関連サブスタンスが [アイテム組成]、[製造元部品の組成] および [部品グループの組成] の各テーブルに自動的に追加されます。これによって、デクラレーションにリストされている部品に含まれたすべての規制サブスタンスを適切に追跡できます。

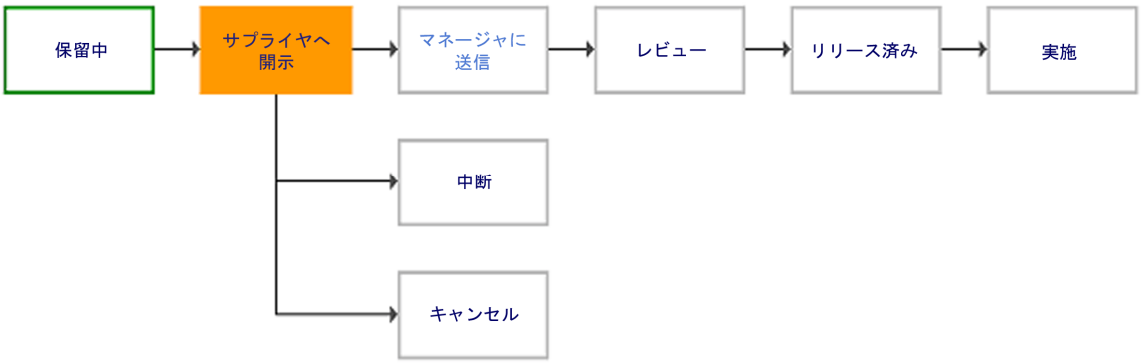
例: [含有基準] テーブルへの含有基準の追加

```
private void addSpecifications(IDeclaration dec, ISpecification[]
specs) throws Exception {
  ITable tableSpecs =
dec.getTable(DeclarationConstants.TABLE_SPECIFICATION);
  for (int i = 0; i < specs.length; ++i) {
    ISpecification spec = specs[i];
    IRow row = tableSpecs.createRow(spec);
  }
}
```

デクラレーションの送信

[デフォルト デクラレーション] ワークフローは、次の図に示すように、簡単な操作で行うことができます。

図 11: [デフォルト デクラレーション] ワークフロー



次の表に、[デフォルト デクラレーション] ワークフローの各ステータスを示します。

ステータス	説明
保留中	適合性管理者は、新規デクラレーションを作成し、新規のアイテム、製造元部品または部品グループを追加します。また、デクラレーションに含有基準を追加します。
サプライヤへ開示	適合性管理者は、デクラレーションをサプライヤに開示し、部品が含有基準に準拠しているかどうかを問い合わせます。 デクラレーションのワークフロー ステータスが [保留中] から [サプライヤへ開示] に変更されると、Agile PLM サーバによって、含有基準にリストされているサブスタンスがデクラレーションの [サブスタンス] テーブルに自動的に挿入されます。
マネージャに送信	サプライヤは電子的に署名し、デクラレーションを適合性管理者に返信します。
レビュー	適合性管理者と他のレビューアは、デクラレーションのコンテンツを確認して承認します。
リリース済み	適合性管理者はデクラレーションをリリースして、マテリアルをプロダクト レコードに公表します。
実施	部品が製造されてフィールドに配布された後、適合性管理者はデクラレーションを実施し、ワークフローを完了します。

デクラレーションを送信するには、その前に、次の 3 つの Cover Page フィールドに値を設定する必要があります。

- Cover Page.Compliance Manager
- Cover Page.Workflow
- Cover Page.Due Date

技術的には、デクラレーションの送信に必要なのは、Compliance Manager フィールドと Workflow フィールドのみです。Due Date フィールドはオプションですが、追跡の目的では指定する必要があります。次の例は、これらの 3 つのフィールドに値を設定する方法を示しています。

例: Compliance Manager、Workflow および Due Date フィールドの値の設定

```
public void setFieldsNeededForRouting(IDeclaration dec) throws
Exception {
  //Set the Compliance Manager field
  IUser user = m_session.getCurrentUser();

dec.setValue(DeclarationConstants.ATT_COVER_PAGE_COMPLIANCE_MANAGE
R, user);
  //Set the Workflow field
  IWorkflow workflow = dec.getWorkflows()[0];
  dec.setWorkflow(workflow);

  //Set the Due Date field
  DateFormat df = new SimpleDateFormat("MM/dd/yy");
  dec.setValue(DeclarationConstants.ATT_COVER_PAGE_DUE_DATE,
df.parse("05/01/05"));

}
```

デクラレーションのステータスを変更するには、IRoutable.changeStatus() メソッドを使用します。デクラレーションがサプライヤに開示された後は、サプライヤのコンタクト ユーザーのみがデクラレーションを編集できます。適合性管理者を含むその他のユーザーに対して、デクラレーションは読み取り専用になります。次の例は、適合性管理者がデクラレーションのステータスを [サプライヤへ開示] に変更する方法を示しています。

例: サプライヤへのデクラレーションの開示

```
public void openToSupplier(IDeclaration dec) throws Exception {
  IStatus status = null;
  // Get the Open to Supplier status type
  IStatus[] stats = dec.getNextStatuses();
  for (int i = 0; i < stats.length; i++) {
    if (stats[i].toString().equals("Open To Supplier")) {
      status = stats[i];
      break;
    }
  }
  // Change to the Open to Supplier status
  dec.changeStatus(status, false, null, false, false, null, null, null,
false);
}
```

ワークフロー プロセスに関連する Agile API の詳細は、161 ページの「ワークフローの管理」を参照してください。

デクラレーションの入力

デクラレーションがサプライヤに開示された場合、サプライヤには、デクラレーションを完成し、提供するコンポーネントとサブアセンブリに規制サブスタンスが含まれている場合は、それらのサブスタンスが含有基準に準拠しているかどうかを公表する責任があります。デクラレーションを完成してサインオフするには、サプライヤの 1 人以上のコンタクト ユーザーに [(限定) マテリアル プロバイダ] の役割が割り当てられている必要があります。

[(限定) マテリアル プロバイダ] ユーザーはデクラレーションを完成するために、次の操作を実行する必要があります。

- [アイテム組成]、[製造元部品の組成] および [部品グループの組成] テーブルにリストされたすべてのサブスタンスの中で、特に含有基準によって制限されているサブスタンスに対して、[質量]、[PPM 宣言値] および [適合性宣言値] フィールドを入力します。
- 必要に応じて、[組成] テーブルのその他のユーザー設定フィールドを完成させます。
- デクラレーションに対してサブスタンスを追加または削除します。

デクラレーションが完成した後、[(限定) マテリアル プロバイダ] ユーザーはサインオフし、デクラレーションを適合性管理者に提出できます。詳細は、次のセクションを参照してください。

適合性管理者へのデクラレーションの提出

サプライヤがデクラレーションのステータスを [サプライヤへ開示] から [適合性管理者に提出済み] に変更するときは、デクラレーションをサインオフする必要があります。したがって、サプライヤは、追加の password パラメータを指定する changeStatus() メソッドを使用する必要があります。

```
changeStatus(IStatus newStatus, boolean auditRelease, String comment,
boolean notifyOriginator, boolean notifyCCB, Object[] notifyList,
Object[] approvers, Object[] observers, boolean urgent, String
password)
```

次の例は、サプライヤがサインオフし、デクラレーションを適合性管理者に提出する方法を示しています。

例: デクラレーションのサインオフおよび適合性管理者への提出

```
public void submitToCM(IDeclaration dec) throws Exception {
  IStatus status = null;
  // Get the Submitted to Compliance Manager status type
  IStatus[] stats = dec.getNextStatuses();
  for (int i = 0; i < stats.length; i++) {
    if (stats[i].toString().equals("Submit To Manager")) {
      status = stats[i];
      break;
    }
  }
  // Change to the Submitted to Compliance Manager status (signoff
password is "agile")
  dec.changeStatus(status, false, null, false, false, null, null, null,
false, "agile");
}
```

デクラレーションの公表

Agile API には、マテリアル デクラレーションをプロダクト レコードに公表するためのメソッドはありません。かわりに、デクラレーションはリリースすると自動的に公表されます。したがって、API に関するかぎりは、アイテム、製造元部品または部品グループの [サブスタンス] テーブルには最新のリリース済みデクラレーションが常に反映されます。ただし、Agile Web クライアントでは、以前のデクラレーションを選択して公表し、プロダクト レコードのサブスタンス情報を更新できます。

重量値の取得および設定

Agile PG&C オブジェクトの質量 (重量) 値をサポートするために、Agile PLM には、単位フィールドが実装されています。単位のデータ タイプは、数値と単位 (グラムやオンスなど) を含む複合データ タイプです。

重量フィールドは、次のインターフェースを使用して設定および管理できます。

- IMeasure
- IUnit
- IUnitOfMeasure
- IUnitOfMeasureManager

Agile PLM 管理者は Agile Java クライアントの UOM ノードから新規の計測を定義できますが、Agile API では、Agile PG&C オブジェクトに対して重量の計測のみをサポートしています。Agile API を使用して新規の計測を定義することはできません。

Agile 9.2.1 で、アイテムの Title Block.Weight フィールドは、Title Block.Mass に変更されました。ただし、このフィールドに対する Agile API 定数は ItemConstants.TITLE_BLOCK_WEIGHT のままです。

次の例は、アイテムの [タイトル ブロック.質量] フィールドに対する値を取得および設定する方法を示しています。

例: アイテムの質量 (重量) 値の取得および設定

```
private IUnitOfMeasure getMassValue(IItem item) throws APIException
{
  IUnitOfMeasure uom =
(IUnitOfMeasure)item.getValue(ItemConstants.ATT_TITLE_BLOCK_WEIGHT
);
  System.out.println("Value: " + uom.getValue());
  System.out.println("Unit: " + uom.getUnit().toString());

  return uom;
}
private void setMassValue(IItem item, double value, String unit) throws
APIException {
  IUnitOfMeasure uom = null;
  IUnitOfMeasureManager uomm =
(IUnitOfMeasureManager)m_session.getManager(
    IUnitOfMeasureManager.class);
  uom = uomm.createUOM(value, unit);
  item.setValue(ItemConstants.ATT_TITLE_BLOCK_WEIGHT, uom);
  System.out.println("Value: " + uom.getValue());
  System.out.println("Unit: " + uom.getUnit().toString());

}
```

アイテムを質量で検索する検索条件を作成すると、数値のみが検索され、単位は検索されません。サーバは、質量値を標準単位に変換してから検索結果を返します。たとえば、次の検索条件は、質量値が 1.0 ～ 2.0 グラム (デフォルト標準単位) の間にあるすべてのアイテムを返します。検索結果には、質量が 1000 ～ 2000 ミリグラムのアイテムも含まれます。

例: アイテムを質量で検索

```
try {
  IQuery query = (IQuery)m_session.createObject(IQuery.OBJECT_TYPE,
    "select * from [Items] where [Title Block.Weight] between (1.0,
2.0)"
  );
  ITable results = query.execute();
} catch (APIException ex) {
  System.out.println(ex);
```

製造元部品のサブスタンス組成の追加

適切な権限を使用すると、デクラレーションを提出せずに、製造元部品の [含有基準]、[組成] および [サブスタンス] テーブルを直接変更できます。この機能は、製造パートナーが自社の部品に関する組成情報を指定する場合に便利です。[含有基準]、[組成] および [サブスタンス] テーブルに行を追加するには、`ITable.createRow(Object)` メソッドを使用します。

注意 製造元部品の [組成] および [サブスタンス] テーブルに追加された行は、更新または削除できません。

製造元部品の [サブスタンス] テーブルに行を追加する手順は、デクラレーションの [組成] テーブルに行を追加する方法に類似しています。サブスタンス組成を製造元部品に追加するには、次の手順に従います。

1. (オプション) [含有基準] テーブルに含有基準を追加します。
2. [組成] テーブルに行を追加します。`ManufacturerPartConstants.ATT_COMPOSITIONS_COMPOSITION_TYPE` 属性に値を指定する必要があります。
3. [サブスタンス] テーブルに 1 つ以上の行を追加します。各行は [組成] テーブルの親行を参照する必要があります。仮想属性 `ManufacturerPartConstants.ATT_PARENT_ROW` を使用して親行を指定します。サブスタンスを追加するときに、サブスタンス名とサブスタンス タイプを指定します。

[サブスタンス] テーブルへのサブスタンスの追加に関するその他のルールは、「サブスタンスの追加に関するルール」を参照してください。

親行の [組成タイプ] 属性によって、[サブスタンス] テーブルに追加できるサブスタンスのタイプが決まります。[組成タイプ] には、次の 3 つの有効値があります。

- 均質材組成 - サブパート、マテリアル、サブスタンス グループおよびサブスタンスが含まれた複数レベルのサブスタンス構成表を作成できます。組成には、サブパートまたはマテリアルを直接の子として含める必要があります。また、サブスタンスとサブスタンス グループを含めることもできますが、それらはサブパートまたはマテリアルにのみ関連付けることができます。
- サブスタンス組成 - [サブスタンス] テーブルには、サブスタンス グループとサブスタンスのみを含めることができます。
- 部品組成 - [サブスタンス] テーブルに行を追加することはできません。

[組成] テーブルの行で参照する含有基準は、その行の [組成タイプ] 属性と一致する必要があります。たとえば、行の [組成タイプ] が [均質材組成] である場合、その行で参照する含有基準の [検証タイプ] は [均質材レベル] である必要があります。

次の例は、製造元部品に対して均質材組成を定義する方法を示しています。[サブスタンス] テーブルには、サブパート、マテリアル、サブスタンス グループおよびサブスタンスの 4 つのレベルがあります。

例: 製造元部品への含有基準、組成およびサブスタンスの追加

```
public void addMfrPartSubs(IAgileSession m_session) throws
APIException {
  try {
    // Create a Manufacturer Part
    HashMap params = new HashMap();

params.put(ManufacturerPartConstants.ATT_GENERAL_INFO_MANUFACTURER
_PART_NUMBER,
     "Widget");

params.put(ManufacturerPartConstants.ATT_GENERAL_INFO_MANUFACTURER
_NAME, "ACME");
    IManufacturerPart mfrPart = (IManufacturerPart)

m_session.createObject(ManufacturerPartConstants.CLASS_MANUFACTURE
R_PART, params);
    // Add a Specification to the Specifications table
    ITable tblSpec =
mfrPart.getTable(ManufacturerPartConstants.TABLE_SPECIFICATIONS);
    ISpecification spec =
(ISpecification)m_session.getObject(ISpecification.OBJECT_TYPE,
     "Lead Spec"); // Homogeneous Material Level
    IRow rowSpec = tblSpec.createRow(spec);
    // Get the Compositions table
    ITable tblComp =
mfrPart.getTable(ManufacturerPartConstants.TABLE_COMPOSITIONS);

    // Add a row to the Compositions table that references the
specification
    params.clear();
```

```
params.put(ManufacturerPartConstants.ATT_COMPOSITIONS_SPECIFICATIO
N, spec.getName());

params.put(ManufacturerPartConstants.ATT_COMPOSITIONS_COMPOSITION_
TYPE,
     "Homogeneous Material Composition");
    IRow rowComp = tblComp.createRow(params);
    // Get the Substances table
    ITable tblSubs =
mfrPart.getTable(ManufacturerPartConstants.TABLE_SUBSTANCES);

    // Add a subpart
    ISubstance subpart = (ISubstance)m_session.

getObject(SubstanceConstants.CLASS_SUBPART, "Steel Casing");
    params.clear();
    params.put(ManufacturerPartConstants.ATT_PARENT_ROW, rowComp);

params.put(ManufacturerPartConstants.ATT_SUBSTANCES_SUBSTANCE_NAME,
subpart);
    IRow rowSubpart = tblSubs.createRow(params);
    // Add a material
    ISubstance material =
(ISubstance)m_session.getObject(SubstanceConstants.CLASS_MATERIAL,
                     "Steel");
    params.clear();
    params.put(ManufacturerPartConstants.ATT_PARENT_ROW,
rowSubpart);

params.put(ManufacturerPartConstants.ATT_SUBSTANCES_SUBSTANCE_NAME,
material);
    IRow rowMaterial = tblSubs.createRow(params);
    // Add a substance group
    ISubstance sg =
(ISubstance)m_session.getObject(SubstanceConstants.CLASS_SUBSTANCE
_GROUP,
     "Lead Componds");
    params.clear();
    params.put(ManufacturerPartConstants.ATT_PARENT_ROW,
rowMaterial);

params.put(ManufacturerPartConstants.ATT_SUBSTANCES_SUBSTANCE_NAME,
sg);
    IRow rowSubGroup = tblSubs.createRow(params);
    // Add a substance
    ISubstance sub =
```

```
(ISubstance)m_session.getObject(SubstanceConstants.CLASS_SUBSTANCE
,
     "Lead");
   params.clear();
   params.put(ManufacturerPartConstants.ATT_PARENT_ROW,
rowSubGroup);

params.put(ManufacturerPartConstants.ATT_SUBSTANCES_SUBSTANCE_NAME,
sub);
   IRow rowSubs = tblSubs.createRow(params);
  } catch (APIException ex) {
     System.out.println(ex);
  }
}
```

適合性データのロールアップ

アイテム、製造元部品および部品グループの適合性データを収集した後、適合性管理者は完成したデクラレーションをレビューして、データをプロダクト レコードに公表する準備ができているかどうかを判断します。デクラレーションが公表され、データが BOM の部品および部品グループに書き込まれた後、適合性管理者は BOM を検査してテストし、アセンブリと製品が準拠していることを確認します。このプロセスは適合性検証と呼ばれ、適合性ロールアップを通して実行されます。ロールアップはシステムに組み込まれています。ロールアップは簡単に使用でき、ロールアップ結果は UI で使用できます。適合性データのロールアップとこのプロセスの背景にあるビジネス ロジックの詳細は、『Product Governance & Compliance ユーザー・ガイド』を参照してください。

SDK では、サーバ側での PG&C ロールアップ機能の呼び出しがサポートされています。この機能は、UI が呼び出すロールアップ機能と同じです。IItem の IPGCRollup インターフェースでこの機能がサポートされています。

IPGCRollup インターフェースの理解

IPGCRollupインターフェースには、適合性データのロールアップをサポートするために次のメソッドが用意されています。

- rollup()
- rollup(Date)

これらのメソッドの一方にはパラメータがなく、他方にはパラメータとして Date が使用されます。ロールアップ API の Date パラメータは、ロールアップの実行時にタイム スタンプを設定するためにシステムで使用されます。

例: IPGCRollup メソッド

```
public interface IPGCRollup {
   public void rollup()
                  throws APIException;
   public void rollup(Date rollupDate)
                  throws APIException;
}
```

注意 rollup(Date) の起動後に、IDataObject.refresh() を呼び出してロールアップ機能が有効であることを確認する必要があります。確認しないと、最新のロールアップのタイム スタンプが Date パラメータと同じ場合に、以前のロールアップで取得した結果が表示されます。

Date パラメータを渡す場合

日付を渡さない場合は、システムが提供する現在の日付が使用されます。一連のアイテムに対してロールアップが実行されるとき、あるアイテムの最新のロールアップのタイム スタンプが、渡された Date パラメータと同じ場合は、そのアイテムに対してロールアップ プロセスは繰り返されません。かわりに、以前のロールアップで取得した結果が表示されます。ロールアップする多数のアイテムがあり、SDK を使用してそれらすべてを呼び出す必要がある場合は、この日付機能を使用できます。この場合は、最初に現在の日付を取得し、その日付を後続の SDK `Rollup(Date)` 呼び出しに対して渡します。たとえば、SDK を使用して Assembly 1 と Assembly 2 のデータをロールアップするとします。この場合は、SDK が 2 回呼び出されます。最初のインスタンスは Assembly 1 のデータをロールアップし、2 番目のインスタンスは Assembly 2 のデータをロールアップします。Assembly 2 のロールアップの実行時は、ロールアップ内にすでにある date パラメータを使用して、Item1 で取得した以前のロールアップ データを再利用します。

```
    Assembly 1
Item1
Iitem2
    Assembly 2
Item1
Item3
```

IPGCRollup インターフェースの使用

次の例では、アイテムと製造元部品に関する集合データをロールアップしています。

- アイテム (最新のリリース済み ECO または MCO)
- MPN (最新のリリース済み ECO または MCO)

アイテムに関する集合データのロールアップ

次の例では、SDK を使用して既存の API を呼び出し、特定アイテム (最新のリリース済み ECO または MCO) のトップ レベルの親を識別しています。次に、直前の API によって返されたトップ レベルの親でロールアップ API を呼び出し、アセンブリと製品が準拠していることを確認します。

例: アイテムのトップ レベルの親の識別

```
public void itemRollup(String itemStr) throws Exception{
  try {
      IItem item =
      (IItem)m_session.getObject(IItem.OBJECT_TYPE,
      itemStr);
      IQuery query =
      (IQuery)m_session.createObject(IQuery.OBJECT_TYPE,
      ItemConstants.CLASS_ITEM_BASE_CLASS);
//      IQuery query = (IQuery)
m_session.createObject(IQuery.OBJECT_TYPE,
ItemConstants.CLASS_PART);

query.setSearchType(QueryConstants.WHERE_USED_TOP_LEVEL);
     query.setCriteria("[1001] Equal To '"+item.getName()+"'");
//
```

```
query.setCriteria("["+SDKWrapper.getString("TITLE_BLOCK")+"."+SDKW
rapper.getString("IQuery_Number")+"] Equal To
'"+item.getName()+"'");
     ITable results=query.execute();
     if (results.size() > 0) {
     Iterator it = results.getReferentIterator();
        if (it.hasNext()) {
        IItem obj = (IItem)it.next();
     IItem tlaItem = (IItem)m_session.getObject(IItem.OBJECT_TYPE,
obj.getName());
     tlaItem.rollup();
        }
     }
     else {
     item.rollup();
     }

     } catch (APIException e) {
     throw e;
     }
  return;
  }
```

MPN に関する集合データのロールアップ

次の例では、SDK を使用して既存の API を呼び出し、特定 MPN (最新のリリース済み ECO または MCO) のトップ レベルの親を識別しています。次に、直前の API によって返されたトップ レベルの親でロールアップ API を呼び出し、アセンブリと製品が準拠していることを確認します。

例: MPN のトップ レベルの親の識別

```
public void testMfrPartRollup() throws Exception{
IManufacturerPartmfrp = (IManufacturerPart)
m_session.getObject(IManufacturerPart.OBJECT_TYPE,
"HARRIS::IS82C55A96");//
ITable
whereused=mfrp.getTable(ManufacturerPartConstants.TABLE_WHEREUSED)
;
Iterator it=whereused.iterator();
while(it.hasNext())
  {
    IRow r = (IRow)it.next();
    // read item number
String itemStr =
r.getValue(ManufacturerPartConstants.ATT_WHERE_USED_ITEM_NUMBER).t
oString();
     try {
     itemRollup(itemStr);
        } catch (APIException e) {
           int error = ((Integer)e.getErrorCode()).intValue();
        }
}
  return;
}
```

[適合性判定値] フィールドの値の設定

次の API を使用して、Item および ManufacturerPart オブジェクトに対する [含有基準] テーブルの [適合性判定値] フィールドに値を設定します。

```
Public void setCalculatedComplianceForPartSpec(Object specName,
Object complianceEntryValue) throws APIException
```

この API で、specName パラメータは Specification オブジェクトの名前であり、complianceEntryValue パラメータは CalculatedCompliance フィールドの実際の値で、Calculated Compliance リストのエントリになります。両方のパラメータとも文字列タイプです。

この値が SDK クライアントによって設定されると、ロールアップ時に上書きされません。この API を使用すると、ユーザーは、システムのデフォルトのロジックを使用するかわりに、独自に定義したロジックに基づいて適合性判定値を設定できます。

例: [適合性判定値] フィールドの値の設定

```
// COMPLIANT, the actual value of the Calculated Compliance field shows
the Specification
          is compliant or not based on customized calculated compliance
result
String COMPLIANT = "Compliant";
// spec_num is the Specification Name in Item object's Specification
Table
String spec_num =
row.getValue(ItemConstants.ATT_SPECIFICATIONS_SPECIFICATION).toStr
ing();
item.setCalculatedComplianceForPartSpec(spec_num, COMPLIANT);
```

第 17 章

管理タスクの実行

扱うトピックは次のとおりです。

Agile PLM 管理について

Agile Java クライアントには、Agile アプリケーション サーバを管理できる管理機能が用意されています。この機能を使用すると、事業の形態にあわせて Agile PLM システムを簡単に調整できます。Agile PLM システムは次の方法でカスタマイズできます。

- Agile PLM データベースのプロパティの変更
- オブジェクト クラスとサブクラスの定義
- プリファレンスの設定
- ユーザー アカウントの作成および設定
- ユーザー グループの定義
- 役割と権限の定義
- スマートルールの定義による変更管理プロセスの管理方法の設定

Agile API では、Agile PLM の管理機能のすべてのノードに対する読み取り/書き込みアクセスが提供されます。これは、ユーザーによる Agile PLM サブクラスの読み取りや変更を可能にする Agile API プログラムを作成したり、Agile PLM ユーザーを追加、変更または削除できることを意味します。Agile API を使用して管理ツリー階層に新しいノードを作成することはできません。したがって、ワークフロー、条件および役割は作成できません。ただし、ユーザーとユーザー グループは作成できます。これは、これらのオブジェクトが、両方とも `IDataObject` を拡張する `IUser` および `IUserGroup` というデータオブジェクトとして実装されているためです。

Agile PLM の管理に必要な権限

Agile アプリケーション サーバを管理するには、適切な権限が必要です。管理機能にアクセスするためには、管理者権限が必要です。[管理者] 役割では、サーバで使用できるすべての管理機能に対する管理者権限が付与されます。[ユーザー管理者] 役割では、ユーザーとユーザー グループに関連する機能に対する管理者権限が付与されます。

管理者権限がないと、管理ノード、ユーザーおよびユーザー グループを変更できません。Agile PLM システムに対する管理者権限が付与されていない場合は、Agile PLM 管理者にお問い合わせください。

ユーザーとユーザー グループを作成するには、これらのオブジェクトの作成権限が必要です。Agile PLM システムに付属する、[管理者]、[ユーザー管理者]、[変更分析者] などの役割には、ユーザーとユーザー グループの作成権限が組み込まれています。

管理インターフェース

次の表に、Agile PLM 管理機能に関連するインターフェースを示します。

インターフェース	説明
IAdmin	Agile PLM のクラス、ノード、ユーザーまたはユーザー グループを取得できるインターフェース。
IAgileClass	オブジェクトが属するカテゴリの識別に使用するクラス定義。
IAgileList	シングルリストおよびマルチリストのすべての属性とプロパティに関する汎用リスト インターフェース。
IAttribute	オブジェクト内の特定のデータ メンバーに関する詳細情報の提供。
IAutoNumber	自動採番ソース。事前定義済みの連続番号で、Agile PLM オブジェクトを自動的に採番する場合に使用します。
ICriteria	主に検索とワークフローに使用する再利用の検索条件セット。
INode	管理階層のノード。各ノードは、Agile Java クライアントのいずれかの管理ノードに相当します。
IProperty	Agile PLM 管理ノードのプロパティ。
IRoutableDesc	IRoutable インターフェースを実装するオブジェクトを説明するメタデータ。IRoutableDesc を使用すると、クラスのオブジェクトをインスタンス化せずに、そのクラスのワークフローを取得できます。
ITableDesc	Agile PLM テーブルを説明するメタデータ。ITableDesc を使用すると、テーブルをロードせずにテーブルの属性を取得できます。
ITreeNode	階層ツリー構造内の一般的なノード。INode や IFolder などの管理インターフェースは ITreeNode のサブインスタンスであるため、ITreeNode の機能を継承します。 **注意:** ITreeNode と同様の機能を提供する ITree インターフェースもありますが、お薦めできません。かわりに、ITreeNode を使用してください。
IUser	Agile PLM ユーザー。

