



Agile Product Lifecycle Management

SDK 開発者ガイド

v9.3.0.2

部品番号: B61299-01

2011 年 3 月

オラクル社の著作権について

Copyright © 1995, 2010, Oracle and/or its affiliates. All rights reserved.

このソフトウェアおよび関連ドキュメントの使用と開示は、ライセンス契約の制約条件に従うものとし、知的財産に関する法律により保護されています。ライセンス契約で明示的に許諾されている場合もしくは法律によって認められている場合を除き、形式、手段に関係なく、いかなる部分も使用、複写、複製、翻訳、放送、修正、ライセンス供与、送信、配布、発表、実行、公開または表示することはできません。このソフトウェアのリバース・エンジニアリング、逆アセンブル、逆コンパイルは互換性のために法律によって規定されている場合を除き、禁止されています。

ここに記載された情報は予告なしに変更される場合があります。また、誤りが無いことの保証はいたしかねます。誤りを見つけた場合は、オラクル社までご連絡ください。

このソフトウェアまたは関連ドキュメントが、米国政府機関もしくは米国政府機関に代わってこのソフトウェアまたは関連ドキュメントをライセンスされた者に提供される場合は、次の Notice が適用されます。

U.S. GOVERNMENT RIGHTS

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

このソフトウェアは様々な情報管理アプリケーションでの一般的な使用のために開発されたものです。このソフトウェアは、危険が伴うアプリケーション（人的傷害を発生させる可能性があるアプリケーションを含む）への用途を目的として開発されていません。このソフトウェアを危険が伴うアプリケーションで使用する際、このソフトウェアを安全に使用するために、適切な安全装置、バックアップ、冗長性（redundancy）、その他の対策を講じることは使用者の責任となります。このソフトウェアを危険が伴うアプリケーションで使用了ことに起因して損害が発生しても、オラクル社およびその関連会社は一切の責任を負いかねます。

Oracle は Oracle Corporation およびその関連企業の登録商標です。その他の名称は、それぞれの所有者の商標または登録商標です。

このソフトウェアおよびドキュメントは、第三者のコンテンツ、製品、サービスへのアクセス、あるいはそれらに関する情報を提供することがあります。オラクル社およびその関連会社は、第三者のコンテンツ、製品、サービスに関して一切の責任を負わず、いかなる保証もいたしません。オラクル社およびその関連会社は、第三者のコンテンツ、製品、サービスへのアクセスまたは使用によって損失、費用、あるいは損害が発生しても、一切の責任を負いかねます。

目次

オラクル社の著作権について	ii
リリース 9.3.0.2 での新機能	xx
導入	1
Agile SDKについて	1
SDKのコンポーネント	2
クライアント側のコンポーネント	2
サーバー側のコンポーネント	2
SDKのアーキテクチャ	3
システム要件	4
Javaの要件	4
メモリー不足例外を回避するためのJava Virtual Memory (JVM) パラメータ	4
Agile SDKインストール・フォルダ	5
Agile PLMシステムの確認	5
Agile PLMビジネス・オブジェクト	5
Agile APIの開始	7
Agile APIの概要	7
Agile APIのクラスとインタフェースのタイプ	7
ネットワーク・クラスのロード	8
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	9
Agile APIプログラムのパッケージ化	9
Agile APIファイルの配布	9
サンプル・プログラム	9
Agile APIプログラムの開始	9
Agile APIライブラリのクラス・パスの設定	10
Agile APIクラスのインポート	10
セッションの作成およびログイン	10
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	12
Agile Webサービスからのセッションの作成	12
Agile PLMオブジェクトのロードおよび作成	13
オブジェクトのロード	13
オブジェクト・タイプの指定	14
オブジェクト・パラメータの指定	14
異なるタイプのオブジェクトのロード	15
オブジェクトの作成	18

Agile PLMクラスの使用.....	19
ユーザー定義サブクラスのオブジェクトの作成	20
自動採番の使用	20
必須フィールドの設定.....	22
異なるタイプのオブジェクトの作成.....	24
Agile PLMオブジェクトの状態の確認.....	28
関連オブジェクトへの値の継承	29
オブジェクトを新規オブジェクトとして保存.....	29
オブジェクトの共有	30
オブジェクトの削除および削除取消	31
セッションを閉じる	33
検索条件の作成およびロード.....	35
検索条件について.....	35
検索条件の作成.....	35
フォルダへの検索条件の保存.....	36
順序付け（並べ替え）された、または順序付けされていない検索結果の生成	37
パラメータ検索の作成.....	38
検索条件作成時の検索属性の指定	39
ワークフロー検索の指定	40
検索条件の指定.....	41
検索条件.....	41
クエリ言語のキーワード	42
検索属性の指定	42
検索可能属性の取得.....	43
関係演算子の使用	44
Unicodeエスケープ・シーケンスの使用	45
検索範囲、検索範囲に含まれない、InおよびNot In演算子の使用	45
ネストされた条件を使用したオブジェクト・リストの値の検索.....	46
条件ライブラリから選択した条件のSDK検索での使用	47
SDK検索での関係とコンテンツの使用	47
添付ファイル内の単語または語句の検索	48
検索条件の日付の書式設定	49
論理演算子の使用.....	49
Like演算子でのワイルドカード文字の使用	50
検索条件での括弧の使用	50
多数のオブジェクトを含むリストに対する検索条件の設定.....	51
検索条件でのSQL構文の使用	51
SQLワイルドカードの使用.....	53
SQL構文の使用による検索結果の並べ替え.....	53
検索条件の結果属性の設定.....	54
結果属性の指定	60

コンテンツ転送作成者名の取得	61
拠点関連オブジェクトとAMLの重複する結果	62
検索結果の使用	62
検索結果の並べ替え	62
検索結果のデータ・タイプ	62
大量の検索結果の管理	63
検索のパフォーマンス	63
使用箇所検索条件の作成	63
検索条件のロード	64
検索条件の削除	65
簡単な検索条件の例	66
テーブルの使用	69
テーブルについて	69
テーブルの取得	70
新規およびマージされた「関係」テーブルへのアクセス	71
「関係」テーブルへのアクセス	71
マージされたテーブルへのアクセス	72
マージされた「関係.影響元」テーブルへのアクセス	72
マージされた「関係.影響先」テーブルへのアクセス	72
マージされた「関係.参照」テーブルへのアクセス	72
読取り専用テーブルの使用	73
テーブルのメタデータの取得	73
テーブル行の追加	73
「BOM」テーブルへのアイテムの追加	74
「添付ファイル」テーブルへの添付ファイルの追加	74
「製造元」テーブルへの製造元部品の追加	75
「対象アイテム」テーブルへのアイテムの追加	75
「スケジュール」テーブルへのタスクの追加	76
複数のテーブル行の追加および更新	76
「BOM」テーブルへの複数アイテムの追加	76
複数のBOM行の更新	77
テーブル行の繰返し処理	78
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	79
多数の検索結果の繰返し時におけるテーブル行の更新	80
テーブル行の並べ替え	80
テーブル行の削除	81
行に対する参照オブジェクトの取得	82
行のステータス・フラグの確認	86
「ページ1」、「ページ2」および「ページ3」の使用	87

レッドライン	88
レッドラインの変更の削除	89
一括モードでのレッドラインの変更の削除	90
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	90
ICell.getOldValueの使用	91
データ・セルの使用	93
データ・セルについて	93
データ・タイプ	93
ユーザーのディスカバリ 権限の確認	94
セルが読取り専用かどうかの確認	95
値の取得 - 9.3	95
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	97
値の設定	97
ロックされたオブジェクトの例外の捕捉	98
リスト値の取得および設定	98
シングルリスト・セルの値の取得および設定	99
マルチリスト・セルの値の取得および設定	99
カスケード・リストの値の取得および設定	100
参照指示セルの使用	102
フォルダの使用	103
フォルダについて	103
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	104
フォルダのロード	105
フォルダの作成	105
フォルダ・タイプの設定	106
フォルダ要素の追加および削除	106
フォルダ要素の追加	107
フォルダ要素の削除	107
フォルダ要素の取得	107
フォルダの削除	110
アイテム、BOMおよびAMLの使用	111
アイテム、BOMおよびテーブルについて	111
アイテムの使用	111
アイテムのリビジョンの取得および設定	111
リビジョンの確定済ステータスの変更	114
BOMの使用	114
BOMへのアイテムの追加	115

BOMの展開.....	115
別のBOMへのBOMのコピー.....	116
BOM関連の製品レポートの作成.....	117
BOMのレッドライン.....	119
リリース済のアセンブリ・アイテムの取得.....	119
変更指示の作成.....	119
変更指示の「対象アイテム」タブへのアイテムの追加.....	120
「BOMのレッドライン」テーブルの変更.....	120
AMLの使用.....	121
「製造元」テーブルへの承認済製造元の追加.....	122
AMLのレッドライン.....	123
APINameフィールドを使用したPLMメタデータへのアクセス.....	125
APINameフィールドについて.....	125
APINameフィールドへの名前の割当て.....	126
APIName検証ルール.....	126
APINameフィールドを使用したメタデータへのアクセス.....	127
APINameフィールドをサポートするAPI.....	127
APINameフィールドを取得するSDK API.....	131
ルート管理者ノードのAPI名.....	132
API名の例.....	133
Agile PLMオブジェクトの確認通知.....	139
ユーザー確認通知について.....	139
確認通知イベント.....	139
確認通知権限.....	140
確認通知.....	140
SDKを使用した通知の送信.....	140
確認通知の対象とするオブジェクトの削除.....	141
オブジェクトに対する確認通知の取得.....	142
オブジェクトに対する確認通知の変更.....	143
確認通知での属性の使用可能化.....	144
親属性と子属性.....	145
「確認通知」テーブルの使用.....	145
製造拠点の管理.....	149
製造拠点について.....	149
拠点へのアクセスの管理.....	149
製造拠点の作成.....	150
製造拠点のロード.....	150
アイテムの「拠点」テーブルの取得.....	151

「拠点」テーブルへの製造拠点の追加	151
アイテムの現在の製造拠点の選択	152
拠点の無効化	153
リストの使用	155
リストについて	155
リスト・ライブラリ	155
シングルのリスト	156
カスケード・リスト	157
マルチリストのリスト	158
IAgileListを使用するメソッド	158
リスト値の選択	159
ライフサイクル・フェーズ・セルの使用	161
動的リストの使用	162
列挙可能リストと列挙不可能リスト	162
列挙不可能なPG&Cリスト	163
リスト・ライブラリからのリストの選択	163
カスタム・リストの作成	165
簡易リストの作成	165
既存リストの変更による新規リストの自動作成	167
カスケード・リストの作成	167
リストのデータ・タイプの確認	170
リストの変更	170
リストへの値の追加	171
リスト値の破棄	171
リスト名と説明の設定	172
カスケード・リストのレベル名の設定	172
リストの有効化または無効化	172
リストの削除	173
リスト値の変更および削除	173
IAgileListオブジェクトのコンテンツの印刷	174
添付ファイルとファイル・フォルダ・オブジェクトの使用	177
添付ファイルとファイル・フォルダについて	177
ファイル・フォルダの使用	178
ファイル・フォルダのクラスとサブクラス	178
ファイル・フォルダのテーブルと定数	179
ファイル・フォルダ・オブジェクトの作成	179
「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・オブジェクトの作成	182
ファイル・フォルダの「ファイル」テーブルの使用	182

IAttachmentFileを使用したAgile PLMファイル格納庫内のファイルへのアクセス	183
オブジェクトの「添付ファイル」テーブルの使用	184
ICheckoutableを使用したファイルのチェックインとチェックアウト	184
アイテムのリビジョンの指定	185
リビジョンが確定済かどうかの確認	185
ファイル・フォルダのチェックアウト	186
ファイル・フォルダのチェックアウトのキャンセル	187
「添付ファイル」テーブルへのファイルおよびURLの追加	188
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	190
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	191
添付ファイルの取得	192
添付ファイルとファイル・フォルダの削除	193
サムネイルの使用	193
サムネイルへのアクセス	193
サムネイルの再生成	194
マスター・サムネイルの設定	195
サムネイルの置換	195
サムネイルの順序付け	196
「添付ファイル」タブへのファイルの追加時におけるサムネイルの生成	196
デザイン・オブジェクトの使用	196
デザイン・オブジェクトの追加およびロード	197
バージョン固有のデザイン・オブジェクト間関係の管理	197
特定バージョンのデザイン・オブジェクト間への関係の追加	197
特定バージョンのデザイン・オブジェクト間への関係の削除	198
特定バージョンのデザイン・オブジェクト間への関係の取得	198
特定バージョンのデザイン・オブジェクト間への関係の編集	198
特定バージョンのデザイン・オブジェクトのページ	199
使用箇所検索条件を使用したデザイン・オブジェクト配置の検索	199
SDKを使用したデータのインポートとエクスポート	201
データのインポートとエクスポートについて	201
インポート・データの検証とデータのインポート	201
SDKを使用した検証の起動およびデータのインポート	202
SDKからのデータのエクスポート	204
SDKからのエクスポートの起動	204
ワークフローの管理	207
ワークフローについて	207
変更管理プロセス	207
動的なワークフロー機能	208
変更のステータスがワークフロー機能に与える影響	208
ユーザー権限がワークフロー機能に与える影響	208
ワークフローの選択	209

承認者の追加および削除.....	210
「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.....	212
ルーティング可能なオブジェクトの承認.....	212
ルーティング可能なオブジェクトの却下.....	213
ルーティング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.....	214
転送元ユーザーを代行するユーザーによるルーティング可能なオブジェクトの承認.....	215
ルーティング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.....	216
ルーティング可能なオブジェクトを承認する 2 番目の署名の指定.....	217
ルーティング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.....	219
変更の承認または却下.....	219
変更のコメント.....	220
変更の検証.....	221
オブジェクトのワークフロー・ステータスの変更.....	222
選択したユーザーへのAgileオブジェクトの送信.....	224
ユーザー・グループへのAgileオブジェクトの送信.....	225
品質の管理および追跡.....	227
品質管理について.....	227
品質関連のAPIオブジェクト.....	227
品質関連の役割と権限.....	228
顧客の使用.....	228
顧客について.....	228
顧客の作成.....	228
顧客のロード.....	229
顧客を別の顧客として保存.....	229
製品サービス依頼の使用.....	230
問題レポートについて.....	230
不具合レポートについて.....	230
製品サービス依頼の作成.....	230
品質分析者への製品サービス依頼の割当て.....	231
製品サービス依頼への対象アイテムの追加.....	231
製品サービス依頼への関連PSRの追加.....	232
品質変更要求の使用.....	233
品質変更要求の作成.....	233
品質管理者への品質変更要求の割当て.....	234
品質変更要求を変更として保存.....	234
PSRおよびQCRでのワークフロー機能の使用.....	235
PSRおよびQCRのワークフローの選択.....	235
プロジェクトの作成および管理.....	237
プロジェクトおよびプロジェクト・オブジェクトについて.....	237
プロジェクト・オブジェクトの動作の相違点.....	238

プロジェクトの作成.....	238
PPMオブジェクトに対するルールの追加	240
プロジェクトのロード.....	241
プロジェクト・テンプレートの使用	242
テンプレートを使用した新規プロジェクトの作成.....	242
プロジェクトの作成および所有権の変更	243
プロジェクトをテンプレートとして保存	244
プロジェクトのスケジュール.....	245
プロジェクトの基準の使用	247
別のユーザーへのプロジェクト所有権の委譲	248
プロジェクトのチームへのリソースの追加	249
プロジェクト・リソースの入れ替え	251
プロジェクトのロックまたはロック解除	252
ディスカッションの使用.....	252
ディスカッションの作成.....	253
ディスカッションへの返信	254
ディスカッションへの参加.....	257
アクション・アイテムの作成	257
Product Cost Managementの使用	259
概要	259
価格オブジェクトの使用.....	260
価格の管理	260
価格オブジェクトの作成	261
デフォルト	261
アイテム・リビジョンの指定.....	261
公表価格の作成	262
価格オブジェクトのロード.....	262
価格ラインの追加.....	262
価格変更の作成.....	264
サプライヤの使用.....	265
サプライヤのロード	265
サプライヤ・データの変更	265
ソーシング・プロジェクトの使用	266
サポートされているAPIメソッド.....	267
既存のソーシング・プロジェクトのロード	268
数量割引の指定によるソーシング・プロジェクトの作成.....	268
数量割引および価格期間の指定によるソーシング・プロジェクトの作成	269
オブジェクト、テーブルおよび属性へのアクセスと変更.....	270

ソーシング・プロジェクトの「カバー・ページ」の値の設定	270
PCMのネスト・テーブルの理解	271
ソーシング・プロジェクトの親テーブルとネスト子テーブルの定数	271
ソーシング・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と変更	272
ソーシング・プロジェクトのステータスへのアクセスと変更	272
ソーシング・プロジェクトでのデータの管理	273
ソーシング・プロジェクトのアイテムの数量の設定	273
ソーシング・プロジェクトでの数量ロールアップの実行	274
ソーシング・プロジェクトでのコスト・ロールアップの実行	274
ソーシング・プロジェクトでの価格検索の実行	275
ソーシング・プロジェクトでのパートナーの設定	280
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	281
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	282
見積依頼の使用	283
サポートされているAPIメソッド	284
ソーシング・プロジェクトに対する見積依頼の作成	284
既存の見積依頼のロード	285
ソーシング・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	285
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	286
見積依頼での価格検索の実行	286
見積依頼回答の処理	287
製品の規制および適合性の管理	291
Agile Product Governance & Complianceについて	291
Agile PG&Cのインタフェースとクラス	292
Agile PG&Cの役割	292
デklarレーション、含有基準およびサブスタンスの作成	293
デklarレーションの作成	293
含有基準の作成	294
サブスタンスの作成	295
サブパートの作成	295
サブスタンス・グループの作成	296
マテリアルの作成	296
サブスタンスの作成	296
デklar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	297
デklarレーションへのサブスタンスの追加	298
BOS（サブスタンス構成表）の構造	299
サブスタンスの追加に関するルール	299
存在しないサブパートとマテリアルの追加	300
サブスタンスを追加する例	300
均質材のデklar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	301
サブスタンス・デklar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	302
含有基準へのサブスタンスの追加	304
デklarレーションへの含有基準の追加	305
含有基準の追加に関するルール	305

デklarレーションの送信	306
デklarレーションの完成	307
適合性管理者へのデklarレーションの提出	308
デklarレーションの公表	308
重量値の取得および設定	309
製造元部品のサブスタンス組成の追加	310
適合性データのロールアップ	312
IPGCRollupインタフェースの理解	313
Dateパラメータを渡す場合	313
IPGCRollupインタフェースの使用	313
アイテムに関する集合データのロールアップ	314
MPNに関する集合データのロールアップ	314
アイテム・オブジェクトに対する「適合性判定値」フィールドの値の設定	315
デklarレーション・オブジェクトに対する「適合性判定値」フィールドの値の設定	316
例外の処理	317
例外について	317
例外定数	318
エラーコードの取得	318
バルクAPIでのエラー・コードの無効化と有効化	318
エラー・メッセージの取得	319
警告メッセージの無効化および有効化	320
APIExceptionがエラーではなく警告であることの確認	321
有効または無効にした警告の状態の保存および復元	321
Agile APIで自動的に無効にした警告の削除	322
管理タスクの実行	323
Agile PLM管理について	323
Agile PLMの管理に必要な権限	324
管理インタフェース	324
IAdminインスタンスの取得	325
ノードの使用	325
「クラス」ノードの使用	330
Agile PLMクラスの管理	330
具象クラスと抽象クラス	332
クラスの参照	333
クラスのターゲット・タイプの識別	334
属性の使用	334
属性の参照	335
属性の参照 - 9.3	336
属性の取得	337

個々の属性の取得.....	338
属性のプロパティの編集.....	338
ユーザー定義属性の使用.....	338
管理ノードのプロパティの使用.....	339
ユーザーの管理.....	340
すべてのユーザーの取得.....	340
ユーザーの作成.....	340
サプライヤ・ユーザーの作成.....	341
ユーザーを新規ユーザーとして保存.....	342
有効期限が切れたパスワードの確認.....	342
ユーザー設定の構成.....	343
ユーザー・パスワードのリセット.....	344
ユーザーの削除.....	344
ユーザー・グループの管理.....	345
すべてのユーザー・グループの取得.....	345
ユーザー・グループの作成.....	345
ユーザーの「ユーザー・グループ」テーブルへのユーザー・グループの追加.....	347
ユーザー・グループ内のユーザーのリスト.....	347
プロセス拡張の開発.....	349
プロセス拡張について.....	349
カスタム自動採番ソースの開発.....	350
カスタム自動採番ソースの定義.....	350
カスタム自動採番ソースのパッケージ化および配置.....	351
Javaクライアントでのカスタム自動採番ソースの設定.....	351
サブクラスへの自動採番ソースの割当て.....	353
カスタム・アクションの開発.....	353
カスタム・アクションの定義.....	353
PLMクライアントでの改行の書式設定.....	354
カスタム・アクションとユーザー・セッション.....	354
カスタム・アクションのパッケージ化および配置.....	355
カスタム・アクションの役割と権限.....	355
プロセス拡張を設定するためのユーザー権限.....	355
Agile Javaクライアントでのカスタム・アクションの設定.....	356
プロセスの拡張ライブラリの使用.....	356
クラスへのプロセス拡張の割当て.....	358
ワークフロー・ステータスへのプロセス拡張の割当て.....	358
URLベースのプロセス拡張の定義と配置.....	359
URLベースのプロセス拡張を作成する前の注意.....	360
URLベースのプロセス拡張の定義.....	360
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.....	361

ターゲット・システムからのAgile PLMセッションの作成.....	361
HTTPリクエストからのAgile PLMオブジェクトの取得.....	363
Agile PLMクラスの識別属性.....	363
外部レポートの作成.....	365
クラスタ環境でのプロセス拡張の配置.....	366
サード・パーティJARファイルをコピーするためのベスト・プラクティス.....	366
プロセス拡張に関するよくある質問.....	369
Webサービス拡張の開発.....	373
Webサービス拡張について.....	373
主な機能.....	374
WSXアーキテクチャ.....	374
Webサービスについて.....	374
Webサービス・アーキテクチャ.....	375
セキュリティ.....	376
ツール.....	376
Webサービスに関する追加情報の検索.....	377
Webサービスの開発および配置.....	377
デプロイメント・ディスクリプタについて.....	378
予約されているWebサービス名.....	378
Webサービスの使用.....	379
Webサービスのエントリ・ポイントの定義.....	379
ユーザーの認証.....	379
クライアント/サーバー・アクセスでのシングル・サインオン・クッキーの使用.....	380
配置アーキテクチャ.....	380
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.....	380
シングル・サインオン・クッキーの取得.....	381
SOAPバインディング・スタブ・コードの変更.....	381
MyFirstWebServiceサンプルの環境の準備.....	382
サンプル作成用ツールのダウンロード.....	382
Java SDKのインストール.....	382
Antのインストール.....	383
MyFirstWebServiceサンプルの作成.....	383
Webサービス・クライアントについて.....	384
クライアント・プログラミング言語.....	384
Webサービスへのアクセス.....	385
MyFirstClientの作成.....	385
SOAPリクエストの生成.....	385
SOAPリクエストの発行.....	386
SOAPレスポンスの処理.....	386

MyFirstClientの実行	387
WSX内部におけるAgileセッションの作成	387
Microsoft .NETの相互運用性	388
Webサービス拡張に関するよくある質問	388
ダッシュボード管理拡張の開発	393
ダッシュボード管理拡張について	393
ダッシュボード管理拡張の役割と権限	394
カスタム・チャート・ダッシュボード管理拡張の開発	394
ChartDataModelおよびChartDataSetの理解	394
カスタム・チャートDXのデータ・ソースの定義	394
カスタム・チャートDXソースのパッケージ化および配置	396
JavaクライアントでのチャートDXの設定	396
Agile Webクライアントでのオプション・タブの表示	397
カスタム・テーブル・ダッシュボード管理拡張の開発	398
CollectionおよびCustomTableConstantsの理解	398
カスタム・テーブルDXのデータ・ソースの定義	399
カスタム・テーブルDXソースのパッケージ化および配置	401
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。	401
JavaクライアントでのテーブルDXの設定	402
タブにテーブルを追加する手順は、次のとおりです。	402
テーブルにデータを追加する手順は、次のとおりです。	403
カスタム（URL）拡張の定義	403
Agile PLMのイベントおよびイベント・フレームワーク	405
Agile PLMのイベントおよびイベント・フレームワークの理解	405
Agile PLMイベントの主要なコンポーネント	405
イベント・タイプ	406
イベント・ハンドラおよびハンドラ・タイプ	407
イベント確認通知受信者	407
イベント・トリガーおよびトリガー・タイプ	407
同期および非同期の実行モード	408
イベント・エラー処理ルール	408
イベントの順序	408
イベントの生成、処理および実行	409
イベント・コンテキスト・オブジェクトの使用	411
イベント・コンテキスト・オブジェクトの理解	411
永続データと一時データ	411
イベント情報オブジェクト	412
イベント・スクリプト・オブジェクト	414
イベント情報オブジェクトおよびイベント・スクリプト・オブジェクトの使用	415

基本イベント・アクションの使用	415
基本イベント情報オブジェクト - Java PX	415
基本イベント・スクリプト・オブジェクト - スクリプトPX	416
一般的なオブジェクト・アクションの使用	418
一般的なオブジェクト・アクション - Java PX	418
一般的な基本イベント・スクリプト・オブジェクトの使用	421
テーブル・アクションと関係アクションの使用	423
テーブル・アクションと関係アクション - Java PX	424
テーブル・アクションと関係アクション - スクリプトPX	426
ワークフロー・オブジェクト・アクションの使用	428
ワークフロー・オブジェクト・アクション - Java PX	428
ワークフロー・オブジェクト・アクション - スクリプトPX	431
特定のオブジェクトベース・アクションの使用	434
特定のオブジェクトベース・アクション - Java PX	434
特定のオブジェクトベース・アクション - スクリプトPX	434
ファイル・アクションおよび添付ファイル・オブジェクト・アクションの使用	434
ファイル・アクションおよび添付ファイル・オブジェクト・アクション - Java PX	435
ファイル・アクションおよび添付ファイル・オブジェクト・アクション - スクリプトPX	435
Product Governance & Complianceアクションの使用	437
Product Governance & Complianceアクション - Java PX	437
Product Governance & Complianceアクション - スクリプトPX	437
その他のオブジェクト・アクションの使用	437
その他のオブジェクト・アクション - Java PX	437
その他のオブジェクト・アクション - スクリプトPX	437
PLMクライアントでのイベント統合ポイントの使用	437
イベント統合ポイント - Java PX	438
イベント統合ポイント - スクリプトPX	438
Java PXハンドラとスクリプトPXハンドラに関するガイドライン	438
Agile PLM管理者との作業	438
イベントJava PXおよびイベント・スクリプトPXのテスト	440
Java PX、スクリプトPXおよび通知ハンドラのトリガーに関するガイドライン	440
一般的なオブジェクト・アクション	440
ワークフロー・アクション	441
ファイル・アクションおよび添付ファイル・アクション	441
イベントに関するよくある質問	442
Agile PLMクライアント機能とAgile APIとのマッピング	447
ログイン機能	447
一般機能	448
検索機能	448
添付ファイル機能	449
ワークフロー機能	449
製造拠点機能	450
フォルダ機能	450
プロジェクト機能	451

管理機能	451
リリース 9.2.1 以前のテーブル定数のリリース 9.2.2 以降への移行.....	453
9.2.2 テーブル定数にマップされたプレリリース 9.2.2 のテーブル定数	453
削除されたプレリリース 9.2.2 のテーブル定数	456
イベント・フレームワークへのカスタム・プロセス拡張の移行.....	457
この付録について.....	457
カスタムPXとJava PXの理解.....	457
PXフレームワークでのカスタムPX	457
イベント・フレームワークでのプロセス拡張.....	457
イベント・フレームワークに移行可能なカスタムPX.....	458
移行タスク・リスト.....	458
タスク 1: カスタムPXコードの変更	458
カスタムPXコード	458
Java PXコード	459
タスク 2: 変更したコードのパッケージ化および配置.....	459
タスク 3: イベント・フレームワークでのイベントの設定.....	460
イベントの作成	460
イベント・ハンドラの作成.....	462
イベント確認通知受信者の作成.....	463
トリガー・タイプ、実行モード、順序およびエラー処理ルールの設定.....	464
タスク 4: 移行したPXのイベント・フレームワークでのテスト.....	466
タスク 5: プロセス拡張ライブラリからのカスタムPXの削除.....	467
タスク 6: PLM管理者への通知.....	467
イベント・フレームワークでのGroovyの実装.....	469
この付録について.....	469
Groovyとは	469
スクリプトPXとJava PXの使用目的	469
情報源.....	469
イベント・フレームワークの実装.....	470
スクリプトの起動.....	470
ユースケース.....	470

はじめに

Agile PLMマニュアル・セットにはAdobe® Acrobat PDFファイルが含まれています。[Oracle Technology Network \(OTN\) Webサイト](http://www.oracle.com/technology/documentation/agile.html) (<http://www.oracle.com/technology/documentation/agile.html>) には、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Agile PLMマニュアル・フォルダにAgile PLM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) (<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) (<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、My Oracle Support (<https://support.oracle.com>) にお問い合わせください。

注意 Agile PLM マニュアルに関する問題について Oracle 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。

Readme

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) (<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は、[Oracle University Webページ](http://www.oracle.com/education/chooser/selectcountry_new.html) (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 JAWS は括弧だけの行を読まない場合があります。

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する評価や言及は行っておりません。

リリース 9.3.0.2 での新機能

このリリースでは、次の新機能と拡張機能が実装されました。

- **有効期限が切れたパスワードの確認** - このリリースでは、有効期限が切れたパスワードに関連する Agile API エラーの確認方法の例 (340 ページの「[ユーザーの管理](#)」内) が変更されました。342 ページの「[有効期限が切れたパスワードの確認](#)」を参照してください。
- **多数のオブジェクトを含むリストに対する検索パフォーマンスの向上** - このリリースでは、多数のオブジェクトを含むリストに対するパフォーマンス向上のために検索条件を設定する方法について、新しいセクションが追加されました。51 ページの「[多数のオブジェクトを含むリストに対する検索条件の設定](#)」を参照してください。
- **Weblogic Server への依存/共有ライブラリの配置** - このリリースでは、依存 JAR ファイルを配置するための適用可能な手順が変更されました。366 ページの「[依存 JAR ファイルの配置](#)」を参照してください。
- **セッションの作成およびログイン** - Weblogic Server で実行されている Agile PLM に LDAP ユーザーがログインする場合の JVM パラメータの設定に関する注意が追加されました。「セッションの作成およびログイン」を参照してください。

リリース 9.3.0.1 リビジョン 2 での変更

380 ページの「[クライアント/サーバー・アクセスでのシングル・サインオン・クッキーの使用](#)」に記載されていた SiteMinder および SAP ポータルへの言及が削除されました。

リリース 9.3.0.1 での変更

このリリースで実装された新機能と拡張機能は、SDK を使用した検索条件の作成とロード、および表の使用に対処しています。これらの変更は、35 ページの第 3 章「[検索条件の作成およびロード](#)」および 69 ページの第 4 章「[テーブルの使用](#)」に記載されています。要約は次のとおりです。

- **SDK によるワークフロー関連の検索条件のサポート** - このリリースでは、PLM プロセスにおけるワークフローのステータスをチェックする検索条件を作成する際に、ワークフロー属性の使用がサポートされます。40 ページの「[ワークフロー検索の指定](#)」を参照してください。
- **SDK によるネストされた条件を使用したオブジェクト・リスト検索実行のサポート** - このリリースでは、この機能の動作とパフォーマンスが向上しています。46 ページの「[ネストされた条件を使用したオブジェクト・リストの値の検索](#)」を参照してください。
- **SDK による検索での関係とコンテンツのサポート** - IQuery インタフェースの機能拡張により、SDK を使用して次の検索条件を作成できます。詳細および使用例は、47 ページの「[SDK 検索での関係とコンテンツの使用](#)」を参照してください。
 - オブジェクトの「関係」属性
 - プログラム・オブジェクトの「コンテンツ」属性
 - 転送オブジェクトの「選択されたコンテンツ」属性

詳細、手順および例は、47 ページの「[SDK 検索での関係とコンテンツの使用](#)」を参照してください。

- **SDK による PLM の条件ライブラリから選択した条件を使用した検索のサポート** - この機能は、以前のリリースの SDK でサポートされていました。このリリースでは、例が変更されています。47 ページの「[条件ライブラリから選択した条件の SDK 検索での使用](#)」を参照してください。
- **SDK によるレッドライン変更の一括削除のサポート** - この操作は、IRedlinedTable と呼ばれる新規インタフェースによってサポートされます。90 ページの「[一括モードでのレッドラインの変更の削除](#)」を参照してください。

注意 Agile PLM リリース 9.2.2 以降の AgileAPI.jar の PG&C 定数と関係テーブル機能は、それ以前のバージョンの AgileAPI.jar と互換性がありません。

この章のトピック

▪ Agile SDKについて	1
▪ SDKのコンポーネント	2
▪ SDKのアーキテクチャ	3
▪ システム要件	4
▪ Javaの要件	4
▪ Agile SDKインストール・フォルダ	5
▪ Agile PLMシステムの確認	5
▪ Agile PLMビジネス・オブジェクト	5

Agile SDKについて

Agile Software Development Kit (SDK) は、Java アプリケーション・プログラミング・インタフェース (API) 、サンプル・アプリケーションおよびマニュアルのコレクションで、Agile アプリケーション・サーバーの機能にアクセスしたり、サーバーの機能を拡張するカスタム・アプリケーションの構築を可能にします。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 Enterprise Resource Planning (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・サーバーに対するバッチ操作の実行

Agile SDK には、次のモジュールが組み込まれています。

- **Agile API** - Agile ビジネス・オブジェクトを公開するインタフェースを備えた 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、次に定義するように、Java スクリプト、Web サービス、PX、sPX または WSX 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 (カスタム 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です。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。これらの PX は、Java プログラミング言語を使用して実装されます。
- **イベント** - このフレームワークは、Java プロセス拡張 (Java PX) およびスクリプト・プロセス拡張 (スクリプト PX) をサポートします。カスタム PX と同様に、Agile PLM の顧客が Agile PLM クライアントの機能を簡単に拡張してイベントを管理できるようにします。Java PX は Java を使用して実装され、スクリプト PX は Groovy と呼ばれるスクリプト言語を使用して実装されます。Groovy は、Java プラットフォーム用のオブジェクト指向のプログラミング言語で、Java プログラミング言語のかわりに使用できます。
- **Web サービス拡張 (WSX)** - Agile PLM の顧客が Agile PLM サーバーの機能を拡張し、顧客固有のソリューションを Web サービスとして公開できるようにするフレームワークです。

SDKのコンポーネント

Agile SDK には、クライアント側のコンポーネントとサーバー側のコンポーネントがあります。

クライアント側のコンポーネント

Agile SDK のクライアント側のコンポーネントは次のとおりです。

ドキュメント

- Agile SDK 開発者ガイド (このマニュアル)
- API リファレンス・ファイル (API メソッドが記述されている、Javadoc で生成された HTML ファイル)
- サンプル・アプリケーション

注意 Agile API HTMLリファレンス・ファイルおよびサンプル・アプリケーションは、SDK_samples.zipフォルダにあります。このフォルダは、Oracle® E-Delivery Webサイト (<http://edelivery.oracle.com/>) のOracle Agile Product Lifecycle Managementメディア・パック内にあります。メディア・パックおよびメディア・パックのコンテンツにアクセスする手順の詳細は、システム管理者に問い合わせるか、Agile PLM インストール・ガイドを参照してください。

インストール

- Agile API ライブラリ (AgileAPI.jar)
- Java プロセス拡張 API ライブラリ (pxapi.jar)
- Apache Axis ライブラリ (axis.jar)

サーバー側のコンポーネント

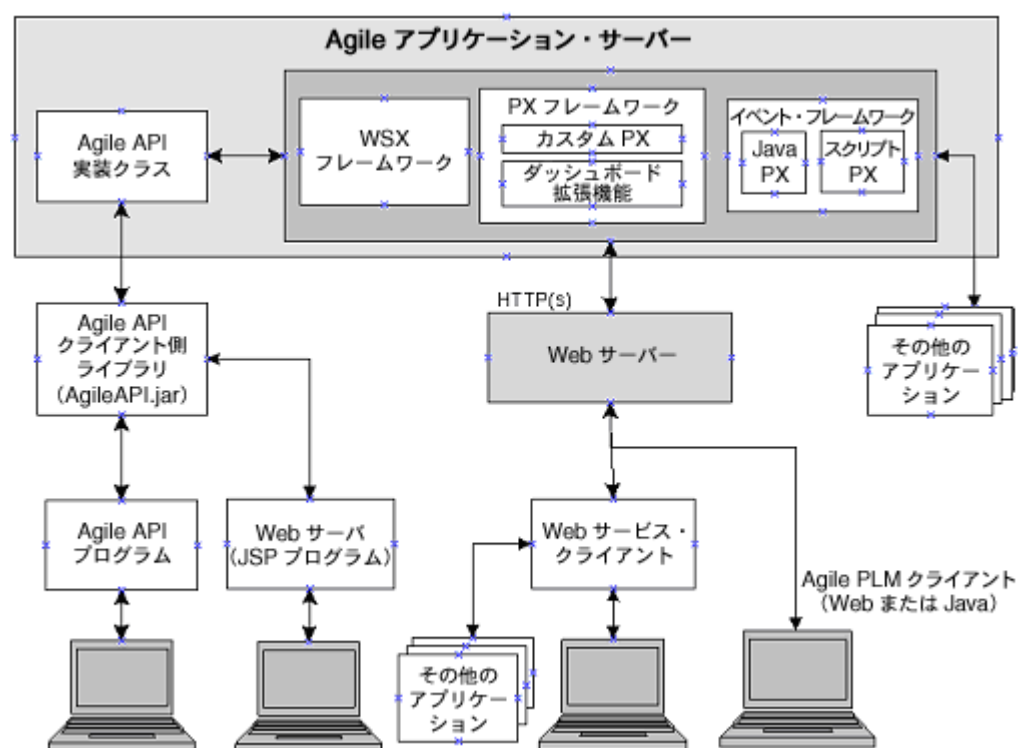
Agile アプリケーション・サーバーに組み込まれている Agile SDK のサーバー側のコンポーネントは、次のとおりです。

- Agile API 実装クラス
- Java およびスクリプト・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

SDKのアーキテクチャ

Agile SDK を使用すると、Agile アプリケーション・サーバー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、サーバー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・サーバーのコンテナ内に Web サービスを配置できます。WSX に使用する Web サーバーは、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、サーバー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。WSX、Java PX およびスクリプト PX の拡張機能にも Agile API を使用できます。これは、Agile SDK のすべての開発プロジェクトで利用できるツールです。拡張機能は、Agile で提供されていない API を使用して開発することもできます。

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・サーバーに接続します。したがって、Agile API プログラムは、企業のファイア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) テクノロジーを使用して企業のファイアウォールを介してサーバーに接続できます。

システム要件

Agile SDK システムの要件は、PLM 容量計画および配置ガイドを参照してください。

Javaの要件

Agile API は、アプリケーション・サーバーがサポートする Java のバージョンと互換性がある必要があります。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・サーバーが使用している 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 では Sun 社の Java Runtime Environment (JRE) 1.5.0_06 を、Oracle WebLogic Server 10.3 では Sun 社の Java Runtime Environment (JRE) 1.6 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・サーバーで 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・サーバー	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 1.5.0
Oracle WebLogic Server 10.3	Windows 2003	Sun JRE 1.6

メモリー不足例外を回避するためのJava Virtual Memory (JVM) パラメータ

メモリー不足例外を回避するには、次の JVM オプションを指示されている場所に追加してください。

注意 この回避策は、シングルスレッドの SDK プログラムにのみ適用できます。

- クライアントがスタンドアロン SDK クライアントの場合は、次のように JVM オプションを追加します。

```
java -Ddisable.agile.sessionID.generation=true pk.sample
```
- クライアントが PX のときに、Agile サーバーでメモリー不足が発生する場合は、
<OAS_HOME>/opmn/conf/opmn.xml に JVM オプションを追加します。

```
<category id="start-parameters">
  <data id="java-options" value="-Xrs -server -
XX:MaxPermSize=256M -ms1280M -mx1280M -XX:NewSize=256M -
XX:MaxNewSize=256M -XX:AppendRatio=3 -
Doracle.xdkjava.compatibility.version=10.1.0 -
Djava.security.policy=$ORACLE_HOME/j2ee/home/config/java2.policy -
Dagile.log.dir=$ORACLE_HOME/j2ee/home/log -
Dcom.sun.management.jmxremote -
Dcom.sun.management.jmxremote.port=9899 -
Dcom.sun.management.jmxremote.authenticate=false -
Dcom.sun.management.jmxremote.ssl=false -Djava.awt.headless=true -
Dhttp.webdir.enable=false -Duser.timezone=GMT -
Ddisable.agile.sessionID.generation=true"/>
  <data id="oc4j-options" value="-verbosity 10 -userThreads"/>
</category>
```

- クライアントが URL PX の場合は、サーバー起動スクリプト (Tomcat の `catalina.bat` など) に次の JVM オプションを追加します。

```
-Ddisable.agile.sessionID.generation=true
```

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、次のフォルダ構造を使用します。

`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 のクラスとインタフェースが含まれています。
- `axis.jar` - Apache Axis ライブラリの Oracle 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。
- `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フェースが含まれています。

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・サーバーの 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 インストール・ガイド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 インタフェースを示します。

オブジェクト	関連する Agile API インタフェース
変更	<code>IChange</code>
顧客	<code>ICustomer</code>
デklarレーション	<code>IDeclaration</code>
デザイン	<code>IDesign</code>
ディスカッション	<code>IDiscussion</code>
ファイル・フォルダ	<code>IFileFolder</code>
アイテム	<code>IItem</code>
製造元部品	<code>IManufacturerPart</code>

オブジェクト	関連する Agile API インタフェース
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
プロジェクト	IProgram
ソーシング・プロジェクト	IProject
品質変更要求	IQualityChangeRequest
レポート	IProductReport
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
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・サーバーにインストールされているサーバー・コンポーネントや、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。

注意 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、一部のレポート・オブジェクトは Agile API 経由ではアクセスできません。

Agile API の開始

この章のトピック

- Agile APIの概要..... 7
- Agile APIプログラムの開始..... 9
- Agile PLMオブジェクトのロードおよび作成..... 13

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 のクラスとインタフェースのタイプ
- Agile API クラスのロード
- 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化
- サンプル・プログラムの検出

Agile APIのクラスとインタフェースのタイプ

Agile API (AgileAPI.jar ライブラリ) には、いくつかのクラスとインタフェースが含まれています。これらのファイルは、サポートする機能に従って次のグループに分類されています。

- **集約インタフェース** - これらのインタフェースは、特定のオブジェクト・タイプに適用可能な機能インタフェースを集約します。たとえば、IItem インタフェースは、IDataObject、IRevisioned、IManufacturingSiteSelectable、IAttachmentContainer、IHistoryManager および IReferenced を拡張します。ほとんどの SDK 機能は、これらのインタフェースに含まれます。これらのインタフェースは、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・インタフェース** - これらのインタフェースは、他のインタ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ための便利な手段を提供します。IAttachmentContainer インタフェースは、このグループの他のインタフェース (IChange、IItem など) 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 ユーザーにルート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・インタフェース** - このグループのクラスは、Agile アプリケーション・サーバーのメタデータ（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・インタフェースには、I AgileClass、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

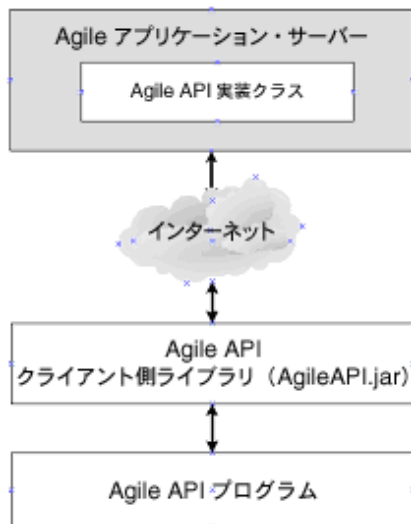
Agile API には、2 つのメイン・ソフトウェア・コンポーネントがあります。

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

サーバー側の実装クラスは、Agile アプリケーション・サーバー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・サーバーに接続され、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。

図2: Agile API のアーキテクチャ



ネットワーク・クラス・ロードには、サーバー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。サーバー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、サーバー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ー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 ファイルの配布

自分で作成した、Agile API をコールする Java アプリケーションまたはアプレットは、すべて自由に配布することができます。作成したアプリケーション・ファイルをパッケージングする際、Agile API ライブラリの `AgileAPI.jar` を含めることができます。

開発環境によっては、プログラムとともに、その他のクラス・ファイルやライブラリの配布も必要な場合があります。開発環境に関するドキュメントを調べて、プログラムとともに配布する必要のあるランタイム・ファイルを確認してください。配布予定のファイルごとに、該当する製造元の使用許諾契約書を確認し、そのファイルをアプリケーションとともに配布する権限があるかどうかを判断してください。

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、`api`、`dx`、`px` および `wsx` フォルダに格納されています。これらのフォルダは、`SDK_samples` (ZIP ファイル) にあります。このファイルにアクセスする方法は、2 ページの「[クライアント側のコンポーネント](#)」の「**注意**」を参照してください。

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` の文書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・サーバー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。次に例を示します。

```
com.agile.api.IItem source =  
(com.agile.api.IItem)m_session.getObject(com.agile.api.IItem.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ため煩雑になります。また、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

注意 ログイン・ユーザーが LDAP ユーザーの場合は、disable.agile.sessionID.generation=true という JVM パラメータを使用してセッションを作成します。これが適用できるのは、Agile が Weblogic Server 上で実行されている場合のみです。クライアントがスタンドアロン SDK クライアントの場合は、JVM オプション java -Ddisable.agile.sessionID.generation=true pk.sample を追加します。または、次に示すように、コードにパラメータを設定することもできます。

```
System.setProperty("disable.agile.sessionID.generation", "true");  
HashMap params = new HashMap();  
params.put(AgileSessionFactory.USERNAME, USERNAME);  
params.put(AgileSessionFactory.PASSWORD, PASSWORD);  
AgileSessionFactory factory = AgileSessionFactory.getInstance(URL);  
IagileSession session = factory.createSession(params);
```

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・サーバー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 サーバーのインスタンスを取得します。サーバーに対して接続 URL を指定する必要があります。指定する URL は、Agile サーバーに直接接続するか、プロキシ Web サーバーを介して接続するかによって異なります。

- Agileサーバーに直接接続するには、URL: <http://appserver:port/virtualPath>を入力します。
- プロキシ Web サーバーを介して Agile サーバーに接続するには、URL: `protocol://webserver:port/virtualPath` を入力します。

ここで、

- `appserver` は Agile サーバーのコンピュータ名です。
- `webserver` は Web サーバーのコンピュータ名です。
- `virtualPath` は Agile PLM サーバーの仮想パスです。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 インストール・ガイド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 サーバー・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 サーバー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);
}
```

1 つのユーザー・アカウントで Agile アプリケーション・サーバーに同時セッションを開くことができる最大数は、Oracle Agile PLM 契約によって決まります。この最大数を超過す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

ファイアウォールを越えて 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・サーバーに指定できます。この場合、サーバー・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()` メソッドを呼び出してサーバー・インスタンスを取得する必要があるため、ログイン・コードはさらに簡単になります。次の例に示すように、`createSessionEx()` メソッドは、1 回の呼び出しでサーバー・インスタンスを取得してセッションを作成します。

例: パスワードで管理された URL へのアクセスによるセッションの作成

```
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);
}
```

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。

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 サーバーの機能にアクセスできます。Web サービスの Agile PLM サーバー・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 `null` 値を指定します。

`AgileSessionFactory` オブジェクトを取得した後は、セッションも作成できます。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 `null` 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 `null` 値を指定すると、Agile PLM サーバーが 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 `null` 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、373 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、次のインタフェースがマップされます。

□ <code>IChange</code>	□ <code>IManufacturer</code>	□ <code>IRequestForQuote</code>
□ <code>ICommodity</code>	□ <code>IManufacturerPart</code>	□ <code>IServiceRequest</code>
□ <code>ICustomer</code>	□ <code>IManufacturingSite</code>	□ <code>ISpecification</code>
□ <code>IDeclaration</code>	□ <code>IPackage</code>	□ <code>ISubstance</code>
□ <code>IDesign</code>	□ <code>IPrice</code>	□ <code>ISupplier</code>
□ <code>IDiscussion</code>	□ <code>IProgram</code>	□ <code>ISupplierResponse</code>
□ <code>IFileFolder</code>	□ <code>IProject</code>	□ <code>ITransferOrder</code>
□ <code>IFolder</code>	□ <code>IQualityChangeRequest</code>	□ <code>IUser</code>
□ <code>IItem</code>	□ <code>IQuery</code>	□ <code>IUserGroup</code>

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。

検索条件およびフォルダの作成方法の詳細は、35 ページの「[検索条件の作成およびロード](#)」および 103 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)`

注意 ユーザーが指定しないかぎり、オブジェクトは常に、サブクラスまたはクラスから導出された基本クラスに従ってロードされます。また、オブジェクトの導出された基本クラスが正しい場合も、オブジェクトは正しくロードされます。ただし、SDK では、オブジェクトに無効なサブクラスが渡された場合でも、無効なクラスから導出された基本クラスとオブジェクトの基本クラスが同一の場合は、オブジェクトがロードされます。

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、332 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。サーバーは、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、363 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

次の例は、様々なタイプの Agile PLM オブジェクトをロードす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_NU
        MBER, "WE10023-45");
        IManufacturerPart mfrPart =
        (IManufacturerPart)m_session.getObject(IManufacturerPart.OBJECT_TYPE,
        params); System.out.println("ManufacturerPart : " + mfrPart.getName());
//Load a manufacturing site
        IManufacturingSite siteHK =
        (IManufacturingSite)m_session.getObject(ManufacturingSiteConstants.CLASS_S
        ITE, "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_TY
        PE, "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)`

注意 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・ステータス属性は設定できません。これは、オブジェクトが作成されるまでは、LCP に必要な設定が有効にならないようにするためです。このことは、UI に対しても同様に適用されます。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。また、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ません。