インターフェース	説明
IUserGroup	ユーザー グループ。ユーザー グループを使用すると、プロジェクト チーム、拠点関連のグループ、部署およびグローバル グループを定義できます。
IWorkflow	ワークフロー ノード。

IAdmin インスタンスの取得

IAdmin インターフェースは、Agile アプリケーション サーバのほとんどの管理機能に対するアクセスを提供します。IAdmin インターフェースを使用するには、最初に、現行セッションから IAdmin のインスタンスを取得します。次の例は、Agile アプリケーション サーバにログインして IAdmin インスタンスを取得する方法を示しています。

例: IAdmin インスタンスの取得

```
public IAgileSession m_session;
public IAdmin m_admin;
public AgileSessionFactory m_factory;

try {
  HashMap params = new HashMap();
  params.put(AgileSessionFactory.USERNAME, "jdassin");
  params.put(AgileSessionFactory.PASSWORD, "agile");
  m_factory =
AgileSessionFactory.getInstance("http://agileserver/virtualPath");
  m_session = m_factory.createSession(params);
  m_admin = m_session.getAdminInstance();
} catch (APIException ex) {
  System.out.println(ex);
}
```

IAdmin インスタンスを取得すると、次のアクションを実行できます。

- サーバ ノードの移動
- フォルダ階層の移動
- Agile PLM クラスとサブクラスの取得
- ユーザーの取得
- ユーザー グループの取得

ノードの使用

INode オブジェクトは、Agile PLM の管理ツリー内の単一ノードまたは単一オブジェクトを表します。Windows エクスプローラ インターフェースと同様に、各 INode を展開して子ノードを表示できます。この階層を使用して、Agile アプリケーション サーバ上の管理ツリー構造をナビゲートできます。ノードの例には、ルート ノード (データベース ノードとも呼ばれます)、クラス、プリファレンス、役割、権限およびスマートルールがあります。

次の表に、Agile Java クライアントのノードと Agile API 管理機能とのマッピングを示します。

Agile Java クライアントのノード	対応する Agile API
データ設定	
クラス	NodeConstants.NODE_AGILE_CLASSES
文字セット	NodeConstants.NODE_CHARACTER_SETS
リスト	サポートされていません
プロセスの拡張	サポートされていません
自動採番	NodeConstants.NODE_AUTONUMBERS
条件	NodeConstants.NODE_CRITERIA_LIBRARY
ワークフロー設定	
ワークフロー	NodeConstants.NODE_AGILE_WORKFLOWS
ユーザー設定	
アカウント規約	サポートされていません
ユーザー	ユーザーの検索条件の作成
ユーザー グループ	ユーザー グループの検索条件の作成
サプライヤ グループ	サポートされていません
役割	NodeConstants.NODE_ROLES
権限	NodeConstants.NODE_PRIVILEGES
ユーザー モニタ	サポートされていません
削除されたユーザー	サポートされていません
削除されたユーザー グループ	サポートされていません
システム設定	
スマートルール	NodeConstants.NODE_SMARTRULES
Viewer とファイル	NodeConstants.NODE_VIEWER_AND_FILES
通知	NodeConstants.NODE_NOTIFICATION_TEMPLATES
全文検索	サポートされていません
UOM	サポートされていません
組織のプロファイル	サポートされていません
通貨換算レート	IAdmin.getConversionRates()
部品分類	サポートされていません
Product Cost Management	
出荷先の場所	サポートされていません
Program Execution	
プログラムの状態	サポートされていません

Agile Java クライアントのノード	対応する Agile API
コスト ステータス	サポートされていません
品質ステータス	サポートされていません
リソース ステータス	サポートされていません
ダッシュボード管理	サポートされていません
デフォルトの役割	サポートされていません
Agile Content Service	
確認通知受信者	NodeConstants.NODE_SUBSCRIBERS
送信先	NodeConstants.NODE_DESTINATIONS
イベント	NodeConstants.NODE_EVENTS
フィルタ	NodeConstants.NODE_FILTERS
パッケージ サービス	サポートされていません
回答サービス	サポートされていません
Product Governance & Compliance	
サインオフ メッセージ	サポートされていません
サーバ設定	
サーバの場所	NodeConstants.NODE_SERVER_LOCATION
データベース	NodeConstants.ROOT
プリファレンス	NodeConstants.NODE_PREFERENCES
ライセンス	NodeConstants.NODE_SERVER_LICENSES NodeConstants.NODE_USER_LICENSES
タスク モニタ	サポートされていません
タスクの設定	サポートされていません
例	
役割の例	サポートされていません
権限の例	サポートされていません
条件の例	サポートされていません

Agile Web クライアントでは、メニューから [管理]、[設定] の順に選択してシステムとユーザーの設定を表示し、編集できます。次の表に、Agile Web クライアントの管理機能と Agile API とのマッピングを示します。

Agile Web クライアントのノード	対応する Agile API
[ツール] > [個人設定]	
ユーザー プロファイル	[ユーザー.一般情報] ページ

Agile Web クライアントのノード	対応する Agile API
パスワードの変更	IUser.changeLoginPassword() および IUser.changeApprovalPassword()
権限委譲	サポートされていません
ブックマークの整理	[私の受信トレイ] フォルダ
検索の整理	[検索] フォルダ
レポートの整理	サポートされていません
パーソナル グループ	[私の受信トレイ] フォルダ
削除されたパーソナル グループ	サポートされていません
パーソナル条件	サポートされていません
パーソナル サプライヤ グループ	サポートされていません
[ツール] > [管理] > [Web クライアント設定]	
テーマ	サポートされていません
[ツール] > [管理者] > [ユーザー設定]	
ユーザー	ユーザーの検索条件の作成
ユーザー グループ	ユーザー グループの検索条件の作成
サプライヤ グループ	サポートされていません
削除されたユーザー	サポートされていません
削除されたユーザー グループ	サポートされていません
ダッシュボードの設定	サポートされていません

Agile PLM クライアントの管理ノードの名前は、それぞれの NodeConstants と完全には一致しません。たとえば、Agile Java クライアントの [通知] ノードは、NodeConstants.NODE_NOTIFICATION_TEMPLATES に相当します。同様に、Agile PLM データベースに表示されるノードの階層は、Agile Java クライアントのノード階層とは正確に一致しません。

Agile API プログラムで Agile PLM 管理ノードのツリー表示が提供される場合は、その表示を使用して INode オブジェクトを対話形式で取得することができます。各 INode オブジェクトから子ノードを取得できます。管理ノード階層を移動し続けると、すべてのノード レベルに到達できます。

次の例は、ルート ノードとその子ノードを取得して、Agile アプリケーション サーバのトップレベル ノードを表示する方法を示しています。

例: トップレベル ノードの取得

```
private void getTopLevelNodes() throws APIException {
  INode root = m_admin.getNode(NodeConstants.ROOT);
  if (null != root) {
    System.out.println(root.getName() + ", " + root.getId());
    Collection childNodes = root.getChildNodes();
```

```
    for (Iterator it = childNodes.iterator();it.hasNext();) {
      INode node = (INode)it.next();
      System.out.println(node.getName() + ", " + node.getId());
    }
  }
}
```

注意 ルート ノードで getChildNodes() を呼び出すと、ドキュメントに記載されていない複数の Agile PLM ノードが結果に含まれます。ドキュメントに記載されていないノードは Agile API でサポートされていません。

より迅速にアクセスするために、ノード ID 定数を指定してノードを取得することもできます。NodeConstants クラスは、直接アクセスできるすべての管理ノードをリストします。次の例は、SmartRules ノードとそのプロパティを取得する方法を示しています。

例: SmartRules 値の取得

```
private void getSmartRules() throws APIException {
  //Get the SmartRules node in Agile Administrator
  INode node = m_admin.getNode(NodeConstants.NODE_SMARTRULES);
  System.out.println("SmartRules Properties");

  //Get SmartRules properties
  IProperty[] props = (IProperty[])node.getProperties();
  for (int i = 0; i < props.length; i++) {
    System.out.println("Name : " + props[i].getName());
    Object value = props[i].getValue();
    System.out.println("Value : " + value);
  }
}
```

ノードを取得する別の方法は、親ノードを特定した後、ITreeNode.getChildNode() メソッドを使用してその子ノードのいずれかを取得することです。getChildNode() メソッドを使用すると、ノードを名前または ID で指定できます。各ノード レベルをスラッシュ (/) で区切って、サブノードへのパスを指定することもできます。次の例は、getChildNode() メソッドを使用してノードを取得する方法を示しています。

例: ITreeNode.getChildNode() を使用したノードの取得

```
private INode getChildNode(INode node, String childName) throws
APIException {
  Node child = (INode)(node.getChildNode(childName));
  return child;

}
```

[クラス] ノードの使用

[クラス] ノードとそのサブノートは、IAdmin.getAgileClasses() メソッドで返される IAgileClass オブジェクトと類似しています。相違点は、getAgileClasses() では、アイテムや変更など、ノードとして表示されない仮想クラスが返されることです。特定ノードの属性のプロパティを変更する場合は、IAdmin.getAgileClasses() または IAdmin.getAgileClass() メソッドの使用をお薦めします。[クラス] ノードとそのサブノードを移動することでサブクラスを変更することも可能ですが、IAgileClass オブジェクトを使用するほうが簡単です。詳細は、280 ページの「Agile PLM クラスの管理」を参照してください。

Agile PLM クラスの管理

Agile の [クラス] ノードは、部品、変更、パッケージなどの Agile PLM オブジェクトを分類するためのフレームワークを提供します。Agile Java クライアントを使用すると、組織の新しいサブクラスを定義できます。Agile API を使用して新しいサブクラスを作成することはできませんが、既存のサブクラスの読み取りや変更はできます。たとえば、各テーブルや各ページに表示される属性を定義することでサブクラスをカスタマイズできます。

Agile PLM クラスのフレームワークは、Agile PLM で作成されるオブジェクトのタイプに基づいています。Agile PLM システムで使用可能なオブジェクトは、自社が購入した Agile PLM サーバ ライセンスによって異なります。

各 Agile PLM クラスには少なくとも 1 つのサブクラスがあります。次の表に、Agile PLM の基本クラス、クラス、および Agile に付属しているサブクラスを示します。Agile PLM システムには、他のユーザー定義サブクラスを組み込むことができます。

基本クラス	クラス	事前定義済みのサブクラス
変更	設計変更	ECO
	設計変更依頼	ECR
	期限付き設計変更	期限付き設計変更
	製造元変更	MCO
	価格変更	PCO
	拠点毎変更	SCO
	出荷停止	出荷停止
顧客	顧客	顧客
デクラレーション	均質材のデクラレーション	均質材のデクラレーション
	IPC 1752-1 デクラレーション	IPC 1752-1 デクラレーション
	IPC 1752-2 デクラレーション	IPC 1752-2 デクラレーション
	JGPSSI デクラレーション	JGPSSI デクラレーション
	部品のデクラレーション	部品のデクラレーション
	サブスタンスのデクラレーション	サブスタンスのデクラレーション
	適合のサプライヤ デクラレーション	適合のサプライヤ デクラレーション
ディスカッション	ディスカッション	ディスカッション
ファイル フォルダ	ファイル フォルダ	ファイル フォルダ
	履歴レポート ファイル フォルダ	スケジュールにより生成
		ユーザーにより保存
アイテム	ドキュメント	ドキュメント
	部品	部品

基本クラス	クラス	事前定義済みのサブクラス
製造元部品	製造元部品	製造元部品
製造元	製造元	製造元
パッケージ	パッケージ	パッケージ
価格	公表価格	契約
		公表価格
	見積履歴	見積履歴
製品サービス依頼	不具合レポート	NCR
	問題レポート	問題レポート
プログラム	アクティビティ	フェーズ
		プログラム
		タスク
	ゲート	ゲート
品質変更依頼	検証	検証
	是正予防処置	CAPA
レポート 1	カスタム レポート	カスタム レポート
	外部レポート	外部レポート
	標準レポート	管理者レポート
		標準レポート
見積依頼	見積依頼	RFQ
見積依頼回答	見積依頼回答	見積依頼回答
拠点	拠点	拠点
ソーシング プロジェクト	ソーシング プロジェクト	ソーシング プロジェクト
含有基準	含有基準	含有基準
サブスタンス	マテリアル	マテリアル
	サブパート	サブパート
	サブスタンス グループ	サブスタンス グループ
	サブスタンス	サブスタンス
サプライヤ	サプライヤ	ブローカー
		部品メーカー
		受託製造業者
		ディストリビュータ

基本クラス	クラス	事前定義済みのサブクラス
		メーカー代表者
転送依頼	自動転送	ATO
	コンテンツ転送	CTO
ユーザー グループ	ユーザー グループ	ユーザー グループ
ユーザー	ユーザー	ユーザー

注意 レポート オブジェクトは Agile API でサポートされていません。

具象クラスと抽象クラス

Agile PLM スーパークラス (アイテムや変更など) は、他の抽象クラス (部品クラス、ドキュメント クラス、設計変更クラスなど) の親クラスとして機能する抽象クラスです。抽象スーパークラスと抽象クラスはインスタンス化できません。

具象クラスは、Agile API でインスタンス化できるユーザー定義サブクラスです。具象クラスの例には、部品、ドキュメント、ECO、ECR などがあります。

`IAgileSession.getObject()` メソッドを使用してオブジェクトをロードする場合は、Agile PLM の具象クラスまたは抽象クラスを指定できます。たとえば、次のメソッドはすべて、指定された同じ部品をロードします。

例: 抽象クラスまたは具象クラスを使用したオブジェクトのロード

```
try {
  IItem item;
  // Load a part using the Item base class
  item =
(IItem)m_session.getObject(ItemConstants.CLASS_ITEM_BASE_CLASS,
"1000-02");
  // Load a part using the Parts class
  item = (IItem)m_session.getObject(ItemConstants.CLASS_PARTS_CLASS,
"1000-02");
  // Load a part using the Part subclass
  item = (IItem)m_session.getObject(ItemConstants.CLASS_PART,
"1000-02");
} catch (APIException ex) {
  System.out.println(ex);
}
```

クラスの配列を取得するには、`IAgileClass.getAgileClasses()` メソッドを使用します。返すクラスの範囲を指定できます。たとえば、`range`パラメータに `IAdmin.CONCRETE` を指定すると具象クラスのみが返され、`IAdmin.ALL` を指定するとすべてのクラスが返されます。

例: クラスの取得

```
private void getConcreteClasses() throws APIException {
  IAgileClass[] classes = m_admin.getAgileClasses(IAdmin.CONCRETE);
  for (int i = 0; i < classes.length; i++) {
    System.out.println("Class Name : " + classes[i].getName());
    System.out.println("ID : " + classes[i].getId());
  }
}
```

```
void getAllClasses() throws APIException {
  IAgileClass[] classes = m_admin.getAgileClasses(IAdmin.ALL);
  for (int i = 0; i < classes.length; i++) {
    System.out.println("Class Name : " + classes[i].getName());
    System.out.println("ID : " + classes[i].getId());
  }
}
```

IAgileSession.createObject() メソッドを使用して新しいオブジェクトを作成する場合は、Agile PLM の具象クラス、つまり、ユーザー定義サブクラスのいずれかを指定する必要があります。抽象クラスはインスタンス化できないことに注意してください。次の例は、部品サブクラスのオブジェクトを作成する方法を示しています。

例: 部品の作成

```
try {
  Map params = new HashMap();
  params.put(ItemConstants.ATT_TITLE_BLOCK_NUMBER, "1000-02");
  IItem item =
(IItem)m_session.createObject(ItemConstants.CLASS_PART, params);
} catch (APIException ex) {
  System.out.println(ex);
}
```

クラスの参照

Agile PLM のクラスは次の方法で参照できます。

- オブジェクト (IAgileClass) で参照。
- クラス ID 定数 (ItemConstants.CLASS_PART や ChangeConstants.CLASS_ECO など) で参照。Agile API のすべての定数は、接尾辞名が「Constants」のクラスに含まれています。たとえば、ItemConstants には、IItem オブジェクトに関連するすべての定数が含まれています。
- クラス名 (「部品」や「ECO」など) で参照。

通常、次の理由により、クラスは名前で参照しないでください。

- クラス名は変更される場合があります。
- クラス名は必ずしも一意ではありません。重複するクラス名が存在する可能性があります。したがって、クラスを名前で参照すると、意図しないクラスを誤って参照する可能性があります。
- クラス名はローカライズされます。つまり、名前は言語によって異なります。

クラスのターゲット タイプの識別

各クラスには特定のターゲット タイプがあります。ターゲット タイプは、クラスが作成できる Agile PLM オブジェクトのタイプです。たとえば、部品サブクラスのターゲット タイプは IItem.OBJECT_TYPE です。ターゲット タイプを使用して、Agile PLM システムに定義されているユーザー定義サブクラスを分類できます。たとえば、アイテム クラスを表示するユーザー インターフェースを作成する場合は、ターゲット タイプ IItem.OBJECT_TYPE を使用してクラスを選択することで、実行時にクラスをリストできます。

例: クラスのターゲット タイプの取得

```
private void getConcreteItemClasses() throws APIException {
  IAgileClass[] classes = m_admin.getAgileClasses(IAdmin.CONCRETE);
  for (int i = 0; i < classes.length; i++) {
    if (classes[i].getTargetType() == IItem.OBJECT_TYPE) {
      System.out.println("Class Name : " + classes[i].getName());
      System.out.println("ID : " + classes[i].getId());
    }
  }
}
```

アイテム クラスには、事前定義済みの 2 つの具象クラス (ドキュメントと部品) があります。自社で Agile PLM システムにアイテム サブクラスを追加していない場合、前述のコード例では、次の結果が印刷されます。

```
Class Name : Document
ID : 9141
Class Name : Part
ID : 10141
```

属性の使用

Agile API プログラムで取得できる各オブジェクトには、一連の属性があります。属性は、特定のビジネス オブジェクトのメタデータを表します。属性には、オブジェクトのプロパティと値が定義されます。たとえば、[タイトル ブロック.番号]、[タイトル ブロック.説明] および [タイトル ブロック.部品カテゴリ] は、部品に関する 3 つのタイトル ブロック属性です。

オブジェクトのインスタンスをプログラムで作成する場合、オブジェクト クラスの各 IAttribute 属性はフィールド (ICell オブジェクト) に相当します。IAttribute オブジェクトは、プログラムで作成または開いているオブジェクトに対する ICell オブジェクトに直接対応しています。ICell オブジェクトの詳細は、89 ページの「データ セルの使用」を参照してください。

属性の参照

Agile PLM の属性は次の方法で参照できます。

- オブジェクト (IAttribute) で参照。
- 属性 ID 定数で参照。属性 ID 定数を含む Agile API のすべての定数は、接尾辞名が「Constants」のクラスに含まれています。たとえば、ItemConstants には、IItem オブジェクトに関連するすべての定数が含まれています。
- 完全修飾名 ([タイトル ブロック.番号] や [カバー ページ.変更カテゴリ] など) で参照。
- 略式名称 ([番号] など) で参照。ただし、属性の略式名称は Agile PLM 内で一意ではありません。複数の属性を参照する場合は、2 つの異なる属性に同じ略式名称が指定されていると競合が発生します。

注意 属性名は変更される場合があるため、属性は ID 番号または定数で参照することをお薦めします。ただし、このマニュアルの多くの例では、読み易さの観点から、属性を名前で参照しています。

次の例は、属性 ID 定数を参照する方法を示しています。

例: 属性 ID 定数の参照

```
Integer attrID = ItemConstants.ATT_TITLE_BLOCK_DESCRIPTION;
try {
  v = item.getValue(attrID);
} catch (APIException ex) {
  System.out.println(ex);
}
```

完全修飾名は、次の形式の文字列です。

テーブル名.属性名

テーブル名は、属性が表示されるテーブルの名前です。属性名は、属性の [名前] プロパティの現在の値です。すべての属性にはデフォルトの名前がありますが、名前は変更可能です。特に、Agile PLM システムに表示される [ユーザー定義 1] 属性と [ユーザー定義 2] 属性は、“Text01”、“List01”、“Date01” ではなく、わかりやすい名前が指定される可能性があります。

[カバー ページ.変更の理由] と [タイトル ブロック.番号] の 2 つは、完全修飾属性名の例です。

次の例は、完全修飾属性名を参照する方法を示しています。

例 17-10: 属性名の参照

```
Object v;
String attrName = "Title Block.Description";
try {
  v = item.getValue(attrName);
} catch (APIException ex) {
  System.out.println(ex);
}
```

注意 属性名では、大文字と小文字が区別されます。

属性の取得

IAttribute オブジェクトは特定のサブクラスに関連付けられています。たとえば、部品の属性は、ECO の属性とは異なります。したがって、オブジェクトのサブクラスを認識している場合は、そのオブジェクトの属性のリストを取得できます。次の表に、属性を取得するために使用できるメソッドを示します。

メソッド	説明
IAgileClass.getAttribute()	クラスの特定の IAttribute オブジェクトを取得します。
IAgileClass.getAttributes()	クラスのすべてのテーブルに対する IAttribute オブジェクトの配列を取得します。
IAgileClass.getTableAttributes()	クラスの特定のテーブルに対する IAttribute オブジェクトの配列を取得します。
ITable.getAttributes()	テーブルに対する IAttribute オブジェクトの配列を取得します。
ICell.getAttribute()	セルの IAttribute オブジェクトを取得します。

次の例は、[BOM] テーブルの属性を取得する方法を示しています。

例: 部品サブクラスの [BOM] テーブルの属性の取得

```
try {
  // Get the Part subclass
  IAgileClass partClass =
   (IAgileClass)m_admin.getAgileClass(ItemConstants.CLASS_PART);
  // Get the collection of BOM table attributes for the Part subclass
  IAttribute[] attrs =
    partClass.getTableAttributes(ItemConstants.TABLE_BOM);
} catch (APIException ex) {
  System.out.println(ex);
}
```

特定のテーブルの属性を取得するための別の方法は、最初にテーブルを取得し、次に、ITable.getAttributes() メソッドを使用してそのテーブルの属性を取得することです。

例: テーブルからの [BOM] テーブルの属性コレクションの取得

```
try {
  // Get Part P200
  IItem item = (IItem)m_session.getObject(IItem.OBJECT_TYPE, "P200");

  // Get the BOM table
  ITable bomTable = item.getTable(ItemConstants.TABLE_BOM);

  // Get BOM table attributes
  IAttribute[] attrs = bomTable.getAttributes();

} catch (APIException ex) {
  System.out.println(ex);
}
```

個々の属性の取得

取得する属性を認識している場合は、IAgileClass.getAttribute() メソッドを使用してその属性を取得できます。次の例は、ECO の [カバー ページ.理由コード] 属性を取得する方法を示しています。

例: [カバー ページ.理由コード] 属性の取得

```
try {
  // Get the ECO subclass
  IAgileClass classECO = m_admin.getAgileClass("ECO");
  // Get the "Cover Page.Reason Code" attribute
  IAttribute attr =
classECO.getAttribute(ChangeConstants.ATT_COVER_PAGE_REASON_CODE);

  // Get available values for Reason Code
  IAgileList availValues = attr.getAvailableValues();

} catch (APIException ex) {
  System.out.println(ex);
}
```

属性のプロパティの編集

Agile PLM クラスには属性があり、属性にはプロパティがあります。特定サブクラスの属性のプロパティを変更するには、次の手順に従います。

1. IAdmin.getAgileClass() メソッドを使用して Agile PLM クラスを取得します。
2. IAgileClass.getAttribute() メソッドを使用して、そのクラスの属性を取得します。
3. IAttribute.getProperty() メソッドを使用して、その属性のプロパティを取得します。
4. IProperty.getValue() メソッドを使用して、そのプロパティの現在の値を取得します。
5. IProperty.setValue() メソッドを使用して、そのプロパティに新しい値を設定します。

ユーザー定義属性の使用

Agile PLM の各サブクラスについて、[ユーザー定義 1] および [ユーザー定義 2] テーブルに追加の属性を定義できます。これらのユーザー定義属性は、顧客設定フィールドとも呼ばれ、Agile PLM に事前に定義されている属性と同様に動作します。ユーザー定義属性を取得して、そのプロパティを編集できます。

ユーザー定義属性は、Agile PLM システムのカスタム拡張機能です。したがって、その ID は CommonConstants クラスには含まれません。ただし、Agile Java クライアントでは、ユーザー定義属性を含むすべての属性のベース ID を表示できます。ユーザー定義属性のベース ID を実行時にプログラムで取得する手順を記述することもできます。

管理ノードのプロパティの使用

Agile API を使用して INode オブジェクトを取得する場合は、INode のプロパティ値も表示できます。IProperty オブジェクトは、管理ノードの単一のプロパティを表します。ノードのすべてのプロパティの配列を返すには、INode.getProperties() メソッドを使用します。

次の例は、[週末の催促/エスカレーションの設定] プリファレンスのプロパティ値を取得する方法を示しています。この例の最後の部分では、このシングルリストのプロパティに使用可能なリスト値が、カンマ区切りの文字列に変換されています。

例: プロパティ値の取得

```
private void getReminderEscalationWeekendProp() throws APIException
{
  //Get the General Preferences node
  INode node = m_admin.getNode(NodeConstants.NODE_PREFERENCES);

  //Get the Reminder/Escalation Weekend Setting property
  IProperty prop = node.getProperty(

PropertyConstants.PROP_REMINDER_ESCALATION_WEEKEND_SETTING
  );

  //Get the Reminder/Escalation Weekend Setting property value
  Object value = prop.getValue();
  System.out.println("Reminder/Escalation Weekend Setting : " +
value);
  IAgileList avail = prop.getAvailableValues();
  if (avail != null) {
    String strAvail = listToString(avail);
    System.out.println("Available Values : " + strAvail);
  }
}
private String listToString(IAgileList list) throws APIException {
  String strList = "";
  Collection children = list.getChildNodes();
  for (Iterator it = children.iterator();it.hasNext();) {
    IAgileList childList = (IAgileList)it.next();
    strList = strList + childList.getValue();
    if (it.hasNext()) {
      strList = strList + ", ";
    }
  }
  return strList;
}
```

シングルリストとマルチリストのプロパティは、他のタイプのプロパティとは異なります。IProperty.getValue() および IProperty.setValue() メソッドを使用して、値リストに含まれるプロパティを直接変更することはできません。かわりに、IAgileList.setSelection() メソッドを使用してリスト ノードを選択し、次に、IProperty.setValue() メソッドを使用して値を設定します。シングルリストとマルチリストのプロパティを変更する方法の詳細は、5-6 ページの「リスト値の取得および設定」を参照してください。

ユーザーの管理

ユーザーは、アイテムや変更と同様に作成できるデータオブジェクトです。したがって、管理者ノード階層を移動することなく直接ユーザーを管理できます。適切な Agile PLM 権限がある場合は、ユーザーを作成、変更および削除できます。たとえば、組織ディレクトリの使用可能なデータと Agile PLM ユーザーを定期的に同期化するプログラムを作成できます。

すべてのユーザーの取得

すべての Agile PLM ユーザーを取得するには、ユーザー オブジェクトに対して検索を実行します。次の例では、すべてのユーザーを取得して、各ユーザーのユーザー名、姓および名を印刷しています。