`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、13 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` オブジェクトも指定できます。Agile アプリケーション・サーバーでは、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・サーバー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・サーバー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・サーバー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、お勧めできません。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」、「ページ 2」および「ページ 3」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 = page2.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_COMMODITY,
        "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_DECLARATION,
        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_MANU
FACTURER, 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_SITE,
"San Jose");
params.put(PriceConstants.ATT_GENERAL_INFORMATION_SUPPLIER, "SUP20013");
```



```

IPrice price =
    (IPrice)m_session.createObject(PricingConstants.CLASS_PUBLISHED_PRICE,
        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);
I AgileList 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_CT
        O, "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 は使用できません。

Agile PLMオブジェクトの状態の確認

IStateful インタフェースは、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、アイテムおよびルーティング可能なオブジェクトです。

ルーティング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 `null`）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 `null`）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 subclass
        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_ROLES);
    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、2 回目の削除を実行します。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
- AML があるアイテム
- リリース済の変更
- 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
- 1 つ以上の製造元部品がある製造元

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、Agile PLM サーバーにより、`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 = JOOptionPane.ShowDialog(null, "This Item is used by another
            Item. " +
                "Would you still like to delete it?", "Item is Used Warning",
            JOOptionPane.YES_NO_OPTION);
        }
    }
}
```

```
    }
    if (i == 0) {
        try {
            // Disable "Item is Used" warning

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

m_session.enableWarning(ExceptionConstants.APDM_DELETECOMPINUSE_WARNING)
;
        } 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();  
}
```


検索条件の作成およびロード

この章のトピック

■ 検索条件について	35
■ 検索条件の作成	35
■ フォルダへの検索条件の保存	36
■ 順序付け（並べ替え）された、または順序付けされていない検索結果の生成	37
■ パラメータ検索の作成	38
■ 検索条件作成時の検索属性の指定	39
■ ワークフロー検索の指定	40
■ 検索条件の指定	41
■ 検索条件でのSQL構文の使用	51
■ 検索条件の結果属性の設定	54
■ 検索結果の使用	62
■ 使用箇所検索条件の作成	63
■ 検索条件のロード	64
■ 検索条件の削除	65
■ 簡単な検索条件の例	66

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単純検索もあります。

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・タイプおよび検索で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（期限付き変更指示、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();
I AgileClass 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・サーバーによって削除されます。

順序付け（並べ替え）された、または順序付けされていない検索結果の生成

35ページの「[検索条件の作成](#)」の例に示すように、`IQuery.execute()` および `IQuery.execute(Object[] params)` メソッドを実行すると、順序付けされた検索結果が `ITable` に返されます。

SDK では、検索パフォーマンスを向上させるために、次のメソッドを提供してデフォルトの順序に並べ替えていない結果を返します。ただし、検索条件に `starts with` 条件が含まれている場合、結果は常にその属性で並べ替えられ、`execute(boolean)` で `skipOrdering` を `true` で渡しても順序付けはスキップされません。

注意 検索結果をデフォルト以外の順序に並べ替える場合は、62ページの「[検索結果の並べ替え](#)」を参照してください。

- `IQuery.execute(boolean skipOrdering)`
- `IQuery.execute(Object[] params, boolean skipOrdering)`

順序付けをスキップまたは実行するには、次の例に示すように、ブールの `skipOrdering` を `true` または `false` に設定します。

例: 検索結果の順序付けのスキップ

```

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'");
    // The boolean is set to true to skip ordering
    ITable results = query.execute(true);
} catch (APIException ex) {
    System.out.println(ex);
}

```

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（実行時などに）指定できます。パラメータは、検索条件に値を渡すための便利な方法です。これによって、時間を短縮して余分なコーディングを削減できます。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、その検索条件には、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、null 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 null 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

ワークフローは、ルーティング可能なオブジェクト（変更など）が従う一連のステータスです。ワークフローにより、多くのプロセスが自動化されます。たとえば、変更がリリースされると、通知リストに指定された Agile ユーザーはこの情報に関する自動電子メールを受信します。

ルーティング可能なオブジェクトは、変更管理プロセスでの変更、転送、製品サービス依頼などのプロセスを表すクラスから作成されます。これらのオブジェクトは、ワークフローを介して、承認や追加入力のために Agile ユーザーにルートできます。

ワークフロー属性を使用してワークフロー検索を指定すると、割り当てられたワークフローでのルーティング可能なオブジェクトのステータスを検索してチェックできます。このような検索は、PLM の Web クライアントと Java クライアントの両方でサポートされています。クライアントを使用すると、検索条件に 1 つのワークフロー属性を指定したり、1 つ以上の属性を検索条件にグループ化することができます。検索条件のグループ化には、Web クライアントおよび Java クライアントの **grouping** オプションを使用できます。ワークフロー検索とグループ化オプションの詳細は、『Agile PLM スタート・ガイド』のワークフローに関する項を参照してください。

SDK の検索条件でサポートされているワークフロー属性は次のとおりです。

- Approver
- Approver Action
- Observer
- Observer Action
- Signoff User
- Notify User

条件へのグループ化が必要なワークフロー属性で、SDK でサポートされていない属性は次のとおりです。

- Duration
- Next Status
- Workflow Status
- Status changed by

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。

例： 検索条件でのワークフロー属性 Approver の使用

```
private void testWorkflowQuery(IAgileSession session) throws Exception
{
    IQuery query = (IQuery)session.createObject(IQuery.OBJECT_TYPE,
        ChangeConstants.CLASS_ECO);
    String criteria = "[Workflow.Approver] contains ([General Info.User ID] ==
        'admin')";
    query.setCriteria(criteria);
    ITable result = query.execute();
    System.out.println(result.size());
}
```

注意 ネストされた条件の例と説明は、46ページの「[ネストされた条件を使用したオブジェクト・リストの値の検索](#)」を参照してください。

例: 検索条件でのワークフロー属性 Approver Action の使用

```
private void testWorkflowQuery(IAgileSession session) throws Exception
{
    IQuery query = (IQuery)session.createObject(IQuery.OBJECT_TYPE,
        ChangeConstants.CLASS_ECO);
    String criteria = "[Workflow.Approver Action] contains 'Approved'";
    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勧めします。

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」管理ノードは、ワークフローに対する再利用可能な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、SDK の現在のリリースではサポートされていません。

- `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、`[TitleBlock.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。ロー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勧めします。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・サーバー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ーで 사용되는検索条件を作成する場合は、名前かわりに、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。

注意 APINameフィールドは検索属性の指定をサポートしていません。詳細は、125ページの「[APIName フィールドを使用したPLMメタデータへのアクセス](#)」を参照してください。

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の参照もできます。

```
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。通常は、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、54 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。

演算子（英語）	表記法	説明
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。

検索範囲、検索範囲に含まれない、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 PLM の Java クライアントと Web クライアントでは直接的には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
<code>[Number] between ('1','6')</code>	<code>[Number] >= '1' and [Number] <= '6'</code>
<code>[Number] not between ('1','6')</code>	<code>[Number] < '1' and [Number] > '6'</code>
<code>[Number] in ('1','2','3','4','5','6')</code>	<code>[Number] == '1' or [Number] == '2' or [Number] == '3' or [Number] == '4' or [Number] == '5' or [Number] == '6'</code>
<code>[Number] not in ('1','2','3','4','5','6')</code>	<code>[Number] != '1' and [Number] != '2' and [Number] != '3' and [Number] != '4' and [Number] != '5' and [Number] != '6'</code>

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。これらの動的リストにあるオブジェクトの検索には、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、カンマも使用できます。このカンマは論理 `OR` と等価です。

次の条件は、`Christopher` という名または `Nolan` という姓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` という名で、かつ `Nolan` という姓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')
```

`Part.Page Three.List01` が有効で、`Part Families` リストに設定されている場合、次の条件は `PartFamily_01` という名前の部品ファミリを検索します。

```
[Page Three.List01] in ([General Info.Name] == 'PartFamily_01')
```

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、次の例に示すように、二次元配列で指定する必要があります。

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)
```

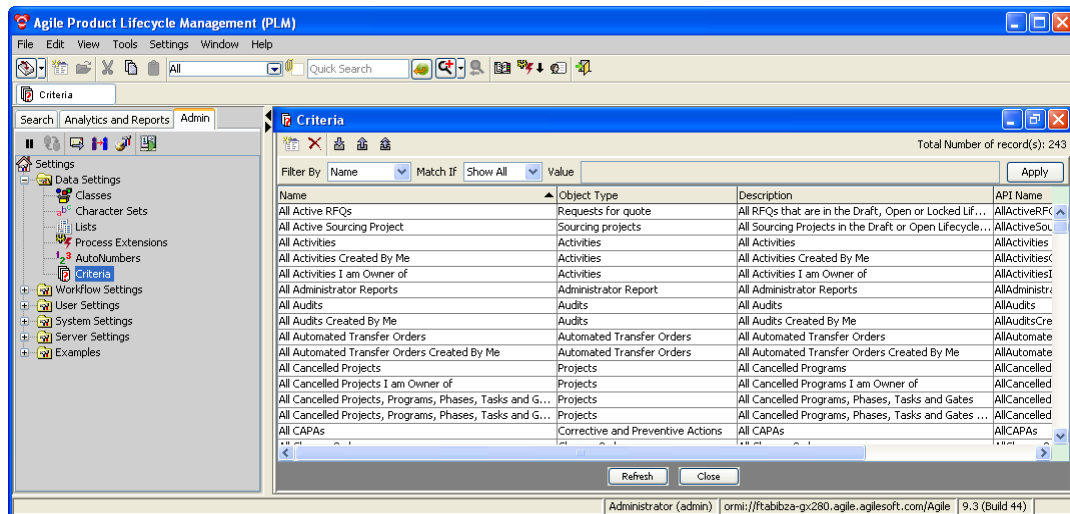
- ただし、プレースホルダではなく文字列値として明示的に指定されている場合は、正常に実行されます。

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

条件ライブラリから選択した条件のSDK検索での使用

Java クライアントまたは Web クライアントの条件ライブラリの「条件」ノードは `ICriteria` オブジェクトです。このオブジェクトは SDK 検索で使用できます。次に示すリストを Java クライアントで表示するには、「管理」>「設定」>「データ設定」>「条件」の順に選択します。

図3: Java クライアントの条件ライブラリの「条件」ノード



次の例では、条件ライブラリから「条件」ノードを取得し、このノードを SDK 検索条件としてロードおよび設定します。

例: 条件ライブラリの条件の SDK 検索での使用

```
IQuery query = (IQuery) session.createObject(IQuery.OBJECT_TYPE,
ItemConstants.CLASS_ITEM_BASE_CLASS);
IAdmin admin = session.getAdminInstance();
// Get the Criteria Library node
INode criteriaLibrary =
admin.getNode(NodeConstants.NODE_CRITERIA_LIBRARY);
// Load the Criteria relevant to the query class (For example it is Items
base class)
ICriteria criteria = (ICriteria) criteriaLibrary.getChild("All Released
Items");
// Set the ICriteria in SDK Query Criteria
query.setCriteria(criteria);
```

SDK検索での関係とコンテンツの使用

Agile SDK には、`IQuery` インタフェースを使用して関係検索やコンテンツ検索を実行する API が用意されています。検索条件には、基本検索クラスおよび関連クラスの両方の属性を含めることができます。

オブジェクトの関係を使用して検索する手順は、次のとおりです。

1. `IQuery.setSearchType(int searchType)` を使用して、`searchType` を `QueryConstants.RELATIONSHIPS` に設定します。
2. `IQuery.setRelatedContentClass(Object relatedClass)` を使用して関連クラスを設定します。

例: 検索条件でのオブジェクトの関係の使用

```
IQuery query1 = (IQuery) session.createObject(IQuery.OBJECT_TYPE,
ItemConstants.CLASS_PART);
query1.setSearchType(QueryConstants.RELATIONSHIPS);
query1.setRelatedContentClass("Substance"); // ID or API Name
query1.setCriteria("[Relationships.Name] Is Not Null and [Title
Block.Number] equals 'P00001' and [Relationships.Substance.General
Info.Name] Is Not Null");
```

プログラム・オブジェクトのコンテンツを使用して検索する手順は、次のとおりです。

1. `IQuery.setSearchType(int searchType)`を使用して、`searchType` を `QueryConstants.RELATIONSHIPS` に設定します。
2. `IQuery.setRelatedContentClass(Object relatedClass)`を使用して関連クラスを設定します。

例: 検索条件でのプログラム・オブジェクトのコンテンツの使用

```
IQuery query1 = (IQuery) session.createObject(IQuery.OBJECT_TYPE,
ProgramConstants.CLASS_ACTIVITIES_CLASS);
query1.setSearchType(QueryConstants.RELATIONSHIPS);
query1.setRelatedContentClass("ECO"); // ID or API Name
query1.setCriteria("[Content.Criteria Met] Is Not Null and
[Content.ECO.Cover Page.Originator] in ([General Info.First Name] ==
'admin')");
```

転送オブジェクトの選択したコンテンツを使用して検索する手順は、次のとおりです。

1. `IQuery.setSearchType(int searchType)`を使用して、`searchType` を `QueryConstants.TRANSFER_ORDER_SELECTED_CONTENT` に設定します。
2. `IQuery.setRelatedContentClass(Object relatedClass)`を使用して関連クラスを設定します。

例: 検索条件での転送オブジェクトの選択したコンテンツの使用

```
IQuery query1 = (IQuery) session.createObject(IQuery.OBJECT_TYPE,
TransferOrderConstants.CLASS_CTO);
query1.setSearchType(QueryConstants.TRANSFER_ORDER_SELECTED_CONTENT);
query1.setRelatedContentClass("ECR"); // ID or API Name
query1.setCriteria("[Selected Content.ECR.Cover Page.Number] equal to
'C0001'");
```

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ー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、さらに次の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

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、`I AgileSession.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・プロフィール」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、63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、アスタリスク (*) および疑問符 (?) の2つのワイルドカード文字を使用できます。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。たとえば、`[Title Block.Description] like '*book*'` 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。たとえば、`[Title Block.Description] like '?al*'` 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	<ul style="list-style-type: none">□ union□ intersection□ minus
2	<ul style="list-style-type: none">□ 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'
```

多数のオブジェクトを含むリストに対する検索条件の設定

SDK を使用して多数のオブジェクトを含むリストを検索する場合は、検索条件にオブジェクト ID を使用してリストの値を設定すると、パフォーマンスが向上します。

たとえば、次のルーチンの場合は、

```
query.setCriteria("[Page Three.List25] equal to 'Administrator
(admin) '");
```

次のように置き換えるとパフォーマンスが向上します。

```
IUser user = (IUser)session.getObject(IUser.OBJECT_TYPE, "admin");
query.setCriteria("[Page Three.List25] equal to
"+user.getObjectId());
```

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、柔軟性がある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`

読みやすさを向上させるために、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勧めします。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.Numeric01] 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。検索属性の使用の詳細は、39 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。

例	説明
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	カバー・ページ.変更タイプ
変更指示	カバー・ページ.番号
ECO	カバー・ページ.説明
変更要求	カバー・ページ.ステータス
ECR	カバー・ページ.ワークフロー
期限付き変更指示	
期限付き変更指示	
製造元依頼	
MCO	
価格変更	
PCO	
拠点毎変更	
SCO	

検索クラス	デフォルトの結果属性
出荷停止 出荷停止	
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
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 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 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

検索クラス	デフォルトの結果属性
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
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

検索クラス	デフォルトの結果属性
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.プロジェクト 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
プロジェクト アクティビティ フェーズ プログラム プロジェクト 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
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	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.プロジェクト 一般情報.顧客 一般情報.ライフサイクル・フェーズ
品質変更要求 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 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.プロジェクト 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、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+"]=='<UserID>')";
または
QueryString = "[Cover Page.Originator] in
(["+UserConstants.ATT_GENERAL_INFO_LAST_NAME+"]=='<Last_name>'"+
"&&
["+UserConstants.ATT_GENERAL_INFO_FIRST_NAME+"]=='<First_name>')";
```

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・サーバー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、78 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 インタフェースを使用して、並べ替えた Iterator を作成できません。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、51 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）である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て、ITable インタフェースを使用すると、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・サーバー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ー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_RELEASE
            D);
        // 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、作成した後に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。

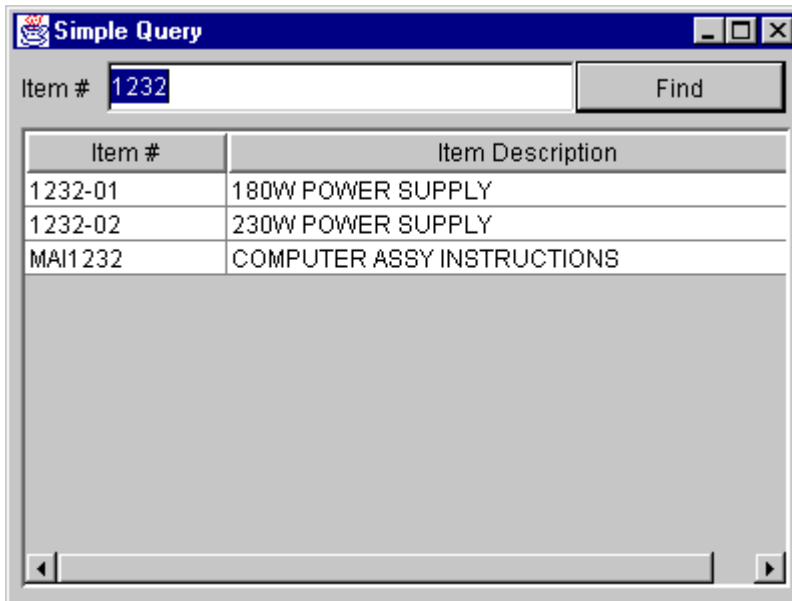
例: 検索条件の削除

```
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4: 「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、「ItemNumber」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_DESCRIPTION).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);
}
```


テーブルの使用

この章のトピック

■ テーブルについて	69
■ テーブルの取得	70
■ 新規およびマージされた「関係」テーブルへのアクセス	71
■ テーブルのメタデータの取得	73
■ テーブル行の追加	73
■ 複数のテーブル行の追加および更新	76
■ テーブル行の繰り返し処理	78
■ テーブル行の並べ替え	80
■ テーブル行の削除	81
■ 行に対する参照オブジェクトの取得	82
■ 行のステータス・フラグの確認	86
■ 「ページ1」、「ページ2」および「ページ3」の使用	87
■ レッドライン	88
■ レッドラインの変更の削除	89
■ 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	90

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限りません。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。70 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5: アイテムの「BOM」タブ

Title Block	Changes	BOM	Manufacturers	Sites	Prices	Quality	Compliance	Suppliers	Relationships	Where Used	Attachments
Effective From: ... to ..											
<div> Edit Remove Add Multi-level... Go To... Selected: 0 of 12 Page: 1 of 2 </div>											
					Item Number	Item Description	Qty	Ref Des ...	Item Rev	Effective	
					MAH1232	Computer Assy Instructions	REF			No Privileg	
					MTI1000	Computer Test Instructions	REF			No Privileg	
					QAT2321	Computer FCC Test Results	REF			No Privileg	
					1543-01	Mid-Size Case	1		B 23478	No Privileg	
					9876-01	IDE Hard Disk Controller	1			No Privileg	
					7654-01	1.0GB IDE Hard Disk	1			No Privileg	
					6598-01	DC Power Cable	3		A 23450	No Privileg	
					6642-01	3.5" Floppy Disk Drive	1			No Privileg	
					8768-01	2MB Video Card	1			No Privileg	
					2543-01	2X Speed CD-ROM Drive	1			No Privileg	
Rows per page: 10 Page: 1 of 2											

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: アイテムまたは変更指示）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klarेशन (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 スタート・ガイド』
- 『Agile PLM 管理者ガイド』
- 『Product Governance & Compliance ユーザー・ガイド』
- 『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

リリース 9.2.2 では、次のテーブルは「関係」テーブルという 1 つのテーブルにマージされました。

- 関係.影響元
- 関係.影響先
- 関係.参照

さらに、これらのテーブルで使用されていた定数 (`TABLE_REFERENCES`、`TABLE_RELATIONSHIPS_AFFECTS` および `TABLE_RELATIONSHIPS_AFFECTED_BY`) も削除されました。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。

注意 1 つの定数にマージおよびマップされたり、新規の定数にマップされたテーブル定数の完全なリストは、453 ページの [「リリース 9.2.1 以前のテーブル定数のリリース 9.2.2 以降への移行」](#) を参照してください。

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 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、『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_RELATIONSHIPS_AFFECTEDBY);
```
- 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_RELATIONSHIPS_AFFECTS);
```
- 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、`I AgileClass` 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;
}

```

テーブルの名前またはIDは、「API名」フィールドを使用して識別することもできます。このフィールドの使用の詳細は、125ページの「[APINameフィールドを使用したPLMメタデータへのアクセス](#)」を参照してください。Agile APIを使用してメタデータを使用する方法の詳細は、323ページの「[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、`IItem.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。76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、`IItem.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、`IItem.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、`IItem.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_PART_NUMBER, "TPS100-256");
    info.put(ManufacturerPartConstants.ATT_GENERAL_INFO_MANUFACTURER_NAME, "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、変更指示の「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) を通してサーバー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・サーバーでは、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、任意の数の行のスキップもできます。`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）はお勧めしません。

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6: 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

`getReferentIterator` を呼び出して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更新した場合、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。最初のページの検索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。

例: 多数の検索結果の繰返し時におけるすべてのテーブル行の更新

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。

多数の検索結果の繰返し時におけるテーブル行の更新

次の例の手順に従います。

多数の検索結果の繰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。

テーブル行の並べ替え

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必要があります。

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 Projects 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、51 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、テーブルから行は削除できません。詳細は、73 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バージョンを使用している場合は、新しいリリースバージョンに対する変更指示を作成するまで、そのアイテムのテーブルから行は削除できません。

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 使用箇所の保留中変更 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の変更指示の場合は、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」、「ページ2」および「ページ3」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）、「ページ2」および「ページ3」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」、「ページ2」および「ページ3」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」、「ページ2」および「ページ3」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「ページ2」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、「ページ2」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、「ページ2」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: 「ページ2」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、以前のリリース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リリースを取得します。
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リースを指定します。
5. SCO および MCO は、アイテムのリリースに影響を与えません。
6. 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）、「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 Preferred 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」テーブル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();
}
}

```

一括モードでのレッドラインの変更の削除

一括モードでのレッドラインの変更の削除

Agile SDK では、IRedlinedTable を使用してレッドラインを削除（取り消す）できます。このインタフェースには、次に示すように、レッドラインの一括取消しを実行する API が用意されています。

```

IRedlinedTable.undoRedline(Collection rows);
IRedlinedTable.undoAllRedline();

```

注意 88ページの「[レッドライン](#)」を参照してください。

例: レッドライン・テーブルの IRedlinedTable インタフェースへのタイプキャスト

```

IItem item = (IItem) session.getObject(ItemConstants.CLASS_PART,
"PART_001");
item.setRevision("B"); // Unreleased change
ITable bomTable = item.getTable(ItemConstants.TABLE_REDLINEBOM);
Iterator it = bomTable.iterator();
List rows = new ArrayList();
while(it.hasNext()) {
    IRow row = (IRow) it.next();
    if(((IRedlined)row).isRedlineModified())
        rows.add(row);
}

// Only Redline tables can be typecasted to IRedlinedTable interface
// Case 1:
((IRedlinedTable)bomTable).undoRedline(rows);
// Case 2:
((IRedlinedTable)bomTable).undoAllRedline();

```

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()` メソッドは、ブール値を返します。セルまたは行がレッドラインされている場合は、`TRUE` の値を返します。

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。

データ・セルの使用

この章のトピック

■ データ・セルについて	93
■ データ・タイプ	93
■ ユーザーのディスカバリ権限の確認	94
■ セルが読取り専用かどうかの確認	95
■ 値の取得 - 9.3	95
■ 値の設定	97
■ リスト値の取得および設定	98
■ 参照指示セルの使用	102

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	I AgileList
TYPE_OBJECT	Object
TYPE_SINGLELIST	I AgileList
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。

ユーザーの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確認できます。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は表示できません。次の例に示すように、`ICell.hasDiscoveryPrivilege()` メソッドを使用すると、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 によって `null` 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、必要な表示権限のないフィールドを表示しようとすると、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取り専用かどうかを確認する必要があります。読取り専用の場合は、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。ユーザーが読取り専用のセルに値を設定しようとする、Agile API で例外が発生します。

例: フィールドが読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 - 9.3

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」、「ページ 2」または「ページ 3」のセルの値を取得します。

セルの値を使用するには、セルを選択する必要があります。Agile PLMのセルは、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、334ページの「[属性の参照](#)」を参照してください。

注意 ICell.getAPIName() を使用すると、データ・セルの属性値にアクセスできます。このフィールドの使用の詳細は、125ページの「[APINameフィールドを使用したPLMメタデータへのアクセス](#)」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、93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、62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、『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
<code>ICell.setValue()</code>	セルの値を設定します。
<code>IRow.setValue()</code>	行内に含まれているセルの値を設定します。
<code>IRow.setValues()</code>	行内に含まれている複数のセルの値を設定します。
<code>IDataObject.setValue()</code>	「ページ 1」、「ページ 2」または「ページ 3」のセルの値を設定します。
<code>IDataObject.setValues()</code>	「ページ 1」、「ページ 2」または「ページ 3」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」、「ページ2」または「ページ3」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、317ページの「[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・セルの利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、171 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、155ページの「[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。範囲セパレータ (-) 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勧めします。

フォルダの使用

この章のトピック

■ フォルダについて	103
■ フォルダのロード	105
■ フォルダの作成	105
■ フォルダ・タイプの設定	106
■ フォルダ要素の追加および削除	106
■ フォルダ要素の取得	107
■ フォルダの削除	110

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ります。ファイル・フォルダはフォルダのサブセットで、IFolder という独自のインタフェースがあります。ファイル・フォルダに格納されているファイルは、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ます。ファイル・フォルダの詳細は、177 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・タイプには設定できません。

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();  
}
```