例: すべてのユーザーの取得

```
private void getAllUsers() throws APIException {
  IQuery q = (IQuery)m_session.createObject(IQuery.OBJECT_TYPE,
"select * from [Users]");
  ArrayList users = new ArrayList();
  Iterator itr = q.execute().getReferentIterator();
  while (itr.hasNext()) {
    users.add(itr.next());
  }
  for (int i = 0; i < users.size(); i++) {
    IUser user = (IUser)users.get(i);
    System.out.println(
      user.getValue(UserConstants.ATT_GENERAL_INFO_USER_ID) + ", " +
      user.getValue(UserConstants.ATT_GENERAL_INFO_FIRST_NAME) + ", "
+

user.getValue(UserConstants.ATT_GENERAL_INFO_LAST_NAME)
    );
  }
}
```

ユーザーの作成

ユーザーは、他のデータオブジェクトと同様に、Agile API を使用して作成できます。ユーザーを作成するには、そのユーザーのパラメータを定義して `IAgileSession.createObject()` メソッドに渡します。指定する必要がある必須パラメータは、ユーザー名とログイン パスワードです。`UserConstants` クラスにリストされているその他のユーザー属性も指定できます。

注意 Agile PLM システムに対するユーザーの認証に LDAP ディレクトリ サーバが使用されている場合は、Agile PLM システムに制限付きでアクセスできるサプライヤ ユーザーのみを作成できます。他のユーザーは、ディレクトリ サーバで作成して管理する必要があります。

新規ユーザーに指定するパスワードはデフォルト値です。承認用パスワードを指定する場合は、ログイン パスワードとは異なるパスワードを指定する必要があります。ただし、`UserConstants.ATT_GENERAL_INFO_USE_LOGIN_PASSWORD_FOR_APPROVAL` セルが「Yes」に設定されている場合を除きます。ユーザーは後でパスワードを変更できます。

例: ユーザーの作成

```
public IAgileSession m_session;
        public IAdmin m_admin;
        public AgileSessionFactory m_factory;

        private void userTest() {
          try {
            //Add code here to log in to the Agile Application Server
            //After logging in, create a new user
            IUser user = createUser("akurosawa");
          } catch (APIException ex) {
            System.out.println(ex);
          }
        }
        private IUser createUser(String newUser) throws APIException {
          //Create the new user
          Map params = new HashMap();
          params.put(UserConstants.ATT_GENERAL_INFO_USER_ID, newUser);
          params.put(UserConstants.ATT_LOGIN_PASSWORD, "agile");
          IUser user = (IUser)session.createObject(UserConstants.CLASS_USER,
        params);

          return user;
        }
```

デフォルトでは、新規ユーザーを作成すると、[同時接続] ユーザー カテゴリと [個人のユーザー プロファイル] 役割の組み合わせが割り当てられます。これによって、ユーザーはオブジェクトを表示できますが、オブジェクトを作成、承認および編集することはできません。オブジェクトを作成および変更するには、適切な作成または変更権限のある役割をユーザーに割り当てる必要があります。ユーザーの役割の設定を変更する方法の例は、292 ページの「ユーザー設定の構成」を参照してください。

サプライヤ ユーザーの作成

サプライヤ ユーザーは、デフォルトで Agile PLM システムへのアクセスを制限する [制限付き] ユーザー カテゴリに割り当てられます。[制限付き] ユーザー カテゴリによって、サプライヤ ユーザーは見積依頼に回答したり、Agile Product Cost Management (PCM) の他の機能を使用することができます。

サプライヤ ユーザーを作成するには、そのユーザーのパラメータを定義して `IAgileSession.createObject()` メソッドに渡します。ユーザー名、ログイン パスワードおよびサプライヤ名を指定する必要があります。`UserConstants` クラスにリストされているその他のユーザー属性も指定できます。

例: サプライヤ ユーザーの作成

```
        private IUser createSupplierUser(String userName, String supplier)
        throws APIException {
          HashMap userParams = new HashMap();
          userParams.put(UserConstants.ATT_GENERAL_INFO_USER_ID, userName);
          userParams.put(UserConstants.ATT_LOGIN_PASSWORD, "agile");
          userParams.put(UserConstants.ATT_SUPPLIER, supplier);
          return (IUser)m_session.createObject(UserConstants.CLASS_USER,
        userParams);
        }
```

ユーザーを新規ユーザーとして保存

IDataObject.saveAs() メソッドを使用すると、既存のユーザーを新規ユーザーとして保存できます。saveAs() メソッドを使用すると、既存のユーザーと同じライセンス、役割、権限および拠点を新規ユーザーに割り当てることができるため便利です saveAs() メソッドを使用してユーザーを保存する場合は、新規ユーザーのユーザー名とログイン パスワードに対するパラメータを指定する必要があります。

例: オブジェクトを新規オブジェクトとして保存

```
private void saveAsUser(IUser user, String newUserName) {
  try {
    //Set parameters for the new user
    Map params = new HashMap();
    params.put(UserConstants.ATT_GENERAL_INFO_USER_ID,
newUserName);
    params.put(UserConstants.ATT_LOGIN_PASSWORD, "agile");

    // Save the new user
    user.saveAs(UserConstants.CLASS_USER, params);
  } catch (APIException ex) {
    System.out.println(ex);
  }
}
```

有効期限が切れたパスワードの確認

Agile PLM のパスワードは、特定の時期に有効期限が切れるように設定できます。ユーザーのログイン パスワードの有効期限が切れると、そのユーザーは Agile アプリケーション サーバにログインできなくなります。ユーザーの承認用パスワードの有効期限が切れると、そのユーザーは変更を承認できなくなります。Agile PLM のパスワードのいずれかまたは両方の有効期限が切れた場合、Agile API プログラムでは、ユーザーが新しいパスワードを指定できるようにする必要があります。

次の例は、有効期限が切れたパスワードに関連する Agile API エラーを確認する方法を示しています。

例: 有効期限が切れたパスワードの確認

```
private void login(String username, String password) {
  try {
    HashMap params = new HashMap();
    params.put(AgileSessionFactory.USERNAME, username);
    params.put(AgileSessionFactory.PASSWORD, password);
    AgileSessionFactory instance =

AgileSessionFactory.getInstance("http://agileserver/virtualPat
h");
    m_session = instance.createSession(params);
  } catch(APIException ex) {
    if
(ex.getErrorCode().equals(ExceptionConstants.API_MUST_CHANGE_BOTH_
PWDS))
     System.out.println("Login Failed. You must change both your login
and approval passwords.");
```

```
    else if
(ex.getErrorCode().equals(ExceptionConstants.API_MUST_CHANGE_LOGIN
_PWD))
     System.out.println("Login Failed. You must change your login
password.");
    else
     System.out.println(ex.getMessage());
    }
  }
}
```

ユーザー設定の構成

IUser オブジェクトはデータオブジェクトであり、管理ノードとは異なります。したがって、IUser オブジェクトにはプロパティではなくデータ セルがあり、ICell インターフェースを使用してユーザーの設定を構成できます。次の例は、ユーザーの [一般情報] および [ユーザー定義 1] テーブルに表示されるセルを取得する方法を示しています。他のユーザーのテーブルにあるセルにアクセスするには、IDataObject.getTable() メソッドを使用してテーブルをロードします。

例: [一般情報] および [ユーザー定義 1] のユーザー セルの取得

```
private void getUserCells(IUser user) throws APIException {
  ICell[] cells = user.getCells();
    for (int i = 0; i < cells.length; i++) {
     System.out.println(cells[i].getName() + " : " +
cells[i].getValue());
    }
}
```

ユーザーに対する 2 つの重要な設定は、[ユーザー カテゴリ] と [役割] です。[ユーザー カテゴリ] 設定では、ユーザーが Agile PLM システムで実行できる広範囲のアクションを定義します。次のユーザー カテゴリ値のいずれかを選択します。

- パワー - いつでもサーバにログインでき、制限なしで Agile PLM システムを使用できます。パワー ユーザーは、同時接続ユーザー数の限定による制約を受けません。
- 同時接続 - 同時接続ユーザー ライセンスがある場合のみ、サーバにログインできます。
- 制限付き - Agile PLM システムに制限付きでアクセスできるユーザー。サプライヤ ユーザーは、デフォルトで [制限付き] カテゴリに割り当てられます。これによって、サプライヤ ユーザーは見積依頼に回答したり、Agile Product Cost Management (PCM) の他の機能を使用することができます。制限付きユーザーは、同時接続ユーザー数の限定による制約を受けません。

[役割] 設定では、役割と権限を割り当てることで、ユーザーの機能をさらに定義します。ユーザーは、適切な役割と権限なしでオブジェクトを作成することはできません。Agile PLM のユーザー ライセンス、役割および権限の詳細は、『Agile PLM 管理者ガイド』を参照してください。

次の例は、ユーザーの [ユーザー カテゴリ] と [役割] 設定を設定する方法を示しています。

例 17-21: ユーザーの [ユーザー カテゴリ] と [役割] 設定の設定

```
private void setCategory(IUser user) throws APIException {
  //Get the User Category cell
  ICell cell =
user.getCell(UserConstants.ATT_GENERAL_INFO_USER_CATEGORY);

  //Get the available values for the cell
  IAgileList license  = cell.getAvailableValues();
```

```
  //Set the selected value to "Concurrent"
  license.setSelection(new Object[] { "Concurrent" });

  //Change the cell value
  cell.setValue(license);
 }
private void setRoles(IUser user) throws APIException {
  //Get the Role cell
  ICell cell = user.getCell(UserConstants.ATT_GENERAL_INFO_ROLES);

  //Get the available values for the cell
  IAgileList roles  = cell.getAvailableValues();

  //Set the selected roles to Change Analyst and Administrator
  roles.setSelection(new Object[] {"Change
Analyst","Administrator","My User Profile"});
  //Change the cell value
  cell.setValue(roles);
 }
```

ユーザー パスワードのリセット

ユーザー管理者権限のある管理者は、他のユーザーのパスワードを新しい値にリセットできます。この権限のないユーザーはユーザー パスワードをリセットできません。UI を使用して一度に 1 つのパスワードを手動でリセットするのではなく、この機能を使用して、多数のパスワードをバッチ モードでリセットすることをお薦めします。

この機能をサポートする changeLoginPassword() メソッドでは、現在のパスワード値のかわりにヌル値を渡すことができます。次の例は、このメソッドで、現在のパスワードのかわりにヌルを使用してユーザーのパスワードをリセットする方法を示しています。

例: パスワードの新しい値へのリセット

```
public void changeLoginPassword(null, String newPassword)
            throws APIException;
```

ユーザーの削除

ユーザーを削除するには、IDataObject.delete() メソッドを使用します。他のデータオブジェクトと同様に、オブジェクトを初めて削除すると、ソフト削除されます。つまり、オブジェクトは無効になりますが、データベースからは削除されません。Agile アプリケーション サーバでは、ユーザーを完全に削除することはできません。

例: ユーザーの削除

```
private void removeUser(IUser user) throws APIException {
  user.delete();
  user = null;
}
```

注意 Agile Java クライアントでは、[管理] > [ユーザー設定] > [削除されたユーザー] の順に選択すると、削除されたユーザーのリストを表示できます。

ユーザー グループの管理

ユーザー グループは、Agile PLM ユーザーのリストを格納する単純なオブジェクトです。ユーザー グループを使用すると、プロジェクト チーム、部署、およびグローバル グループと、そのグループに割り当てられたユーザーを定義できます。ユーザー グループは、アイテムや変更と同様に、拠点に関連しませんが、それぞれの場所に基づいてユーザーのグループを作成できます。ユーザー グループにユーザーを追加すると、その変更内容が必ずユーザーの [グループ] 設定に反映されます。この設定の属性 ID は `UserConstants.ATT_GENERAL_INFO_GROUPS` です。

注意 Agile Web クライアントなどの Agile クライアントでは、変更などのオブジェクトをユーザー グループに送信できます。Agile API では、ユーザー グループへのオブジェクトの送信はサポートされていません。ただし、ユーザー グループ オブジェクトの [ユーザー] テーブルからユーザーを取得して、そのユーザーにオブジェクトを送信することはできます。

すべてのユーザー グループの取得

すべての Agile PLM ユーザー グループを取得するには、ユーザー グループ オブジェクトに対して検索を実行します。ユーザー グループで処理を繰り返して、特定のグループを検索できます。次の例では、すべてのユーザーを取得して、各ユーザー グループの名前、説明、ユーザーの最大数および有効なステータスを印刷しています。

例: すべてのユーザー グループの取得

```
private void getAllUserGroups() throws APIException {
  IQuery q = (IQuery)m_session.createObject(IQuery.OBJECT_TYPE,
"select * from [User Groups]");
  ArrayList groups = new ArrayList();
  Iterator itr = q.execute().getReferentIterator();
  while (itr.hasNext()) {
    groups.add(itr.next());
  }
  for (int i = 0; i < groups.size(); i++) {
    IUserGroup ug  = (IUserGroup)groups.get(i);
    System.out.println(
      ug.getValue(UserGroupConstants.ATT_GENERAL_INFO_NAME) + ", " +

ug.getValue(UserGroupConstants.ATT_GENERAL_INFO_DESCRIPTION) + ", "
+

ug.getValue(UserGroupConstants.ATT_GENERAL_INFO_MAX_NUM_OF_NAMED_U
SERS) + ", " +

ug.getValue(UserGroupConstants.ATT_GENERAL_INFO_STATUS)
    );
  }
}
```

ユーザー グループの作成

ユーザーと同様に、ユーザー グループはデータオブジェクトであり、Agile アプリケーション サーバの管理ノードではありません。ユーザー グループを作成するには、そのユーザー グループのパラメータ (グループの名前など) を定義して `IAgileSession.createObject()` メソッドに渡します。指定する必要がある必須パラメータは、属性 ID `UserGroupConstants.ATT_GENERAL_INFO_NAME` の名前のみです。

UserGroupConstants クラスにリストされているその他のユーザー属性も指定できます。ユーザー グループを有効化するには、[有効] セルを [はい] に設定します。

ユーザー グループを作成する際は、複数のユーザーを [ユーザー] テーブルに追加して、グループを意味のあるものにします。[ユーザー] テーブルに新しい行を作成するには、ITable.createRow(java.lang.Object) メソッドを使用します。

例: ユーザー グループの作成

```
public IAgileSession m_session;
public IAdmin m_admin;
public AgileSessionFactory m_factory;
private void userGroupTest() throws APIException {
  //Add code here to log in to the Agile Application Server
  //After logging in, create a new user group
  IUserGroup group = createGroup("Swallowtail Project");

  //Add users to the Western project group
  IUser[] selUsers = new IUser[] {
    m_session.getObject(IUser.OBJECT_TYPE, "jford"),
    m_session.getObject(IUser.OBJECT_TYPE, "hhawkes"),
    m_session.getObject(IUser.OBJECT_TYPE, "speckinpah")
  };
  addUsers(group, selUsers);
}
private IUserGroup createGroup(String groupName) throws APIException
{
  //Create the user group
  IUserGroup group =

(IUserGroup)m_session.createObject(UserGroupConstants.CLASS_USER_G
ROUP, groupName);
  //Enable the user group
  ICell cell =
group.getCell(UserGroupConstants.ATT_GENERAL_INFO_STATUS);
  IAgileList list = cell.getAvailableValues();
  list.setSelection(new Object[] { "Active" });
  cell.setValue(list);

  return group;
}
private void addUsers(IUserGroup group, IUser[] users) throws
APIException {
  ITable usersTable =
group.getTable(UserGroupConstants.TABLE_USERS);
  for (int i = 0; i < users.length; i++) {
    IRow row = usersTable.createRow(users[i]);
  }
}
```

ユーザー グループは、グローバルまたはパーソナルにできます。グローバル ユーザー グループには、すべての Agile PLM ユーザーがアクセスできます。パーソナル ユーザー グループにアクセスできるのは、そのグループを作成したユーザーのみです。次の例は、ユーザー グループをグローバルにする方法を示しています。

例: ユーザー グループのグローバル化

```
private void setGlobal(IUserGroup group) throws APIException {
  //Get the Global/Personal cell
  ICell cell =
group.getCell(UserGroupConstants.ATT_GENERAL_INFO_GLOBAL_PERSONAL)
;

  //Get the available values for the cell
  IAgileList values  = cell.getAvailableValues();

  //Set the selected value to "Global"
  values.setSelection(new Object[] { "Global" });

  //Change the cell value

group.setValue(UserGroupConstants.ATT_GENERAL_INFO_GLOBAL_PERSONAL,
values);
}
```

ユーザー グループ内のユーザーのリスト

ユーザー グループ内のユーザーは、[ユーザー] テーブルにリストされます。したがって、ユーザー グループ内のユーザーのリストを取得するには、IDataObject.getTable() メソッドを使用し、次に、取得したテーブル行で処理を繰り返して、各ユーザーのデータにアクセスします。次の例は、ユーザー グループ内のユーザーをリストする方法を示しています。

例: ユーザー グループ内のユーザーのリスト

```
private void listUsers(IUserGroup group) throws APIException {
  ITable usersTable =
group.getTable(UserGroupConstants.TABLE_USERS);
  Iterator it = usersTable.iterator();
  while (it.hasNext()) {
    IRow row = (IRow)it.next();

System.out.println(row.getValue(UserGroupConstants.ATT_USERS_USER_
NAME));
  }
}
```

第 18 章

例外の処理

扱うトピックは次のとおりです。

例外について

Java プログラムにおける問題発生の原因となるエラーを例外と呼びます。検出できない例外が Java で発生した場合は、プログラムが終了したり、画面にエラーが表示される場合があります。例外を適切に処理するために、プログラムでは次の事項に注意する必要があります。

- 例外発生の原因と思われるメソッドを含んだコードを `try` ブロックに配置して保護します。
- `catch` ブロック内で発生した例外をテストおよび処理します。

Agile API には、`APIException` という `Exception` のサブクラスが用意されています。このサブクラスは、Agile API 全体にわたる Agile PLM ランタイム エラーの処理に使用する汎用例外クラスです。Agile API HTML リファレンスには、各メソッドで発生する例外のタイプが示されます。通常、Agile アプリケーション サーバとの相互作用が必要な Agile API メソッドでは、`APIException` が発生します。次の表に、例外を処理するための `APIException` クラス メソッドを示します。

メソッド	説明
`getErrorCode()`	`APIException` に関連付けられているエラー コードの番号を返します。
`getMessage()`	`APIException` に関連付けられているエラー メッセージを返します。
`getRootCause()`	`APIException` の根本原因を返します (ある場合)。
`getType()`	例外のタイプを返します。

例外定数

ExceptionConstants クラスには、すべての Agile アプリケーション サーバ用のリテラルおよび Agile API のランタイム エラーと警告コードが含まれています。各定数の説明は、Agile API HTML リファレンスを参照してください。

いくつかの ExceptionConstants は、Agile PLM 警告メッセージをアクションの完了前に表示するために使用される例外用です。警告メッセージ用の定数はすべて接尾辞 WARNING で終わります。プログラムで使用しない Agile PLM 警告メッセージは、無効にすることができます。詳細は、299 ページの「警告メッセージの無効化および有効化」を参照してください。

エラー コードの取得

警告エラーを正しく捕捉するには、例外のエラー コードを取得して適切に処理する必要があります。通常は、アクションを完了するかどうかをユーザーが選択できるように、確認ダイアログ ボックスを表示します。次の例は、catch ブロックで例外のエラー コードを確認する方法を示しています。

例: Agile PLM エラー コードの取得

```
private void removeApprover(IChange change, IUser[] approvers, IUser[]
observers, String comment) {
  try {
    // Remove the selected approver
    change.removeApprovers(change.getStatus(), approvers, observers,
comment);
  } catch (APIException ex) {
    if
(ex.getErrorCode().equals(ExceptionConstants.APDM_RESPONDEDUSERS_W
ARNING))
     JOptionPane.showMessageDialog(null, ex.getMessage(), "Warning",
JOptionPane.YES_NO_OPTION);
  }
}
```

エラー メッセージの取得

プログラムで Agile PLM ランタイム エラーを示す APIException が発生した場合は、エラー メッセージを表示できます。getMessage() メソッドを使用すると、次の例に示すように、エラー メッセージ文字列を返して、その内容をメッセージ ダイアログ ボックスに表示できます。

例: エラー メッセージの取得

```
// Display an error message dialog
void errorMessage(APIException ex) {
  try {
    JOptionPane.showMessageDialog(null, ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
  } catch (Exception e) {}
}
```

Agile PLM エラー メッセージのリストについては、Agile API HTML リファレンスの ExceptionConstants を参照してください。

警告メッセージの無効化および有効化

Agile PLM エラー メッセージの一部は警告であり、これによって、操作の停止または続行を選択できます。デフォルトでは、警告メッセージも含めて、ほとんどのエラー メッセージは有効になっています。警告のトリガーとなるアクションを実行すると、例外が発生します。例外の発生を回避するために、アクションの実行前に警告メッセージを無効にすることができます。

次の例は、変更のリリースによって例外が発生するかどうかを確認する方法を示しています。例外のエラーコードが ExceptionConstants.APDM_UNRESPONDEDCHANGE_WARNING の場合は、警告が表示されます。ユーザーが警告ダイアログ ボックスで [はい] をクリックすると、変更がリリースされます。

例: エラー コードの無効化および有効化

```
private void releaseChange(IAgileSession m_session, IChange chgObj)
{
  IStatus nextStatus = null;
  try {
    // Get the default next status
    nextStatus = chgObj.getDefaultNextStatus();

    // Release the Change
    chgObj.changeStatus(nextStatus, false, "", false, false, null, null,
null, false);
  } catch (APIException ex) {
    // If the exception is error code
    // ExceptionConstants.APDM_UNRESPONDEDCHANGE_WARNING,
    // display a warning message
    if (ex.getErrorCode() ==
ExceptionConstants.APDM_UNRESPONDEDCHANGE_WARNING) {
      int i = JOptionPane.showConfirmDialog(null, ex.getMessage(),
            "Warning", JOptionPane.YES_NO_OPTION);
      if (i == 0) {
        // If the user clicks Yes on the warning, disable the error code
and release the change
        try {
          // Disable the warning
```

```
m_session.disableWarning(ExceptionConstants.APDM_UNRESPONDEDCHANGE
_WARNING);
         // Release the Change
         chgObj.changeStatus(nextStatus, false, "", true, true, null,
null, null, false);
         // Enable all warnings

m_session.enableWarning(ExceptionConstants.APDM_UNRESPONDEDCHANGE_
WARNING);
       } catch (APIException exc) {}
     }
    }
  }
}
```

APIException がエラーではなく警告であることの確認

前述したように、警告のトリガーとなる操作を実行すると、例外が発生します。警告メッセージは、Agile Web クライアントのような対話型の GUI クライアントで役立ちますが、Agile API プログラムでは使用しないほうが適切な場合があります (特にバッチ処理を実行する場合)。

`APIException.isWarning()` を使用すると、Agile PLM 例外が警告かどうかを確認できます。警告の場合は、その警告を無効にして操作を続行できます。

例: APIException が警告かどうかの確認

```
private void checkIfWarning(IAgileSession m_session) {
  boolean gotWarning = true;
  while (gotWarning) {
    try {
      // Add some API code here that throws an exception
      m_session.doNothing();

      gotWarning = false;
    } catch (APIException e) {
      try {
        if (e.isWarning())

m_session.disableWarning((Integer)e.getErrorCode());
      } catch (Exception ex) {}
      continue;
    }
    break;
  }
}
```

Agile API で自動的に無効にした警告の削除

Agile Web クライアントでは、オブジェクトを削除しようとすると警告メッセージが表示されます。Agile API プログラムでは、バッチ処理に対する警告メッセージは適切ではありません。このため、Agile API では次の警告を暗黙で無効にし、ユーザーがコード内で警告を無効にする手間を省きます。

- `ExceptionConstants.APDM_HARDDELETE_WARNING`
- `ExceptionConstants.APDM_SOFTDELETE_WARNING`

オブジェクトの削除の詳細は、34 ページの「オブジェクトの削除および削除取消」を参照してください。

有効または無効にした警告の状態の保存および復元

特定操作の開始前に警告メッセージが有効か無効かを追跡するかわりに、`IAgileSession.pushWarningState()` を使用して、有効または無効にした警告の現在の状態を保存できます。操作が完了した後に、`IAgileSession.popWarningState()` を使用して、有効または無効にした警告を前の状態に復元できます。

例: pushWarningState() および popWarningState() の使用

```
private void pushPopWarningState(IAgileSession m_session, IItem item)
throws APIException {
  // Save the current state of enabled/disabled warnings
  m_session.pushWarningState();

  // Disable two AML warnings

m_session.disableWarning(ExceptionConstants.APDM_WARN_MFRNAMECHANG
E_WARNING);

m_session.disableWarning(ExceptionConstants.APDM_ONEPARTONEMFRPART
_WARNING);
  // Get the Manufacturers table
  ITable aml = item.getTable(ItemConstants.TABLE_MANUFACTURERS);

  // Create a new row and set a value for the row
  HashMap amlEntry = new HashMap();
  amlEntry.put(ItemConstants.ATT_MANUFACTURERS_MFR_NAME,
"MFR_TEST3");

amlEntry.put(ItemConstants.ATT_MANUFACTURERS_MFR_PART_NUMBER,
"MFR_PART3");
  IRow rowAML1 = aml.createRow(amlEntry);

rowAML1.setValue(ItemConstants.ATT_MANUFACTURERS_REFERENCE_NOTES,
"new note");
  // Restore the previous state of enabled/disabled warnings
  m_session.popWarningState();
}
```

第 19 章

プロセス拡張の開発

扱うトピックは次のとおりです。

プロセス拡張について

プロセス拡張 (PX) は、Agile PLM システムの機能を拡張するためのフレームワークです。機能の拡張には、サーバ側の拡張 (カスタム ワークフロー アクション、カスタム自動採番など) とクライアント側の拡張 (外部レポート、[アクション] メニューや [ツール] メニューに追加された新規コマンドなど) があります。すべてのカスタム アクションは、プロセス拡張によって提供される機能のタイプにかかわらず、ローカル クライアントではなく Agile アプリケーション サーバで起動されます。

プロセス拡張によって、Agile PLM サーバと Agile PLM ユーザーは外部システムに接続できます。また、プロセス拡張を使用すると、標準の Agile PLM クライアントでは提供されない機能を追加できます。さらに、プロセス拡張によって、簡単かつ強力な方法で Agile PLM システムをオープンにし、ビジネス要件に応じてシステムを調整できます。

プロセス拡張は、Agile アプリケーション サーバに配置されている Java クラス、または URL へのリンクのいずれかです。URL は、単なる Web サイトでも、Web ベースのアプリケーションの場所でも構いません。

プロセス拡張を使用すると、次の内容を作成できます。

- カスタム レポート
- ユーザー主導型/ワークフロー起動型カスタム アクション
- Agile PLM クライアントからアクセスできるカスタム ツール
- カスタム自動採番

プロセス拡張フレームワーク内では、どのようなタイプのカスタム アクションやツールを作成できるのでしょうか。技術的には、カスタム アクションを定義する際、カスタム アクションで実行できる操作に制限はほとんどありません。つまり、オープンエンドのソリューションです。Agile ソリューション担当および Agile パートナーの協力のもとに、必要なプロセス拡張を開発できます。

複数のプロセス拡張を 1 つのチェーンにまとめて、個別のビジネス機能を実行する各プロセス拡張にリンクできます。また、プロセス拡張を使用すると、Web サービス (Agile の Web サービス拡張フレームワークを使用して構築したサービスなど) への依頼を作成することもできます。