アイテム、BOM および AML の使用

この章のトピック

■ アイテム、BOMおよびテーブルについて	111
■ アイテムの使用	111
■ BOMの使用	114
■ AMLの使用	121

アイテム、BOMおよびテーブルについて

アイテムの使用

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、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 nullString 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、IRevised インタ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);
    }
}
```

IRevised.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 null 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);
```

a change number (a String) associated with a particular revision:

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

revision identifier (a String such as "Introductory", "A", "B", "C", and so on): `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リビジョンおよび保留中の変更がない場合、
IRevised.getRevision() メソッドは null String を返し、
IRevised.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ステータスの変更

アイテムの各リビジョンは確定済にできます。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	<code>ItemConstants.TABLE_BOM</code>
「BOM のレッドライン」テーブル	<code>ItemConstants.TABLE_REDLINEBOM</code>

「BOM」テーブルの取得方法を示す例は、70ページの「[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().getVal
ue()

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

製造拠点の詳細は、149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、アイテムの新規バー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関連の製品レポートの作成

SDK には、次の BOM 関連の製品レポートを準備するために、ProductReportConstants で定義した定数を使用する IProductReport API が用意されています。次のレポートは XML 形式で作成されます。

- **BOM 展開レポート** - BOM 展開レポートには、指定した 1 つ以上のアセンブリの部品構成表 (BOM) にあるアイテムが、指定のレベル数まで表示されます。
- **BOM 比較レポート** - BOM 比較 XML レポートには、2 つの異なる BOM を比較した結果が、指定のレベル数まで表示されます。

たとえば、ベース BOM とターゲット BOM の比較では、次の結果が表示されます。

- 「BOM」 ノードに **d** が表示される場合 - ベース・アセンブリにのみ BOM があることを示します。
- 「BOM」 ノードに **a** が表示される場合 - ターゲット・アセンブリにのみ BOM があることを示します。
- 「BOM」 ノードに **u** が表示される場合 - 両方のルート・アセンブリに同じ BOM があることを示します。
- 「BOM」 ノードに **m** が表示される場合 - 両方のルート・アセンブリに BOM がありますが、いくつかの相違があることを示します。

ベース・アセンブリおよびターゲット・アセンブリともに、**第 1 レベル**のすべての BOM は別のノード **BOMs** に分類されます。**BOMs** の下にある「BOM」ノードは、最初に FindNum で並べ替えられ、次に ItemNumber で並べ替えられます。

これらのレポートについては、いくつかの使用ケースがあります。たとえば、ERP システムの出力を使用したアーカイブや比較分析などです。

製品レポートを作成するには、IAgileSession オブジェクトを使用する必要があります。次の例は、IAgileSession および ProductReportConstants を使用して、BOM 展開レポートおよび BOM 比較レポートの準備する方法を示しています。

例: Agile セッションの作成

```
AgileSessionFactory factory =
    AgileSessionFactory.getInstance("http://agileServer/virtualPath");
Map params = new HashMap();
params.put(AgileSessionFactory.URL,
    "http://agileServer/virtualPath");
params.put(AgileSessionFactory.USERNAME, "username");
params.put(AgileSessionFactory.PASSWORD, "pwd");

IAgileSession session = factory.createSession(params);
```

例: BOM 比較レポートの準備

```
Map param = new HashMap();
param.put(ProductReportConstants.REPORTPARAM_REPORT_TYPE,
    ProductReportConstants.REPORT_BOM_COMPARISON);
```

```

param.put(ProductReportConstants.REPORTPARAM_ITEMREVSITE,
"item1;item2");
param.put(ProductReportConstants.BOMCOMP_BOM_ATTRS,
ProductReportConstants.BOM_ATT_ITEM_NUM + ";" +
ProductReportConstants.BOM_ATT_FIND_NUM);
param.put(ProductReportConstants.BOMCOMP_BOMLEVEL, "4");
IProductReport report = (IProductReport)
session.createObject(IProductReport.OBJECT_TYPE, "My BOM Comparison
Report");
String xmlReport = report.execute(param);

```

ProductReportConstants.BOMCOMP_BOM_ATTRS に値を指定しないと、"Find Num;Item Number;Sites"とみなされます。

例: BOM 展開レポートの準備

```

Map param = new HashMap();
param.put(ProductReportConstants.REPORTPARAM_REPORT_TYPE,
ProductReportConstants.REPORT_BOM_EXPLOSION);
param.put(ProductReportConstants.BOMEXP_OBJTYPE, "Document;Part;");
param.put(ProductReportConstants.REPORTPARAM_ITEMREVSITE,
"MM75-01|23450|India;");
param.put(ProductReportConstants.BOMEXP_MAXLEVEL, "5");
IProductReport report = (IProductReport)
session.createObject(IProductReport.OBJECT_TYPE, "My BOM Explosion
Report");
String xmlReport = report.execute(param);

```

BOM 展開レポートでは、ProductReportConstants.REPORTPARAM_ITEMREVSITE に次のように値を指定できます。

- <Item_number>|<Change_number>|<Site_number>。次の場合、<Change_number>と<Site_number>はオプションです。
 - <Change_number>を指定しないと、最新リビジョンとみなされます。
 - <Site_number>を指定しないと、共通拠点とみなされます。
- 値には、セミコロンで区切られた 1 つ以上のアイテムを指定できます。
- Item1;Item2;Item3 は、共通拠点の Item1、Item2 および Item3 の最新リビジョンです。
- Item1|ECO1;Item2;Item3 (ECO1 リビジョンの Item1、および Item2 と Item3 の最新リビジョン)
- Item1|ECO1|Site1;Item2|ECO2 (Site1 固有の BOM を持つ ECO1 リビジョンの Item1、および ECO2 リビジョンの Item2)
- Item1|Site1;Item2 (Site1 固有の BOM を持つ Item1、および共通拠点の Item2 の最新リビジョン)

BOM 比較レポートでは、ProductReportConstants.REPORTPARAM_ITEMREVSITE に次のように値を指定できます。

- <Item_number>|<Change_number>|<Site_number>。次の場合、<Change_number>と<Site_number>はオプションです。
 - <Change_number>を指定しないと、最新リビジョンとみなされます。
 - <Site_number>を指定しないと、共通拠点とみなされます。
- 値には、セミコロンで区切られた 2 つのアイテムを指定する必要があります。
- Item1;Item2 (Item1 および Item2 の最新リビジョン、およびすべての拠点)

- Item1|ECO1;Item2 (ECO1 リビジョンの Item1、および Item2 の最新リビジョン)
- Item1|ECO1|Site1;Item2|ECO2 (Site1 固有の BOM を持つ ECO1 リビジョンの Item1、および ECO2 リビジョンの Item2)
- Item1|Site1;Item2 (Site1 固有の BOM を持つ Item1、および共通拠点の Item2 の最新リビジョン)

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変更指示 (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。89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であり、BOM があることを確認してください。

例: リリース済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;
}
```

変更指示の作成

BOM をレッドラインするには、ECO などの変更指示を作成する必要があります。次の例は、ECO を作成し、選択した ECO 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;
}
```

変更指示の「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: 変更指示の「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製造元の追加

「BOM」テーブルと同様に、「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。承認済製造元は、拠点別であるか、またはすべての拠点到共通です。

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_PART_
NUMBER, mfrPartNum);

    map.put(ManufacturerPartConstants.ATT_GENERAL_INFO_MANUFACTURER_NAME,
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_PART_
NUMBER, mfrPartNum);

    map.put(ManufacturerPartConstants.ATT_GENERAL_INFO_MANUFACTURER_NAME,
mfrName);
    IManufacturerPart mfrPart = (IManufacturerPart)m_session.getObject(
ManufacturerPartConstants.CLASS_MANUFACTURER_PART, map
```



```
);  
  
item.setManufacturingSite(((IAgileList)m_session.getCurrentUser().get  
Value(  
    UserConstants.ATT_GENERAL_INFO_DEFAULT_SITE)).getSelection()[0]  
));  
  
item.getTable(ItemConstants.TABLE_MANUFACTURERS).createRow(mfrPart);  
}
```

製造拠点の詳細は、149ページの「[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、新規変更指示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。変更指示を使用すると、「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。89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バー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リースを指定します。SCO および MCO は、アイテムのリリース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。

APIName フィールドを使用した PLM メタデータへのアクセス

この章のトピック

- APINameフィールドについて 125
- APINameフィールドへの名前の割当て 126
- APIName検証ルール 126
- APINameフィールドを使用したメタデータへのアクセス 127

APINameフィールドについて

APIName フィールドの主要な目的は、SDK アプリケーションの開発時に SDK 開発者が Agile 内部メタデータに簡単にアクセスできるようにすることです。このフィールドについて説明する前に、SDK 定数ファイルで定義されている表示名または数値 ID を使用して、クラス、テーブル、属性、リストなどのオブジェクトの Agile 内部メタデータにアクセスする方法について考えてみます。この方法のデメリットは、数値を記憶するのが困難なことと、表示名が変更される可能性があることです。これに対して、オブジェクトの APIName は一意で記憶しやすく、DisplayName とは異なり、コードが使用不可になるような変更はありません。

たとえば、SDK アプリケーションでリストの内部値を参照する場合は、リスト ID の 6820、またはリスト表示名の「検証結果」を使用するかわりに、リストの APIName である「AuditResult」を使用できます。

図7: 「API 名」フィールドを介した属性値へのアクセス

ID	Name	Description	API Name	Enabled	Editable	Cascade	Disp
4682	AttachType List	AttachType List	AttachTypeList	Yes	Yes	No	List
6820	Audit Result	Audit Result	AuditResult	Yes	Yes	No	List
8934	Buyer	Buyer	Buyer	Yes	Yes	No	List
2000001019	Calculated Compliance	Calculated Compliance	CalculatedCompliance	Yes	No	No	List

次に、名前を APIName フィールドに割り当てるルール、および SDK アプリケーションで APIName フィールドを使用して内部データにアクセス可能な SDK インタフェースについて説明します。

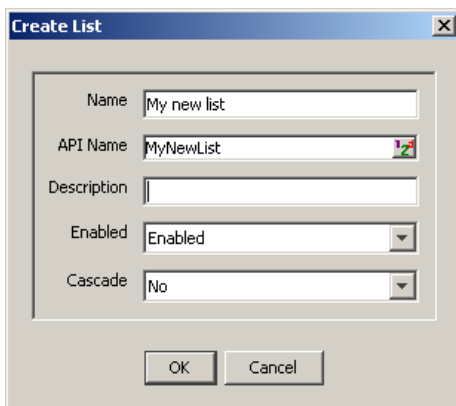
注意 リリース 9.3 にアップグレードする際は、以前のユーザー定義フィールドに割り当てられた APIName を複製できます。たとえば、P2 および P3 にユーザー定義フィールド「Foo」がある場合は、アップグレード・ツールによって両方のフィールドに APIName 「Foo」が割り当てられます。複製しない場合は、Java クライアントでいずれかのフィールドの APIName を変更します。

APINameフィールドへの名前の割当て

権限のあるユーザーが Java クライアントで新しいデータ・オブジェクトを作成すると、PLM アプリケーションによって、APIName フィールドに名前が自動的に割り当てられます。APIName をサポートするオブジェクトでは、「作成」ダイアログ・ボックスに「API 名」フィールドが追加されます。このフィールドには、オブジェクトの「名前」フィールドに名前が入力されると、PLM によって即時に API 名が挿入されます。

PLM によって割り当てられる APIName は、キャメルケース命名規則を使用して「名前」フィールドの内容が変換された名前です。キャメルケース規則はすべてのコア API の Java JDK で踏襲され、API 名によく似た名前に変換されます。たとえば、「Manufacturer Parts」というクラスは「ManufacturerParts」に、「My new list」というリストは「MyNewList」に変換されます。

図8: 「API 名」フィールド



APIName検証ルール

APIName の命名規則は、次のルールに従う必要があります。

- APIName にはシングルスバイト文字のみ含めることができ、Unicode やマルチバイト文字は使用できません。
- 有効な Java/XML 識別子である必要があります。
 - 許可される文字は a～z、A～Z、0～9、_（アンダースコア）です。
- 1～255 文字である必要があります。
- 大文字と小文字が区別されます。
- 1 つのコンテキスト内で一意である必要があります。次に例を示します。
 - 属性「Number」は、「Parts」クラスと「Changes」クラスのそれぞれのカバー・ページ・テーブルに存在できます（異なるコンテキスト、「Parts」と「Changes」）。
 - APIName が「Number」の属性は、「Parts」のカバー・ページ・テーブルに 2 つ存在することはできません（同じコンテキスト、「Parts」カバー・ページ）。
 - APIName が「Number」の属性は、「カバー・ページ」、「ページ 2」および「ページ 3」に 2 つ存在することはできません（「カバー・ページ」、「ページ 2」および「ページ 3」は単一コンテキストであるため）。

APIName フィールドを使用したメタデータへのアクセス

Agile メタデータの API 名を使用すると、次の操作を実行できます。

- Agile PLM のメタデータ（ノード、クラス、属性）へのアクセス
- データ・オブジェクトのメタデータ（属性およびテーブル属性）の値へのアクセスおよび操作

Java クライアントで、ノード、クラスおよび属性の API 名を表示できます。次の表に、APIName フィールドをサポートする SDK インタフェースを示します。

APIName フィールドをサポートするAPI

API	例
IAdmin	
IAdmin	
getAgileClass(Object id)	getAgileClass("Parts")
getNode(Object id)	getNode ("Part.TitleBlock")
IAgileClass	
getAttribute (Object key)	getAttribute("TitleBlock.number") または、 getAttribute("number")
getTableAttributes(Object tableId)	getTableAttributes("TitleBlock")
getTableDescriptor (Object id)	getTableDescriptor("TitleBlock")
isSubclassOf(Object cls)	isSubclassOf("Parts")
IAgileList	
getChild(Object child)	getChild("UNITED_STATES") UNITED_STATES is the APIName for entry 'United States' in 'Country' list
getChildNode(Object child)	getChildNode("UNITED_STATES")
setSelection (Object[] childNodes)	setselection(new Object[] { "UNITED_STATES" , "INDIA" })
IAgileSession	
createObject(Object objectType, Object params)	Map map = new HashMap(); String partNumber = "P00001" map.put("TitleBlock.number", partNumber);

	<pre>IDataObject dObj = (IDataObject) (m_session.createObject("Part", map));</pre>
<pre>createObject(int objectType, Object params)</pre>	<pre>Map map = new HashMap(); String partNumber = "P00001" map.put("number", partNumber); IDataObject dObj = (IDataObject) (m_session.createObject("Part", map));</pre>
<pre>getObject(Object objectType, Object params)</pre>	<pre>Map map = new HashMap(); map.put("TitleBlock.number", "1000-01"); IDataObject dObj = (IDataObject) (m_session.getObject("Part", map));</pre>
<pre>getObject(int objectType, Object params)</pre>	<pre>Map map = new HashMap(); map.put("TitleBlock.number", "1000-01"); IDataObject dObj = (IDataObject) (m_session.getObject(IItem. OBJECT_TYPE, map));</pre>
IDataObject	
<pre>getCell(Object key)</pre>	<pre>getCell("PageTwo.list11") or getCell("list11")</pre>
<pre>getTable(Object tableId)</pre>	<pre>getTable("AffectedItems")</pre>
<pre>getValue(Object attribute)</pre>	<pre>getValue ("PageTwo.list11") or getValue ("list11")</pre>
<pre>setValue(Object key,Object value)</pre>	<pre>setValue("PageTwo.text01","test")</pre>
<pre>saveAs(Object type,Object params)</pre>	<pre>Map params = new HashMap(); params.put("number", number); IItem part2 = (IItem) part.saveAs("Document", params);</pre>
<pre>setValues(Map map)</pre>	<pre>Map map = new HashMap(); map.put("TitleBlock.number", "1000-01"); part.setValues(map);</pre>
ILibrary	

<code>getAdminList(Object listId)</code>	<code>getAdminList("ActionStatus")</code>
<code>createAdminList(Map map)</code>	<pre> map.put(IAdminList.ATT_NAME, "My List"); map.put(IAdminList.ATT_APINAME, "MyList"); map.put(IAdminList.ATT_DESCRIPTION, "My desc"); map.put(IAdminList.ATT_ENABLED, new Boolean(false)); map.put(IAdminList.ATT_CASCADE, new Boolean(false)); IAdmin admin = m_session.getAdminInstance(); IListLibrary listLibrary = admin.getListLibrary(); IAdminList newList = listLibrary.createAdminList(map); </pre>
INode	
<code>getChild(Object child)</code>	<pre> IAdmin admin = m_session.getAdminInstance(); INode node = admin.getNode(NodeConstants.NODE_AGILE_C LASSES); INode partsClass = node.getChild("Parts"); </pre>
<code>getChildNode(Object child)</code>	<pre> IAdmin admin = m_session.getAdminInstance(); INode node = admin.getNode("AgileClasses"); INode partsClass = node.getChildNode("Parts"); </pre>
IProgram	
<code>saveAs (Object type, Object[] tablesToCopy, Object params)</code>	<pre> HashMap map = new HashMap(); map.put("name", new_number); map.put("scheduleStartDate", new Date()); Object[] objects = new Object[]{"PageTwo", "PageThree", "Team"}; IProgram program2 = (IProgram)program.saveAs("Program", objects, map); </pre>

<pre>saveAs(Object type, Object[]tablesToCopy, Object params, boolean applyToChildren)</pre>	<pre>HashMap map = new HashMap(); map.put("name", new_number); map.put("scheduleStartDate", new Date()); Object[] objects = new Object[]{"PageTwo", "PageThree", "Team"}; IProgram program2 = (IProgram)program.saveAs("Program", objects, map, true);</pre>
IProject	
<pre>assignSupplier(Object supplierParams)</pre>	<pre>HashMap map = new HashMap(); map.put("Responses.itemNumber", item.getName()); map.put("Responses.supplier", supplier.getName()); rfq.assignSupplier(map);</pre>
IQuery	
<pre>setResultAttributes (Object[] attrs)</pre>	<pre>String[] attrs = new String[3]; attrs[0] = "TitleBlock.number"; attrs[1] = "TitleBlock.description"; attrs[2] = "TitleBlock.lifecyclePhase"; query.setResultAttributes(attrs);</pre>
IRequestForQuote	
<pre>assignSupplier (Object supplierParams)</pre>	<pre>HashMap map = new HashMap(); map.put("Responses.itemNumber", item.getName()); map.put("Responses.supplier", supplier.getName()); rfq.assignSupplier(map);</pre>
ITable	
<pre>createRow(Object param)</pre>	<pre>HashMap params = new HashMap(); params.put("itemNumber", "P0001"); params.put("newRev", "A"); ITable affectedItems = change.getTable("AffectedItems");</pre>

	<pre> IRow affectedItemRow = affectedItems.createRow(params); </pre>
<code>createRows(Object[] rows)</code>	
<code>getAvailableValues(Object attr)</code>	<code>getAvailableValues("PageTwo.list01")</code>
<code>updateRows (Map rows)</code>	<pre> HashMap[] mapx = new HashMap[5]; Map rows = new HashMap(); mapx[0] = new HashMap(); mapx[0].put("newRev", "A"); mapx[0].put("text01", "sdk test1"); rows.put(rowArray[0], mapx[0]); mapx[1] = new HashMap(); mapx[1].put("newRev", "A"); mapx[1].put("text01", "sdk test2"); rows.put(rowArray[1], mapx[1]); tab.updateRows(rows); </pre>
ITableDesc	
<code>getAttribute (Object key)</code>	<code>getAttribute("number")</code>

APINameフィールドを取得するSDK API

インタフェース	メソッド
IAdminList	<code>getAPIName()</code>
IAgileClass	<code>getAPIName()</code>
IAgileList	<pre> getAPIName() addChild(Object child, String apiName) </pre>
IAttribute	<code>getAPIName()</code>
ICell	<code>getAPIName()</code>
INode	<code>getAPIName()</code>
IProperty	<code>getAPIName()</code>
ITableDesc	<code>getAPIName()</code>

ルート管理者ノードのAPI名

次の表に、Agile Java クライアントでは公開されていないトップレベルの管理者ノードの API 名を示します。トップレベルの管理者ノードとは、単独で存在する管理者ノードです。つまり、トップレベルの管理者ノードがある場合、別の管理者ノードは存在できません。たとえば、「クラス」と「役割」はトップレベルのノードですが、「ライフ・サイクル・フェーズ」と「属性」は、別の管理者ノードに属しているため、トップレベルのノードではありません。同様に、「ワークフロー・ステータス」は「ワークフロー」に属しているため、トップレベルのノードではありません。

ルート・ノード	API 名
ACS の回答	ACSResponses
アカウント規約	AccountPolicy
アクティビティ・ステータス	ActivityStatuses
アクティビティ・ヘルス	ActivityHealths
Agile クラス	AgileClasses
Agile ワークフロー	AgileWorkflows
Agile eHub	AgileEHubs
添付ファイル・ページ設定	AttachmentPurgeSetting
自動採番	AutoNumbers
キャッチャ	Catchers
文字セット	CharacterSet
クラスタ	Cluster
組織のプロファイル	CompanyProfile
条件ライブラリ	CriteriaLibrary
ダッシュボード管理	DashboardManagement
デフォルトの役割の設定	DefaultRoleSettings
送信先	Destinations
イベント・ハンドラ・タイプ	EventHandlerTypes
イベント・ハンドラ	EventHandlers
イベント確認通知受信者	EventSubscribers
イベント・タイプ	EventTypes
イベント	Events
役割/権限の例	ExampleRolePrivilege
フィルタ	Filters
全文検索設定	FullTextSearchSettings
インポート・プリファレンスの設定	ImportPreferenceSetting

LDAP 設定	LDAPConfig
ライフサイクル・フェーズ	LifeCyclePhases
私の割当て	MyAssignments
通知テンプレート	NotificationTemplates
PGC スマートルール	PGCSmartRules
パッケージ・ファイル・タイプ	PackageFileTypes
ポータル	Portals
プリファレンス	Preferences
権限	Privileges
プロセス拡張ライブラリ	ProcessExtensionLibrary
クエリー・クリーン・アップ	QueryCleanup
見積依頼取引条件	RFQTermsAndConditions
レポート	Reports
役割	Roles
サーバーの場所	ServerLocation
サインオフ・メッセージ	SignOffMessage
スマートルール	SmartRules
確認通知受信者	Subscribers
タスクの設定	TaskConfiguration
UOM シリーズ	UOMFamilies
Viewer とファイル	ViewerAndFiles
wCM サーバー	WCMServers

API名の例

次の例は、Agile PLM サーバーへのログイン方法、2 つの部品の作成方法、「ページ 2」の「テキスト 01」および「リスト 20」の有効化と値の設定方法、最初の部品の「BOM」テーブルへの 2 番目の部品の追加方法を示しています。

例: APIName フィールドを使用したメタデータへのアクセス

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

/**
 * This sample code shows how to use the API name.
 * It uses some of the SDK APIs with the API name.
 * For a list of API names for attributes and classes,
```

```
* refer to Agile Java Client.
* Some API names in Agile Java Client may differ from the ones
* in this example. This is because a duplicate conflict
* was detected in the API name in the same context.
* If detect this conflict, be sure change the API name
* in this sample before compiling and executing the code.
*/

public class APIName
{
    public static final String      USERNAME = "admin";
    public static final String      PASSWORD = "agile";
    public static final String      URL      =
        "http://localhost:8080/Agile";

    public static IAgileSession      session      = null;
    public static IAdmin              admin        = null;
    public static AgileSessionFactory factory = null;
    public static IListLibrary        listLibrary  = null;

    /**
     * @param args
     */
    public static void main(String[] args) {
        try {
            // Create an IAgileSession instance
            session = connect(session);
            admin    = session.getAdminInstance();
            listLibrary = admin.getListLibrary();

            // Create two parts
            IItem itemParent = createItem(getAutonumber());
            IItem itemChild = createItem(getAutonumber());
            // enable Page Two tab for Part
            enableP2();

            // enable Page Two Text 01 and set value
            setP2Text(itemParent);

            // create a new AdminList
            createAdminList();

            // enable Page Two List 20 and set value
            setP2List(itemParent);

            // Add the child part to the BOM table of the parent part
            ITable bomTable = addBOM(itemParent, itemChild);
        }
        catch (Exception e) {
            e.printStackTrace();
        }
        finally {
            session.close();
        }
    }
}
```

```

/**
 * @throws APIException
 */
private static void createAdminList() throws APIException {
    Map listParams = new HashMap();
    listParams.put(IAdminList.ATT_APINAME, "MY_LIST"); // Specify
    the API name of the List
    listParams.put(IAdminList.ATT_NAME, "My List");
    listParams.put(IAdminList.ATT_ENABLED, new Boolean(true));
    IAdminList myList = listLibrary.createAdminList(listParams);
    IAgileList values = myList.getValues();
    values.addChild("Value A", "VAL_A"); // Specify the API name
    along with the value
    values.addChild("Value B", "VAL_B");
    values.addChild("Value C", "VAL_C");

    myList.setValues(values);
    System.out.println("Created Admin List " + myList.getName());
}

/**
 * @throws APIException
 */
private static void enableP2() throws APIException {
    INode p2 = admin.getNode("Part.PageTwo"); // Fully qualified
    API name
    IProperty visible =
    p2.getProperty(PropertyConstants.PROP_VISIBLE);
    IAgileList values = visible.getAvailableValues();
    values.setSelection(new Object[]{ "Yes" });
    visible.setValue(values);
    System.out.println("Page two enabled for Part class");
}

/**
 * @param itemParent
 * @throws APIException
 */
private static void setP2Text(IItem itemParent) throws APIException
{
    IAgileClass clazz = itemParent.getAgileClass();
    ITableDesc p2TableDesc = clazz.getTableDescriptor("PageTwo");
    // 'PageTwo' is the API name of the Page Two tab
    IAttribute text01 = p2TableDesc.getAttribute("text01"); //
    'text01' is the API name of the Text01 field
    IProperty visible =
    text01.getProperty(PropertyConstants.PROP_VISIBLE);
    IAgileList values = visible.getAvailableValues();
    values.setSelection(new Object[]{ "Yes" });
    visible.setValue(values);
    itemParent.setValue("PageTwo.text01", "SDK test"); //
    'PageTwo.text01' is the fully qualified APIName for
    ItemConstants.ATT_PAGE_TWO_TEXT01
    System.out.println("Set P2 Text01 " +
    itemParent.getValue("PageTwo.text01") + " for Part " +
    itemParent.getName());
}

/**

```

```

    * @param itemParent
    * @throws APIException
    */
private static void setP2List(IItem itemParent) throws APIException
{
    IAgileClass clazz = itemParent.getAgileClass();
    ITableDesc p2TableDesc = clazz.getTableDescriptor("PageTwo");
    // 'PageTwo' is the API name of the Page Two tab
    IAttribute text01 = p2TableDesc.getAttribute("list20"); //
    // 'list20' is the API name of the List20 field
    IProperty visible =
    text01.getProperty(PropertyConstants.PROP_VISIBLE);
    IAgileList values = visible.getAvailableValues();
    values.setSelection(new Object[]{ "Yes" });
    visible.setValue(values);
    IAdminList myList = listLibrary.getAdminList("MY_LIST"); //
    MY_LIST is the API name of the List 'My List'
    IAgileList listValues = myList.getValues();
    listValues.setSelection(new Object[] { "VAL_B" } ); // VAL_B
    // is the API name of the list value 'Value B'
    itemParent.setValue("PageTwo.list20", listValues); //
    // 'PageTwo.list20' is the fully qualified APIName for
    ItemConstants.ATT_PAGE_TWO_TEXT01
    System.out.println("Set P2 List20 " +
    listValues.getSelection()[0].getValue() + " for Part " +
    itemParent.getName());
}

/**
 * <p> Create an IAgileSession instance </p>
 * @param session
 * @return IAgileSession
 * @throws APIException
 */
private static IAgileSession connect(IAgileSession session)
    throws APIException {
    factory = AgileSessionFactory.getInstance(URL);
    HashMap params = new HashMap();
    params.put(AgileSessionFactory.USERNAME, USERNAME);
    params.put(AgileSessionFactory.PASSWORD, PASSWORD); session =
    factory.createSession(params);
    return session;
}

/**
 * <p> Create a part </p>
 * @param parent
 * @return IItem
 * @throws APIException
 */
private static IItem createItem(String number) throws APIException {
    HashMap map = new HashMap();
    map.put("TitleBlock.number", number); // 'number' or
    // 'TitleBlock.number' is the APIName for
    ItemConstants.ATT_TITLE_BLOCK_NUMBER
    map.put("description", "test"); // 'description'
    // or 'TitleBlock.description' is the APIName for
    ItemConstants.ATT_TITLE_BLOCK_DESCRIPTION
    String p = "P" + System.currentTimeMillis();

```

```
        IItem item = (IItem)session.createObject("Part",map); //
        'Part' is the API name of the Part class
        System.out.println("Created Part " + number);
        return item;
    }

    /**
     * <p> Add the child parts to the BOM table of the parent part </p>
     * @param itemParent
     * @param itemChild1
     * @param itemChild2
     * @return ITable
     * @throws APIException
     */
    private static ITable addBOM(IItem itemParent, IItem itemChild)
    throws APIException {
        ITable table = itemParent.getTable("BOM");
        // 'BOM' is APIName for ItemConstants.TABLE_BOM
        IRow row1 = table.createRow();
        String number =
            (String)itemChild.getValue("TitleBlock.number");
        row1.setValue("itemNumber", number);
        // 'itemNumber' is APIName for 'Item Number' on BOM Table
        System.out.println("Added Part " + itemChild.getName() + " to
        BOM of the Part " + itemParent.getName());
        return table;
    }

    /**
     * @return
     * @throws APIException
     */
    private static String getAutonumber() throws APIException{
        IAgileClass cls = admin.getAgileClass("Part");
        // 'Part' is the API name of the Part class
        IAutoNumber auto[] = cls.getAutoNumberSources();
        String number = null;
        if(auto != null && auto.length > 0)
            number = auto[0].getNextNumber();
        else
            number = "PART" + System.currentTimeMillis();
        return number;
    }
}
```


Agile PLM オブジェクトの確認通知

この章のトピック

- ユーザー確認通知について 139
- オブジェクトに対する確認通知の取得 142
- オブジェクトに対する確認通知の変更 143
- 確認通知での属性の使用可能化 144
- 「確認通知」テーブルの使用 145

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。

ルーティング可能なオブジェクトとルーティング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（ルーティング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（ルーティング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 Projects 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。

ほとんどのルーティング可能なオブジェクトおよびルーティング不可能なオブジェクトで、前述の表にリストされている確認通知イベント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、次の 2 種類の Agile PLM 通知のトリガーとなります。

- **電子メール** - 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。ユーザー・プリファレンスおよびシステム・プリファレンスの詳細は、『Agile PLM 管理者ガイド』を参照してください。
- **受信トレイ** - 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。

管理者権限があるユーザーは、Java クライアントでこれらの通知を作成して設定できます。Java クライアントには、この目的で使用する類似した 2 つのダイアログが用意されています。2 つのダイアログが必要なのは、電子メール通知と受信トレイ通知の 2 つのセットがあるためです。

- 「送信先」フィールドが灰色表示されている通知
- 確認通知イベントがトリガーされると通知を受信する受信者を管理者ユーザーが選択できる通知

SDKを使用した通知の送信

通知については、SDK ガイドのイベント通知に関する項で簡単に説明しています。SDK では、Agile PLM オブジェクト用の指定のテンプレートを使用して特定の通知者に通知を送信するために、次の API が公開されています。通知の詳細は、『Agile PLM 管理者ガイド』を参照してください。

<code>sendNotification</code>	<code>(IDataObject</code>	<code>object</code>	通知の発行対象になるオブジェクト
	<code>String</code>	<code>template</code>	通知テンプレートの名前
	<code>Collection</code>	<code>notifiers</code>	通知者のリスト
	<code>boolean</code>	<code>urgent</code>	緊急の場合は <code>True</code>
	<code>String</code>	<code>comments</code>	通知に関するコメント
	<code>)</code>		<code>throws APIException</code>

各パラメータは次のように定義されます。

- **object** - 通知の発行対象になるオブジェクト
- **template** - 通知テンプレートの名前
- **notifiers** - 個々の IDataObjects (IUser、IUserGroup など) であるユーザーのリストを含むコレクション
- **urgent** - 緊急に送信する場合は値を true に設定し、そうでない場合は false に設定する
- **comments** - 通知に関するコメント

これらのパラメータおよびAPIの詳細は、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。これらのファイルは、SDK_samples (ZIPファイル) のHTMLフォルダにあります。このファイルにアクセスするには、2ページの「[クライアント側のコンポーネント](#)」の注意を参照してください。

次の例は、sendNotification を使用して、ECO C0001 から user1 と user2 に通知コメントとテンプレートとともに通知を送信する方法を示しています。

例: Agile SDK を使用した通知の送信

```
IAgileSession session =
    AgileSessionFactory.createSessionEx(loginParams);
List notifyList =
    new ArrayList();
IDataObject user =
    (IDataObject) session.getObject(com.agile.api.IUser.OBJECT_TYPE, "John
    Doe");
notifyList.add(user);
user =
    (IDataObject) session.getObject(com.agile.api.IUser.OBJECT_TYPE, "Jane
    Doe");
notifyList.add(user);
IDataObject agileObject =
    (IDataObject) session.getObject(com.agile.api.IChange.OBJECT_TYPE, "C000
    1");
boolean urgent = true;
String comment =
    "Add ECO approver, Notify CA";
String template =
    "Automated SDK process added ECO approver";
session.sendNotification(obj, templateName, notifyList, urgent, comment);
```

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_CHANGE)) {
    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、サーバーで設定され、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も変更できます。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、「ページ 2」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_TWO_L
                IST01)) ||
                (subs[i].getAttribute().getId().equals(CommonConstants.ATT_PAGE_TWO_TE
                XT01)))) )
                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「ページ 2」のすべての属性、および「ページ 3」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「ページ 2」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「ページ 2」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は追加できません。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は削除できます。

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のクリアもできます。

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);
}
```


製造拠点の管理

この章のトピック

■ 製造拠点について	149
■ 拠点へのアクセスの管理	149
■ 製造拠点の作成	150
■ 製造拠点のロード	150
■ アイテムの「拠点」テーブルの取得	151
■ 「拠点」テーブルへの製造拠点の追加	151
■ アイテムの現在の製造拠点の選択	152
■ 拠点の無効化	153

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、『Agile PLM Product Collaboration ガイド』を参照してください。

拠点へのアクセスの管理

組織で製造拠点を使用するには、Agile 管理者の拠点モジュールが有効である必要があります。Agile PLM で有効となるモジュールは、Oracle との契約内容によって決定されます。

製造拠点へのユーザーのアクセスは、そのユーザーに割り当てられている役割と権限、およびそのユーザーのプロファイルの拠点プロパティによって管理されます。組織は、必要な数の製造拠点を作成できますが、ユーザーのユーザー・プロファイルの拠点プロパティにすべての拠点が指定されてないかぎり、そのユーザーはすべての拠点にアクセスできません。組織によっては、ユーザー・プロファイルの拠点プロパティによる決定で、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 システムを実装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。拠点および拠点の有効化の詳細は、『Agile PLM 管理者ガイド』を参照してください。

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.C
    LASS_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、より簡単に迅速に取得できます。`getManufacturingSites()` の使用例は、152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().getV
    alue(
    UserConstants.ATT_GENERAL_INFO_DEFAULT_SITE)).getSelection()[0].getVal
    ue());
    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);
}
```


リストの使用

この章のトピック

■ リストについて	155
■ リスト値の選択	159
■ リスト・ライブラリからのリストの選択	163
■ カスタム・リストの作成	165
■ リストのデータ・タイプの確認	170
■ リストの変更	170
■ IAgileListオブジェクトのコンテンツの印刷	174

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、「ページ 2」および「ページ 3」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9: リスト・ライブラリ

ID	Name	Description	API Name	Enabled	Editable	Cascade	Display Type
18414	Action Status	Action Status	ActionStatus	Yes	No	No	List
2000008548	Agile Script Log Leve...		AgileScriptL...	Yes	Yes	No	List
2249	AML Preferred Status	AML Preferred Status	AMLPreferre...	Yes	Yes	No	List
4682	AttachType List	AttachType List	AttachTypeList	Yes	Yes	No	List
6820	Audit Result	Audit Result	AuditResult	Yes	Yes	No	List
8934	Buyer	Buyer	Buyer	Yes	Yes	No	List
2000001019	Calculated Compliance	Calculated Compliance	CalculatedC...	Yes	No	No	List
2000011160	Cas Number Match f...	Cas Number Match f...	CasNumber...	Yes	No	No	List
2624820	Cascade1	Cascade1	Cascade1	Yes	Yes	Yes	List
2000000192	Category 10 List	Category 10 List	Category10...	Yes	Yes	No	List
2000000189	Category 7 List	Category 7 List	Category7List	Yes	Yes	No	List

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 つの値のみを選択できます。次の図は、Agile Web クライアントの部品タイプに対するシングルリストのセルを示しています。

図10: Agile Web クライアントのシングルリストのセル

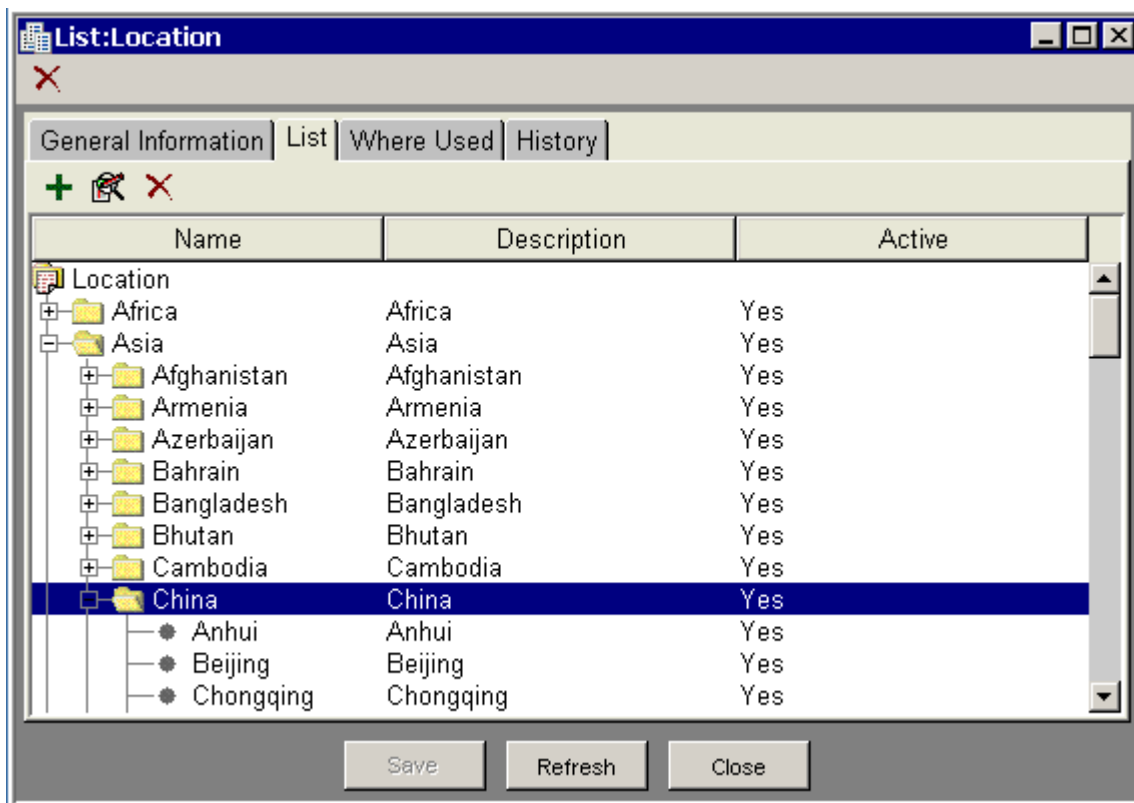
* Part Type:

- Capacitor
- Connector
- Diode
- Fuse
- Hard Disk
- Hardware
- IC**
- Inductor
- Part
- Resistor
- Resistor Network
- Socket
- Switch
- Transistor

カスケード・リスト

Agile Java クライアントでは、シング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11: Agile Java クライアントでのカスケード・リストの設定

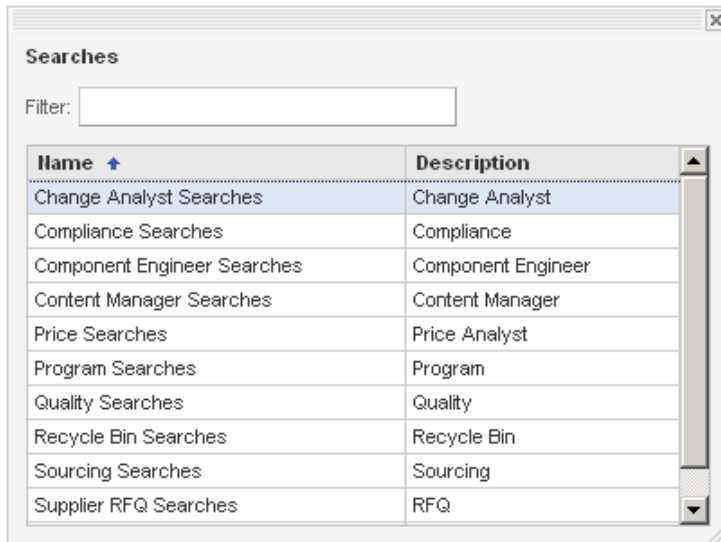


注意 「場所」リストは、Agile PLM で出荷する唯一のカスケード・リストです。ただし、独自のカスケード・リストも定義できます。

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てマルチリストのセルの値を選択できます。

図12: Agile Web クライアントの「複数の値の選択」ウィンドウ



IAgileListを使用するメソッド

IAgileList インタフェースには、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。

注意 `IAgileList getAPIName()` を使用すると、リストの利用可能な値を取得できます。詳細は、125 ページの「[APIName フィールドを使用した PLM メタデータへのアクセス](#)」を参照してください。

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、「ページ 2」属性の利用可能な値のリストを再利用して、「ページ 3」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);
}
```

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);
}
```

動的リストの使用

Agile サーバーには、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、「ページ2」と「ページ3」フィールドに対して使用できますが、これらのリストに対して値を列挙することはできません。

列挙可能リストと列挙不可能リスト

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ー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 `null` を返した場合、リストの値は列挙できません。ただし、リストの値は選択できます。

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);
    }
}
}

```

列挙不可能なPG&Cリスト

以前のリリースの SDK で列挙可能であった次の PG&C リストは、このリリースでは列挙不可能になりました。

- デクラレーション
- サブスタンス
- 含有基準
- 部品ファミリ
- 部品ファミリの部品分類

リスト・ライブラリからのリストの選択

IListLibrary インタ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も指定できます (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、「ページ 2」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、「ページ 2」および「ページ 3」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できます。さらに、リストを重ねて表示し、複数のレベルも設定できます。

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_CASCADE` - 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_CASCADE` の値は変更できません。

リストを作成した後は、`IAdminList` インタ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・リストには変更できません。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_CASCADE, 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は共有できません。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 `null` 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、サーバーでは、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は再利用できません。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、167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_CASCADE, 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);
}
```


添付ファイルとファイル・フォルダ・オブジェクトの使用

この章のトピック

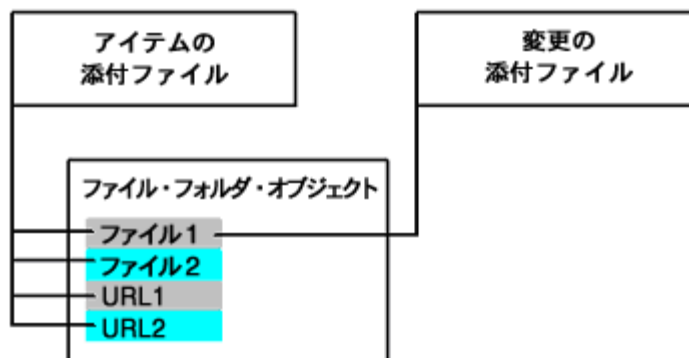
- 添付ファイルとファイル・フォルダについて 177
- ファイル・フォルダの使用 178
- オブジェクトの「添付ファイル」テーブルの使用 184
- ファイル・フォルダのチェックアウト 186
- ファイル・フォルダのチェックアウトのキャンセル 187
- 「添付ファイル」テーブルへのファイルおよびURLの追加 188

添付ファイルとファイル・フォルダについて

オブジェクトの添付ファイルには、オブジェクト（製造過程）に関する情報が含まれています。ファイルと URL は、ファイル・フォルダ・オブジェクト内にあるそれらを参照することで添付できます。ファイル・フォルダ・オブジェクトでは、関連コンテンツ（添付ファイル）が保持されます。ほとんどの主要な Agile API オブジェクト（IItem、IChange、IManufacturer、IManufacturerPart、IPackage、ITransferOrder、IUser、IUserGroup など）には、「添付ファイル」テーブル（Java クライアントのタブ）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。

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です。

図13: 「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、ファイルをダウンロードした後は、別のアプリケーションを使用して添付ファイルを表示、編集または印刷できます。

ファイル・フォルダは、ファイルの PLM サーバー格納庫に保存されている 1 つ以上のファイルまたは URL を

指定するビジネス・オブジェクトです。また、ファイル・フォルダには独自のテーブルのセットがあります。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。

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」プロパティを設定してください。「管理」>「設定」>「サーバー設定」>「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」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サーバー設定の詳細は、『Agile PLM管理者ガイド』を参照してください。

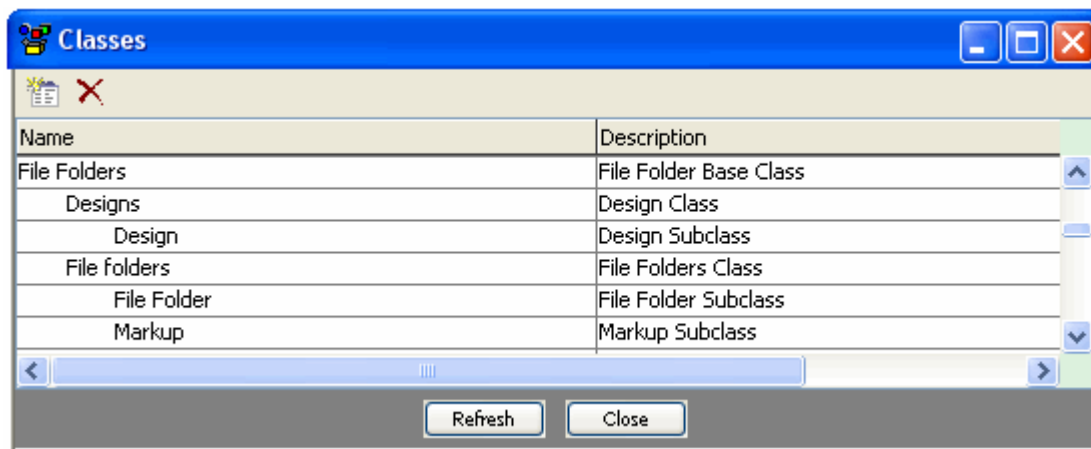
ファイル・フォルダの使用

添付ファイルと同様に、SDK では、ファイル・フォルダ関連のタスクを実行するための API が公開されています。このタスクには、「添付ファイル」テーブルの行のオブジェクトに関連するファイルのチェックインとチェックアウト、「添付ファイル」テーブルへのファイルおよび URL の追加、添付ファイルの削除などがあります。このセクションでは、これらの機能と、SDK を使用したタスクの実行に必要な手順について説明します。

ファイル・フォルダのクラスとサブクラス

ファイル・フォルダの基本クラスには、2つのクラスがあり、各クラスには独自のサブクラスがあります。次の図に、ファイル・フォルダの基本クラス、クラスおよびサブクラスを示します。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。手順については、「ファイル・フォルダの使用」を参照してください。

図14: ファイル・フォルダのクラスとサブクラス



次の表に、これらのクラスとオブジェクトの説明を示します。

基本クラス	クラス	サブクラス	説明
ファイル・フォルダ	デザイン	デザイン	CAD でのモデル構造の構築を許可するオブジェクト。

	ファイル・フォルダ	ファイル・フォルダ マークアップ	ファイルまたは URL が含まれているオブジェクト。 このクラスには、履歴レポート・ファイル・フォルダを除く、すべてのファイル・フォルダ・オブジェクトが含まれています。
--	-----------	---------------------	---

これらのオブジェクトの送信の詳細は、28ページの「[Agile PLMオブジェクトの状態の確認](#)」を参照してください。

ファイル・フォルダのテーブルと定数

ファイル・フォルダ・オブジェクトでは、次のテーブルとそれに対応する定数がサポートされています。

テーブル	定数	読取り/書込みモード
タイトル・ブロック	TABLE_TITLEBLOCK	読取り/書込み
ページ 2	TABLE_PAGETWO	読取り/書込み
ページ 3	TABLE_PAGETHREE	読取り/書込み
ファイル	TABLE_FILES	読取り/書込み
構造	TABLE_STRUCTURE	読取り/書込み
送信スリップ/ワークフロー	TABLE_WORKFLOW	読取り/書込み
関係	TABLE_RELATIONSHIPS	読取り専用
履歴	TABLE_HISTORY	読取り専用
使用箇所	TABLE_WHEREUSED	読取り/書込み
デザイン使用箇所	TABLE_WHEREUSEDDESIGN	読取り専用

ファイル・フォルダ・オブジェクトの作成

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);
    ff.checkOutEx();
}
```

注意 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・オブジェクトが自動的に作成されます。182ページの「[「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・オブジェクトの作成](#)」を参照してください。

ファイル・フォルダ・デザイン・クラスは、CAD オブジェクト用の「構造」テーブル（Java クライアント UI のタブ）が追加されたファイル・フォルダ・クラスに類似しています。次の例は、デザイン・オブジェクトの作成方法、構造ツリーへのデザイン・オブジェクトの追加方法および構造テーブルのロード方法を示しています。

例: デザイン・オブジェクトの作成