Agile PLM クライアントで利用可能なプロセス拡張には、5 つの統合ポイントがあります。プロセス拡張は、以下の領域から起動することができます。

- 外部レポート
- [アクション] メニュー
- [ツール] メニュー
- ワークフロー ステータス
- 自動採番ソース

カスタム自動採番ソースの開発

このセクションでは、カスタム自動採番ソースの開発方法について説明します。

ほとんどの Agile PLM オブジェクト クラスには少なくとも 1 つのデフォルト自動採番ソースがあり、新規オブジェクトを作成して次の番号を自動的に割り当てることができます。自動採番には、英数字の接頭辞または接尾辞 (あるいはその両方) を指定できます。自動採番 (文字列) の長さ、および使用する数字を指定することもできます。

自動採番は柔軟性を備えていますが、一部の企業には、Agile PLM の標準の自動採番機能では対応できない特定の要件があります。このような企業では、プロセス拡張フレームワークを使用して、カスタム自動採番ソースを定義し、Agile PLM システムに追加できます。

管理者権限があるユーザーは、Agile Java クライアントで自動採番ソースを定義できます。自動採番ソースでクライアントの標準の自動採番機能を使用するか、自動採番ソースをカスタム自動採番ソースに関連付けることができます。Agile PLM クライアントでカスタム自動採番ソースを使用して新規オブジェクトを作成すると、Agile アプリケーション サーバでは、番号を生成するためのカスタム Java コードが起動します。

カスタム自動採番ソースの定義

カスタム自動採番ソースを定義するには、`com.agile.px` パッケージのサーバ側 API である `ICustomAutoNumber` インターフェースを実装する Java クラスを作成します。コードでは、自動採番のロジック (接頭辞、接尾辞、桁数、文字セットなど) および永続性メカニズムを定義する必要があります。カスタム自動採番ソースで番号を保存する場所は、永続性に関係なくプログラムによって決まります。たとえば、番号は Oracle のような SQL データベースやファイルに保存できます。

Agile PLM サーバは、`getAutoNumber()` メソッドを呼び出してカスタム自動採番ソースから次の番号を取得します。このメソッドはクラスで指定する必要があります。次の例は、カスタム自動採番ソースに対して Java クラスを実装する方法を示しています。

例: カスタム自動採番ソースに対するクラスの定義

```
package autonumbers;
import com.agile.px.*;
import com.agile.api.*;

public class ResistorNumber implements ICustomAutoNumber
{
```

```
   public ActionResult getAutoNumber(IAgileSession session, INode
actionNode)
   {
      String num;
      // Write code here to define the custom autonumber source for
Resistors
      return new ActionResult(ActionResult.STRING, num);
   }
}
```

カスタム自動採番ソースのパッケージ化および配置

カスタム自動採番ソースのクラスを開発した後は、次の手順に従ってカスタム自動採番ソースを正しくパッケージ化して配置します。

カスタム自動採番ソースをパッケージ化して配置する手順は、次のとおりです。

1. Java 開発環境または Java アーカイブ ツール (または JAR ツール) を使用して、カスタム自動採番ソース用の JAR ファイルを 1 つ以上作成します。JAR ファイルに、com.agile.px.ICustomAutoNumber という名前のファイルが格納された META-INF/services ディレクトリが含まれていることを確認します。このファイルは、カスタム自動採番ソース用の Java の完全修飾クラス名を 1 行に 1 クラスずつリストしたテキスト ファイルです。

   1 つのパッケージに複数のカスタム自動採番ソースを含めることができます。たとえば、com.agile.px.ICustomAutoNumber ファイルには、次のように指定できます。

   autonumbers.ResistorNumber

   autonumbers.CapacitorNumber

   autonumbers.DiodeNumber

   注意 JAR ファイル内のパスでは、大文字と小文字が区別されます。したがって、JAR ファイル内の META-INF フォルダの名前は、すべて大文字か、すべて小文字である必要があります。そうでない場合、カスタム自動採番ソースは配置されません。

2. Agile アプリケーション サーバがインストールされているコンピュータの agile_home/integration/sdk/extensions フォルダに JAR ファイルを格納します。

   注意 クラスタ環境に複数のアプリケーション サーバがある場合は、クラスタ内の各サーバにプロセス拡張ファイルを配置する必要があります。

Agile Java クライアントでのカスタム自動採番ソースの設定

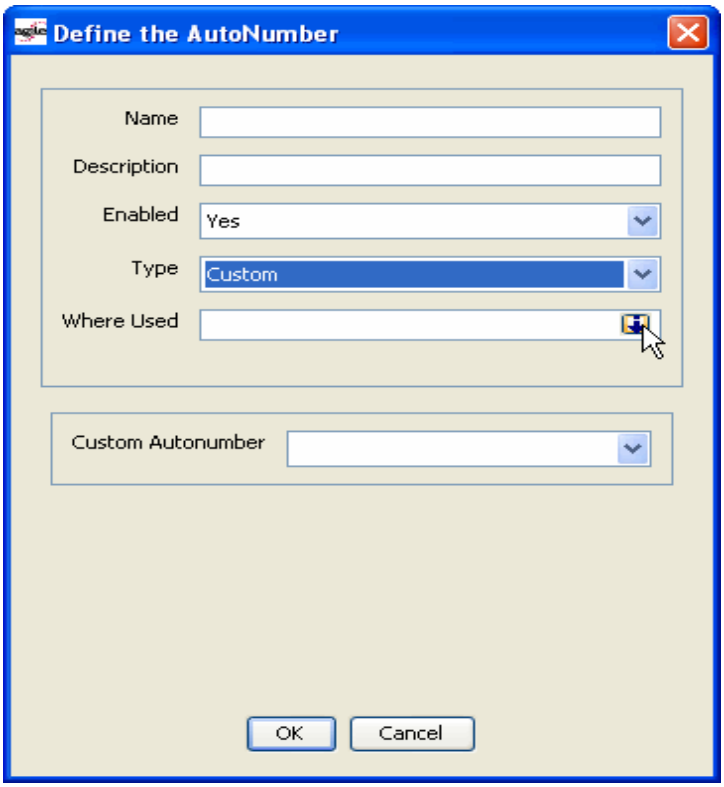
Agile Java クライアントでは、管理モジュールに自動採番ソースを定義できます。Agile PLM システム設定を構成するには、管理者アカウントが必要です。

カスタム自動採番ソースを追加する手順は、次のとおりです。

1. 管理者として Agile Java クライアントへログインします。
2. [管理] タブをクリックします。
3. [設定] > [データ設定] > [自動採番] の順に進みます。
4. [自動採番] ノードをクリックします。

5. [自動採番] ウィンドウで  をクリックします。[自動採番の定義] ダイアログ ボックスが表示されます。

図 12: [自動採番の定義] ダイアログ ボックス



6. 次の情報を入力します。
   - 名前 - 自動採番ソースの名前を入力します。
   - 説明 - 自動採番ソースの簡単な説明を入力します。
   - 有効 - [はい] または [いいえ] を選択します。
   - 自動採番のタイプ - [カスタム] を選択します。[カスタム自動採番] フィールドがアクティブになります。
   - 使用箇所 - 自動採番ソースを使用できるサブクラスを選択します。
   - カスタム自動採番 - リストからカスタム自動採番ソースを選択します。
7. [OK] をクリックして自動採番定義を保存します。

サブクラスへの自動採番ソースの割り当て

自動採番ソースを定義するときは、その自動採番ソースを使用するサブクラスを [使用箇所] フィールドに指定できます。自動採番ソースは、[クラス] ノードで、サブクラスに割り当てることもできます。

自動採番ソースをサブクラスに割り当てる手順は、次のとおりです。

1. 管理者として Agile Java クライアントへログインします。
2. [管理] タブをクリックします。
3. [データ設定] フォルダを開きます。

4. [クラス] ノードを開きます。
5. [クラス] ウィンドウで、サブクラスをダブルクリックします。サブクラス ウィンドウが表示されます。
6. [自動採番ソース] フィールドで  をクリックします。ポップアップ ウィンドウが表示されます。
7. [選択肢] リストで自動採番ソースを選択し、  をクリックして [選択済み] リストに移動します。完了したら、[OK] をクリックします。
8. [保存] をクリックして設定を保存します。

カスタム アクションの開発

このセクションでは、Java クラスでカスタム アクションを開発する方法について説明します。Agile PLM クライアントでは、これらのクラスに対してメソッドを直接呼び出してアクションを実行できます。

カスタム アクションは、Agile PLM クライアントの次の領域から起動できます。

- [アクション] メニュー
- [ツール] メニュー
- 外部レポート
- ワークフロー ステータス

カスタム アクションの定義

カスタム アクションを定義するには、com.agile.px パッケージのサーバ側 API である ICustomAction インターフェースを実装する Java クラスを作成します。実行するアクションは、コードで定義する必要があります。Agile PLM サーバは、doAction() メソッドを呼び出してアクションを起動します。このメソッドはクラスで指定する必要があります。

次の例は、HelloWorld クラスのコードを示しています。doAction() メソッドが呼び出されると、このメソッドは “Hello World” を返します。[アクション] メニューから HelloWorld カスタム アクションを起動すると、文字列 “Hello World” がオブジェクトの [履歴] テーブルに記録されます。ワークフローから HelloWorld カスタム アクションを起動すると、そのワークフローが適切なワークフロー ステータスに入ったときに、文字列 “HelloWorld” が設計変更の [履歴] テーブルに記録されます。

例: カスタム アクションに対する HelloWorld クラスの定義

```
package actions;
import com.agile.px.*;
import com.agile.api.*;

public class HelloWorld implements ICustomAction
{
   public ActionResult doAction(IAgileSession session, INode
actionNode,
                                IDataObject affectedObject)
   {
      return new ActionResult(ActionResult.STRING, "Hello World");
   }
}
```

前述の HelloWorld クラスは特定のアクションを実行しません。この例は、単にカスタム アクションにクラスを実装する方法を示しています。

カスタム アクションとユーザー セッション

Agile PLM クライアントがプロセス拡張を起動するときは、現在のユーザーのセッション内で起動します。したがって、プロセス拡張コード内、またはプロセス拡張から直接起動するコード内では、Agile API を使用して追加の `IAgileSession` オブジェクトを作成できません。つまり、プロセス拡張で新規の Agile PLM セッションが直接作成されることはありません。

Web サービス拡張 (WSX) を作成し、そのコードをプロセス拡張内から使用する場合は、Web サービス インフラストラクチャを使用せずに、WSX クラスに含まれる Java メソッドを直接起動できます (ただし、このメソッドで新規の `IAgileSession` オブジェクトを作成しない場合)。

プロセス拡張 (PX) の起動と Web サービス拡張 (WSX) の起動を混同しないでください。特に、PX と WSX が同じアプリケーション コンテナに混在している場合は、PX コードで WSX コードを直接起動しないでください。プロセス拡張で Web サービスを使用する場合、該当する WSX では新規の Agile PLM セッションが作成される可能性がありますが、このセッションは、プロセス拡張で使用するセッションとは異なります。

WSX を作成し、そのコードをプロセス拡張内から使用する場合は、Web サービス インフラストラクチャを使用せずに、WSX クラスに含まれる Java メソッドを直接起動できます (ただし、このメソッドで新規の `IAgileSession` オブジェクトを作成しない場合)。

URL ベースのプロセス拡張では、Agile PLM サーバと通信して現在選択されているビジネス オブジェクトに対してアクションを実行する外部アプリケーションを呼び出すことができます。このようなアクションを実行するために、外部アプリケーションでは、Agile API を使用して別の Agile PLM セッションを作成できます。詳細は、315 ページの「ターゲット システムからの Agile PLM セッションの作成」を参照してください。

カスタム アクションのパッケージ化および配置

カスタム アクションのクラスを開発した後は、次の手順に従ってカスタム アクションを正しくパッケージ化して配置します。

カスタム アクションをパッケージ化して配置する手順は、次のとおりです。

1. Java 開発環境または Java アーカイブ ツール (または JAR ツール) を使用して、カスタム アクション用の JAR ファイルを 1 つ以上作成します。JAR ファイルに、com.agile.px.ICustomAction という名前のファイルが格納された META-INF/services ディレクトリが含まれていることを確認します。このファイルは、カスタム アクション用の Java の完全修飾クラス名を 1 行に 1 クラスずつリストしたテキスト ファイルです。

   1 つのパッケージに複数のカスタム アクションを含めることができます。たとえば、com.agile.px.ICustomAction ファイルは次のような形式になります。

   ```
   actions.HelloWorld
   actions.RFQConsolidation
   actions.RefreshCustomerFromCRM
   actions.StartMfg
   actions.ObsoletePartReplacer
   actions.WorkflowConflictResolver
   ```

注意 JAR ファイル内のパスでは、大文字と小文字が区別されます。したがって、JAR ファイル内の META-INF フォルダの名前は、すべて大文字か、すべて小文字である必要があります。そうでない場合、カスタム アクションは配置されません。

2. Agile アプリケーション サーバがインストールされているコンピュータの agile_home/integration/sdk/extensions フォルダに JAR ファイルを格納します。

注意 クラスタ環境に複数のアプリケーション サーバがある場合は、クラスタ内の各サーバにプロセス拡張ファイルを配置する必要があります。

カスタム アクションの役割と権限

カスタム アクションを Agile Java クライアントに設定した後は、そのカスタム アクションで使用する役割を指定できます。デフォルトでは、カスタム アクションでは現在のユーザーの役割と権限を使用します。ただし、カスタム アクションには拡張した権限を設定することもできます。これは、プロセス拡張の重要な機能の 1 つです。通常のユーザーより多くの権限を付与することによって、カスタム アクションのビジネス ロジックを実施できます。カスタム アクションが Agile PLM クライアントで起動されると、そのカスタム アクションの役割と権限は、現在のユーザーの役割と権限より優先されます。そのカスタム アクションが完了した時点で、クライアントはユーザーの役割と権限に戻ります。

プロセス拡張を設定するためのユーザー権限

プロセス拡張を設定するには、ユーザーの言語設定を取得するためのユーザー権限が必要です。プロセス拡張が失敗した場合は、エラー メッセージをユーザーの現在の言語で表示する必要があります。ユーザーの役割が、現在のユーザーのオブジェクト情報をロードする権限を含めて設定されていない場合、サーバでは、すべてのメッセージがデフォルトのシステム言語で表示されます。

Agile Java クライアントでのカスタム アクションの設定

Agile Java クライアントでは、管理モジュールにカスタム アクションを定義できます。Agile PLM システム設定を構成するには、管理者権限があるユーザーとしてログインする必要があります。

プロセスの拡張ライブラリの使用

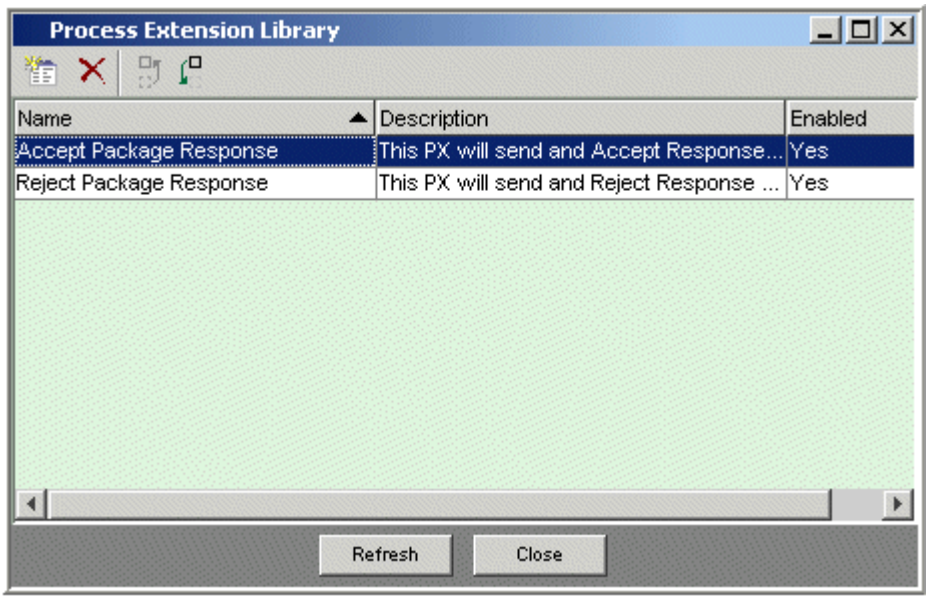
Agile PLM クライアントで使用するカスタム アクションを定義する場所はプロセスの拡張ライブラリです。カスタム アクションをプロセスの拡張ライブラリに追加するときには、そのアクションをクライアントから起動する方法を指定します。

カスタム アクションをプロセスの拡張ライブラリに追加する手順は、次のとおりです。

1. 管理者として Agile Java クライアントへログインします。
2. [管理] タブをクリックします。
3. [データ設定] フォルダを開きます。

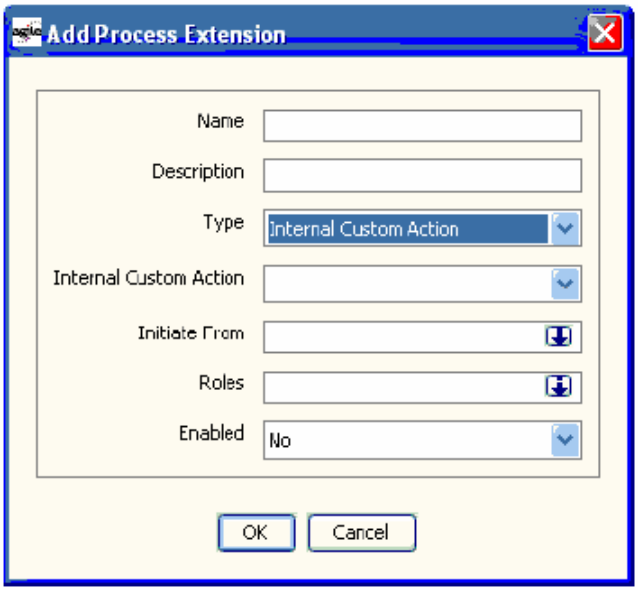
4. [プロセス拡張] ノードを開きます。

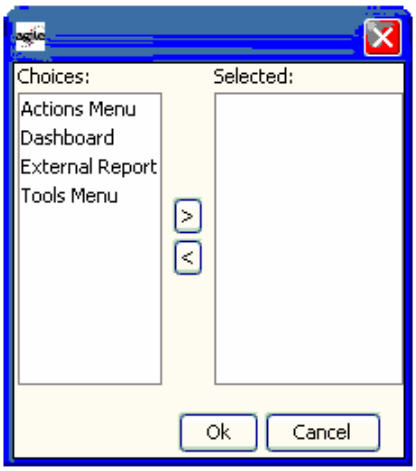
図 13: 拡張ライブラリ



5. [拡張ライブラリ] ウィンドウで、[プロセス拡張の追加] ボタン  をクリックして [プロセス拡張の追加] ダイアログ ボックスを開きます。

図 14: [プロセス拡張の追加] ダイアログ





6. 次の情報を入力します。
   - 名前 - プロセス拡張の名前を入力します。
   - 説明 - プロセス拡張の簡単な説明を入力します。
   - タイプ - [内部カスタム アクション] を選択します。[内部カスタム アクション] フィールドがアクティブになります。

- 内部カスタム アクション - リストからカスタム アクションを選択します。
- 起動先 - プロセス拡張を起動する場所を 1 つ以上選択します。次のオプションから選択してください。
  - [アクション] メニュー - 正しく設定されたクラスの [アクション] メニューからカスタム アクションを選択できます。
  - 外部レポート - 外部リソースまたは URL にアクセスしてレポートを生成できます。プロセス拡張が内部カスタム アクションの場合、[外部レポート] オプションは利用できません。
  - [ツール] メニュー - [ツール] メニューからカスタム アクションを選択できます。
  - ワークフロー ステータス - 正しく設定されたワークフローが特定のステータスに入ったときに、常にカスタム アクションが起動します。

  プロセス拡張を [アクション] メニューまたはワークフロー ステータスから起動するように指定した場合は、そのプロセス拡張を使用するサブクラスまたはワークフローを設定できます。プロセス拡張を使用して外部レポートを作成するように指定した場合は、Agile Web クライアントを使用してレポートを作成できます。プロセス拡張を [ツール] メニューから起動するように指定した場合、そのプロセス拡張は Agile PLM クライアントで常に使用できます。
- 役割 - カスタム アクションに使用する役割を 1 つ以上選択します。現在のユーザーの役割および権限を使用する場合、このフィールドは空白にしておきます。現在のユーザーの役割および権限を一時的に無視する場合には、1 つ以上の役割を選択します。カスタム アクションが完了した時点で、クライアントは現在のユーザーの役割および権限に戻ります。
- 有効 - [はい] または [いいえ] を選択します。

7. [OK] をクリックして新規のプロセス拡張を保存します。

クラスへのプロセス拡張の割り当て

カスタム アクションを Agile PLM のオブジェクト (部品、ECO など) の [アクション] メニューに追加するには、そのオブジェクトのクラスを設定します。各基本クラス、クラスおよびサブクラスには [プロセスの拡張] タブがあります。クラスに割り当てるカスタム アクションは、プロセスの拡張ライブラリで事前に定義しておく必要があります。

プロセス拡張は、クラスおよび基本クラスから継承されます。したがって、プロセス拡張が基本クラスに割り当てられた場合、その基本クラスの下位にあるクラスおよびサブクラスにも割り当てられることになります。

注意 プロセス拡張は、クラス階層のひとつのレベルにのみ割り当てることができます。たとえば、プロセス拡張が [部品] サブクラスに割り当てられている場合、その拡張を [アイテム基本クラス] に割り当てることはできません。

プロセス拡張をクラスに割り当てる手順は、次のとおりです。

1. 管理者として Agile Java クライアントへログインします。
2. [管理] タブをクリックします。
3. [データ設定] フォルダを開きます。
4. [クラス] ノードを開きます。
5. [クラス] ウィンドウで、基本クラス、クラスまたはサブクラスをダブルクリックします。
6. [プロセスの拡張] タブをクリックします。

7. ツールバーの  をクリックします。[プロセス拡張の割り当て] ダイアログ ボックスが表示されます。
8. [選択肢] リストでカスタム アクションを選択し、  をクリックして [選択済み] リストに移動します。完了したら、[OK] をクリックします。
9. [OK] をクリックして設定を保存します。

ワークフロー ステータスへのプロセス拡張の割り当て

「保留中」ステータス以外の各ワークフロー ステータスには、ワークフローがそのステータスに入ったときに起動するカスタム アクションを 1 つ以上割り当てることができます。ワークフロー ステータスに割り当てるカスタム アクションは、プロセスの拡張ライブラリで事前に定義しておく必要があります。

注意 自動転送依頼 (ATO) は、ワークフロー起動のプロセス拡張に対応していません。

プロセス拡張をワークフロー ステータスに割り当てる手順は、次のとおりです。

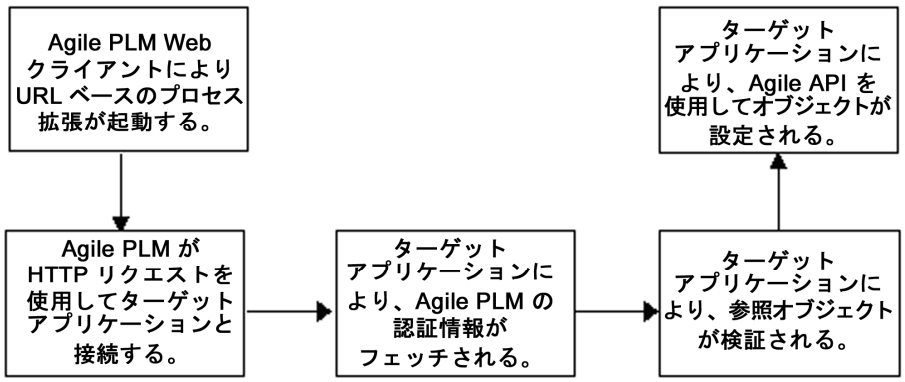
1. 管理者として Agile Java クライアントへログインします。
2. [管理] タブをクリックします。
3. [ワークフロー設定] フォルダを開きます。
4. [ワークフロー] ノードを開きます。
5. [ワークフロー] ウィンドウで、ワークフローをダブルクリックします。
6. [ステータス] タブをクリックします。
7. 「保留中」以外のステータスを選択します。ステータス テーブルの下に表示される [ワークフロー条件] プロパティ テーブルが、選択したステータスによって更新されます。
8. [ワークフロー条件] プロパティ テーブルで、選択したステータスをダブルクリックします。
9. [プロセスの拡張] リストで  をクリックします。ポップアップ ウィンドウが表示されます。
10. [選択肢] リストでカスタム アクションを選択し、  をクリックして [選択済み] リストに移動します。完了したら、[OK] をクリックします。
11. [保存] をクリックして設定を保存します。

URL ベースのプロセス拡張の定義

Agile Web クライアントで URL ベースのプロセス拡張を使用すると、Web クライアントから外部アプリケーションにアクセスできます。URL を参照するプロセス拡張を Agile PLM Web クライアントが起動すると、Web ページが新しいブラウザ ウィンドウに表示されます。

URL ベースのプロセス拡張には、どのようなタイプの Web ベースのアプリケーションが使用できるのでしょうか。前述したように、制限はほとんどありません。Agile PLM オブジェクトに対してビジネス ルール検証を実行し、その結果に従ってオブジェクトを更新する Web ベースのアプリケーションはその一例です。次の図は、このようなプロセス拡張のプログラム フローを示しています。

図 1: URL ベースのプロセス拡張のプロセス フロー例



URL ベースのプロセス拡張は、Web ベースのレポート エンジンの参照にも使用できます。URL ベースのプロセス拡張を使用する外部レポートを作成するには、Agile Web クライアントで [作成] > [レポート] > [外部] の順に選択します。詳細は、319 ページの「外部レポートの作成」を参照してください。

URL ベースのプロセス拡張は、ワークフロー ステータスの変更によって起動することはできません。これは、ステータスの変更が発生したときに Agile PLM クライアントがアクティブにならないためです。

注意 URL ベースのプロセス拡張は、ソーシング プロジェクト (`IProject`) ではサポートされていません。

URL ベースのプロセス拡張を定義する手順は、次のとおりです。

1. 管理者として Agile Java クライアントへログインします。
2. [管理] タブをクリックします。
3. [データ設定] フォルダを開きます。
4. [プロセス拡張] ノードを開きます。
5. プロセスの [拡張ライブラリ] ウィンドウで  をクリックします。[プロセス拡張の追加] ダイアログ ボックスが表示されます。
6. 次の情報を入力します。
   - 名前 - プロセス拡張の名前を入力します。
   - 説明 - プロセス拡張の簡単な説明を入力します。
   - タイプ - [URL] を選択します。
   - アドレス - Web ページのアドレスを指定します。プロトコルも含めて完全な URL を指定する必要があります。たとえば、Agileの Web サイトを指定するには、www.agile.com ではなく、http://www.agile.com と入力します。
   - 起動先 - Web ページを起動する場所を 1 つ以上選択します。次のオプションから選択してください。

- [アクション] メニュー - 正しく設定されたクラスの [アクション] メニューから Web ページを選択できます。
- ダッシュボード - 第 20 章「ダッシュボード管理拡張の開発」を参照してください。
- 外部レポート - Web ページにアクセスしてレポートを生成できます。
- [ツール] メニュー - [ツール] メニューから Web ページを選択できます。

プロセス拡張を [アクション] メニューから起動するように指定した場合は、そのプロセス拡張を使用するサブクラスを設定できます。プロセス拡張を使用して外部レポートを作成するように指定した場合は、Agile Web クライアントを使用してレポートを作成できます。プロセス拡張を [ツール] メニューから起動するように指定した場合、そのプロセス拡張は Agile PLM クライアントで常に使用できます。

- 有効 - [はい] または [いいえ] を選択します。

7. [OK] をクリックして新規のプロセス拡張を保存します。

エンコードされた Agile PLM 情報を他のアプリケーションに渡す場合

Agile SDK 9.2.2 では、パスワードで保護された外部アプリケーション サーバを介するシングル サインオンはサポートされていません。

重要 Agile Web クライアントは、エンコードされたユーザーのアカウント情報を継承できます。このユーザーのアカウント情報は、PX アプリケーションで Agile SDK を使用するときに SDK で再利用できます。外部アプリケーション サーバに対してパスワードで保護されたアクセスを実行する場合は、外部サーブレットにアクセスするためのユーザー名とパスワードをコード内にハードコードする必要があります。

URL ベースのプロセス拡張をオブジェクトの [アクション] メニューから起動する場合、そのオブジェクトの結合キーとクラス ID、および現在のユーザー名は、GET メソッドを使用して URL にエンコードされます。クライアントは、データを ID= 値のペアとしてエンコードし、このペアを URL の最後に追加します。次の例に示すように、各 ID には接頭辞「agile」が付きます。

```
http://www.acoolwebsite.com/?agile.username=wangsh&agile.classId=1
0141&agile.1001=
➥1000-02&agile.1014=A&agile.siteName=Taipei
```

注意 [アクション] メニューとは異なり、[ツール] メニューのコマンドに関連付けられた Agile PLM オブジェクトはありません。したがって、URL ベースのプロセス拡張を [ツール] メニューから起動した場合、エンコードされたオブジェクト データは URL に追加されません。

URL ベースのプロセス拡張の URL にエンコードされた情報に加えて、暗号化されたユーザー名と関連するパスワードがそれぞれ j_username と j_password クッキーから使用できます。これらのクッキーは、次の条件を満たす場合に、ターゲット システムに自動的に渡されます。

- ユーザーが URL ベースのプロセス拡張を Agile Web クライアントから起動する場合

注意 Web アプリケーションは agile.properties の cookie.domain プロパティで指定したドメインに存在している必要があります。そうでない場合、セキュリティ クッキーは継承されません。

- ターゲット システムでクッキーを受け取ることが許可されている場合
- ターゲット システムが Agile PLM システムと同じドメイン内にある場合

注意 ターゲット システムが企業のファイヤウォールの外側にある場合、そのシステムは、SSL を使用するセキュアな Web サーバである必要があります。

ターゲット システムからの Agile PLM セッションの作成

Agile Web クライアントで起動した URL ベースのプロセス拡張から受信した HTTP リクエストの認証情報を使用することによって、ターゲット アプリケーションは Agile API を使用して IAgileSession を作成できます。さらに、Agile API クライアントは、HTTP リクエストで参照される Agile PLM オブジェクトを取得して設定できます。

ユーザーが Agile Web クライアントにログインすると、認証プロセスによって、そのユーザーの暗号化されたユーザー名とパスワードが保存されているサーバ コンピュータ上にクッキーのペア (j_username と j_password) が作成されます。Agile Web クライアントから URL ベースのプロセス拡張を起動すると、ターゲット システムは、クッキーを使用して Agile PLM セッションを作成できます。実際には、ターゲット システム上の Agile Web クライアントと Agile API クライアントは、シングル サインオンを共有できます。

注意 Agile Java クライアントでは、Web クライアントと異なり、クライアント側のクッキーが作成されません。したがって、プロセス拡張のシングル サインオン機能はサポートされません。

クッキーは、同じドメイン内のコンピュータ間で共有するように設計されています。たとえば、Agile PLM のインストール中にドメインを「.agile.agilesoft.com」に設定すると、「.agile.agilesoft.com」で終わるすべてのコンピュータが j_username と j_password クッキーを使用できます。

クッキーの詳細は、次の Web サイトを参照してください。
http://wp.netscape.com/newsref/std/cookie_spec.html

次の例は、Agile API を使用して、HTTP サーブレット リクエストからクッキー情報を抽出し、その情報を使用して IAgileSession を生成する方法を示しています。AgileSessionFactory.PX_REQUEST フィールドの値はセッション作成時に使用されるキーで、サーブレット リクエストと同じになるように設定されます。

例: PX_REQUEST フィールドの使用による、サーブレット リクエストからの IAgileSession の作成

```
private IAgileSession connect(HttpServletRequest request) throws
ServletException {
  factory =
AgileSessionFactory.getInstance("http://agileserver/Agile");
  HashMap params = new HashMap();
  params.put(AgileSessionFactory.PX_REQUEST, request);
  session = factory.createSession(params);
  return session;
}
```

ターゲット アプリケーションがサーブレットベースでない場合は、別の方法でクッキー情報を使用してセッションを作成します。AgileSessionFactory.PX_REQUEST を使用せずに、AgileSessionFactory.PX_USERNAME と AgileSessionFactory.PX_PASSWORD フィールドを HashMap のキーとして使用できます。これらのフィールドの値は、それぞれ j_username と j_password クッキーの値である必要があります。

例: PX_USERNAME および PX_PASSWORD フィールドの使用による IAgileSession の作成

```
private IAgileSession connect(Cookie[] cookies) throws Exception {
  factory =
AgileSessionFactory.getInstance("http://agileserver/Agile");
  HashMap params = new HashMap();
  String username = null;
  String pwd = null;
```

```
  for (int i = 0; i < cookies.length; i++) {
    if (cookies[i].getName().equals("j_username"))
      username = cookies[i].getValue();
    else if (cookies[i].getName().equals("j_password"))
      pwd = cookies[i].getValue();
  }
  params.put(AgileSessionFactory.PX_USERNAME, username);
  params.put(AgileSessionFactory.PX_PASSWORD, pwd);
  session = factory.createSession(params);

  return session;
}
```

HTTP リクエストからの Agile PLM オブジェクトの取得

オブジェクトの [アクション] メニューから URL ベースのプロセス拡張を起動する場合は、ターゲット アプリケーションで Agile PLM オブジェクトを取得して変更できます。オブジェクトの結合キーとクラス ID は、GET メソッドを使用して URL にエンコードされます。ターゲット アプリケーションで参照先の IAgileObject を簡単に取得するために、Agile API では、次の例に示すように、オーバーロードされた IAgileSession.getObject() メソッドを使用できます。SDK では、リクエストからオブジェクト ID 情報を抽出し、その情報を使用して指定のオブジェクトを取得します。

例: HTTP リクエストからの Agile PLM オブジェクトの取得

```
private IAgileObject getAgileObject(HttpServletRequest request)
throws ServletException {
  IAgileObject obj = session.getObject(null, request);
  return obj;
}
```

ターゲット アプリケーションがサーブレットベースでない場合は、通常の IAgileSession.getObject() メソッドを使用して参照先のオブジェクトを取得できます。getObject() の params パラメータには、オブジェクトのクラスの必須属性がすべて含まれた HashMap を指定します。必要な属性/値のペアは、エンコードされた URL に含まれます。各 Agile PLM クラスの識別属性のリストは、次のセクションを参照してください。

Agile PLM クラスの識別属性

各 Agile PLM クラスには、異なる一連の識別属性があり、エンコードされた URL にパラメータとして渡すことができます。たとえば、変更オブジェクトは、クラス ID と [カバー ページ.番号] 属性を渡します。次の表に、各 Agile PLM クラスの識別属性を示します。

クラス	パラメータ	説明
変更	agile.classID	選択したオブジェクトのクラス ID
	agile.1047	カバー ページ.番号
顧客	agile.classID	選択したオブジェクトのクラス ID
	agile.5110	一般情報.顧客番号
部品分類	agile.classID	選択したオブジェクトのクラス ID
	agile.agile.2000004284	タイトル ブロック.名前

クラス	パラメータ	説明
デクラレーション	agile.classID	選択したオブジェクトのクラス ID
	agile.agile.2000002615	タイトル ブロック.名前
ディスカッション	agile.classID	選択したオブジェクトのクラス ID
	agile.18417	カバー ページ.番号
ファイル フォルダ	agile.classID	選択したオブジェクトのクラス ID
	agile.6173	タイトル ブロック.番号
	agile.7951	タイトル ブロック.バージョン
アイテム	agile.classID	選択したオブジェクトのクラス ID
	agile.1001	タイトル ブロック.番号
	agile.1014	タイトル ブロック.リビジョン
	agile.siteName	拠点名 - [すべて] が選択されている場合、このパラメータは省略されます。
製造元部品	agile.classID	選択したオブジェクトのクラス ID
	agile.1647	一般情報.製造元名
	agile.1648	一般情報.製造元部品番号
製造元	agile.classID	選択したオブジェクトのクラス ID
	agile.1754	一般情報.製造元名
パッケージ	agile.classID	選択したオブジェクトのクラス ID
	agile.3110	カバー ページ.パッケージ番号
価格	agile.classID	選択したオブジェクトのクラス ID
	agile.10355	一般情報.番号
	agile.10357	一般情報.リビジョン
プログラム	agile.classID	選択したオブジェクトのクラス ID
	agile.18041	一般情報.番号
プロジェクト	agile.classID	選択したオブジェクトのクラス ID
	agile.14824	一般情報.番号
PSR	agile.classID	選択したオブジェクトのクラス ID
	agile.4856	カバー ページ.番号
QCR	agile.classID	選択したオブジェクトのクラス ID
	agile.4029	カバー ページ.QCR 番号

クラス	パラメータ	説明
レポート 1	agile.classID	選択したオブジェクトのクラス ID
	agile.8071	一般情報.名前
RFQ	agile.classID	選択したオブジェクトのクラス ID
	agile.13925	カバー ページ.RFQ 番号
見積依頼回答	agile.classID	選択したオブジェクトのクラス ID
	agile.14472	カバー ページ.RFQ 番号
	agile.14452	カバー ページ.サプライヤ名
拠点	agile.classID	選択したオブジェクトのクラス ID
	agile.11882	一般情報.名前
含有基準	agile.classID	選択したオブジェクトのクラス ID
	agile.2000001969	タイトル ブロック.名前
サブスタンス	agile.classID	選択したオブジェクトのクラス ID
	agile.2000001124	タイトル ブロック.名前
サプライヤ	agile.classID	選択したオブジェクトのクラス ID
	agile.17761	一般情報.番号
転送	agile.classID	選択したオブジェクトのクラス ID
	agile.12673	カバー ページ.転送番号
ユーザー	agile.classID	選択したオブジェクトのクラス ID
	agile.11617	一般情報.ユーザー名
ユーザー グループ	agile.classID	選択したオブジェクトのクラス ID
	agile.12077	一般情報.名前

注意 プロセス拡張フレームワークでは URL にレポート情報をエンコードできますが、Agile API では、レポート オブジェクトはサポートされていません。したがって、Agile API を使用して、URL で参照されるレポート オブジェクトを取得することはできません。

SDK ネットワーク クラスローダと Weblogic Server の操作の設定

SDK ベースのサーブレットを Weblogic Server に配置した場合は、SDK クラスローダを目的に従って設定しないかぎり、URL のプロセス拡張は操作に失敗します。

Weblogic Server に対して SDK ネットワーク クラスローダを設定する手順は、次のとおりです。

1. Application.ear ファイルから次のファイルを抽出します。
   - `AgileAPI.jar`
   - `agileclasses.jar`

- fsclient.jar
- sdk.jar

注意 これらのファイルは、<AgileHOME>¥agileDomain¥SERVERNAME-AgileServer¥.wlnotdelete¥Application_SERVERNAME-AgileServer¥APP-INF¥lib フォルダにあります。

- wlsuth.jar

注意 このファイルは、カスタム アプリケーションが Agile サーバにリモートでアクセスする場合のみ必要です。このファイルは <AgileHOME>¥agileDomain¥lib にあります。

2. 最初の 4 つのファイルをアプリケーション ライブラリ ディレクトリ (.war ファイル) にパッケージ化します。
3. Weblogic Application Server を停止します。
4. システムの一時ディレクトリ内で AgileSDK.cache フォルダを検索し、検出した場合は削除します。
5. サーバを再起動します。

注意 Web アプリケーションでアプリケーションをリモート サーバに対して実行している場合は、Weblogic JVM の classpath に wlsauth.jar を追加する必要があります。

外部レポートの作成

Agile Web クライアントで、外部リソースまたは URL に接続して外部レポートを生成できます。外部レポートを作成する前に、レポートに関連付けられた URL をプロセスの拡張ライブラリに追加する必要があります。詳細は、前述の「URL ベースのプロセス拡張の定義」を参照してください。

Agile Web クライアントでレポートを作成するには、レポートの作成権限が必要です。

外部レポートを作成する手順は、次のとおりです。

1. Agile Web クライアントにログインします。

注意 Agile Java クライアントでは外部レポートを作成できません。

2. [作成] > [レポート] > [外部] の順に選択します。[レポート作成ウィザード] が表示されます。
3. レポートの名前を入力します。[次へ] をクリックします。
4. 次の一般情報を入力します。
   - 説明 - レポートの簡単な説明を入力します。
   - プロセスの拡張 - プロセス拡張を選択します。選択したプロセス拡張は、URL (Web ベース レポート エンジンの場所など) に関連付けられます。
   - フォルダ – レポートの親フォルダを選択します。
5. [完了] をクリックします。

クラスタ環境でのプロセス拡張の配置

Agile PLM インストーラがアプリケーション サーバ クラスタ内のサーバで実行されていない場合は、そのサーバに、/agile_home/integration/sdk/extensions フォルダは存在しません。存在しない場合は、フォルダを手動で作成し、プロセス拡張の JAR ファイルをそのフォルダにコピーする必要があります。

プロセス拡張の配置フォルダを手動で作成する手順は、次のとおりです。

1. 次のフォルダをクラスタ内のすべてのアプリケーション サーバ上に作成します (存在しない場合)。
   /agile_home/integration/sdk/extensions
2. プロセス拡張の全 JAR ファイルを、クラスタ内の各サーバの
   /agile_home/integration/sdk/extensions フォルダに配置します。

プロセス拡張に関するよくある質問

このセクションでは、プロセス拡張に関する一般的な質問について回答します。

プロセス拡張とは、どのようなものですか。

カスタム アクション、外部レポート、カスタム自動採番およびツールを介して Agile PLM クライアントの機能を拡張し、顧客の業務にあわせてシステムを調整できます。プロセス拡張を使用すると、Agile PLM サーバと Agile PLM ユーザーは外部システムに接続できます。

プロセス拡張を使用すると、どのようなタイプのアクションを定義できますか。

プロセス拡張では、2 つのタイプのプロセス拡張アクションがサポートされています。それは、カスタム自動採番ソースとカスタム アクションです。カスタム自動採番ソースは、オブジェクトのクラスで使用する連続番号を定義します。カスタム アクションは、Agile PLM クライアントから実行できるプログラムです。

プロセス拡張は、URL への参照である場合もあります。URL は、単なる Web サイトでも、Web ベースのアプリケーションの場所でも構いません。

プロセス拡張は、非同期処理をサポートできますか。

Agile のプロセス拡張でサポートしているのは、同期処理のみです。プロセス拡張で非同期処理が必要な場合は、PX コードを変更して、選択した非同期ソリューションを実装する必要があります。たとえば、スレッドを起動することができます。

Agile の Java API をプロセス拡張プログラム内で使用できますか。

はい。Agile の Java API および他の外部 Java API を使用できます。唯一の要件は、拡張のタイプに応じて、ICustomAutoNumber または ICustomAction インターフェースのいずれかを実装することです。

プロセス拡張は Agile PLM クライアントでどのように起動するのですか。

カスタム アクションは、次の方法で起動できます。

- ワークフロー ステータスの変更
- [ツール] メニューからのカスタム アクションの選択

- オブジェクトの [アクション] メニューからのカスタム アクションの選択
- カスタム アクションを使用する外部レポートの選択
- カスタム自動採番ソースを使用するクラスのオブジェクトの作成

プロセス拡張には、特別なセキュリティ要件がありますか。

いいえ。プロセス拡張のスタックは Agile アプリケーション サーバ上にあるため、カスタム アクションとカスタム自動採番ソースは、認証済のユーザーに権限がある環境内で実行されます。

カスタム アクションには、どのように役割と権限を定義するのですか。

デフォルトでは、カスタム アクションでは現在のユーザーの役割と権限を使用します。ただし、拡張した権限を持つようにカスタム アクションを設定する場合は、カスタム アクションに必要な役割を Agile Java クライアントのプロセスの拡張ライブラリに指定できます。Agile PLM クライアントでカスタム アクションを使用するときは、そのカスタム アクションに対して指定した役割と権限が、現在のユーザーの役割と権限より優先されます。そのカスタム アクションが完了した時点で、クライアントはユーザーの元の役割と権限に戻ります。

プロセス拡張は、どのように設定して配置するのですか。

アプリケーション サーバの agile_home/integration/sdk/extensions フォルダに、プロセス拡張の JAR ファイルを格納します。この JAR ファイルには、META-INF/services ディレクトリ内に com.agile.px.ICustomAutoNumber または com.agile.px.ICustomAction という名前のファイルが含まれている必要があります。ファイルの内容は、カスタム自動採番ソースまたはカスタム アクション用の Java の完全修飾クラス名で、1 行に 1 クラスずつリストされています。

プロセス拡張プログラムをアプリケーション サーバに配置した後は、どのようにプロセス拡張を有効にするのですか。

配置した後のプロセス拡張は、Agile PLM クライアント内で使用するように設定できます。Agile Java クライアントでは、カスタム アクションをプロセスの拡張ライブラリに追加し、カスタム自動採番を [自動採番] テーブルに追加できます。

カスタム アクションまたはカスタム自動採番ソースの JAR ファイルを配置した後に、アプリケーション サーバのクラスパスを更新する必要がありますか。

いいえ。クラスパスは専用のクラスローダによって自動的に更新されます。クラスローダは、agile_home/integration/sdk/extensions (または agile.properties ファイルの sdk.extensions プロパティで指定した場所) にあるクラスを使用して、アプリケーション サーバのクラスパスを拡張します。

カスタム自動採番ソースは、どのように作成するのですか。

com.agile.px パッケージのサーバ側 API である ICustomAutoNumber インターフェースを実装する Java クラスを作成します。コードでは、自動採番のロジック (接頭辞、接尾辞、桁数など) および永続性メカニズムを定義します。Agile PLM システムでは、getAutoNumber() メソッドを呼び出してカスタム自動採番ソースから次の番号を取得します。

Agile Java クライアントでは、カスタム自動採番ソースをどのように割り当てるのですか。

[クラス] ノードで、自動採番ソースを特定のサブクラスに割り当てます。[自動採番] ノードでは、サブクラスを自動採番ソースに割り当てることもできます。

カスタム アクションは、どのように作成するのですか。

com.agile.px パッケージのサーバ側 API である ICustomAction インターフェースを実装する Java クラスを作成します。コードでは、カスタム アクションを定義し、現在のオブジェクトを変更するか、外部レポートを作成するか、Agile PLM クライアントを外部システムと統合するか、またはその他のビジネス ロジックを実行するかを定義します。Agile PLM クライアントでカスタム アクションを起動するとき、Agile PLM システムは doAction() メソッドを呼び出します。

カスタム アクションを [ツール] メニュー、[アクション] メニュー、ワークフロー ステータスおよび外部レポートにどのように関連付けるのですか。

Agile Java クライアントで [プロセス拡張] ノードを開き、カスタム アクションを追加して設定します。カスタム アクションは、ワークフロー ステータス、[ツール] メニュー、クラスの [アクション] メニュー、および外部レポートに関連付けることができます。ワークフロー ステータスに関連付けられたカスタム アクションは、ワークフローがそのステータスに入ると自動的に起動します。[起動先] プロパティが [[ツール] メニュー] に設定されている場合、カスタム アクションは [ツール] メニューに表示されます。カスタム アクションをサブクラスの [プロセスの拡張] タブに追加すると、そのカスタム アクションはオブジェクトの [アクション] メニューに表示されます。外部レポートに関連付けられたカスタム アクションは、そのレポートが実行されると自動的に起動します。

プロセス拡張は、Agile PLM クライアントの [ツール] メニューまたは [アクション] メニューにどのような順序で表示されますか。

プロセス拡張を [ツール] メニューまたはオブジェクトの [アクション] メニューに追加すると、標準のメニュー コマンドの後に作成順に表示されます。[ツール] メニューまたは [アクション] メニューのコマンドに対して、並べ替えなどの管理操作は実行できません。

クラスに割り当てられたカスタム アクションの継承モデルとは、どのようなものですか。

カスタム アクションは、基本クラス レベル、クラス レベルまたはサブクラス レベルで定義できます。基本クラス レベルで定義されたカスタム アクションは、基本クラスの下位にあるすべてのクラスおよびサブクラスで使用できます。サブクラス レベルで定義されたカスタム アクションは、そのサブクラスでのみ使用できます。

PX および WSX 設定プロパティ ファイルはどこに配置するのですか。

Agile PLM リリース 9.2.2.2 での配置変更によって、agileDomain¥config ディレクトリはクラスパスに含まれません。PX および WSX プロパティ ファイルは、次のディレクトリに配置できます。
¥oas¥j2ee¥home¥applications¥Agile¥APP-INF¥classes¥

第 20 章

Web サービス拡張の開発

扱うトピックは次のとおりです。

Web サービス拡張について

Web サービス拡張 (WSX) は、内部および外部の異種システムと Agile PLM の間の通信を可能にする Web サービス エンジンです。これらの異種システムには、Enterprise Resource Planning (ERP) システム、Customer Resource Management (CRM) システム、Business-to-Business Integration (B2Bi) システム、他の Agile PLM システムおよびサプライ チェーン パートナーが含まれます。WSX によって、新製品投入 (NPI)、製品変更および製造リソースの急速な増加に対応するプロセスを簡素化することができます。また、未加工の製品コンテンツを集計し、重要な製品コンテンツを他のコア システムに対してリアルタイムで提供するプロセスを簡素化することもできます。WSX には、Agile PLM の新しい Web サービスを開発するためのツールとフレームワークが含まれています。

WSX を使用すると、次の内容を実行できます。

- Enterprise Application Integration (EAI) システムに対して製品コンテンツを提供します。これによって、多岐にわたる内部アプリケーションにデータを供給できるようになります。
- 製品コンテンツを、製品設計、製造計画、製造現場、Enterprise Resource Planning (ERP) および Customer Relationship Management (CRM) の各アプリケーションと共有します。
- 製品コンテンツを Business-to-Business (B2B) システムに提供します。これによって、Agile アプリケーション サーバのデータを、広範囲にわたる外部アプリケーションに企業の境界を超えて転送できます。
- 交換、レポートおよびカスタム アプリケーションに対してコンテンツを提供し、ERP および他のサプライ チェーン アプリケーションから製品コンテンツ データをインポートします。

注意 Agile Integration Services (AIS) は、単独のライセンス製品です。AIS は、WSX 技術を使用して構築された一連の Web サービスで、プログラムによるインポートとエクスポートの機能を Agile PLM システムに提供します。AIS の詳細は、『Agile Integration Services Developer Guide』を参照してください。

主な機能

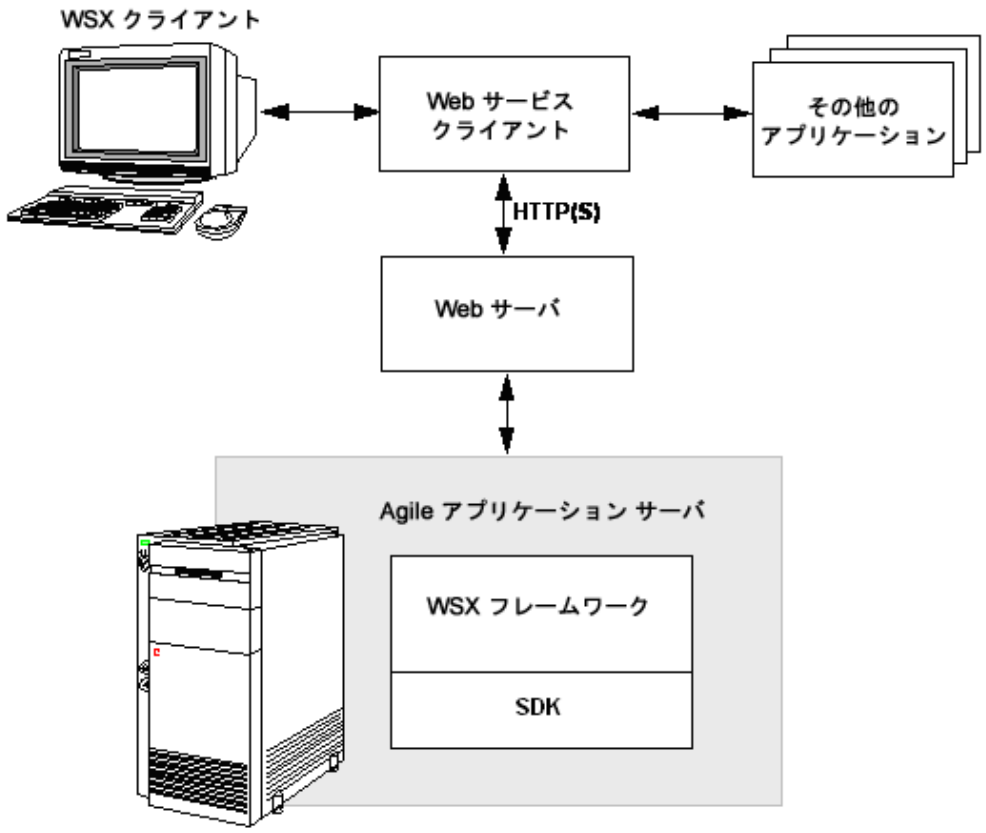
WSX の主な機能は、次のとおりです。

- プログラムによるデータ アクセス - Agile PLM システムおよび他のデータ リソースに格納されているデータへのプログラムによるアクセスを提供します。これによって、開発者は、コンテンツの転送を自動化するためのカスタム アプリケーションを作成できます。
- アクセスの容易性 - 標準的な HTTP(S) 技術を使用して、企業のファイヤウォール外部にある Agile PLM 製品コンテンツに関するアクセスの容易性を提供します。
- 複数のプログラミング言語のサポート - Simple Object Access Protocol (SOAP) または Web Services Description Language (WSDL)、あるいはその両方を作成および理解できるあらゆる言語をサポートしています。
- 複数の出力形式のサポート - aXML および PDX 1.0 をサポートしています。XSL を使用して、XML データを任意の形式に変換したり、データを任意の形式で返す Web サービスを開発することもできます。
- セキュリティ - インターネット標準の通信プロトコルとセキュリティ プロトコル (HTTP と SSL) を使用して、XML に準拠したアプリケーションと通信します。したがって、インターフェースはファイヤウォール フレンドリで安全です。

WSX アーキテクチャ

Agile PLM および WSX フレームワークに接続するには、Web サービスの標準的な起動方法を使用します。

図 15: WSX アーキテクチャ



Web サービスについて

Web サービスは、分散アプリケーションを構築するための技術です。インターネットを介して使用できるこれらのサービスでは、標準化された XML メッセージング システムが使用されるため、1 つのオペレーティング システムまたはプログラミング言語に制限されることはありません。Web サービスによって、企業では、既存のビジネス プロセスをカプセル化してサービスとして公表し、他のサービスを検索してサブスクライブし、企業全体または企業の境界を超えて情報を交換できます。Web サービスは、構造化されたデータ交換、メッセージング、サービスのディスカバリ、インターフェース記述およびビジネス プロセス設計の仕様に関する一般的な合意に基づいています。

Web サービスでは、インターネットを介してリモート プロシージャによる呼び出しが実行されます。また、HTTP(S) または他のプロトコルを使用してリクエストとレスポンスを転送し、Simple Object Access Protocol (SOAP) を使用してリクエストとレスポンスの情報を通信します。