```
// autoNum is autoNumber as usual
IFileFolder obj = (IFileFolder) m_session.createObject(
    FileFolderConstants.CLASS_DESIGN, autoNum);
```

例: 構造ツリーへのデザイン・オブジェクトの追加

```
IFileFolder obj = // some Design object
IFileFolder childObj1 = // some Design object
IFileFolder childObj2 = // some Design object

obj.checkOutEx();
ITable table = obj.getTable(FileFolderConstants.TABLE_STRUCTURE);
// add row
Object[] vers = childObj1.getVersions();
IRow row = table.createRow(childObj1);
row.setValue(FileFolderConstants.ATT_STRUCTURE_LABEL,
    "label modified by creating row 1");
row = table.createRow(childObj2);
row.setValue(FileFolderConstants.ATT_STRUCTURE_LABEL,
    "label modified by creating row 2");
obj.checkIn();
```

例: 構造テーブルのロード

```
public void testLoadingDesignStructureTable() throws Exception {
    addCaseInfo("Design Object", "load Structure table", "");
    // assuming Design object Design00004 existed with some data in Structure
    IFileFolder obj = (IFileFolder) m_session.getObject(
        FileFolderConstants.CLASS_DESIGN, "Design00004");
    IAgileClass agileClass = obj.getAgileClass();
    // load Structure table
    ITable table = obj.getTable(FileFolderConstants.TABLE_STRUCTURE);
    Integer tableId = (Integer) table.getTableDescriptor().getId();
    // ITable performs related tasks
}
```

例: ツリーとしての構造テーブルのロード

```
public void testLoadingDesignStructureTree()
throws Exception
{
    addCaseInfo("Design Object", "load Structure tree", "");
    // assuming Design object Design00004 existed with some data in Structure
    IFileFolder obj = (IFileFolder) m_session.getObject(
        FileFolderConstants.CLASS_DESIGN, "Design00004");
    IAgileClass agileClass = obj.getAgileClass();
    // load Structure table
    ITable table = obj.getTable(FileFolderConstants.TABLE_STRUCTURE);
    Integer tableId = (Integer) table.getTableDescriptor().getId();
    ITreeNode root = (ITreeNode) table;
    Collection topLevelChildren = root.getChildNodes();
}
```

```

        Iterator it;
        ITreeNode row;
        if (topLevelChildren != null) {
            it = topLevelChildren.iterator();
            int level = 0;
            while (it.hasNext()) {
                row = (ITreeNode) it.next();
                if(row instanceof IRow) {
                    IRow aRow = (IRow) row;
                    IDataObject referent =
                        aRow.getReferent();
                    if(referent != null) {
                        System.out.println(
                            "Row Referent Object ID/row:
                            "+ referent.getObjectId()+ " / "
                            + referent.getName());
                    }
                }
                iterateTreeNode(agileClass, true,
                    tableId, (ITreeNode) row);
                count++;
            }
        }

        System.out.println("The number of rows in top level is " + count);
    }

    private void iterateTreeNode(IAgileClass agileClass, boolean print,
        Integer tableId, ITreeNode node) throws APIException {
        Collection childNodes = node.getChildNodes();
        printRow(agileClass, print, tableId, (IRow) node);
        if (childNodes == null || childNodes.size() <= 0) {
            return;
        }

        Iterator it = childNodes.iterator();
        ITreeNode childNode;
        IRow row;

        while (it.hasNext()) {
            childNode = (ITreeNode) it.next();
            if (childNode instanceof IRow) {
                row = (IRow) childNode;
                if(row instanceof IRow) {
                    IDataObject referent =
                        row.getReferent();
                    if(referent != null) {
                        System.out.println
                            ("Row Referent Object ID/row:
                            "+ referent.getObjectId()+ " / "
                            + referent.getName());
                    }
                }
            }
        }
    }

```

```

        }
        iterateTreeNode(agileClass, print, tableId, (ITreeNode) childNode);
    }
}

```

「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・オブジェクトの作成

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「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を使用したAgile PLMファイル格納庫内のファイルへのアクセス

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フェースです。このインタフェースは、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	<code>getFile()</code> 戻り値	<code>isSecure()</code> 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 <code>InputStream</code> (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 <code>InputStream</code> 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 <code>true</code> 。
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 <code>InputStream</code> 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 <code>true</code> 。
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 <code>InputStream</code> が返されます。	参照先ファイルが URL でない場合は <code>true</code> 。

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;
}
```

オブジェクトの「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。`ITable.getTableIterator()` メソッドを使用すると、テーブルの双方向の `Iterator` を取得できます。

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
        InputStream 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を使用したファイルのチェックインとチェックアウト

`ICheckoutable` は、オブジェクトに関連付けられているファイルのチェックインおよびチェックアウトに使用できるインタフェースです。このインタフェースは、「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、111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かどうかの確認

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。

確定した添付ファイルは表示できますが、変更することはできません。確定した添付ファイルを変更するには、次の例に示すように添付ファイルを未確定にするか、新規の変更指示を発行して新規リビジョンを作成する必要があります。リビジョンが確定済かどうかを確認する方法の詳細は、113ページの「[リビジョンの確定済ステータスの変更](#)」を参照してください。

例: 添付ファイルの確定

```
class IncorporateItem implements ICustomAction {
    public ActionResult doAction(IAgileSession session, INode
        actionNode,
            IDataObject affectedObject)    {
        try {
            System.out.println("Workflow action kicked off....");
            IItem object = (IItem)affectedObject;
            System.out.println("Incorporating...");
            IItem loItem = (IItem) session.getObject(IItem.OBJECT_TYPE,
                object.getName());
            //this will get the latest version. Make sure the latest is against
            a MCO
                loItem.setIncorporated(true);
            //incorporate the attachment
            System.out.println("Attachment added.");
            String message = ("Incorporated "+object);
            return new ActionResult(ActionResult.STRING, message);
        } catch (APIException ae) {
            ae.printStackTrace();
            return new ActionResult(ActionResult.EXCEPTION, ae);
        }
    }
}
```

例: 添付ファイルの未確定

```
class IncorporateItem implements ICustomAction {
    public ActionResult doAction(IAgileSession session, INode
        actionNode,
            IDataObject affectedObject)    {
        try {
            System.out.println("Workflow action kicked off....");
            IItem object = (IItem)affectedObject;
            System.out.println("Incorporating...");
            IItem loItem = (IItem) session.getObject(IItem.OBJECT_TYPE,
                object.getName());
            //this will get the latest released version
            loItem.setIncorporated(false);
            //Unincorporate the attachment
            System.out.println("Attachment added.");
            String message = ("Incorporated "+object);
            return new ActionResult(ActionResult.STRING, message);
        } catch (APIException ae) {
            ae.printStackTrace();
            return new ActionResult(ActionResult.EXCEPTION, ae);
        }
    }
}
```

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。184ページの「[ファイルのチェックインとチェックアウト](#)」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。184ページの「[ファイルのチェックインとチェックアウト](#)」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、サーバーでは、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、サーバー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 では、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の MCO がある場合。
- 現在のリビジョンが確定済の場合。

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。

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;
}
```

次の例は、addAttachment() メソッドのいくつかのインスタンスを使用して行を「添付ファイル」テーブルに追加するそれぞれの方法を示しています。

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

```

I AgileClass 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);

```

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、プログラムに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・ウィンドウに表示できます。

添付ファイルとファイル・フォルダの削除

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、31 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・サーバー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ペ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、81 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、関連するファイル・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。

サムネイルの使用

Agile PLM では、グラフィカル・オブジェクトを表すキー・オブジェクト、またはイメージが必要なキー・オブジェクトへの小さな静的グラフィカル・イメージ（サムネイル）の追加がサポートされています。たとえば、キー・オブジェクトには、Excel ワークシート、テキスト・ファイル、PDF ファイル、CAD ファイルなどのファイルとして添付されるドキュメントがありますが、サムネイルには、これらのイメージ・ファイルの縮小バージョンが表示され、部品オブジェクトの場合は、それらのファイルの相互の関連性が示されます。

SDK では、次のサムネイル関連機能をサポートしています。

- サムネイルの再生成
- サムネイルの順序付け
- マスター・サムネイルの設定
- 「添付ファイル」タブへのファイルの追加時におけるサムネイルの生成

サムネイルへのアクセス

Agile SDK では、`IThumbnailContainer` インタフェースによって、ファイル・フォルダ・オブジェクトおよびビジネス・オブジェクトに対するサムネイル関連操作への一般化されたアクセスが提供されます。このインタフェースは、次の API オブジェクトでサポートされています。

- `IFileFolder` オブジェクト
- `IItem` オブジェクト

□ IManufacturerPart オブジェクト

IFileFolder オブジェクトの場合は、前述の API をコールする前に IFileFolder.setCurrentVersion を使用して適用可能なバージョンを設定します。デフォルトのバージョンは LATEST_VERSION です。IItem オブジェクトまたは IManufacturerPart オブジェクトの場合は、これらのオブジェクトにすでに設定されているリビジョンを使用します。

次の例では、IItem オブジェクトまたは IFileFolder オブジェクトの TitleBlock からサムネイル詳細を取得します。

例: IItem オブジェクトまたは IFileFolder オブジェクトの TitleBlock からのサムネイル詳細の取得

```
IItem dataObj =
    (IItem)session.getObject(IItem.OBJECT_TYPE, "P00015");
ITable titleBlockTable =
    this.itemObj.getTable(TableTypeConstants.TYPE_PAGE_ONE);
Iterator i =
    titleBlockTable.getTableIterator();
while (i.hasNext()) {
    IRow row = (IRow)i.next();
    Object thumbnailIDDetails =
        row.getValue(ThumbnailConstants.ATT_THUMBNAIL_ATTACHMENT_TAB);
    IAgileList[] nodes =
        ((IAgileList)thumbnailIDDetails).getSelection();
    for(int ii=0; ii<nodes.length; ii++) {
        IAgileList childNode = nodes[ii];
        IThumbnailID thumbnailID = (IThumbnailID)childNode.getValue();
    }
}
```

サムネイルの再生成

サムネイルの再生成の意図は、ファイル・フォルダ・オブジェクトおよびアイテム・オブジェクトの既存のサムネイル（生成済）に対応するサムネイルの生成にあります。この機能は特に、アセンブリ構造に関連があります。この構造では、アセンブリ構造の子の変更が再生成後のサムネイルに反映されます。

この目的のために、Agile SDK には、IThumbnailContainer.generateThumbnail(IThumbnailID) API が用意されています。この API を起動すると、新規のサムネイルが生成されて返されます。IFileFolder オブジェクトの場合、API では現在のバージョンのオブジェクトが使用されます。IItem オブジェクトまたは IManufacturerPart オブジェクトの場合、API では現在のリビジョンのオブジェクトが使用されます。API が、指定した thumbnailID パラメータのサムネイルの再生成に失敗した場合は、APIException 例外が発生します。

例: IFileFolder オブジェクトのサムネイルの再生成

```
IFileFolder ff =
    (IFileFolder)session.getObject(IFileFolder.OBJECT_TYPE,
    "FOLDER00037");
ff.setCurrentVersion(new Integer(1));
IThumbnailID oldThumbnailID = "";
//get this id from row of supported tables like Title Block
ff.generateThumbnail(oldThumbnailID);
```


例: アイテム・オブジェクトのサムネイルの再生成

```

IItem itemObj =
    (IItem)session.getObject(IItem.OBJECT_TYPE, "P00015");
IThumbnailID oldThumbnailID = "";
//get this id from row of supported tables like Title Block
itemObj.generateThumbnail(oldThumbnailID);

```

マスター・サムネイルの設定

Agile PLM では、ファイル・フォルダ・オブジェクトがサムネイル・ファイルで表されます。このサムネイル・ファイルの「**ファイル**」タブには、複数のファイルを格納できます。**SetMasterThumbnail** を使用すると、ユーザーは、選択したサムネイルのファイル・フォルダを表す「**ファイル**」タブの行を決定できます。

SDK には、ファイル・フォルダ・オブジェクトにマスター・サムネイルを設定するための void `setMasterThumbnail (IRow masterRow) throws APIException` API が用意されています。この機能が `masterRow` パラメータが示すマスター・サムネイルの設定に失敗した場合は、例外が発生します。

例: マスター・サムネイルの設定

```

IFileFolder ff =
    (IFileFolder)session.getObject(IFileFolder.OBJECT_TYPE,
    "FOLDER00037");
ff.setCurrentVersion(new Integer(1));
ITable attachmentTable =
    ff.getTable(FileFolderConstants.TABLE_FILES);
Iterator i =
    attachmentTable.getTableIterator();
while (i.hasNext()) {
    IRow row = (IRow)i.next();
    IRow masterRow = null;
    //set one of the rows as the master row
}

ff.setMasterThumbnail(masterRow);

```

サムネイルの置換

ファイル・フォルダ・オブジェクトおよびアイテム・オブジェクトの場合は、Agile PLM で生成されたサムネイルをユーザー提供のイメージに置換できます。この目的のために、SDK には次の API が用意されています。

```

IThumbnailID replaceThumbnail (IThumbnailID oldThumbnailID, byte[] bytes)
throws APIException

```

この API は、`oldThumbnailID` で指定されたサムネイルを入力ストリームで指定されたイメージ・ファイルに置換します。つまり、置換されたサムネイルの `thumbnailID` が返されます。

`IFileFolder` オブジェクトの場合、API では、オブジェクトにすでに設定されているバージョンが使用されます。`IItem` オブジェクトまたは `IManufacturerPart` オブジェクトの場合は、オブジェクトにすでに設定されているリビジョンが使用されます。`oldThumbnailID` パラメータで指定したサムネイルの置換に失敗した場合は、`APIException` 例外が発生します。

例: IFileFolder オブジェクトのサムネイルの置換

```

IFileFolder ff =
    (IFileFolder)session.getObject(IFileFolder.OBJECT_TYPE,
    "FOLDER00037");
ff.setCurrentVersion(new Integer(1));

```

```
IThumbnailID oldThumbnailID = "";
//get this id from row of supported tables like Title Block
String filePath = "C:\\Earth.bmp";
File file1_tmp = new File(filePath);
byte[] b1 = new byte[(int)file1_tmp.length()];
FileInputStream fileInputStream = new FileInputStream(file1_tmp);
fileInputStream.read(b1);
IThumbnailID newThumbnailID = ff.replaceThumbnail(oldThumbnailID, b1);
```

例: アイテム・オブジェクトのサムネイルの置換

```
IItem itemObj =
    (IItem)session.getObject(IItem.OBJECT_TYPE, "P00015");
IThumbnailID oldThumbnailID = "";
//get this id from row of supported tables like Title Block
String filePath = "C:\\Earth.bmp";
File file1_tmp = new File(filePath);
byte[] b1 = new byte[(int)file1_tmp.length()];
FileInputStream fileInputStream = new FileInputStream(file1_tmp);
fileInputStream.read(b1);
IThumbnailID newThumbnailID = itemObj.replaceThumbnail(oldThumbnailID,
b1);
```

サムネイルの順序付け

Web クライアントのユーザーは、添付ファイルをビジネス・オブジェクトに追加するときに、サムネイル・ナビゲータで表示順序を設定することもできます。Agile PLM には、この機能を SDK で有効にするための `void setThumbnailSequence (IThumbnailID[] thumbnailIDs) throws APIException` API が用意されています。IItem オブジェクトまたは IManufacturerPart オブジェクトの場合、API では、オブジェクトにすでに設定されているリビジョンが使用されます。API は、`thumbnailIDs` パラメータを使用して表示順序をソート（順序付け）します。この機能がマスター・サムネイルの設定に失敗した場合は、例外が発生します。

例: サムネイルの順序付け

```
IItem itemObj =
    (IItem)session.getObject(IItem.OBJECT_TYPE, "P00015");
IThumbnailID[] thumbnailIDs = null;
//get this id from row of Title Block table
IThumbnailID[] newSeqOfThumbnailIDs = null;
//generate new order using thumbnail IDs
itemObj.setThumbnailSequence(newSeqOfThumbnailIDs);
```

「添付ファイル」タブへのファイルの追加時におけるサムネイルの生成

この目的のための特別な API はありません。Web クライアントでサムネイル・サポートが有効な場合は、アイテムの「添付ファイル」タブにファイルを追加すると、そのファイルに対するサムネイルが生成されます。

デザイン・オブジェクトの使用

デザイン・オブジェクトは、Agile PLM のファイル管理サーバーに格納されている 1 つ以上の URL またはファイルを指定するビジネス・オブジェクトです。デザイン・オブジェクトには、添付されているバイナリ・ファイルに関する情報が含まれています。他の Agile PLM のビジネス・オブジェクトと同様に、デザイン・オブジェクトは、Agile PLM のクラス階層に個別の基本クラスとして表示されます。

デザイン・クラス・オブジェクトは、Agile PLM で CAD データの管理に使用する Agile PLM の Engineering Collaboration (EC) モジュールで使用されます。このクラスで作成されたオブジェクトには、ファイル・フォルダと同じプロパティと動作が多数含まれています。管理者権限のあるユーザーは、デザイン・オブジェクトを Java クライアントで開いて使用できます。Agile PLM ユーザーは、このオブジェクトに Web クライアントでアクセスし、排他的に使用できます。

Agile SDK では、次のデザイン・オブジェクト関連機能がサポートされています。

- バージョン固有の 2 つのデザイン・オブジェクト間関係の管理（追加、削除、取得および編集）。
- Agile PLM クラス構造のデザイン・オブジェクト配置に対する使用箇所検索条件の使用。使用箇所検索条件の詳細は、63 ページの「[使用箇所検索条件の作成](#)」を参照してください。

デザイン・オブジェクトの追加およびロード

IDesign オブジェクトを作成または取得するには、IAgileSession.createObject() または IAgileSession.getObject() を使用します。次の例は、デザイン・オブジェクトを作成および取得するために SDK が提供する様々なメソッドを示しています。

例: クラス名によるデザインの作成

```
IDesign des = (IDesign)
    m_session.createObject("Design", "DESIGN00133");
```

例: クラス ID によるデザインの作成

```
IDesign des = (IDesign)
    m_session.createObject(FileFolderConstants.CLASS_DESIGN,
        "DESIGN00133");
```

例: IAgileClass 参照によるデザインの作成

```
IDesign des = (IDesign)
    m_session.createObject(desClass, "DESIGN00133");
```

例: デザイン・オブジェクトのロード

```
IDesign des = (IDesign)
    m_session.getObject(IDesign.OBJECT_TYPE, "DESIGN00133");
```

バージョン固有のデザイン・オブジェクト間関係の管理

Agile SDK では、デザイン・オブジェクト間に対して次のバージョン固有関係機能をサポートしています。

注意 次のバージョン固有機能が適用されるのは、デザイン・オブジェクトのみです。

- デザイン・オブジェクト間へのバージョン固有関係の追加
- デザイン・オブジェクト間のバージョン固有関係の削除
- 特定バージョンのデザイン・オブジェクトのバージョン固有関係の取得
- デザイン・オブジェクトのバージョン固有関係の編集

特定バージョンのデザイン・オブジェクト間への関係の追加

SDK では、2 つの特定バージョンのデザイン・オブジェクト間に関係を追加するために、次の API が用意されています。

```
IDesign.addVersionSpecificRelationship(Object versionNum, IDesign
relatedDesign, Object relatedVersionNum)
```

パラメータは、次のとおりです。

- **versionNum** - デザイン・オブジェクトのバージョン番号を示す整数。
- **relatedDesign** - 関係の作成対象のデザイン・オブジェクト。
- **relatedVersionNum** - 関係の作成対象のデザイン・オブジェクトのバージョン番号を示す整数。

2 つのデザイン・オブジェクト間のバージョン固有関係が作成されなかった場合は、`APIException` 例外が発生します。

あるいは、次の `params` によって、オブジェクトの `RelationshipsTable` をロードしたり、`createRow(Object params)` をコールすることができます。

```
HashMap params = new HashMap();
params.put(DesignConstants.ATT_RELATIONSHIPS_REV_VERSION, versionNum);
params.put(DesignConstants.ATT_RELATIONSHIPS_NAME, relatedDesign);
params.put(DesignConstants.ATT_DESIGN_VERSION, relatedVersionNum);
```

特定バージョンのデザイン・オブジェクト間の関係の削除

次に、`IDesign` のバージョン固有関係を削除する方法を示します。

```
IDesign des1 = (IDesign)session.getObject(IFileFolder.OBJECT_TYPE,
"DESIGN00001");
des1.setCurrentVersion(new Integer(4));
ITable relationshipTable = des1.getRelationship();
relationshipTable.removeRow(row);
```

特定バージョンのデザイン・オブジェクト間の関係の取得

次に、特定バージョンの `IDesign` の関係を取得する方法を示します。

```
IDesign des1 = (IDesign)session.getObject(IFileFolder.OBJECT_TYPE,
"DESIGN00001");
des1.setCurrentVersion(new Integer(4)); //set desired version
ITable relationshipTable = des1.getRelationship();
```

特定バージョンのデザイン・オブジェクト間の関係の編集

次に、特定バージョンの `IDesign` オブジェクト間の関係を編集する方法を示します。

```
IDesign des1 = (IDesign)session.getObject(IFileFolder.OBJECT_TYPE,
"DESIGN00001");
des1.setCurrentVersion(new Integer(4));
ITable relationshipTable = des1.getRelationship();
HashMap mapForUpdate=new HashMap();
HashMap rowUpdateMap = new HashMap();
rowUpdateMap.put(DesignConstants.ATT_DESIGN_VERSION, new Integer(1));
```

```
mapForUpdate.put(row1, rowUpdateMap);
relationshipTable.updateRows(mapForUpdate);
```

特定バージョンのデザイン・オブジェクトのパージ

SDK には、特定バージョンのデザイン・オブジェクトと関連バージョンのその子オブジェクトをパージするために、`IDesign.purgeVersions(Object[] versions)` API が用意されています。整数の値である `versions` パラメータを使用して、パージするバージョン番号を指定します。API がオブジェクトのパージに失敗した場合は、例外が発生します。

使用箇所検索条件を使用したデザイン・オブジェクト配置の検索

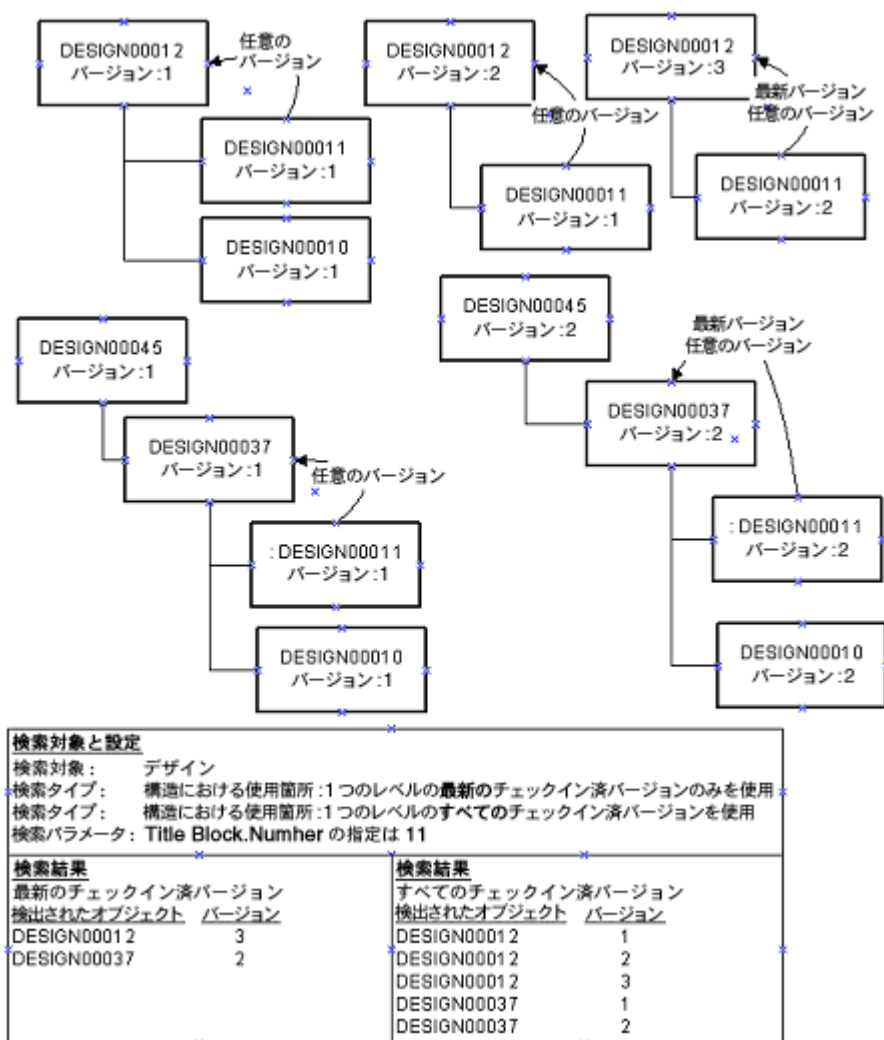
デザイン・オブジェクトの「構造」タブを使用すると、ユーザーは、個別のデザイン・オブジェクトが複数のバージョンを持つ構造を作成できます。SDK では、Agile PLM クラス構造でのデザイン・オブジェクトの使用について、次の検索条件と検索条件定数を使用した最新のチェックイン済バージョンとすべてのチェックイン済バージョンの検索をサポートしています。

- `WHERE_USED_IN_STRUCTURE_ONE_LEVEL_LATEST_CHECKEDIN` - この `WHERE_USED` 検索条件では、入力したデザイン・オブジェクトをデザイン構造で子として使用している直近の親であるデザイン・オブジェクトの最新バージョンが返されます。この検索は、定数 `QueryConstants.WHERE_USED_IN_STRUCTURE_ONE_LEVEL_LATEST_CHECKEDIN` によってサポートされています。
- `WHERE_USED_IN_STRUCTURE_ALL_LEVEL_LATEST_CHECKEDIN` - この `WHERE_USED` 検索条件では、入力したデザイン・オブジェクトをデザイン構造で子として使用している直近の親であるデザイン・オブジェクトのすべてのバージョンが返されます。この検索は、定数 `QueryConstants.WHERE_USED_IN_STRUCTURE_ONE_LEVEL_ALL_CHECKEDIN` によってサポートされています。

SDK_samples.zip フォルダの `QueryConstants` を使用したコード・サンプルは、Oracle® E-Delivery の Web サイト (<http://edelivery.oracle.com/>) で入手できます。このドキュメント・フォルダには、Javadoc で生成された HTML ファイルが含まれています。

次の図を使用して、2つの検索とそれぞれの結果について説明します。次の図は、デザイン・オブジェクトと1レベルの構造を示しています。検索パラメータ Title Block.Number には11が指定されています。

図15: デザイン・オブジェクトと検索結果



SDK を使用したデータのインポートとエクスポート

この章のトピック

- データのインポートとエクスポートについて 201
- インポート・データの検証とデータのインポート 201
- SDKからのデータのエクスポート 204

データのインポートとエクスポートについて

SDK を使用すると、外部データベースから PLM システムにデータをインポートおよびエクスポートできます。ソースは、Agile データベース、サード・パーティの Product Data Management (PDM) システムまたは Enterprise Resource Planning (ERP) システムのいずれかです。次に、バックグラウンド情報、手順、および Agile SDK を使用したこれらのタスクの実行例を示します。

インポート・データの検証とデータのインポート

データをインポートする際は、そのデータを検証するオプションがあります。検証しない場合は、このステップを無視します。インポート検証の目的は、データが適切なサーバー・ルール（長さの許容範囲、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・プロセスを開始する前に、インポートされないデータがわかります。

SDK では、次のインポート関連タスクをプログラムで実行するために、2つのメソッドを公開しています。

- インポートするデータをサーバーのビジネス・ルールへの準拠に関して検証する
`IImportManager.validateData(byte[], String, byte[], byte[], String[], List)` メソッド。このアクションは、入力ソース・データの無効なアイテムを識別するために、データをインポートする前に実行されます。
- データの PLM データベースへのインポートをサポートする `IImportManager.importData(byte[], String, byte[], byte[], String[], List)` メソッド。このアクションは、サーバーのビジネス・ルールに適合するデータを選択して PLM システムへのインポートを可能にするように、
`IImportManager.validateData()` メソッドの実行後に実行されます。

データのインポートの詳細は、『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。

次に、これらのメソッドを使用して、インポートするデータを準拠の点から検証し、検証後にそのデータを PLM システムにインポートする例を示します。

SDKを使用した検証の起動およびデータのインポート

例: データの検証と PLM へのデータのインポート