Web サービスの主なメリットは、次のとおりです。

- サービス指向のアーキテクチャ - Web サービスは、パッケージ製品とは異なり、あらゆるプラットフォームからアクセスできる一連のサービスとして配信できます。複数のコンポーネントはそれぞれが単独で存在します。公開が必要なのは、ビジネスレベルのサービスのみです。
- 相互運用性 - システム間の完全な相互運用性を保証します。
- 統合 - 特に、異なるプラットフォーム上のアプリケーションまたは異なる言語で記述されているアプリケーションを接続する場合は、柔軟な統合ソリューションを提供します。
- モジュール方式 - プログラミングにモジュール的アプローチを提供します。アプリケーションの各ビジネス機能を個別の Web サービスとして公開できます。モジュールを小型化することでエラーが削減され、より多くのコンポーネントが再利用可能になります。
- アクセスの容易性 - ビジネス サービスを完全に分散化できます。これらのビジネス サービスはインターネット経由で配信されるため、多様な通信機器でアクセスできます。
- 効率性 - 内部での使用が目的のアプリケーションから構築した Web サービスを、コード変更なしで、外部向けに使用できます。Web サービスは人間が解読可能な形式で宣言および実装されるため、Web サービスを使用した増分開発は比較的簡単です。

あらゆる技術と同様に、Web サービスにも若干の制限があります。Web サービスを開発する際は、次の点を考慮する必要があります。

- SOAP は、転送媒体を介してデータとリクエストを処理するための簡単なメカニズムです。分散ガーベッジ コレクト、オブジェクトのアクティブ化、参照による呼び出しなど、高度な操作を処理するようには設計されていません。
- Web サービスはネットワークを基盤としているため、ネットワーク トラフィックの影響を受けます。Web サービスを起動する際の待ち時間は、多くの場合、数百ミリ秒になります。したがって、このサービスで提供される機能の価値は、長い起動待ち時間を正当化できるほど十分に有意であることが必要です。
- Web サービスは、会話型プログラミングには適していません。したがって、公開するサービスを設計するときは、可能なかぎりその独立性を確保するように努める必要があります。

Web サービス アーキテクチャ

Web サービス アーキテクチャは、役割とプロトコル スタックの観点で考えることができます。

- Web サービスの役割:
  - サービス プロバイダ - サービスを実装しインターネット上で使用可能にすることでサービスを提供します。
  - サービス リクエスタ - サービスのユーザーです。ネットワーク接続を開いて XML リクエストを送信することでサービスにアクセスします。
  - サービス レジストリ - サービスの集中ディレクトリです。開発者は、検出した既存のサービスに関する新しいサービスをこのディレクトリで公表できます。
- Web サービスのプロトコル スタック:
  - サービス転送レイヤー - HTTP を使用して、アプリケーション間でメッセージを転送します。その他の転送は、AIS の将来のリリースでサポートされます。
  - XML メッセージ レイヤー - SOAP を使用して、メッセージを XML 形式にエンコードします。SOAP は、プラットフォームに依存しない XML プロトコルであり、コンピュータ間での情報交換に使用されます。転送するカプセル化されたデータのエンベロープ仕様、データ エンコード ルールおよび RPC 規則を定義します。

- サービス記述レイヤー - Web Service Description Language (WSDL) プロトコルを使用して、特定の Web サービスへのパブリック インターフェースを記述します。WSDL は、ネットワーク サービスを、メッセージを交換できる通信エンドポイントの集合として記述するための XML 構文を定義します。これらのメッセージには、ドキュメント指向またはプロシージャ指向の情報が記述されています。操作とメッセージは抽象的に記述されてから、ネットワーク プロトコルとメッセージ形式にそれぞれバインドされます。WSDL を使用すると、通信に使用するメッセージ形式やネットワーク プロトコルに関係なく、エンドポイントとそのメッセージを記述できます。WSDL ドキュメントは、サービスをネットワーク エンドポイント (ポートと呼ばれる) の集合として定義します。ポートは、ネットワーク アドレスを再利用可能なバインディングに関連付けることで定義され、複数ポートの集合で 1 つのサービスを定義します。
- サービス ディスカバリ レイヤー - Universal Description, Discovery, and Integration (UDDI) プロトコルを使用して、複数のサービスを共通レジストリに集中化します。

注意 WSX は、現在、UDDI または他のサービス ディスカバリ レイヤーをサポートしていません。

セキュリティ

WSX は、インターネット標準の通信プロトコルとセキュリティ プロトコル (HTTP と SSL) を使用して、XML に準拠したアプリケーションと通信します。WSX とそのクライアント間の (Web サーバを介した) 通信は、セキュア ソケット レイヤー (SSL) とサーバ側の証明書を介して暗号化できるため、認証、プライバシおよびメッセージ整合性が提供されます。標準的な Java 暗号化ライブラリを使用して、ファイルの暗号化と復号化、セキュリティ キーの作成、ファイルへのデジタル署名の作成およびデジタル署名の検証を実行できます。

Web サービス拡張フレームワークによって、ファイヤウォール外部から受信した起動リクエストの安全性が確保されます。つまり、WSX への外部リクエストはすべて、HTTPS または同等のプロトコルを使用して保護されます。WSX への内部リクエストは、セキュリティなしで (つまり、HTTP を使用して) 実行できます。

Web サービスを起動する際は、ユーザー名とパスワードによるセキュリティを実施する複数の方法があります。Agile API を使用して Web サービスを開発する場合は、他の API プログラムの場合と同様に、`createSession()` のパラメータにユーザー名とパスワードを指定できます。

Java のセキュリティと暗号化のサポートに関する詳細は、
http://java.sun.com/j2se/1.3/docs/guide/security/index.html を参照してください。

ツール

Web サービスへのアクセスに必要なツール セットは複数あります。選択するツールは、クライアントの開発に使用する環境によって大きく異なります。基本的には、XML および HTTP リクエスト/レスポンス メッセージを生成して処理できるツールが必要になります。

WSX フレームワークは、SOAP プロセッサである Apache eXtensible Interaction System (AXIS) に基づいています。ただし、ソース言語に関係なく SOAP ツールの他の実装を使用して Web サービス クライアントを構築できます。

注意 Agile SDK には、AXIS の使用方法を示す WSX Java サンプルが付属しています。AXIS とその機能および使用方法の詳細は、AXIS の Web サイトを参照してください。http://xml.apache.org/axis

Web サービスに関する追加情報の検索

次に、検討する Web サイトの一部を示します。

- WebServices.Org - http://www.webservices.org/
- Web Services Architect - http://www.webservicesarchitect.com/
- Web Services Journal - http://www.sys-con.com/webservices/
- webservices.xml.com - http://webservices.xml.com/
- O'Reilly Web Services - http://webservices.oreilly.com/
- Apache Axis - http://ws.apache.org/axis/
- Java Web Services Developer Pack 1.1 - http://java.sun.com/webservices/webservicespack/html
- Sun ONE Web Services Platform Developer Edition - http://sunonedev.sun.com/building/development/developer_platform_overview.html
- Microsoft .Net Framework - http://msdn.microsoft.com/netframework/
- SOAP::Lite for Perl - http://www.soaplite.com
- Soap Tutorial - http://www.w3schools.com/soap/default.asp

Web サービスの開発および配置

独自の Web サービスを記述する作業は、数段階の手順で構成される簡単なタスクです。

1. Web サービスのエントリ ポイントを定義します。Web サービスのエントリ ポイント (または操作) は、Java クラスのパブリック メソッドに対応しています。
2. Web サービス操作のロジックをコーディングします。Web サービス操作のロジックをコーディングする際に従う必要がある特別なルールはありません。Agile が提供するライブラリ (Agile API を含む) に加え、サード パーティのコード ライブラリを活用することもできます。
3. Java コードを通常と同様にコンパイルします。
4. コンパイルした JAR ファイルを Agile アプリケーション サーバ コンピュータの AGILE_HOME¥integration¥sdk¥extensions にコピーします。Web サービスのデプロイメント ディスクリプタは、META-INF/services/com.agile.wsx.Deployment.wsdd という名前のファイルにある JAR ファイルにも必要です。

注意	クラスタ環境に複数のアプリケーション サーバがある場合は、クラスタ内の各サーバに、Web サービス ファイルを配置する必要があります。

Agile アプリケーション サーバによって、デプロイメント ディスクリプタにリストされているすべての Web サービスが自動的に配置され、最新の変更内容が確実に適用されます。

デプロイメント ディスクリプタについて

Web サービスのデプロイメント ディスクリプタ ファイル (Deployment.wsdd) は、Axis の Web Service Deployment Descriptor (WSDD) 形式に従ってフォーマットされた XML ファイルです。このデプロイメント ディスクリプタは、WSX を介して公開される一連の Web サービスと Web サービス操作を宣言および記述します。WSDD ファイルには、着信 SOAP リクエスト (認証など) またはレスポンス (送信データの再フォーマットなど) を処理する際に使用する必要がある追加の動作も定義されています。

Axis のマニュアルには、WSDD 形式の概要と使用方法が記載されています。ただし、Axis のマニュアルを参照する前に、WSX 内の次の制約事項に注意してください。

- Web サービスのデプロイメント ディスクリプタには、グローバル WSX 設定情報を挿入しないでください。Deployment.wsdd 内で宣言する設定情報は、サービス固有の宣言に制限する必要があります。
- WSX は、Axis の .jws ベースの Web サービスをサポートしていません。これらのサービスは準備段階では有用ですが、開発環境では、Web サービスを再配置する当社のメカニズムが、さらに堅牢で使用しやすいことが判明しています。
- セキュリティ上の理由から、Axis AdminServlet は WSX に含まれていません。

Axis デプロイメント ディスクリプタの詳細は、次の Axis のマニュアルを参照してください。

- 『Axis User's Guide』 - http://cvs.apache.org/viewcvs.cgi/~checkout~/xml-axis/java/docs/user-guide.html

  「Custom Deployment - Introducing WSDD」および「Service Styles - RPC, Document, Wrapped, and Message」の各セクションを参照してください。
- 『Axis Reference Guide』 - http://cvs.apache.org/viewcvs.cgi/~checkout~/xml-axis/java/docs/reference.html

  「Deployment (WSDD) Reference」のセクションを参照してください。

予約されている Web サービス名

次の Web サービス名は、Agile Integration Services (AIS) によって使用されるため、予約されています。これらの名前は、作成した Web サービスに対して使用しないでください。

- Export
- Importer
- 予約されているサービス名:
  - FSHelper、DmsService (ファイル マネージャおよび Viewer)
  - Export、Importer (AIS)
  - ResponseService、PackageService、AcsStatusService (ACS)

Web サービスの使用

カスタム Web サービスを開発して配置した後は、そのサービスを使用します。Web サービスには、`http://<hostname>:<port>/<virtualPath>/integration/ws/<WebServiceName>` 形式の URL を使用してアクセスできます。

注意 Agile で変更した `axis.jar` ファイルを使用する必要があります。このファイルは Agile API に付属しています。このファイルは、Agile の API コンポーネントをインストールすると、次の場所にインストールされます。*agile_home*¥`integration¥sdk¥lib¥axis.jar`

Web サービスのエントリ ポイントの定義

Web サービスのエントリ ポイント (または操作) は、Java クラスのパブリック メソッドに対応しています。クラスのすべてのパブリック メソッドを操作として公開する必要はありませんが、すべての操作はパブリック メソッドに対応しています。したがって、2 つのパブリック メソッド (methodOne と methodTwo など) を公開する Java クラス (MyClass など) がある場合は、一方または両方を Web サービス操作として公開できます。

通常、パラメータと戻りタイプに使用するデータ タイプが簡単であるほど、Web サービス操作の相互運用性は高くなります。データ タイプが複雑になると、Web サービス フレームワークからカスタム シリアライザ/デシリアライザまたは追加のサポートが必要になります。Axis が提供する追加のシリアライザ/デシリアライザの詳細は、
http://cvs.apache.org/viewcvs.cgi/~checkout~/xml-axis/java/docs/user-guide.html#DataMapping を参照してください。

注意 原則的には、Web サービスから Agile API オブジェクト (IAgileSession や IItem など) を返さないでください。Web サービスが返すのは、データ構造のみにしてください。

ユーザーの認証

デフォルトの Web サービスのバージョンとユーザーがカスタマイズしたバージョンはすべて、アプリケーション サーバによって保護されます。保護されている Web サービスにアクセスするには、次の行を Web サービス クライアントのスタブ コードに追加します。

```
        // Configure the stub with the necessary authentication information
    stub.setUsername(cl.getOptionValue(USER_SHRT));
    stub.setPassword(cl.getOptionValue(PASSWORD_SHRT));
    stub.setMaintainSession(true);
```

特定の Web サービスに対する Web コンテナの保護を解除するには、以下の行を次のアプリケーションに追加します。

```
 application.ear#application.war/WEB-INF/web.xml
```

および

```
application.ear#integration.war/WEB-INF/web.xml files:
        <security-constraint>
            <web-resource-collection>
               <web-resource-name>Unprotect web
        services</web-resource-name>
               <url-pattern>/ws/<web service name></url-pattern>
               <url-pattern>/services/<web service name></url-pattern>
            </web-resource-collection>
        </security-constraint>
```

クライアント/サーバ アクセスでのシングル サインオン クッキーの使用

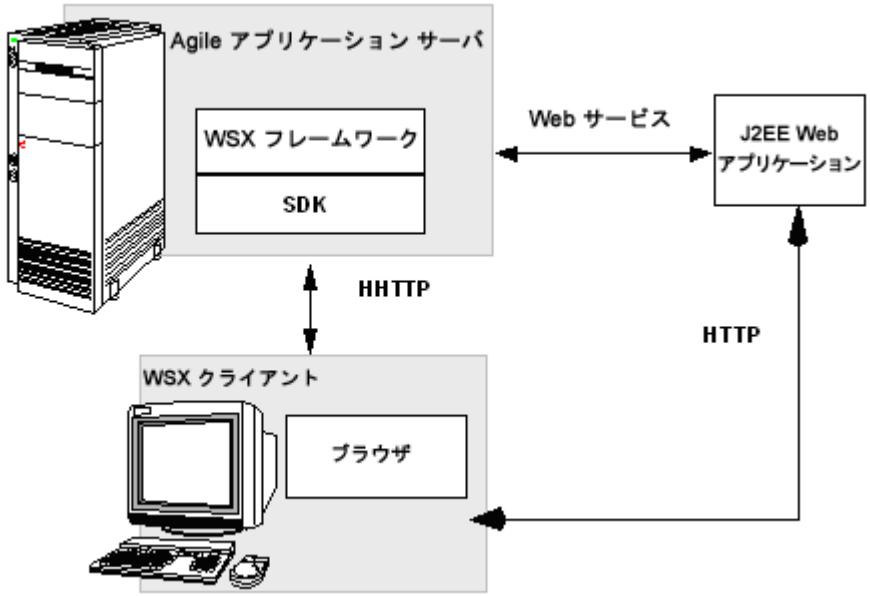
WSX クライアントのユーザーが、サード パーティのシングル サインオン製品 (SiteMinder や SAP ポータルなど) で保護されている Agile 9.X サーバで認証されると、ブラウザにはシングル サインオン クッキーが割り当てられます。このクッキーは、カスタム j2ee Web アプリケーションが Agile 9.X サーバと同じ DNS ドメインにある場合、そのアプリケーションに送信されます。SiteMinder を実行している場合は、2 つのアプリケーションが同じ DNS ドメインにない場合でも、この製品は SSO をサポートしているため、SiteMinder 管理者に確認してくださいこれで、Agile 9.X サーバに配置されている Web サービスを起動する場合は、ユーザー名とパスワードのかわりにシングル サインオン クッキーを有効なアカウント情報として渡すことができます。

注意 ユーザー名/パスワードとシングル サインオン クッキーの両方を使用している場合は、シングル サインオン クッキーがユーザー名/パスワードより優先されます。

配置アーキテクチャ

次の図は、Agile サーバと WSX クライアント間の相互作用とリクエストの流れを要約しています。

図 16: 配置アーキテクチャ



シングル サインオン クッキーを使用した Web サービス クライアントの起動

最初に HTTP リクエストからシングル サインオン クッキーを取得し、次に SOAP バインディング スタブコードを変更します。

シングル サインオン クッキーの取得

Web サービス クライアント スタブを起動するには、その前に、HTTP リクエストのシングル サインオン クッキーを取得する必要があります。デフォルトでは、SiteMinder が提供するシングル サインオン クッキーは、SMSESSION と呼ばれます。このクッキーを http://www.ietf.org/rfc/rfc2965.txt の RFC2965 で指定されている形式に変更します。最も簡単な形式は、「名前 = 値」です。この名前と値には、`javax.servlet.http.Cookie object` メソッドを呼び出すことでアクセスします。

SOAP バインディング スタブ コードの変更

Web サービスの SOAP バインディング スタブ クラスを検索します。これは Axis の wsdl2java ユーティリティによって生成されたクラスです。通常は、*<service-name>*`SoapBindingStub.java` という名前が付きます。値を設定するには、次に示すように、cookies という名前の変数とメソッドを追加します。

SOAP バインディング スタブ コードを変更する手順は、次のとおりです。

1. 次のコードを追加します。

```
private String cookies = "";

public void setCookies(String cookies) {
     this.cookies = cookies;
}
```

2. 太字の行を createCall() メソッドに追加します。

```
if  (super.cachedPortName != null) {
     _call.setPortname(super.cachedPortName);
}
_call.setProperty(org.apache.axis.transport.http.HTTPConstants.HEADER_COOKIE,
this.cookies);
java.util.Enumeration keys = super.cachedProperties.keys();
```

3. このクラスを再コンパイルし、次のサンプルに従って Web サービス スタブを起動します。

```
((<soaping binding stub class name>)stub).setCookies(<sso cookies you got in step
2.1>);
stub.setMaintainSession(true);
```

4. 次のサンプルと比較してください。このサンプルには、有効なアカウント情報としてユーザー名とパスワードが必要です。

```
stub.setUsername(<username>);
stub.setPassword(<password>);
stub.setMaintainSession(true);
```

5. Web サービス クライアントを j2ee Web アプリケーションの一部としてテストします。

MyFirstWebService サンプルの環境の準備

Web サービスの開発の容易性を示すために、開発プロセスに焦点を絞ったサンプルが用意されています。このサンプル (MyFirstWebService) は、Web サービスの作成方法を示す比較的簡単な例です。この Web サービスは、Agile SDK を使用して、特定のアイテムに関する情報を取得し、Web サービス操作の結果として返します。

必要な操作をサポートするために、次のエントリ ポイントが定義されています。

```
public String getItemField(String[] args) throws RemoteException
```

引数は、Jakarta Commons CLI というサード パーティ ライブラリを使用して、一連のコマンドライン引数であるかのように解析されます。それらの引数に基づいて、結果が `String` として返されます。実装の詳細情報は、*AGILE_HOME*¥integration¥sdk¥samples¥wsx¥src¥first のサンプルを参照してください。このセクションでは、実装の詳細ではなく配置プロセスについて説明します。

サンプル作成用ツールのダウンロード

`MyFirstWebService` サンプルを作成して配置するには、その前に、次のツールをダウンロードする必要があります。

ツール	ダウンロード拠点
Java 2 SDK SE バージョン 1.4.2	http://java.sun.com/j2se/1.4.2/download.html
Apache Project の Ant ビルド ツール、バージョン 1.6.5	http://ant.apache.org/

Java SDK のインストール

このセクションでは、Windows および Solaris プラットフォームで Java SDK をインストールする手順について説明します。適切なバージョンの Java がすでにインストールされている場合は、このセクションをスキップできます。

Windows で Java SDK をインストールする手順は、次のとおりです。

1. ディストリビューションをダブルクリックし、インストール手順に従います。
2. システム変数 JAVA_HOME に Java SDK インストールのホーム ディレクトリ (たとえば、D:¥j2sdk142) を設定します。

Solaris で Java SDK をインストールする手順は、次のとおりです。

1. ディストリビューション (たとえば、$ ./ j2sdk-1_4_2-solaris-sparc.sh) を実行し、インストール手順に従います。
2. 環境変数 JAVA_HOME に Java SDK インストールのホーム ディレクトリ (たとえば、/home/<user>/j2sdk142) を設定します。
3. 使用しているシェルに従って .profile または .cshrc ファイルを実行し、環境設定を再初期化します。

Ant のインストール

このセクションでは、Windows および Solaris で Ant をインストールする手順について説明します。

Windows で Ant をインストールする手順は、次のとおりです。

1. Zip アーカイブの内容をローカル ディレクトリに抽出し、インストール手順に従います。

   Windows 用の Ant ディストリビューションは zip ファイル (apache-ant-1.6.5-bin.zip など) です。

2. コマンド プロンプト ウィンドウを開き、次のコマンドを入力して Ant を起動できることを確認します。

   %ANT_HOME%¥bin¥ant -version

   次の出力が表示されます。

   ```
   Apache Ant version 1.6.5 compiled on date
   ```

Solaris で Ant をインストールする手順は、次のとおりです。

1. tar アーカイブの内容をローカル ディレクトリ (/home/user/ant など) に抽出し、インストール手順に従います。

   UNIX 用の ANT ディストリビューションは tar ファイル (apache-ant-1.6.5-bin.tar.gz など) です。

2. 使用しているシェルに従って .profile または .cshrc ファイルを実行し、環境設定を再初期化します。

3. コマンド プロンプトから、次のコマンドを入力して Ant を起動できることを確認します。

   $ANT_HOME/bin/ant -version

   次の出力が表示されます。

   ```
   Apache Ant version 1.6.5 compiled on date
   ```

MyFirstWebService サンプルの作成

Agile には、MyFirstWebService というサンプル Web サービスも含めて、SDK に関する複数のサンプル プログラムが用意されています。サンプル プログラムは、http://docs.agile.com からダウンロードできます。MyFirstWebService サンプルは、SDK サンプルの wsx フォルダにあります。

Ant ツールは、build.xml スクリプトを読み取り、WSX サンプルを実行しているサーバ上で、次の順序ですべてのターゲットを作成します。

1. Web サービスの Java コードを MyFirstWebService.jar にコンパイルします。
2. 結果の MyFirstWebService.jar ファイル (これには Deployment.wsdd が含まれています) と commons-cli.jar ファイルを .../sdk/extensions フォルダにコピーします。
3. クライアントの実行に使用するスクリプト (runner.bat または runner.sh) を生成します。(これによって、クライアントの実行に必要な CLASSPATH が設定されます。)
4. クライアント側のスタブ ファイルを生成し、次のフォルダに格納します。

   AGILE_HOME¥integration¥sdk¥samples¥wsx¥built¥src¥client

5. クライアント クラスをコンパイルし、次のフォルダに格納します。

   AGILE_HOME¥integration¥sdk¥samples¥wsx¥built¥classes¥client

サーバ マシンで WSX サンプルを作成する手順は、次のとおりです。

1. Agile ドキュメントの Web サイト (http://docs.agile.com) から、SDK サンプル フォルダをコピーします。
2. samples/WSX フォルダに移動します。

注意 このフォルダに AgileAPI.jar がない場合は、WSX サンプルをコンパイルできません。次のようにします。

3. サーバの $AGILE_HOME/sdk/samples/wsx に移動します。
4. http://archive.apache.org/dist/ws/axis/1_2/ (axis-bin-1_2.zip#/lib) から wsdl4j-1.5.1.jar をダウンロードし、lib フォルダにコピーして、ファイル名を wsdl4j.jar に変更します。
5. サンプルの build.xml ファイルを使用して、MyFirstWebService サンプルを作成します。
   - Windows の場合 - `%ANT_HOME%/bin/ant`
   - Solaris/Linux の場合 - `$ANT_HOME/bin/ant`

   重要 Web サービス サンプルを $AGILE_HOME/sdk/samples/wsx に作成しない場合は、wsx/built/MyFirstWebService.jar を $AGILE_HOME/integration/sdk/extensions にアップロードします。このディレクトリは、サーバの agile.properties で設定できます。http://hostname:port/virtualPath/ws/MyFirstWebService?wsdl を起動すると、SDK では WSDL ファイルまたは WSX が生成されないため、必要な WSDL ファイルが返されません。これらのファイルを生成するには、次の手順のとおりに実行してください。
6. 前述のパッケージ wsdl4j.jar ファイルを Agile の application.ear#APP-INF/lib フォルダにコピーし、ear ファイルを再配置します。
7. WSX フォルダで、次の適切なコマンドを起動し、WSX スタブを生成します。
   - Windows の場合 - `%ANT_HOME%/bin/ant -Dwsx.url=http://webserver/virtualPath/ws -Dusername=<username> -Dpassword=<password>`
   - Solaris/Linux の場合 - `$ANT_HOME/bin/ant -Dwsx.url=http://webserver/virtualPath/ws -Dusername=<username> -Dpassword=<password>`

Web サービス クライアントについて

このセクションでは、クライアント アプリケーションの開発に使用できるツールと、XML ファイルおよび HTTP リクエスト/レスポンス メッセージを生成して処理できる言語について説明します。

クライアント プログラミング言語

AIS クライアントの開発に使用する Java は、Agile でテストおよび認定されますが、SOAP メッセージはプラットフォームや言語には依存しません。これは、事実上、XML を生成および処理し、HTTP リクエスト/レスポンス メッセージを処理できるすべてのクライアント プログラミング言語が使用できることを意味します。たとえば、クライアントは Java、Visual Basic.Net、C++、C または Perl で開発できます。

Java、.Net、Perl、Python、C++、C およびその他の環境に対応した便利なライブラリもあります。次に、詳細を確認できるいくつかの Web サイトを示します。

- Apache Axis - Java 用オープン ソース SOAP 実装。次の Web サイトを参照してください。
  http://ws.apache.org/axis/
- Java Web Services Developer Pack (JWSDP) - Sun 社の SOAP プロトコルの Java 実装。次の Web サイトを参照してください。
  http://java.sun.com/webservices/webservicespack.html

- Microsoft .Net - Web サービス クライアントの作成に使用できる Microsoft Windows 用 XML Web サービス プラットフォーム。次の Web サイトを参照してください。
  http://msdn.microsoft.com/net
- SOAP::Lite for Perl - SOAP プロトコルの Perl 実装。次の Web サイトを参照してください。
  http://www.soaplite.com/

注意 他の SOAP 実装の包括的なリストは、次の Web サイトを参照してください。
http://www.soapware.org/

Web サービスへのアクセス

通常、Web サービスにアクセスするには、次の操作を実行する必要があります。

1. SOAP リクエストを生成する - 多くの場合、Web サービスを意識したコード ライブラリでは、適切な形式の SOAP リクエストを生成するクライアント側スタブを生成できます。
2. WSX へのリクエストを HTTP または HTTPS で発行する - 一連の適切なクライアント側スタブを生成した後は、クライアント アプリケーションがこれらのスタブを使用してリクエストを発行できます。
3. SOAP レスポンスを処理する - 通常は、クライアント側のスタブが SOAP レスポンスを処理し、レスポンスを一連の適切な戻りオブジェクトに変換する役割を担います。

WSX サンプルには、この処理を SOAP および Web サービスの関連ライブラリによって簡素化する方法が例示されています。次のセクションでは、MyFirstWebService サンプルを使用して、前述の手順を詳細に説明します。

MyFirstClient の作成

MyFirstWebService を作成して配置するときは、クライアント側のスタブとクライアント クラスも自動的に生成します。このセクションでは、Web サービス クライアントの作成方法に関する一般的な側面を説明するために、MyFirstClient を例として使用します。

SOAP リクエストの生成

ほとんどの場合、適切な SOAP リクエストの生成は、クライアント側のスタブを利用する場合と同じように簡単です。Web サービスを意識した多くのコード ライブラリでは、クライアント側のスタブを生成できます。この生成には、目的の Web サービスの WSDL に加え、コード生成ユーティリティの使用が伴います。

Axis には、クライアント側のスタブを生成する際に使用できる WSDL2Java ユーティリティが用意されています。Web サービスを意識した他のライブラリには、クライアント側のスタブを生成する独自の機能があります。Microsoft .Net には、wsdl.exe ユーティリティがあります。WSX サンプルの場合、クライアント側スタブは、そのサンプルの作成処理の中で生成されます。

build.xml ファイルには、次の Ant ターゲットがあります。

```
 <target name="generate-stubs" depends="init"
unless="stubs.present">
   <fail unless="wsx.url">wsx.url must be defined</fail>
   <axis-wsdl2java output="${built.dir}/src"
                  url="${wsx.url}/MyFirstWebService?wsdl">
     <mapping namespace="http://www.agile.com/ws/SampleWsx"
package="client"/>
   </axis-wsdl2java>
 </target>
```

前述の Ant ターゲットは、MyFirstWebService のクライアント側スタブを生成する役割を担います。この起動によって、${ws.url}/MyFirstWebService?wsdl から MyFirstWebService WSDL が取得され、クライアント Java パッケージで Java コードが生成され、${built.dir}/src ディレクトリにソース コードが配置されます。WSDL2Java ユーティリティの詳細は、次の Web サイトで Axis のマニュアルを参照してください。
http://xml.apache.org/axis

クライアント側のスタブが生成された後、ユーザーは、生成されたオブジェクト定義を使用することで、適切な SOAP リクエストをより簡単に生成できます。ユーザーは、有効な SOAP リクエストの作成方法を理解する必要がなく、これらのスタブを使用して、ターゲット Web サービス操作の機能に集中できます。..¥samples¥wsx¥src¥client に格納されている MyFirstClient.java を参照すると、SOAP リクエストの生成に使用されるすべてのコードが主要なメソッドに含まれていることがわかります。

SOAP リクエストの発行

Web サービス操作を使用する次の手順は、生成された SOAP リクエストを Web サービス エンジンに適切に発行することです。生成されたクライアント側スタブを処理する場合、通常、この手順は、スタブを目的のサーバに示して、そのスタブのメソッドを起動するなどの簡単なものです。接続を開いたり、ワイヤーにデータを手動で配列する必要はありません。このような詳細な処理は、生成されたスタブがかわりに処理します。

..¥samples¥wsx¥src¥client にある MyFirstClient.java サンプルでは、次の 2 箇所で SOAP リクエストを発行する方法が示されています。

- `getStub()` メソッドは、必要な Web サービス エンジンにクライアント側のスタブを示します。
- 主要なメソッドに含まれている stub.`getItemField()` メソッドを起動することで、Web サービス エンジンにリクエストを発行します。リクエストの発行は、スタブ自体で管理されます。接続、発行または個々の配列について心配する必要はありません。

必要な Web サービス エンジンにスタブを示す方法およびリクエストを発行する方法の詳細は、コード ライブラリごとに異なります。詳細は、Web サービスを意識したコード ライブラリのマニュアルを参照してください。

SOAP レスポンスの処理

SOAP レスポンスは、通常、生成されたクライアント側のスタブを介して処理されます。これらのスタブが生成されていない場合、XML ベースの SOAP レスポンスの解析、およびフォーマットや非整列化などで発生する多くの問題には、開発者による対処が必要となります。ただし、生成されたスタブで処理する場合は、これらの詳細すべてが考慮されるため、開発者は、適切に処理された Java オブジェクトを入手できます。XML ドキュメントの解析や戻りデータの識別は、開発者ではなく、スタブによって自動的に実行されます。

SOAP レスポンスが処理される仕組みの詳細は、コード ライブラリごとに異なります。一部の SOAP サーバでは、クライアントが、なんらかの別の手段 (多くの場合 WSDL) でデータ タイプを認識することが求められます。詳細は、Web サービスを意識したコード ライブラリのマニュアルを参照してください。

MyFirstClient の実行

MyFirstWebService サンプルを作成して配置するには、そのサンプルを Web サービス クライアントで実行するために必要な CLASSPATH の初期設定が含まれているファイルを *AGILE_HOME*¥integration¥sdk¥samples¥wsx ディレクトリに配置します。

- Windows の場合、このファイルは runner.bat です。
- Solaris の場合、このファイルは runner.sh です。

MyFirstClient の使用説明を出力するには、次のコマンドを入力します。

```
> runner client.MyFirstClient
```

次の使用説明では、部品 1000-02 の [タイトル ブロック.説明] フィールドが返されます。

```
> runner client.MyFirstClient -T 15000 -a "<attribute name>"
-e <virtual path> -h <host> -l <port> -n <item number> -p <password>
-u <username>
> runner client.MyFirstClient -T 15000 -a "Title Block.Description"
  -e Agile -h localhost -l 80 -n 1000-02 -p agile -u jeffp
```

WSX 内部における Agile セッションの作成

デフォルトでは、WSX は Web コンテナによって保護されます。したがって、WSX 内部に Agile セッションを作成する場合は、ユーザーのアカウント情報を指定する必要があります。次の例では、保護されている WSX 内に Agile セッションを作成しています。

例: WSX 内部におけるセッションの設定

```
AgileSessionFactory factory = AgileSessionFactory.getInstance(null);
IAgileSession session = factory.createSession(null);
```

注意 暗黙的なセッションは上書きしないでください。

異なるユーザーを指定するには、リモート クライアントから接続する場合のように、明示的な SDK セッションを作成する必要があります。つまり、AgileSessionFactory.getInstance() メソッドに引数を指定します。

例 19-2: 暗黙的なセッションに依存しない明示的なセッションの作成

```
AgileSessionFactory factory = AgileSessionFactory.getInstance
("http://...");
HashMap params = new HashMap();
     params.put(AgileSessionFactory.USERNAME, ...);
     params.put(AgileSessionFactory.PASSWORD, ...);
     IAgileSession session = factory.createSession(params);
```

インポート データのサーバ ルール準拠の検証

インポート検証の目的は、インポート データが適切なサーバ ルール (最大長、許容値、その他の制約など) に準拠していることを確認することです。この検証プロセスによって、インポートする前に、正常にインポートされないデータがわかります。SDK には、AIS のための 2 つのメソッドがあります。これらのメソッドは、Agile PLM システムにインポートする前のソース データをプログラムによって検証し、検証済みのデータをインポートします。AIS の概要と、PLM データベースにインポートする前にデータを検証する方法は、『Agile Integration Services Developer Guide』および『Agile インポートおよびエクスポート・ガイド』を参照してください。

データの検証 (インポート前)

SDK には、サーバのビジネス ルールへの準拠に関して、データをプログラムによって検証する `IImportManager.validateData(byte[], String, byte[], byte[], String[], List)` メソッドがあります。このアクションは、無効なデータを識別するために、データをインポートする前に実行されます。

データのインポート (検証後)

SDK には、データをプログラムによってインポートする `IImportManager.importData(byte[], String, byte[], byte[], String[], List)` メソッドがあります。このアクションは、サーバのビジネス ルールに適合するデータを選択してから PLM システムにインポートするように、`IImportManager.validateData()` の実行後に実行されます。

例: データ検証メソッドの起動

```
String _url="http://localhost/Agile";
     String _user="admin";
     String _pwd="agile";
     String srcFilePath="itemp1p2p3.csv";

     //Supported file
types:aXML,IPC2571,ExcelFile,DelimitedTextFile. Note these are the
same constants as in AIS command.
     String srcFileType="DelimitedTextFile";
     //Null = load the default mapping file. The default mapping file
mappingPath is loaded if one is not provided for aXML/PDX source. The
default mapping does not include p3 fields.
     String mappingPath="NewMapFile.xml";
     //Null = no transform. The transformation tempalte file can be
downloaded from import wizard.
     String transformPath=null;
     //all supported operations in 9221
release :"items.bom","items.aml","items.attachments","items.relati
onships", "manufacturers",

//"manufacturers.relationships","manufacturers.attachments",
"manufacturerParts", "manufacturerParts.relationships",
     //"manufacturerParts.attachments" "partgroups",
"partgroups.relationships","partgroups.attachments". Note this is
same constants of AIS command.
     String [] operations=new String[]{"items"};
     List options=new ArrayList();
```

```
     //The string element in options array is same as preference option
of AIS command.

options.add("BusinessRuleOptions|ChangeMode=Authoring");
     AgileSessionFactory factory =
AgileSessionFactory.getInstance(_url);
     HashMap params = new HashMap();
     params.put(AgileSessionFactory.USERNAME, _user);
     params.put(AgileSessionFactory.PASSWORD, _pwd);
     IAgileSession session = factory.createSession(params);
     IImportManager imgr = (IImportManager)
session.getManager(IImportManager.class);

      //need one help method to convert the file stream into byte array.
     //byte[] logData=imgr.importData(convert2Byte(srcFilePath),
srcFileType, convert2Byte(mappingPath),null,operations, options);
     byte[] logData=imgr.validateData(convert2Byte(srcFilePath),
srcFileType, convert2Byte(mappingPath), null,operations, options);
```

Microsoft .NET の相互運用性

Microsoft 社の .NET フレームワーク技術は、アプリケーション プログラミング インターフェース (API) を標準的な Windows オペレーティング システムのサービスおよび API に提供する開発フレームワークで、1990 年代後期に Microsoft 社から登場した多様な技術 (ASP、COM+、XML、SOAP など) の集大成です。

.NET も、Visual Basic、J++、C++ など、Microsoft 社が提供する Visual Studio 環境で提供されるすべての言語の集大成です。また、C# (C シャープと読みます) や .NET ファミリには比較的新しい言語である J# (J シャープと読みます) など、新しい言語も開発されています。J# は、実質的には .NET フレームワークへの Java の統合を提供する Microsoft 風の Java です。J# は、現時点では Java VM では動作しません。J# は、本質的には、Microsoft 社独自の仮想マシンである .NET Common Language Runtime (CLR) で実行される Java 対応コードを含むラッパーとして機能します。

CLR は、.NET フレームワークにとって最も重要なコンポーネントです。この CLR によって、オブジェクトのアクティブ化、セキュリティ チェック、メモリ管理、オブジェクトの実行、およびオブジェクトが使用されなくなった場合のメモリ クリーンアップ (ガーベッジ コレクト) が提供されます。

.NET には、前述したいずれかの言語を使用して Windows ベースのアプリケーションまたは (ASP.NET を介した) Web ベースのアプリケーションを記述するのみでなく、これらの言語を 1 つの共通 API に統合できるという要素もあります。これは、開発者が言語に依存しないコードを記述し、クラスから継承し、例外を捕捉し、.NET フレームワーク全体で様々な言語による多様性のメリットを最大限に活用できることを意味します。

重要 WSX フレームワーク (AXIS SOAP プロセッサ) は、AXIS Web サービス クライアントでは問題なく機能しますが、.NET との互換性は完全ではありません。Microsoft 社も Apache グループも、AXIS と .Net に対する相互運用性テストは実施していません。簡単なデータ タイプの場合、AXIS ベースの Web サービスは .Net ベースの Web サービス クライアントで問題なく機能します。一部の複雑なデータ タイプ (バイナリ添付ファイルなど) の場合は、相互運用上の問題が発生する可能性があります。Agile アプリケーション サーバの外部に配置された非 AXIS Web サービスの実装に関する相互運用性については、各 Web サービス ベンダーにお問い合わせください。

Web サービス拡張に関するよくある質問

このセクションでは、Web サービス拡張に関する一般的な質問について回答します。

Web サービス拡張 (WSX) とは、どのようなものですか。

WSX は、Agile 顧客が Web サービスを使用して Agile PLM サーバの機能を拡張するためのフレームワークです。

Web サービスとは、どのようなものですか。

Web サービスは、SOAP メッセージ プロトコルを使用して、インターネットを介してソフトウェア サービスを提供します。これによって、複数のソフトウェア コンポーネントが世界中で相互に情報を受け渡しできます。Web サービスは、いずれのオペレーティング システムまたはプログラミング言語にも制限されません。サービスのパブリック インターフェースの記述には WSDL が使用され、Web サービスは原則的に自己記述となるため、比較的簡単に使用できます。

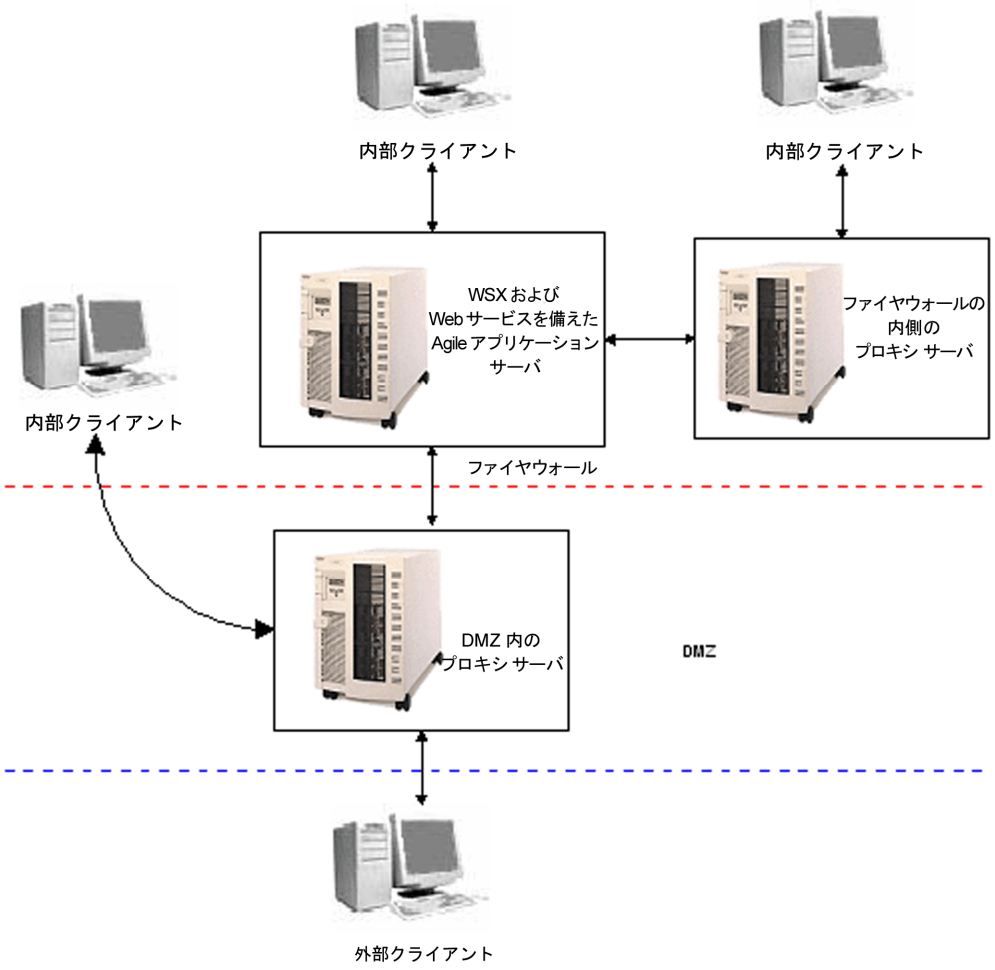
Agile の Java API のみでは不可能で、WSX では可能な操作とは、どのようなものですか。

WSX は、標準的な HTTP(S) プロトコルを使用して、ファイヤウォール フレンドリな XML ベースの統合を Agile PLM データに提供します。WSX は、SOAP に準拠するあらゆるプログラミング言語をサポートしています。たとえば、Web サービスの Perl または .Net クライアントを作成できます。WSX を使用すると、異なる企業の複数のシステムで簡単かつ安全に情報を相互に受け渡しできます。WSX 内部に配置されたサービスは、アプリケーション サーバが提供するすべての拡張性、フェイルオーバーおよびクラスタリング機能を活用できます。アプリケーション サーバ上で動作するサービスにとっても、大きなパフォーマンス上のメリットがあります。

WSX は、保護されている接続と保護されていない接続の両方をサポートしていますか。

はい。ファイヤウォール外部から Web サービスに発行されるリクエストには、ファイヤウォール内部で発行されるリクエストとは異なるセキュリティ要件が適用されます。外部 (ファイヤウォール外部) または内部の各 WSX に対して 2 つの個別のエントリ ポイントが提供されます。外部のリクエストはプロキシ サーバに対して実行された後、アプリケーション サーバに転送されます。プロキシ サーバは DMZ に設置されています。内部リクエストは、次の図に示すように、保護されている同じプロキシ サーバ、DMZ に設置されていない別のプロキシ サーバ、または直接アプリケーション サーバに対して実行できます。

図 17: Web サービス クライアントが Agile PLM サーバに接続する方法



WSX には、どのようなユーザー認証サービスがありますか。

デフォルトでは、WSX はアプリケーション サーバによって保護されます。WSX クライアントが保護されているサービスを起動するたびに、ユーザー名とパスワードによるセキュリティが実施されます。詳細は、330 ページの「ユーザーの認証」を参照してください。

WSX では、どのような SOAP エンジンが使用されますか。

WSX は、Apache Axis (SOAP のオープン ソース実装) に基づいています。Axis の詳細は、http://ws.apache.org/axis/ の Axis Web サイトを参照してください。

WSX は SOAP 添付ファイルを処理しますか。

はい。実際、Agile Integration Services には、バイナリ添付ファイルのエクスポートとインポートを実行できる exportData と importData サービスが用意されています。

WSX は、ステートフル セッションをサポートしていますか。

はい。WSX の中心である Axis Web サービス エンジンが、各接続間のセッションの状態を維持します。セッションは、HTTP クッキーまたは SOAP ヘッダをベースにすることができます。これは、簡単な 1 回かぎりの処理ではなく、より持続的なアプリケーションをサポートするサーバ側のコードを生成する際に有効です。Web サービス セッションの詳細は、Axis のマニュアルを参照してください。参照は、http://ws.apache.org/axis/faq.html の「Axis FAQ」から開始できます。

WSX は、HTTP 以外のプロトコルをサポートしていますか。

いいえ。WSX がサポートしているのは、HTTP 関連のプロトコルのみです。安全性をさらに高めるためには、HTTPS および SSL を使用して Web サービスに接続できます。将来は、必要に応じて他のプロトコルをサポートする可能性もあります。

WSX は、Perl、Python、PHP または他の Web スクリプト言語をサポートしていますか。

WSX は、SOAP メッセージを送信できるすべてのクライアント プログラミング言語をサポートしています。Agile SDK は、WSX クライアントの例を Perl、Python または PHP で提供していませんが、SOAP メッセージは、これらのスクリプト言語で確実に送信できます。

WSX は UDDI をサポートしていますか。

いいえ。UDDI は、ソフトウェアで他のサービスを自動的に検出して統合できるように設計された Web サービスの一般的なビジネス レジストリの仕様です。現在は、インターネット上の Agile PLM Web サービスの登録に UDDI を使用する必要はありません。通常、Agile PLM Web サービスは、内部のソフトウェア システムとの統合、またはデータをパートナーまたはサプライヤと交換するために作成されます。ただし、UDDI に対するサポートは、技術の進展に伴って考慮される可能性があります。

Web サービスは、どのように配置するのですか。

アプリケーション サーバ コンピュータの agile_home/integration/sdk/extensions フォルダに、サービスの JAR ファイルを格納します。Web サービスの JAR ファイルには、META-INF/services/com.agile.wsx.Deployment.wsdd という名前のデプロイメント デスクリプタ ファイルが含まれている必要があります。

このデプロイメント ディスクリプタ ファイルは、Axis の Web Service Deployment Descriptor (WSDD) 形式に従ってフォーマットされた XML ファイルです。このデプロイメント ディスクリプタは、WSX を介して公開される一連の Web サービスと Web サービス操作を宣言および記述します。WSDD ファイルには、着信 SOAP リクエスト (ユーザー認証など) またはレスポンス (送信データの再フォーマット) を処理する際に使用する必要がある追加の動作も定義されています。WSDD 形式の詳細は、『Axis Reference Guide』を参照してください。このマニュアルは、http://ws.apache.org/axis/ から入手できます。

Web サービスとその JAR ファイルを配置するときは、アプリケーション サーバのクラスパスを更新する必要がありますか。

いいえ。クラスパスは専用のクラスローダによって自動的に更新されます。クラスローダは、agile_home/integration/sdk/extensions (または agile.properties ファイルの sdk.extensions プロパティで指定した場所) にあるクラスを使用して、アプリケーション サーバのクラスパスを拡張します。

Web サービスを変更して再配置した場合は、アプリケーション サーバを再起動する必要がありますか。

いいえ。専用のハンドラが、配置された最新のファイルを使用して Web サービス スタックを更新します。Web サービス リクエストが発行されるたびに、このハンドラによって、agile_home/integration/sdk/services 内の JAR ファイルが更新、追加または削除されたかどうかが確認されます。変更があった場合は、Web サービス スタック全体が再設定されます。この機能によって、コードを再コンパイルし、アプリケーション サーバを再起動せずに Web サービスを再配置できるため、貴重な開発時間を節約できます。

WSX フレームワークを使用している Agile 製品はありますか。

はい。Agile Content Service (ACS) と Agile Integration Services (AIS) は両方とも、WSX フレームワークに依存して Agile PLM サーバとデータを交換します。

Agile Integration Services とは、どのようなものですか。

Agile Integration Services (AIS) は、インポート、エクスポートおよび部品リスト機能を提供する Web サービスです。これらの Web サービスには、サンプル Java クライアントが付属していますが、別の SOAP 準拠の AIS クライアントを他のプログラミング言語で作成できます。

基本認証とは、どのようなものですか。

基本認証は簡易的な認証方法です。基本認証を使用すると、クライアント プログラムでは、リクエストを発行する際に、暗号化されていないユーザー名とパスワードの形式のアカウント情報を提供できます。Web サービス リスナーの配置に基本認証を使用する新しい Web モジュールがあります。基本認証で Web サービスにアクセスするには、次の URL を使用します。
http://<host>:<port>/Agile/integration/ws/xxxx

たとえば、MyFirstWebService のサンプルにアクセスするには、次の URL を使用します。

```
http://<hostname>/Agile/integration/ws/MyFirstWebService?wsdl
```

第 21 章

ダッシュボード管理拡張の開発

扱うトピックは次のとおりです。

ダッシュボード管理拡張について

プロセス拡張と同様に、ダッシュボード管理拡張 (DX) は、Agile PLM システムの機能を拡張します。この製品を使用するには、Agile ダッシュボード ライセンスが必要です。この機能拡張では、PLM データにアクセスして Agile PLM ダッシュボードに表示する際に、次の形式を使用できます。

- ChartDataModel (チャートの場合)
- Collections (テーブルの場合)

これらの形式を使用して定義されたデータは、Agile サーバによって解釈と処理が実行され、管理者権限を持つ Agile Java クライアント ユーザーがダッシュボードの各タブを定義して、次のいずれかの表示またはレイアウトでデータを表示できます。

- チャート
- テーブル
- カスタム (URL)

SDK には、Agile PLM サーバが、必要なコンテンツを取得して、DX が必要とする形式に整えて Agile PLM ダッシュボードにデータを表示できるように、内部の Agile データベースに接続する API が用意されています。同様に、JDBC などの他の Java API を使用して、外部のデータベースに接続してコンテンツを取得することもできます。

要約すると、DX がデータを提供し、Agile Java クライアントがダッシュボードのタブおよびデータ (テーブル、チャートおよび URL) を表示するための形式を設定します。最終的に、適切な権限を持つ Agile PLM ユーザーが [ダッシュボート] タブを表示することで、Agile Web クライアントに表示されます。

この章では、背景情報およびこれらのメソッドを開発する手順について説明します。

ダッシュボード管理拡張の役割と権限

管理者は、[管理] > [ユーザー設定] > [権限] の順に選択して、[ダッシュボード] タブ表示権限を設定する必要があります。その結果、PLM ユーザーは、タブおよび関連データを Web クライアントに表示できるようになります。さらに、[ダッシュボード] タブは権限によって管理されるため、Agile PLM ユーザーには、タブにデータを表示するための適切な役割と権限が必要です。表示を設定し、権限を割り当てる手順の詳細は、『Agile PLM 管理者ガイド』の第 10 章および第 11 章を参照してください。

カスタム チャート ダッシュボード管理拡張の開発

`ICustomChart` インターフェースを使用して、必要なデータをチャート形式で表示する DX を作成できます。このインターフェースは、public `ChartDataModel getChart(IAgileSession session, Map params)` のインスタンスを返すメソッドを公開します。

注意 このインターフェースの実装には、引数のないコンストラクタが必要で、再入可能であることが必要です。

ChartDataModel および ChartDataSet の理解

ChartDataModel クラスは、入力データをチャート形式に編成します。このクラスは、DX に公開される具象クラスで、チャートの構成に必要な 1 つ以上の ChartDataSet が含まれます。

ChartDataSet は、DX に公開されるもう 1 つの具象クラスです。このクラスには、チャートの描画に必要なデータが格納されます。たとえば、X 軸と Y 軸の値とラベルが含まれます。ChartDataModel は、すべてのデータセットのプレースホルダです。

注意 ChartDataModel および ChartDataSet クラスは、`com.agile.px` パッケージで公開されています。

カスタム チャート DX のデータ ソースの定義

前述のように、チャート DX では、データがチャート形式で表示されます。次の例のコードでは、ICustomChart および公開クラス (ChartDataModel と ChartDataSet) を使用して、チャート形式で事前定義されている入力データから、各曜日の朝夕の温度差を表示しています。

例 20-1: チャート形式でデータを表示するための DX の定義

```
package dashboard.chart;
import java.util.Map;
import com.agile.api.IAgileSession;
import com.agile.px.ICustomChart;
import com.agile.px.ChartDataModel;
import com.agile.px.ChartDataSet;

/**
 * A Sample Dashboard DX for Charts with predefined data. This Example
is
```

```
 * to display a comparison chart between Morning and Evening Temperatures
 * for each day in the week with some predefined data.
 */

public class TemperatureComparisionChart implements ICustomChart(

   /**
    * Returns custom ChartDataModel. ChartDataModel is a placeholder
to hold all the
    * ChartDataSet(s) and any other relevant information related to
the charts.
    * @param session current user session.
    * @param params
    * @return com.agile.px.ChartDataModel
    */
public ChartDataModel getChart(IAgileSession session,Map params)
throws Exception{
// Create a ChartDataModel
ChartDataModel chartDataModel = new ChartDataModel("Temperatures");

// Create a ChartDataSet's for Morning and Evening Temperatures
ChartDataSet chartdataSet[] = new ChartDataSet[2];
// Create a ChartDataSet for Morning Temperatures
chartdataSet[0] = new ChartDataSet("Morning Temperatures",7);

// fill in the Morning Temperatures
double[] morTempValues = {10, 8, 12, 19, 10, 14, 13};
chartdataSet[0].setValues(morTempValues); // or setYValues can be used
instead

// Set the Labels
String[] labels = {"Monday", "Tuesday", "Wednesday", "Thursday",
"Friday", "Saturday", "Sunday"};
chartdataSet[0].setLabels(labels);

// Create a ChartDataSet for Evening Temperatures
chartdataSet[1] = new ChartDataSet("Evening Temperatures",7);

// Fill in the Evening Temperatures
double[] eveTempValues = {16, 12, 20, 15, 18, 24, 26};
chartdataSet[1].setValues(eveTempValues);
chartdataSet[1].setLabels(labels);

// Set the ChartDataSets in the Chart Model
chartDataModel.setDataSets(chartdataSet);

return chartDataModel;
}
}
```