```
import com.agile.api.*;
import java.util.*;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

public class ImportClient {
    public static IAgileSession session = null;
    public static AgileSessionFactory factory;
    public static void main(String[] args) {
        try {
            String _url="http://localhost/Agile";
            String _user="admin";
            String _pwd="agile";
            String srcFilePath="bom.txt";

/* Supported file types: aXML,IPC2571, ExcelFile, DelimitedTextFile
 * The value of "-f" parameter is the same
 * in Import AIS sample command
 */

            String srcFileType="DelimitedTextFile";
            // Null implies loading the default mapping
            String mappingPath="NewMapFile.xml";
            // Null implies do not transform
            String transformPath=null;
            /* The value used by operations is the
             same as the value of the "-t" parameter in the import AIS sample
             command.
             all supported
             operations:"items.bom","items.aml","items.attachments",
             "items.relationships", "manufacturers",
             "manufacturers.relationships",
             "manufacturers.attachments", "manufacturerParts",
             "manufacturerParts.relationships",
             "manufacturerParts.attachments" "partgroups",
             "partgroups.relationships",
             "partgroups.attachments", "productServiceRequests"
             "productServiceRequests.affectedItems",
             "productServiceRequests.relatedPSR"
             "productServiceRequests.relationships"
             "productServiceRequests.attachments" "qualityChangeRequests"
             "qualityChangeRequests.affectedItems"
             "qualityChangeRequests.relationships"
             "qualityChangeRequests.attachments"
 */
            String [] operations=new String[]{"items", "items.bom","items.aml"};
            /* The value used by options is the
             * same as the "-n" parameter in the import AIS sample command
```



```

*/
    List options=new ArrayList();
    options.add("BusinessRuleOptions|ChangeMode=Authoring");
    options.add("BusinessRuleOptions|BehaviorUponNonExistingObjects=Accept");
    String _output="log.xml";
    FileOutputStream fop=new FileOutputStream(_output);
    // Create an instance of IAgileSession
    session = connect(_url, _user, _pwd);
    IImportManager imgr = (IImportManager)
        session.getManager(IImportManager.class);
    byte[] logData=null;
    /* Sample code to import data
    * Remove comments to run the importData example.
    * byte[] logData=imgr.importData(stream2byte
    * (new FileInputStream(srcFilePath)),
    * srcFileType, convertFiletoStream(mappingPath),
    * convertFiletoStream(transformPath),
    * operations, options);
    * Sample code to validate data
    * Remove comments to run the validateData example
    * logData=imgr.validateData(stream2byte
    * (new FileInputStream(srcFilePath)),
    * srcFileType, convertFiletoStream(mappingPath),
    * convertFiletoStream(transformPath), * operations, options);
    * byte buf[]=new byte[1024*4];

    int n=0;
    InputStream logStream=byte2stream(logData);
    while((n=logStream.read(buf))!=-1){
        fop.write(buf, 0, n);
    }
    fop.close();
    }
    catch (Exception e) {e.printStackTrace();
    }
    finally {session.close();
    }
}

/*
* <p> Create an IAgileSession instance </p>
*
* @return IAgileSession
* @throws APIException
*/
private static IAgileSession connect(String _url,String _user,String
    _pwd) throws APIException {
    factory = AgileSessionFactory.getInstance(_url);
    HashMap params = new HashMap();
    params.put(AgileSessionFactory.USERNAME, _user);
    params.put(AgileSessionFactory.PASSWORD, _pwd);
    session = factory.createSession(params);
    return session;
}

private static byte[] stream2byte(InputStream stream) throws
    IOException {

```

```
    ByteArrayOutputStream outputStream=new ByteArrayOutputStream();
    byte buf[]=new byte[1024*4];
    int n=0;
    while((n=stream.read(buf))!=-1){
        outputStream.write(buf, 0, n);
    }
    byte[] data=outputStream.toByteArray();
    outputStream.close();
    return data;
}

private static InputStream byte2stream(byte[] data) throws
IOException{
    ByteArrayInputStream stream=new ByteArrayInputStream(data);
    return stream;
}

private static byte[] convertFiletoStream(String path) throws
IOException{
    if(path==null || path.equals(""))
        return null;

    return stream2byte(new FileInputStream(path));
}
}
```

SDKからのデータのエクスポート

SDK では、データを PLM データベースからプログラムでエクスポートするために、`exportData()` メソッドを公開しています。このメソッドは、SDK プログラムへの大規模な BOM のロード時に発生するパフォーマンスとメモリーの問題を解決するために設計されています。この問題を解決するには、このエクスポート機能を起動して BOM をロードします。SDK プログラムでは、抽出した XML ファイルからデータを読み込んでエクスポートできます。

データのエクスポートの詳細は、『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。

SDKからのエクスポートの起動

SDK のエクスポート機能を起動するには、次のコールを使用します。

```
public byte[] exportData (Object[], Integer, String[])
```

次に、このコールの内容を説明します。

- `exportData` - エクスポート・データをバイト配列で返すメソッドです。このバイト配列は、aXML または PDX 形式のエクスポート XML ファイルが格納された ZIP ファイル、およびエクスポート・パッケージに含まれている添付ファイルを表します。
- `Object[]` - PLM から外部システムにエクスポートされるオブジェクトの配列です。これらのオブジェクトは、`IDataObject` オブジェクトとして渡されます。
- `Integer` - 出力するエクスポート形式が aXML か PDX かを識別するためのインジケータ (`ExportConstants.java` で提供される定数) です。SDK でサポートされている形式は、この 2 種類です。

- `String[]` - エクスポートで使用する ACS フィルタ名の配列です。フィルタ名は大文字と小文字を区別しませんが、ACS の管理ツールで定義されたフィルタ名と一致する必要があります。

次に、**exportData** メソッドで例外が発生する条件と個別の例外を示します。

- **Invalid Data Format** - メソッドがコールされましたが、エクスポート・データ形式で認識されない値が含まれています。有効な値は、aXML 値（定数ラベルを提供）と PDX 値（定数ラベルを提供）のみです。
- **No Filter Specified** - メソッドがコールされましたが、フィルタが指定されていません。有効なフィルタを少なくとも 1 つ指定する必要があります。
- **Specified Filter Not Found** - メソッドがコールされましたが、システムに指定のフィルタがありません。

例: SDK を使用した PLM からのデータのエクスポート

```
... //

IItem item = (IItem) session.getObject(IItem.OBJECT_TYPE, "P0001");
if (item == null) {
    ... // throw an error, the part wasn't found
}
IDataObject[] expObjs = {item};
String[] filters = {"Default Item Filter"};

...

IExportManager eMgr = (IExportManager)
session.getManager(IExportManager.class);
try {
    byte[] exportData = eMgr.exportData(expObjs,
ExportConstants.EXPORT_FORMAT_PDX, filters);
    if (exportData != null) {
        String fileName = createOutputFileName();
        FileOutputStream outputFile = new FileOutputStream(fileName);
        outputFile.write(exportData);
        outputFile.close();
        System.out.println("Data exported to file: " + fileName);
    }
} catch (Throwable t) {
    ... // error handling
}
...
```


ワークフローの管理

この章のトピック

■ ワークフローについて	207
■ ワークフローの選択	209
■ 承認者の追加および削除	210
■ 変更の承認または却下	219
■ 変更のコメント	220
■ 変更の検証	221
■ オブジェクトのワークフロー・ステータスの変更	222
■ 選択したユーザーへのAgileオブジェクトの送信	224
■ ユーザー・グループへのAgileオブジェクトの送信	225

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ルートします。
- 承認者およびオブザーバに、変更がルート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、ルーティング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、ルーティング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 > 検証済 > 発行済 > 改善済 > 検証済 > 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 > 終了
デフォルトの変更指示	保留中 > 提出済 > CCB > リリース済 > 実施
デフォルトの変更要求	保留中 > 提出済 > CCB > リリース済 > 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 > 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 > 実施
デフォルトの期限付き変更指示	保留中 > 提出済 > CCB > リリース済 > 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 > CCB > リリース済 > 検収済
デフォルトの不具合レポート	保留中 > 提出済 > レビュー > リリース済 > 終了
デフォルトのパッケージ	保留中 > 提出済 > レビュー > 受諾済 > 終了
デフォルトの価格変更	保留中 > 提出済 > 価格のレビュー > リリース済 > 実施
デフォルトの問題レポート	保留中 > 提出済 > レビュー > リリース済 > 終了
デフォルトの拠点毎変更	保留中 > 提出済 > CCB > リリース済 > 実施
デフォルトの出荷停止	保留中 > 提出済 > CCB > リリース済 > 再開

動的なワークフロー機能

特定のルーティング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、そのルーティング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、リリース済の変更に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_RELEASED));
}
```

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	<code>IRoutable.changeStatus()</code>
コメント	<code>IRoutable.comment()</code>
送信	<code>DataObject.send()</code>

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指示を作成する場合は、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、ルーティング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。ルーティング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当てにできます。変更を未割当てにするには、`IRoutable.setWorkflow()` メソッドを使用してワークフロー・パラメータに `null` を指定します。

例: 変更の未割当て

```
private void unassign(IChange change) throws APIException {
    change.setWorkflow(null);
}
```

承認者の追加および削除

変更がルートされ、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ある場合があります。承認者やオブザーバを追加または削除するユーザーには、ルート権限が必要です。

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（設計変更指示）などのルーティング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、およびオブジェクト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、Oracle® E-DeliveryのWebサイト (<http://edelivery.oracle.com/>) にあるAPIリファレンス・ファイル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 null 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、変更指示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。

ルーティング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、サーバーの「プリファレンス」設定に設定されている `secondSignature`、`escalations`、`transfers` または `signoffForSelf` パラメータの指定もできます。

```
// 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。

- 「サーバー設定」>「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:「デフォルトの変更指示」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」ステータスに追加します。

例: ルーティング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);
```

ルーティング可能なオブジェクトの却下

このメソッドは、ルーティング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、サーバーの「プリファレンス」設定に設定されている secondSignature、escalations、transfers または signoffForSelf パラメータの指定もできます。

```
// 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。

- 「サーバー設定」>「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:「デフォルトの変更指示」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」ステータスの承認者として追加します。

例: ルーティング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);

```

ルーティング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。

- 「サーバー設定」>「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:「デフォルトの変更指示」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[])null, (
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ルーティング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。

- 「サーバー設定」>「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:「デフォルトの変更指示」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);

```

ルーティング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ケーレションの追加

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ケーレ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「エスケーレ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 Workflowstatus.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。

- 「サーバー設定」>「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:「デフォルトの変更指示」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);

```

ルーティング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、ルーティング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 219 ページの「[ルーティング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](#)」、212 ページの「[ルーティング可能なオブジェクトの承認](#)」および 213 ページの「[ルーティング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。

例

```

/*
Parameters
    Status: The status (IStatus) of the object checked for the next
    Workflowstatus.
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（変更指示）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。

- 「サーバー設定」>「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:「デフォルトの変更指示」の「二重識別が必要」を「はい」に設定します。

例: ルーティング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 Req'd =
    "+ chg.isSecondSignatureRequired(st));
    System.out.println("IS Second Signature UserId = "
    + chg.isSecondSignatureUserId(st));

    secondSigReqd = chg.isSecondSignatureRequired(st);
}
```


ルーティング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、ルーティング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、217ページの「[ルーティング可能なオブジェクトを承認する 2 番目の署名の指定](#)」で説明されているメソッドを組み合わせで使用します。これらのメソッドには、secondSignature パラメータがあります。212ページの「[ルーティング可能なオブジェクトの承認](#)」および 213ページの「[ルーティング可能なオブジェクトの却下](#)」を参照してください。

```
/*
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（変更指示）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。

- 「サーバー設定」>「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:「デフォルトの変更指示」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ルートされると、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ユーザーには、作成および管理権限または依頼および承認権限が必要です。詳細は、『Agile PLM 管理者ガイド』を参照してください。

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、Oracle® E-Delivery Web サイト (<http://edelivery.oracle.com/>) にある API リファレンス・ファイルを参照してください。

次に、特定のルーティング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、212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);
    }
}
```

変更根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せん。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、ユーザー通知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、Oracle® E-Delivery Web サイト (<http://edelivery.oracle.com/>) にある API リファレンス・ファイル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、Oracle® E-Delivery の Web サイト (<http://edelivery.oracle.com/>) にある API リファレンス・ファイル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。変更に関係がない場合、ICell キーは null 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、null オブジェクトがキーとして含まれる場合があります。null 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_CAUSE
S)){
                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、Oracle® E-Delivery Web サイト (<http://edelivery.oracle.com/for>) にある API リファレンス・ファイル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、`null` 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。`null` 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、ルーティング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() メソッドはルーティング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 = new IUser[userList.size()];
    System.arraycopy(userList.toArray(), 0, users, 0, userList.size());
    // 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() メソッドはルーティング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);
    }
}

```


品質の管理および追跡

この章のトピック

■ 品質管理について	227
■ 顧客の使用	228
■ 製品サービス依頼の使用	230
■ 品質変更要求の使用	233
■ PSRおよびQCRでのワークフロー機能の使用	235

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、次の新規インタフェースが含まれています。

- ICustomer - Customer クラスのインタフェース。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 クラスのインタフェース。IServiceRequest は IRoutable のサブインタフェースです。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要求に類似した QualityChangeRequest クラスのインタフェース。このインタフェースは、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。

このオブジェクトは、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ルートできます。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ルートできます。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。指定する必要が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・フィールドで利用可能な値は、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、複数のアイテムに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_CHANG
E);
        IAgileList listvalues = attr.getAvailableValues();
        listvalues.setSelection(new Object[] { new Integer(0)});
        params.put(ServiceRequestConstants.ATT_AFFECTED_ITEMS_LATEST_CHANGE,
listvalues);
        //Set the Affected Site value
        attr =
cls.getAttribute(ServiceRequestConstants.ATT_AFFECTED_ITEMS_AFFECTED_SIT
E);
        IAgileList listvalues = attr.getAvailableValues();
        listvalues.setSelection((new Object[] { "Hong Kong" }));
        params.put(ServiceRequestConstants.ATT_AFFECTED_ITEMS_AFFECTED_SITE,
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;
    }
}

```

品質変更要求の使用

品質分析者は、品質変更要求（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ルートでき、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要求があります。

- **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。

品質変更要求の作成

QCR を作成するには、IAgileSession.createObject() メソッドを使用します。指定する必要が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要求の割当て

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);
}
```

品質変更要求を変更として保存

`IDataObject.saveAs()` メソッドを使用すると、QCR を別の QCR または ECO（つまり、変更指示の別のタイプ）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。ワークフローは、QCR で `saveAs()` を使用するための必須の入力パラメータです。

注意 QCR を品質変更要求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(IAgileSession session,
    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();
    // Get Workflow for the ECO class
    IWorkflow ecoWf =
        ((IRoutableDesc)session.getAdminInstance().getAgileClass(ChangeConstants.CLASS_ECO)).getWorkflows()[0];
    // Save the QCR as an ECO
    HashMap map = new HashMap();
```



```

        map.put(ChangeConstants.ATT_COVER_PAGE_NUMBER, numbers[0]);
        map.put(ChangeConstants.ATT_COVER_PAGE_WORKFLOW, ecoWf);
        IChange eco = (IChange)qcr.saveAs(ChangeConstants.CLASS_ECO, map);
        // 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()

PSRおよびQCRの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要求を作成する場合は、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要求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要求にワークフローの設定もできます。

例: 「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);
}
```

プロジェクトの作成および管理

この章のトピック

■ プロジェクトおよびプロジェクト・オブジェクトについて	237
■ プロジェクト・オブジェクトの動作の相違点	238
■ プロジェクトの作成	238
■ PPMオブジェクトに対するルールへの追加	240
■ プロジェクトのロード	241
■ プロジェクト・テンプレートの使用	242
■ プロジェクトのスケジュール	245
■ プロジェクトの基準の使用	247
■ 別のユーザーへのプロジェクト所有権の委譲	248
■ プロジェクトのチームへのリソースの追加	249
■ プロジェクト・リソースの入れ替え	251
■ プロジェクトのロックまたはロック解除	252
■ ディスカッションの使用	252

注意 リリース 9.3 では、プログラム基本クラスの名前がプロジェクトに変更され、プロジェクトがソーシング・プロジェクトに変更されました。ただし、SDK 開発者ガイドおよび Javadoc リファレンスでは、プロジェクト基本クラスのインタフェースは、今までどおり IProgram と呼ばれています。

プロジェクトおよびプロジェクト・オブジェクトについて

Agile Product Portfolio Management (PPM) のプロジェクト管理機能を使用すると、プロジェクト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当て、ボトルネックの識別、割当て超過および割当て不足のリソース状況への対応が可能です。また、プロジェクト・テンプレートを作成し、再利用することもできます。

プロジェクトをスケジュールして実行するには、プロジェクト・オブジェクトを使用します。各プロジェクトには、スケジュール情報の他に、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、プロジェクトの内容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、プロジェクトの内容の作成、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、プロジェクトの内容であるすべてのオブジェクトを表します。

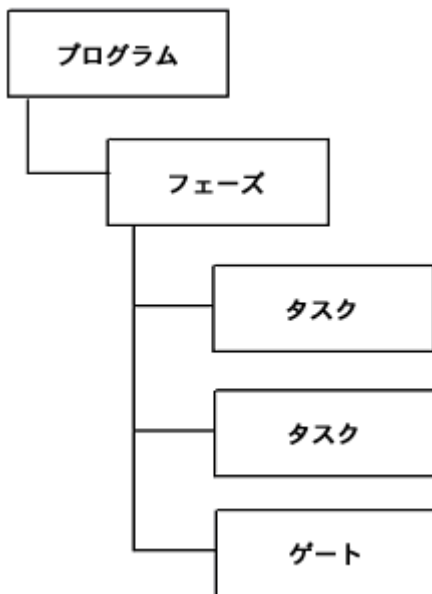
他の Agile PLM ビジネス・オブジェクトと同様に、IProgram インタフェースには IRouteable が実装されています。したがって、プロジェクトの内容のワークフロー・ステータスの変更、および他のユーザーへのルートには、同じ IRouteable.changeStatus() メソッドが使用されます。詳細は、222 ページの「[オブジェクトのワークフロー・ステータスの変更](#)」を参照してください。

プロジェクト・オブジェクトの動作の相違点

IProgram インタフェースには、他の Agile PLM オブジェクトで一般的に使用されるインタフェースがいくつか実装されています。ただし、プロジェクト・オブジェクトを他のオブジェクトと区別する次の固有の機能も提供されます。

- プロジェクト・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ジェクト・オブジェクトのコンテナです。基礎となるプロジェクト・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（通常はプロジェクト）と関連付けられます。
- プロジェクトには、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。
- プロジェクトはアーカイブ可能です。ルート・プロジェクトをアーカイブすると、プロジェクト・ツリー全体がシステムからソフト削除されます。
- プロジェクトは、ロックまたはロック解除可能です。

プロジェクトの作成



プロジェクトを作成するには、IAgileSession.createObject() メソッドを使用します。プロジェクト・パラメータを指定するときは、プロジェクトのサブクラス（例: プロジェクト、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。プロジェクト、フェーズおよびタスクについては、次の必須プロジェクト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、新規プロジェクトを作成し、必須属性を指定する方法を示しています。

例: プロジェクトの作成

```
try {
    // Create a Map object to store parameters
    Map params = new HashMap();
    // Set Projects name
    String name = "APOLLO PROJECTS";
    // Set Projects start date
    Date start = new Date();
    start.setTime(1);

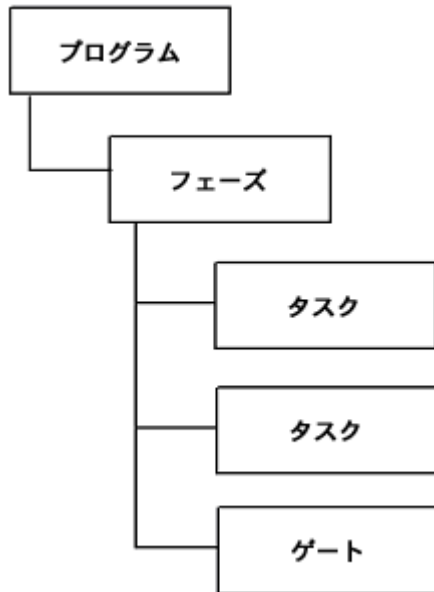
    // Set Projects end date
    Date end = new Date();
    end.setTime(1 + 2*24*60*60*1000);

    // Set Projects 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 Projects
    IProgram program =
(IProgram)m_session.createObject(ProgramConstants.CLASS_PROGRAM,
params);
} catch (APIException ex) {
    System.out.println(ex);
}
```

プロジェクトには、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、タスクまたはプロジェクト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、プロジェクト・オブジェクトの階層を示しています。

図16: 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、他のプロジェクト・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、プロジェクト・オブジェクトの「スケジュール」テーブルに追加できます。詳細は、245ページの「[プロジェクトのスケジュール](#)」を参照してください。

PPMオブジェクトに対するルールの追加

PLM では、ライフサイクル・フェーズがあるオブジェクトまたはワークフローが割り当てられているオブジェクトはすべて成果物として追加できます。ディスカッション、ユーザーおよびユーザー・グループのみ例外です。

PPM のルールでは、ワークフローまたはライフサイクル・フェーズに設定されているとおりに、前のアクティビティが完了してから次のアクティビティが完了することが保証されます。たとえば、アクティビティが完了してからゲートを開く場合は、そのアクティビティを成果物として開くゲートに追加できます。前のゲートを成果物として後続の開くゲートに追加することで、あるゲートが別のゲートの前に開くことを制限することもできます。詳細は、『Product Portfolio Management ユーザー・ガイド』を参照してください。