カスタム チャート DX ソースのパッケージ化および配置

新しいチャートに必要なクラスを開発した後は、次の手順を使用してクラスをパッケージ化し、配置します。

チャート DX ソースをパッケージ化して配置する手順は、次のとおりです。

1. Java 開発環境または Java アーカイブ ツール (または JAR ツール) を使用して、カスタム アクション用の JAR ファイルを 1 つ以上作成します。JAR ファイルに、com.agile.px.ICustomChart という名前のファイルが格納された META-INF/services ディレクトリが含まれていることを確認します。このファイルは、カスタム アクション用の Java の完全修飾クラス名を 1 行に 1 クラスずつリストしたテキスト ファイルです。

   1 つのパッケージに複数のチャートを含めることができます。たとえば、`com.agile.px.ICustomChart` は次のような形式になります。

   ```
   dashboard.chart.TemperatureComparisionChart
   dashboard.chart.AgileObjectsCountChart
   dashboard.chart.ActualVsBudgetedLaborCostChart
   ```

   注意 JAR ファイル内のパスでは、大文字と小文字が区別されます。したがって、JAR ファイル内の META-INF フォルダの名前は、すべて大文字か、すべて小文字である必要があります。そうでない場合、カスタム アクションの配置が失敗となります。

2. Agile アプリケーション サーバがインストールされているコンピュータの agile_home/integration/sdk/extensions フォルダに JAR ファイルを格納します。

   注意 クラスタ環境に複数のアプリケーション サーバがある場合は、クラスタ内の各サーバにダッシュボード拡張ファイルを配置する必要があります。

Java クライアントでのチャート DX の設定

Agile Java クライアントでは、管理モジュールにチャート データ ソースを定義できます。Agile PLM システム設定を構成するには、管理者アカウントが必要です。これについては、後ほど簡単に説明します。詳細は、『Agile PLM 管理者ガイド』を参照してください。

DX に対して用意したデータは、レイアウトに関係なく、[ダッシュボード管理] タブに表示されます。[エグゼクティブ]、[ファイナンシャル] などのデフォルトのタブには新しいテーブルを定義できないため、新規のタブを定義し、そのタブにテーブルを設定して DX を設定する必要があります。

オプションの [ダッシュボード管理] タブを追加する手順は、次のとおりです。

1. Java クライアントで、[管理] > [システム設定] > [ダッシュボード管理] の順に選択し、[ダッシュボード管理] の [新規ダッシュボード タブ]  アイコンをクリックします。
2. [ダッシュボード タブの作成] ダイアログで、[名前] フィールドと [説明] フィールドに値 (たとえば、「ダッシュボード拡張機能」など) を入力して、[表示] フィールドを [はい] に設定し、[OK] をクリックします。[ダッシュボード管理] のエントリとして、[ダッシュボード拡張機能] が表示されます。
3. ダッシュボードのタブを並べ替える  アイコンをクリックし、表示するタブを必要に応じて並べ替えます。

Agile Web クライアントでのオプション タブの表示

管理者は、Agile Web クライアントに新規のオプション タブを表示できます。役割および権限の要件を満たすユーザーには、タブおよび対応するデータが表示されます。必要な手順は、『Agile PLM 管理者ガイド』を参照してください。

オプション タブにチャート タイプのテーブルを設定する手順は、次のとおりです。

1. 新しいタブ (たとえば、前述の [ダッシュボード拡張機能] など) を定義します。
2. 新しいタブ ([ダッシュボード拡張機能]) で、 

をクリックします。[ダッシュボード管理 - ダッシュボード拡張機能] ページが表示されます。
3. このページで、[新規ダッシュボード テーブル]  アイコンをクリックして [ダッシュボード テーブルの作成] ダイアログを開き、新しいテーブルを定義します。
4. [リスト タイプの表示] ドロップダウン リストから、[チャート] を選択します。

ダッシュボード テーブル	説明	可能な設定
名前	テーブルの名前を入力します。	文字列
説明	テーブルの説明を入力します。	文字列 (オプション)
リスト タイプの表示	テーブルのタイプがリストされます。[チャート] を選択します ([チャート] を選択すると、オプションがさらに表示されます)。	[チャート]、[テーブル]、[カスタム]、[詳細検索]
ダッシュボード拡張機能	チャート タイプのリスト用に作成されたすべてのプロセス拡張がリストされます。目的のチャート用のプロセス拡張を選択します。	
表示	Web クライアントで使用できるようにするかどうかを選択します。	[はい] または [いいえ]
チャート タイプ	表示するチャートのタイプを選択します。	[面]、[棒]、[線]、[円]、[極]、[散布]、[積み上げ面]、[積み上げ棒]、[テーブル]
X 軸	X 軸ラベルを入力します。	(オプション)
Y 軸	Y 軸ラベルを入力します。	(オプション)
表題の表示	チャートの表題を画面に表示するかどうかを選択します。	[はい] または [いいえ]
表題の位置	表題を表示する位置を選択します。	[一番下]、[デフォルト]、[左]、[右]、[一番上]
3D スタイル	グラフを 3D で表示するかどうかを選択します。	[はい] または [いいえ]
ヘッダ	必要に応じてヘッダ メモを入力します。	(オプション)
フッタ	必要に応じてフッタ メモを入力します。	(オプション)

5. フィールドを完成し、[OK] をクリックします。[ダッシュボード管理 - ダッシュボード拡張機能] ビューに新しいチャート名が表示されます。

カスタム テーブル ダッシュボード管理拡張の開発

ICustomTable インターフェースを定義して、必要なデータをテーブル形式で表示する DX を作成します。このインターフェースは、Collection クラスのインスタンスを返す getTable(IAgileSession session, Map params) メソッドを公開します。

```
public Collection getTable(IAgileSession session, Map params);
```

注意 このインターフェースの実装には、引数のないコンストラクタが必要で、再入可能であることが必要です。

Collection および CustomTableConstants の理解

DX のテーブル データは Java HashMap のコレクションです。各マップ キーはテーブル表示の属性を表し、マップはテーブルの行を表します。

表示の [属性] プロパティ列には、データ モデルとテーブル表示とのマッピングを定義します。このプロパティの値は、HashMap エントリのキーに相当します。

- HashMap キー - HashMap エントリでは、属性はテーブル表示で定義されます。たとえば、キー値に「name」を使用する HashMap エントリでは、この属性のプロパティ [属性] の値が「name」となります。get('name') メソッドは、この属性に対する表示データを提供します。
- リンク、画像、通貨、テキスト、日付および数値データ - これらのデータ タイプは、テーブル DX 形式でサポートされ、次のプロパティが設定されているオブジェクトが返されます。
  - テキスト - 日付および数値データ タイプには、その他のプロパティは必要ありません。
  - リンク - 有効な URL (文字列など) が、表示のターゲットおよびラベルとして使用されます。リンク データ タイプに想定されるプロパティは、内部および外部リンクとも同じです。DX ユーザーが、内部リンクの URL を解決して、URL プロパティに追加します。DX ユーザーは、内部リンクに RightPane のターゲット プロパティを指定できます。デフォルトでは、リンクのターゲットは新しいウィンドウに設定されます。
  - 画像 - 画像では、画像 URL (文字列など) および画像のツール チップとして表示するラベルを返すことが想定されます。
  - 通貨 - 通貨データ タイプには、通貨コード (文字列) と値 (数値) を指定する必要があります。

注意 リンク、通貨および画像データのプロパティをサポートするキーは、CustomTableConstants クラスに定数として用意されています。このクラスには、定数 SERVER_URL があります。この定数を使用して、パラメータから DX のサーバ URL を取得できます。

カスタム テーブル DX のデータ ソースの定義

次の例のサンプル ダッシュボード DX では、ダッシュボードに表示するための事前定義データが格納された行のコレクションを作成しています。各行は、テーブルの各列に相当するキーと値のペアが指定されている Java Map オブジェクトです。テーブルの列の各セルには、値が表示されます。キーは、表示におけるマッピング属性名です。表示に新しい属性 (列) を作成する場合は、このキーを [属性] フィールドに指定する必要があります。この DX の属性名および対応するデータ タイプは、次のとおりです。

属性	対応するデータ タイプ
myString	テキスト
myExternalLink	リンク
myDate	日付
myMoney	通貨
myNumber	数値
myImage	画像

例: テーブル形式でデータを表示するためのダッシュボード拡張機能の定義

```
package dashboard.table;
import java.util.*;
import com.agile.api.IAgileSession;
import com.agile.px.ICustomTable;
import com.agile.px.CustomTableConstants;

/** This Sample Dashboard DX creates a collection of rows with predefined
data
* in the format to be displayed in the Dashboard.
* Each row is a Java Map object which has key-value pairs corresponding
to
* each column in the Table. The value is displayed in each Cell of the
* column in the table. The key is the mapping Attribute name in the
View.
* While creating new Attributes (Columns) in the View, you must supply
this key
* in the Attribute field.
* The corresponding Attribute Names and Data type for this DX are listed
below.
*  <table border="1">
*  <tr><td><b>Attribute</b> </td><td><b>Data Tye<b></td></tr>
*  <tr><td>myString        </td><td>Text     </td></tr>
*  <tr><td>myExternalLink   </td><td>Link     </td></tr>
*  <tr><td>myDate          </td><td>Date     </td></tr>
*  <tr><td>myMoney         </td><td>Money    </td></tr>
*  <tr><td>myNumber        </td><td>Numeric   </td></tr>
*  <tr><td>myImage         </td><td>Image    </td></tr>
*  </table>
*
*/
```

```
public class DashboardSampleTable  implements ICustomTable {
    /**
     * Returns custom table data in form of collection of rows.  Row
is assumed
     * to be a java Map object.
     * @param session the user session
     * @param params
     * @return : java.util.Collection
     */
   public Collection getTable (IAgileSession session,Map params) throws
Exception{
String serverUrl =
(String)params.get(CustomTableConstants.SERVER_URL);
String baseUrl = serverUrl.substring(0,serverUrl.lastIndexOf('/'));
ArrayList result = new ArrayList();
// 1st Row Entry
HashMap row1 = new HashMap();

// For Text type
row1.put("myString","Manoj Yeturu");

// For Numeric type
row1.put("myNumber",new Double(10000));

// For Date Type
row1.put("myDate",new Date());

// For Image Type. The url for image and label (for tooltip) properties
sre set
HashMap hm1Image = new HashMap();
hm1Image.put(CustomTableConstants.URL,baseUrl+"/images/action_nosh
ad.gif");

// Tool Tip
hm1Image.put(CustomTableConstants.LABEL,"Action_Noshad");
row1.put("myImage",hm1Image);

// For Money Type. The Currency and value properties are set
HashMap hm1Money = new HashMap();
hm1Money.put(CustomTableConstants.MONEY_CURRENCY_CODE,"USD");
hm1Money.put(CustomTableConstants.MONEY_VALUE,new Integer(3000));
row1.put("myMoney",hm1Money);

// For External Link, url, label (display string) and target
(Rightpane,_new etc) are set
HashMap externalLink1 = new HashMap();
externalLink1.put(CustomTableConstants.URL,"http://www.agile.com")
;
externalLink1.put(CustomTableConstants.LABEL,"Agile");
externalLink1.put(CustomTableConstants.TARGET,"_new");
row1.put("myExternalLink",externalLink1);
result.add(row1);
```

```
// 2nd Row Entry
HashMap row2 = new HashMap();

// For Text type
row2.put("myString","Venkat Tipparam");

// For Numeric type
row2.put("myNumber",new Double(50000));

// For Date Type
row2.put("myDate",(new Date()));

// For Image Type
HashMap hm2Image = new HashMap();
hm2Image.put(CustomTableConstants.URL,baseUrl +
"/images/addressdown.gif");

// Tool Tip
hm2Image.put(CustomTableConstants.LABEL,"Addressdown");
row2.put("myImage",hm2Image);

// For Money Type
HashMap hm2Money = new HashMap();
hm2Money.put(CustomTableConstants.MONEY_CURRENCY_CODE,"INR");
hm2Money.put(CustomTableConstants.MONEY_VALUE,new Integer(4000));
row2.put("myMoney",hm2Money);

// For External Link
HashMap externalLink2 = new HashMap();
externalLink2.put(CustomTableConstants.URL,"http://www.agile.com/s
ervices/support.asp");
externalLink2.put(CustomTableConstants.LABEL,"Supprt");
externalLink2.put(CustomTableConstants.TARGET,"_new");
row2.put("myExternalLink",externalLink2);
result.add(row2);

return result;
}
}
```

カスタム テーブル DX ソースのパッケージ化および配置

新しいテーブルに必要なクラスを開発した後は、次に示すようにクラスをパッケージ化して、配置します。

テーブル DX ソースをパッケージ化して配置する手順は、次のとおりです。

1. Java 開発環境または Java アーカイブ ツール (または JAR ツール) を使用して、カスタム アクション用の JAR ファイルを 1 つ以上作成します。JAR ファイルに、com.agile.px.ICustomTable という名前のファイルが格納された META-INF/services ディレクトリが含まれていることを確認します。このファイルは、カスタム アクション用の Java の完全修飾クラス名を 1 行に 1 クラスずつリストしたテキスト ファイルです。

   1 つのパッケージに複数のチャートを含めることができます。たとえば、com.agile.px.ICustomtable ファイルは次のような形式になります。

```
dashboard.chart.ActualVsBudgetedLaborCostTable
dashboard.chart.DashboardSampleTable
dashboard.chart.QueryDashboardPrograms
```

注意 JAR ファイル内のパスでは、大文字と小文字が区別されます。したがって、JAR ファイル内の META-INF フォルダの名前は、すべて大文字か、すべて小文字である必要があります。そうでない場合、カスタム アクションは配置されません。

2. Agile アプリケーション サーバがインストールされているコンピュータの agile_home/integration/sdk/extensions フォルダに JAR ファイルを格納します。

注意 クラスタ環境に複数のアプリケーション サーバがある場合は、クラスタ内の各サーバにダッシュボード拡張ファイルを配置する必要があります。

Java クライアントでのテーブル DX の設定

チャート タイプの DX と同様に、既存の [ダッシュボード管理] タブを使用することも、独自のオプション タブを作成してテーブル DX を追加することもできます。

タブにテーブルを追加する手順は、次のとおりです。

1. 新しいタブ (たとえば、前述の [ダッシュボード拡張機能] など) を定義します。
2. 新しいタブ ([ダッシュボード拡張機能]) で、 

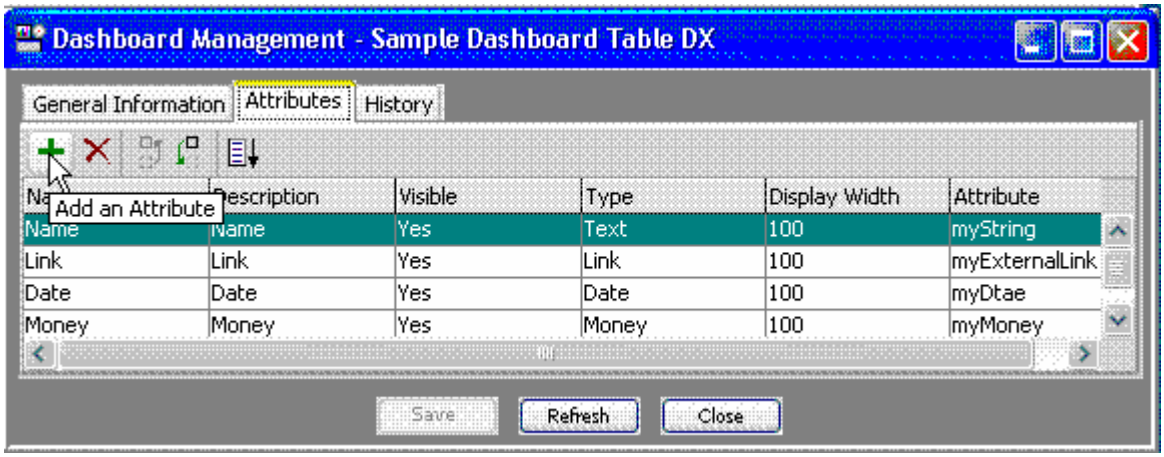
をクリックします。[ダッシュボード管理 - ダッシュボード拡張機能] ページが表示されます。
3. このページで、[新規ダッシュボード テーブル]  アイコンをクリックして [ダッシュボード テーブルの作成] ダイアログを開き、新しいテーブルを定義します。
4. [リスト タイプの表示] ドロップダウン リストから、[テーブル] を選択します。[ダッシュボード テーブルの作成] ダイアログに、次のフィールドが表示されます。
5. これらのフィールドを完成し、[OK] をクリックします。新しいテーブルが作成されます。

ダッシュボード テーブル	説明	可能な設定
名前	テーブルの名前を入力します。	文字列
説明	テーブルの説明を入力します。	文字列
リスト タイプの表示	テーブルのタイプがリストされます。[テーブル] を選択します。	[チャート]、[テーブル]、[カスタム]、[詳細検索]
ダッシュボード拡張機能	テーブル タイプの表示用に作成されたすべてのプロセス拡張がリストされます。	これらは、「カスタム テーブル DX ソースのパッケージ化および配置」で定義した属性です。
変数	Web クライアントで使用できるようにするかどうかを選択します。	[はい] または [いいえ]

テーブルにデータを追加する手順は、次のとおりです。

1. 「タブにテーブルを追加する手順」 (354 ページの「タブにテーブルを追加する手順」) で作成した新しいテーブルをダブルクリックします。
2. [属性] をクリックし、次に [属性の追加] アイコンをクリックして、新しい属性を作成します。

注意 現在、Agile では、テキスト、数値、画像、日付、通貨およびリンク タイプのデータがテーブル属性としてサポートされています。これらは、353 ページの「カスタム テーブル DX ソースのパッケージ化および配置」にリストおよび定義されています。



3. [一般情報] タブで、DX の属性名に [属性] フィールドをマップします。

注意 ここでは、[ダッシュボード] タブのテーブルに表示されることになる属性 (列) を定義しています。[属性] プロパティには、データ モデルと表示とのマッピングを定義します。たとえば、DX の属性名が myString で、選択されている属性タイプが [テキスト] の場合は、属性名が myString の [属性] フィールドをマップします。

4. 詳細は、『Agile PLM 管理者ガイド』の第 11 章を参照してください。

カスタム (URL) 拡張の定義

URL タイプのダッシュボード PX は、[ダッシュボード管理] から起動するように設定されます。カスタム拡張を定義する場合は、テーブル タイプとして [カスタム] を選択します。20-10 ページの「タブに URL を追加する手順」を参照してください。URL タイプのダッシュボード PX には、その他のマッピングは必要ありません。

注意 URL プロセス拡張は、[ダッシュボード管理] から起動されるように、プロセスの拡張ライブラリに定義されます。

タブに URL を追加する手順は、次のとおりです。

1. 新しいタブ (たとえば、前述の [ダッシュボード拡張機能] など) を定義します。
2. 新しいタブ ([ダッシュボード拡張機能]) で、 

をクリックします。[ダッシュボード管理 - ダッシュボード拡張機能] ページが表示されます。
3. このページで、[新規ダッシュボード テーブル]  アイコンをクリックして [ダッシュボード テーブルの作成] ダイアログを開き、新しいテーブルを定義します。
4. [リスト タイプの表示] ドロップダウン リストから、[カスタム] を選択します。[ダッシュボード テーブルの作成] ダイアログに、次のフィールドが表示されます。
5. [ダッシュボード テーブルの作成] ダイアログの各フィールドを完成し、[OK] をクリックします。

ダッシュボード テーブル	説明	可能な設定
名前	URL の名前を入力します。	文字列
説明	説明を入力します。	文字列
リスト タイプの表示	テーブルのタイプがリストされます。[カスタム] を選択します。	[チャート]、[テーブル]、[カスタム]、[詳細検索]
ダッシュボード拡張機能	カスタム タイプのリスト用に作成されたすべてのプロセス拡張がリストされます。	従業員ポータル、Yahoo、Google、プロセス拡張 URL
表示	Web クライアントで使用できるようにするかどうかを選択します。	[はい] または [いいえ]

第 22 章

Agile PLM クライアント機能と Agile API とのマッピング

扱うトピックは次のとおりです。

ログイン機能

次の表に、Agile アプリケーション サーバにログインするための一般的な機能を示します。

機能	対応するメソッド
Agile アプリケーション サーバ セッションのインスタンスを取得する	`AgileSessionFactory.getInstance()`
セッションを作成して、Agile アプリケーション サーバにログインする	`AgileSessionFactory.createSession()`
セッションを閉じて、Agile アプリケーション サーバとの接続を切断する	`IAgileSession.close()`

一般機能

次の表に、すべての Agile PLM ビジネス オブジェクトに適用される一般的な機能を示します。

機能	対応するメソッド
新規オブジェクトを作成する	IAgileSession.createObject()
既存のオブジェクトをロードする	IAgileSession.getObject()
オブジェクトを別のオブジェクトとして保存する	IDataObject.saveAs()
オブジェクトを削除する	IDataObject.delete() IFolder.delete() IQuery.delete()
オブジェクトの削除を取り消す	IDataObject.undelete()
オブジェクトのセル値を取得する	IDataObject.getValue()
オブジェクトにセル値を設定する	IDataObject.setValue()
オブジェクトのテーブルを取得する	IDataObject.getTable()
テーブルに行を追加する	ITable.createRow()
テーブルから行を削除する	ITable.removeRow()
オブジェクトに対する確認通知を取得する	ISubscribable.getSubscriptions()
確認通知イベントを有効にする	ISubscription.enable()
オブジェクトに対する確認通知を変更する	ISubscribable.modifySubscriptions()

検索機能

次の表に、サポートされている検索機能を示します。

機能	対応するメソッド
検索の名前を設定する	IQuery.setName()
検索をパブリックまたはプライベートにする	IQuery.setQueryType()
検索に検索タイプ (オブジェクト検索または使用箇所検索) を設定する	IQuery.setSearchType()
検索条件を設定および取得する	IQuery.setCriteria() IQuery.getCriteria()
検索を実行する	IQuery.execute()
検索で大文字と小文字を区別する	IQuery.setCaseSensitive()
検索を削除する	IQuery.delete()
検索を別の検索として保存する	IQuery.saveAs()

添付ファイル機能

次の表に、添付ファイルおよびファイル フォルダを使用するための機能を示します。

機能	対応するメソッド
ファイル フォルダ内のすべてのファイルをダウンロードする	`IFileFolder.getFile()`
[添付ファイル] タブにリストされている単一のファイルをダウンロードする	`IAttachmentFile.getFile()`
ファイル フォルダをチェックアウトする	`IFileFolder.checkOut()`
ファイル フォルダをチェックインする	`IFileFolder.checkIn()`
チェックアウトをキャンセルする	`IFileFolder.cancelCheckOut()`
アイテムを確定または未確定にし、そのアイテムの添付ファイルをロックまたはロック解除する	`IAttachmentContainer.setIncorporated()`

ワークフロー機能

次の表に、Agile PLM の送信可能なオブジェクトに対するワークフロー機能を示します。

機能	対応するメソッド
送信可能なオブジェクトを検証する	`IRoutable.audit()`
送信可能なオブジェクトのステータスを変更する	`IRoutable.changeStatus()`
オブジェクトを他の Agile PLM ユーザーに送信する	`IDataObject.send()`
送信可能なオブジェクトを承認する	`IRoutable.approve()`
送信可能なオブジェクトを却下する	`IRoutable.reject()`
送信可能なオブジェクトにコメントする	`IRoutable.comment()`
送信可能なオブジェクトの承認者およびオブザーバを追加または削除する	`IRoutable.addApprovers()` `IRoutable.removeApprovers()`

製造拠点機能

次の表に、製造拠点を使用するための機能を示します。

機能	対応するメソッド
アイテムに対して選択されている現在の製造拠点を取得する	IManufacturingSiteSelectable.getManufacturingSite()
アイテムに対するすべての製造拠点を取得する	IManufacturingSiteSelectable.getManufacturingSites()
すべての製造拠点を使用するようにアイテムを設定する	IManufacturingSiteSelectable.setManufacturingSite(ManufacturingSiteConstants.ALL_SITES)
アイテムは拠点別ではなく、すべての拠点に対して共通であることを指定する	IManufacturingSiteSelectable.setManufacturingSite(ManufacturingSiteConstants.COMMON_SITE)
特定の製造拠点を使用するようにアイテムを設定する	IManufacturingSiteSelectable.setManufacturingSite(site)

フォルダ機能

次の表に、フォルダを使用するためのフォルダ機能を示します。

機能	対応するメソッド
フォルダにアイテム (検索条件など) を追加する	IFolder.addChild()
フォルダのタイプ (パブリックまたはプライベート) を設定する	IFolder.setFolderType()
フォルダ名を設定する	IFolder.setName()
現在のユーザーのフォルダを取得する	IUser.getFolder()
フォルダからアイテムを削除する	IFolder.removeChild()
フォルダからすべてのオブジェクトをクリアする	IFolder.clear()
フォルダを削除する	IFolder.delete()

プログラム機能

次の表に、プログラムを使用するための機能を示します。

機能	対応するメソッド
プログラムを別のプログラムまたはテンプレートとして保存する	IProgram.saveAs()
プログラムを再スケジュールする	IProgram.reschedule()
リソース プールからユーザーを割り当てる	IProgram.assignUsersFromPool()
プログラムの所有権を別のユーザーに委譲する	IProgram.delegateOwnership()
プログラム リソースを入れ替える	IProgram.substituteResource()
基準を作成する	IProgram.createBaseline()
プログラムの基準表示を選択する	IProgram.selectBaseline()
プログラムをロックまたはロック解除する	IProgram.setLock()
ディスカッションに返信する	IMessage.reply()

管理機能

次の表に、Agile Java クライアントの管理ノードとプロパティを使用するための機能を示します。

機能	対応するメソッド
管理ノードを取得する	IAdmin.getNode()
管理ノードのすべてのサブノード (子ノード) を取得する	ITreeNode.getChildNodes()
管理ノードのすべてのプロパティを取得する	INode.getProperties()
管理ノードのプロパティに対する値を取得する	IProperty.getValue()
リスト フィールドに対する可能な値を取得する	IProperty.getAvailableValues()
すべての Agile PLM クラスを取得する	IAdmin.getAgileClasses(ALL)
トップレベルのすべての Agile PLM クラスを取得する	IAdmin.getAgileClasses(TOP)
インスタンス化可能なすべての Agile PLM クラスを取得する	IAdmin.getAgileClasses(CONCRETE)
特定のクラスに対するサブクラスのリストを取得する	IAgileClass.getSubclasses()
サブクラスの自動採番ソースを取得する	IAgileClass.getAutoNumberSources()
テーブルに対する属性の配列を取得する	IAgileClass.getTableAttributes()
テーブルのメタデータを取得する	IAgileClass.getTableDescriptor()

機能	対応するメソッド
Agile PLM リスト ライブラリを取得する	`IAdmin.getListLibrary()`
新しい Agile PLM リストを作成する	`IListLibrary.createAdminList()`
Agile PLM リストを取得する	`IListLibrary.getAdminList()`
すべての Agile PLM ユーザーを取得する	ユーザーの検索条件の作成
すべての Agile PLM ユーザー グループを取得する	ユーザー グループの検索条件の作成
ユーザーまたはユーザー グループを作成する	`IAgileSession.createObject()`
ユーザーまたはユーザー グループのプロパティを設定する	`IProperty.setValue()`
ユーザー パスワードを変更する	`IUser.changeApprovalPassword()` `IUser.changeLoginPassword()`