次の例に示すように、SDK では、この機能を `IProgram` インタフェースを使用してサポートしています。

例: PPM オブジェクトに対するルールの設定

```
try{
    //Get Program
    IProgram pgm =
        (IProgram)session.getObject
        (ProgramConstants.CLASS_PROGRAM, "PGM00239");
    //Get Object and add as relationship under Content tab
    IChange eco = (IChange)session.getObject
```

```

(ChangeConstants.CLASS_ECO, "C00060");
    ITable table = pgm.getTable
(ProgramConstants.TABLE_RELATIONSHIPS);
    IRow row = table.createRow(eco);
    //Get the Control object status
    IStateful state = (IStateful)pgm;
    IStatus[] statuses = state.getStates();
    IStatus ctl_status = null;
    for(int i=0; i<statuses.length; i++){
        if(statuses[i].getName().equals("In Process")){
            ctl_status = statuses[i];
            break;
        }
    }
    //Get the Affected object status
    state = (IStateful)eco;
    statuses = state.getStates();
    IStatus aff_status = null;
    for(int i=0; i<statuses.length; i++){
        if(statuses[i].getName().equals("Submitted")){
            aff_status = statuses[i];
            break;
        }
    }

    //Add Rule
    HashMap map = new HashMap();
    map.put(CommonConstants.ATT_RELATIONSHIPS_RULE_CONTROLOBJECT, pgm);
    map.put(CommonConstants.ATT_RELATIONSHIPS_RULE_AFFECTEDOBJECT, eco);
    map.put(CommonConstants.ATT_RELATIONSHIPS_RULE_CONTROLOBJECTSTATUS,
    ctl_status);
    map.put(CommonConstants.ATT_RELATIONSHIPS_RULE_AFFECTEDOBJECTSTATUS,
    aff_status);
    row.setValue(CommonConstants.ATT_RELATIONSHIPS_RULE, map);
    System.out.println(row.getCell
(CommonConstants.ATT_RELATIONSHIPS_RULE));
} catch (APIException ex) {
    System.out.println(ex);
}
}

```

プロジェクトのロード

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。プロジェクト・オブジェクトを一意に識別するためには、General Info.Number 属性に値を指定します。また、プロジェクト・オブジェクトを名前で検索して、その検索結果から選択する方法でプロジェクト・オブジェクト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: プロジェクトのロード

```

public IProgram loadProgram(String number) throws APIException {
    IProgram program =
    (IProgram)m_session.getObject(IProgram.OBJECT_TYPE, number);
    return program;
}

```

注意 プロジェクトに対する「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。

プロジェクト・テンプレートの使用

プロジェクト・テンプレートを使用すると、新しいプロジェクト・オブジェクト、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ジェクトです。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、新規プロジェクトを作成できます。

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ジェクトを作成し、コピーするテーブルを指定できます。
- プロジェクトの所有者および子の所有者を変更できます。
- プロジェクトをテンプレートとして保存して、新規プロジェクト・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ジェクトの作成

この特別なバージョンの saveAs() メソッドを使用すると、元のプロジェクトから新規プロジェクトにコピーするプロジェクト・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、「ページ 2」、「ページ 3」（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ジェクトの作成

```
try {
    // Get the Projects 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);
}
}

```

プロジェクト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ジェクトを作成する場合は、プロジェクトの所有権を変更し、その変更をプロジェクト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、プロジェクト所有権を変更するためには必須です。`OWNER` 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ジェクトを作成するとき、プロジェクトの所有権を変更し、その変更を子に適用できます。この場合、SDK では、UI 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ジェクトを作成するには、元のプロジェクト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、プロジェクトはテンプレートです。

例: テンプレートからのプロジェクトの作成、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 ;
}
```

プロジェクトをテンプレートとして保存

プロジェクトを作成するとき、テンプレート属性 (ProgramConstants.ATT_GENERAL_INFO_TEMPLATE) の値を「Template」に設定すると、プロジェクト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、プロジェクトを作成するとき、または新規プロジェクトとして保存するときのみです。既存のプロジェクトを、「Active」または「Proposed」の状態から「Template」には変更できません。次の例は、プロジェクト・オブジェクトを開いて、テンプレートとして保存する方法を示しています。

例: プロジェクト・オブジェクト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(P
rogramConstants.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);
}
}

```

プロジェクトのスケジュール

プロジェクト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 Projects
    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);
}
}

```

プロジェクト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ジェクトを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 - スケジュールされている開始日を指定した日付に移動します。
- ENDDATE - スケジュールされている終了日を指定した日付に移動します。
- BACKWARD_DAYS - 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。

- FORWARDDDAYS - 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: プロジェクトの再スケジュール

```
try {
    // Get a Projects
    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.BACKWARDDDAYS, new Integer(3));
        // Move the schedule forward two days
        program.reschedule(IProgram.RESCHEDULE.FORWARDDDAYS, new
        Integer(2));
    }
} catch (Exception ex) {
    System.out.println(ex);
}
```

プロジェクトの基準の使用

プロジェクト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、プロジェクト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・プロジェクト・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 Projects
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ジェクト所有権の委譲

プロジェクト・オブジェクトの所有者またはマネージャは、プロジェクト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、受諾または拒否できる依頼を受け取ります。受諾すると、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、委譲されたプロジェクト・オブジェクトに対する「プロジェクト・マネージャ」役割が自動的に付与されます。

プロジェクトの所有権を委譲するには、`IProgram.delegateOwnership()` メソッドを使用します。プロジェクト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ジェクトの子にも適用するかどうかを指定できます。

例: プロジェクト・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);
}
}

```

プロジェクト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、プロジェクト・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・メンバーの割当ての変更を行うことができます。プロジェクトの「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、そのプロジェクト・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、Projects 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次の 2 つの属性があります。

- ProgramConstants.ATT_TEAM_NAME
- ProgramConstants.ATT_TEAM_ROLES.

これらはそれぞれ、シングルの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利用可能な値を取得するには、IAttribute.getAvailableValues()ではなく、ITable.getAvailableValues()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: プロジェクト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,
"albert1");
        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 Projects
        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", "Projects 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[]{"Projects
Administrator"});
            map.put(ProgramConstants.ATT_TEAM_NAME, user2);
            IRow row2 = teamTable.createRow(map);

            // Add the resource pool to the team
            attrRolesValues.setSelection(new Object[]{"Projects 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、プロジェクトの「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;
                }
            }
        }
    }
}
```

プロジェクト・リソースの入れ替え

リソースの可用性は、過負荷、再割当て、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ジェクトに対してのみです。プロジェクト・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、リソース割当てをプロジェクトの子に適用するかどうかも指定できます。

例: プロジェクト・リソースの入れ替え

```
try {
    // Get a Projects
    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);
    // Substitute u4 with a user group, and apply to children
    program.substituteResource(u4, ug, true);
}
} catch (APIException ex) {
    System.out.println(ex);
}
```

プロジェクトのロックまたはロック解除

プロジェクトの所有者は、プロジェクト・オブジェクトをロックまたはロック解除できます。プロジェクト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。プロジェクト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、プロジェクトは自動的にロックされます。

例: プロジェクトのロック

```
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、プロジェクトとは異なりルーティング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。「ページ 2」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**

また、「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、プロジェクトの「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_MESSAGE)
;
    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_NOTIFY
_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 Projects
    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 Projects
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_MESS
                AGE);
        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17: 展開された「返信」テーブル

Subject	Creator
☐ Stress testing	Dassin, Jules (jdassin)
RE: Stress testing	Hitchcock, Alfred (ahitchcock)
☐ RE: Stress testing	Reed, Carol (creed)
☐ RE: Stress testing	Huston, John (jhuston)
RE: Stress testing	Dassin, Jules (jdassin)

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、プロジェクトの「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、そのプロジェクトのチーム・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、返信は表示できません。ただし、プロジェクトの「チーム」テーブルにリストされているユーザーは、そのプロジェクト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 Projects
    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、プロジェクト・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 Projects
    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
        Table table = program.getTable(ProgramConstants.TABLE_ACTIONITEMS);
        // Add the new Action Item to table
        table.createRow(map);
    }
} catch (APIException ex) {
    System.out.println(ex);
}
```


Product Cost Management の使用

この章のトピック

■ 概要.....	259
■ 価格オブジェクトの使用.....	260
■ 価格の管理.....	260
■ サプライヤの使用.....	265
■ ソーシング・プロジェクトの使用.....	266

概要

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され、強化および簡素化されます。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）
- ソーシング・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 クラスのインタフェース。価格変更（PCO）が含まれます。
- IPrice - Price クラスのインタフェース。公表価格と履歴価格の両方が処理されます。
- IProject - Sourcing Project クラスのインタフェース。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

価格オブジェクトの使用

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** - 以前のソーシング・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る見積履歴オブジェクト。
- **公表価格** - 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る公表価格履歴。

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** - 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る見積履歴オブジェクト。

□ **公表価格** - 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る公表価格。

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）、プロジェクト、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」、「ページ 2」、「ページ 3」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 `Program==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(PricingConstants.CLASS_PUBLISHED_PRICE);
    IAutoNumber an =
        cls.getAutoNumberSources()[0];
    params.put(PricingConstants.ATT_GENERAL_INFORMATION_NUMBER, an);
    params.put(PricingConstants.ATT_GENERAL_INFORMATION_CUSTOMER,
customer);
    params.put(PricingConstants.ATT_GENERAL_INFORMATION_ITEM_NUMBER,
"1000-02");
    params.put(PricingConstants.ATT_GENERAL_INFORMATION_ITEM_REV,
"CO-35884");
    params.put(PricingConstants.ATT_GENERAL_INFORMATION_PROGRAM,
"PROGRAM0023");
    params.put(PricingConstants.ATT_GENERAL_INFORMATION_MANUFACTURING_SITE,
"San Jose");
    params.put(PricingConstants.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。これらのファイルは、`SDK_samples` (ZIPファイル) のHTMLフォルダにあります。このファイルにアクセスするには、2ページの「[クライアント側のコンポーネント](#)」の注意を参照してください。

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_DATE,
    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_TIME);
    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: 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
- 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: 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」のすべての読取り/書込みフィールドを読取りおよび更新できます。「一般情報」、「ページ 1」および「ページ 3」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://www.globalpartscorp.com");
    //Update Corporate Currency (List field)
```

```
        cell =
supplier.getCell(SupplierConstants.ATT_GENERAL_INFO_CORPORATE_CURRENCY
);
        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。ソーシング・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。複数のユーザーがソーシング・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）、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ソーシング・プロジェクトの作成
- 価格期間の指定によるソーシング・プロジェクトの作成
- ソーシング・プロジェクトの開閉
- アイテムの追加（ソーシング・プロジェクト・アイテムへの AML の追加を含む）
- ソーシング・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
- ソーシング・プロジェクト・ステータスへのアクセスおよび変更
- ソーシング・プロジェクト AML の更新
- ソーシング・プロジェクト内の「ページ 1」、「ページ 2」および「ページ 3」の更新

- ソーシング・プロジェクト内のネストされた「価格」テーブルの読取りおよび更新
- 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ソーシング・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
- ソーシング・プロジェクト・テーブルのフィルタリング
- ソーシング・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

SDKでは、ソーシング・プロジェクトに対して次のAPIメソッドをサポートしています。これらのインタフェースの詳細は、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。ファイルはSDK_samples (ZIPファイル) のHTMLフォルダにあります。このファイルにアクセスするには、2ページの「[クライアント側のコンポーネント](#)」の注意を参照してください。

- `IAgileSession.createObject(Object, Object)`
- `IAgileSession.createObject(int, Object)`
- `IAgileSession.getObject(Object, Object)`
- `IAgileSession.getObject(int, Object)`
- `IProject.assignSupplier (Object partnerParams)`
- `IProject.Costrollup()`
- `IProject.lookupPrices()`
- `IProject.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。

既存のソーシング・プロジェクトのロード

既存のソーシング・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には、「カバー・ページ | 番号」属性に値を指定します。

例: ソーシング・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ソーシング・プロジェクトの作成

ソーシング・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: ソーシング・プロジェクトの作成

```
IAgileObject createObject (Object objectType, Object params)
    throws APIException;
```

ソーシング・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。

- ソーシング・プロジェクト番号と数量割引

または

- ソーシング・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ソーシング・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ソーシング・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_BREAKS,
new Integer(4));
IProject prj = (IProject) m_session.createObject(agClass, map);

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ソーシング・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）を指定してソーシング・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ソーシング・プロジェクトを作成しています。

注意 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、ソーシング・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ソーシング・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_BREAKS,
    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、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」）、「ページ2」、「ページ3」、「アイテム」、「AML」、「分析」およびネストされた「価格」テーブルの読取り
- 「ページ1」（「カバー・ページ」）、「ページ2」、「ページ3」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。

- ソーシング・プロジェクト・クラス
 - デフォルトの並べ替え順がある PCM 固有のテーブルの並べ替え。該当するテーブルは、「プロジェクト・アイテム」、「プロジェクト AML」、「プロジェクト変更」、「プロジェクト分析」、「プロジェクトの見積依頼」、「見積依頼回答」、「見積依頼変更」、「サプライヤ回答」および「サプライヤ変更」です。
- 見積依頼クラスおよび見積依頼回答クラス
 - 「回答」テーブルおよび「変更」テーブル
- 「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）

ソーシング・プロジェクトの「カバー・ページ」の値の設定

ソーシング・プロジェクトのすべての読取り/書込みセルを読取りおよび更新できます。次の例では、ソーシング・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: ソーシング・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_DESCRIPTION);
    cell.setValue("Sourcing project for Odyssey III");
    //Update Manufacturing Site (List field)
cell =
    project.getCell(ProjectConstants.ATT_GENERAL_INFORMATION_MANUFACTURING
        _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_LOCAT
        ION);
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。

ソーシング・プロジェクトの親テーブルとネスト子テーブルの定数

親ソーシング・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、「親ソーシング・プロジェクト・テーブルと対応するネスト・ソーシング・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	読取り/書き込みモード
TABLE_ITEMS	ATT_ITEMS_AML	読取り/書き込み
TABLE_ITEMS	ATT_ITEMS_PRICING	読取り/書き込み
TABLE_AML	ATT_AML_PRICETABLE	読取り/書き込み
TABLE_ITEM	ATT_ITEM_PRICE_TABLE	読取り/書き込み
TABLE_ITEM	ATT_ITEM_BOM_TABLE	読取り
TABLE_ANALYSIS	ATT_ANALYSIS_AML	読取り
TABLE_ANALYSIS	ATT_ANALYSIS_PRICING	読取り

ソーシング・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と変更

次の例は、ネスト・テーブルにアクセスする読取りの例です。ネスト・テーブルを変更/更新するには、286ページの「[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](#)」に記載されている「ネストされた「見積依頼」テーブルの更新」の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、パ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);
```

ソーシング・プロジェクトのステータスへのアクセスと変更

ソーシング・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、ソーシング・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」へのソーシング・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。

ソーシング・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。ソーシング・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、ソーシング・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。ソーシング・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: ソーシング・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_LOCATION);
    ship2List.setSelection(new Object[] {sj});
    prj.setValue(ProjectConstants.ATT_GENERAL_INFORMATION_SHIP_TO_LOCATION,
    ship2List);
    // open project //
    prj.changeStatusToOpen();

    // close project //
    prj.changeStatusToClose();

```

ソーシング・プロジェクトでのデータの管理

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。

注意 指定したソーシング・プロジェクト開始日は、PLM データベースでのストレージ用に GMT フォーマットに変換されます。この変換のため、
`IProject.getValue(ProjectConstants.ATT_GENERAL_INFORMATION_PERIOD_START_DATE)` から戻るデータ値は、ユーザーの想定値と同じになる保証はありません。

ソーシング・プロジェクトのアイテムの数量の設定

SDK を使用すると、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_PROJECT, "PRJ
00001");
prj.rollupQuantity();
```

注意 更新された数量の値を取得するには、265 ページの「[サブライヤ・データの変更](#)」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、数量を設定した後、getValue() が対象アイテムの更新された値を返さないために必要な操作です。

ソーシング・プロジェクトでのコスト・ロールアップの実行

コスト・ロールアップ（ロールアップ・コスト）では、使用可能な価格に基づいてアセンブリ・コスト・レポート（ACR）が生成されます。このプロセスでは、フィルタリングされたデータから最小コストを選択し、「最良に設定」（ユーザー定義またはデフォルトのパラメータで）およびコスト付き BOM のロールアップ（集約）を実行して ACR を生成します。UI でのロールアップ・コストでは、非 PCM ユーザーに直感的なメカニズムを提供し、PCM 手順を実行せずに BOM のコストを計算します。

注意 コスト・ロールアップは既存のソーシング・プロジェクト価格で実行されます。検索した価格に対してコスト・ロールアップを実行する必要がある場合は、costRollup() を実行する前に lookupPrices() を起動する必要があります。ソーシング・プロジェクトにアセンブリがない場合は、ExceptionConstants.PCM_NO_ASSEMBLY_IN_PROJECT が発生します。

PCM SDK では、次の API によるコスト・ロールアップ機能をサポートしています。

```
public void costRollup()
throws APIException, RemoteException, Exception;
```


例: Using the costRollup API

```

IProject prj = (IProject)
m_session.getObject(ProjectConstants.CLASS_SOURCING_PROJECT,
"PRJ0001");
prj.costRollup();

```

注意 コスト・ロールアップの直後に数量ロールアップを実行する必要がある場合は、コスト・ロールアップの結果がデータベース内で更新できるように、若干の遅延（例: `Thread.currentThread().sleep(10000);`）を確保してください。

ソーシング・プロジェクトでの価格検索の実行

SDK を使用すると、指定の期間と数量に関する価格算出ケースがアイテム・マスターに存在していることを確認できます。アイテムの価格情報を使用することも、その価格情報を変更して再見積のためにサプライヤに見積依頼を送信することもできます。

Agile PCM には、3 タイプの価格オブジェクトがあります。

- **契約** - 指定期間のアイテム価格に関するサプライヤとの事前定義の契約。
- **公表価格** - 他のソーシング・プロジェクトから公表されたアイテム価格情報。
- **見積履歴** - アイテムについて以前に受け取った見積価格。

ソーシング・プロジェクトの価格オブジェクトおよび価格検索の詳細は、『Product Cost Management ユーザー・ガイド』を参照してください。

価格検索APIおよび価格検索オプション

- サポートされている API

SDK では、次の IProject の API によって価格検索をサポートしています。

```

public void lookupPrices(Object lookupParams)
throws APIException, RemoteException, Exception;

```

- 価格検索オプション

この API は、別のソーシング・プロジェクトの価格履歴および価格検索から、価格検索を実行します。

注意 `lookupPrices()` は、アイテムまたは MPN をオブジェクトごとに検索します。複数のアイテム/MPN を検索するには、API をアイテムまたは MPN ごとに実行する必要があります。

次の例は、ソーシング・プロジェクト履歴および別のソーシング・プロジェクトからの価格検索を示しています。さらに、適用可能なパラメータを、価格検索タイプに固有なパラメータと価格検索タイプに必須のパラメータに分類して示しています。

履歴または別のソーシング・プロジェクトからの価格検索のパラメータ

この例は、ソーシング・プロジェクト履歴および別のソーシング・プロジェクトからの価格検索を示しています。この例には、2 つの価格検索の固有のパラメータと必須パラメータのリストが記載されています。

例: 履歴および別のソーシング・プロジェクトからの価格検索

```

ArrayList priceTypes = new ArrayList();
priceTypes.add(PriceConstants.CLASS_PUBLISHED_PRICE);

```

```
priceTypes.add(PriceConstants.CLASS_QUOTE_HISTORY);
priceTypes.add(PriceConstants.CLASS_CONTRACT);

ArrayList suppliers = new ArrayList();
suppliers.add(supplier1);
suppliers.add(supplier2);
//supplier1, supplier2 are objects of ISupplier or String

ArrayList customers = new ArrayList();
customers.add(customer1);
customers.add(customer2);
//customer1, customer2 are objects of ICustomer or String

ArrayList programs = new ArrayList();
programs.add(program1);
programs.add(program2);
//program1, program2 are objects of String

String shipTo = "berlin";
HashMap itemMap = new HashMap();
itemMap.put("IPN1", "REV1");//itemMap.put("IPN1", null) if no revision
または
itemMap.put(item); //item is an object of IItem

HashMap mpnMap = new HashMap();
mpnMap.put("MPN1", "MFR1");
または
mpnMap.put(mfrPart); //mfrPart is and object of IManufacturerPart

IProject srcPrj = (IProject)
    m_session.getObject(ProjectConstants.CLASS_SOURCING_PROJECT,
        "PRJ_SRC");
Boolean isLookupFromPrice = new Boolean(false);
String priceScenario = null;
Map priceScenarios = new HashMap();
//if lookup from price history
priceScenario = "QuantityBreak1";
//if lookup from project
String destPricePoint1 = "QuantityBreak1";
String destPricePoint2 = "QuantityBreak2";
String srcPricePoint1 = "QuantityBreak1";
String srcPricePoint2 = "QuantityBreak2";
priceScenarios.put(destPricePoint2, srcPricePoint1);
priceScenarios.put(destPricePoint1, srcPricePoint2);

Boolean ignoreQtyRange = new Boolean(true);
```

```

Double qtyPercentRange = new Double(15);

Boolean ignoreDateRange = new Boolean(true);
Integer dateRange = new Integer(20);

HashMap map = new HashMap();
map.put(PricingConstants.ATT_GENERAL_INFORMATION_PRICE_TYPE,priceTypes);
map.put(PricingConstants.ATT_ANALYSIS_SUPPLIER,suppliers);
map.put(PricingConstants.ATT_GENERAL_INFORMATION_CUSTOMER,customers);
map.put(PricingConstants.ATT_GENERAL_INFORMATION_PROGRAM,programs);
map.put(PricingConstants.ATT_GENERAL_INFORMATION_SHIP_TO_LOCATION,shipTo
);
map.put(PricingConstants.ATT_ITEMS_NUMBER,itemMap);
map.put(PricingConstants.ATT_ITEMS_AML,mpnMap);
map.put(PricingConstants.ATT_GENERAL_INFORMATION_NUMBER,srcPrj);
map.put(PricingConstants.FLAG_IGNORE_ITEM_REVISION,ignoreItemRev);
map.put(PricingConstants.FLAG_CONSIDER_BEST_PRICES,considerBestPrices);
map.put(PricingConstants.FLAG_ALL_PRICE_SCENARIOS,allPriceScenarios);
map.put(PricingConstants.FIELD_PRICE_SCENARIO,priceScenario);
map.put(PricingConstants.FIELD_PRICE_SCENARIOS,priceScenarios);
map.put(PricingConstants.FLAG_IGNORE_QUANTITY,ignoreQtyRange);
map.put(PricingConstants.FIELD_QUANTITY_RANGE,qtyPercentRange);
map.put(PricingConstants.FLAG_IGNORE_DATE_RANGE,ignoreDateRange);
map.put(PricingConstants.FIELD_DATE_RANGE,dateRange);
map.put(PricingConstants.FIELD_SELECT_RESPONSE_BY,
    PricingConstants.OPTION_LOWEST_PRICE);
map.put(PricingConstants.FIELD_LOOKUP_TYPE,
    PricingConstants.OPTION_LOOKUP_FROM_PRICE);
prj.lookupPrices(map);

```

価格履歴からの価格検索に固有のパラメータ

```

PricingConstants.ATT_GENERAL_INFORMATION_PRICE_TYPE
PricingConstants.ATT_GENERAL_INFORMATION_CUSTOMER
PricingConstants.ATT_GENERAL_INFORMATION_PROGRAM
PricingConstants.ATT_GENERAL_INFORMATION_SHIP_TO_LOCATION
PricingConstants.FLAG_ALL_PRICE_SCENARIOS
PricingConstants.FIELD_PRICE_SCENARIO
PricingConstants.FLAG_IGNORE_QUANTITY
PricingConstants.FIELD_QUANTITY_RANGE
PricingConstants.FLAG_IGNORE_DATE_RANGE
PricingConstants.FIELD_DATE_RANGE
PricingConstants.FIELD_SELECT_RESPONSE_BY

```

ソーシング・プロジェクトからの価格検索に固有のパラメータ

```

PricingConstants.ATT_GENERAL_INFORMATION_NUMBER
PricingConstants.FLAG_ALL_PRICE_SCENARIOS
PricingConstants.FLAG_IGNORE_ITEM_REVISION
PricingConstants.FLAG_CONSIDER_BEST_PRICES

```

注意 残りのパラメータは両方の場合に共通です。

価格履歴からの価格検索に必須のパラメータ

```
PriceConstants.ATT_GENERAL_INFORMATION_PRICE_TYPE  
ProjectConstants.ATT_ITEMS_NUMBER or ProjectConstants.ATT_ITEMS_AML  
LookupConstants.FIELD_QUANTITY_RANGE if  
LookupConstants.FLAG_IGNORE_QUANTITY is 'false'  
LookupConstants.FIELD_DATE_RANGE if  
LookupConstants.FLAG_IGNORE_DATE_RANGE is 'false'  
LookupConstants.FIELD_PRICE_SCENARIO if  
LookupConstants.FLAG_ALL_PRICE_SCENARIOS is 'false'
```

ソーシング・プロジェクトからの価格検索に必須のパラメータ

```
ProjectConstants.ATT_GENERAL_INFORMATION_NUMBER  
LookupConstants.FLAG_ALL_PRICE_SCENARIOS  
ProjectConstants.ATT_ITEMS_NUMBER or ProjectConstants.ATT_ITEMS_AML
```

注意 たとえば、履歴など、ある価格検索では必須でないパラメータが、別のソーシング・プロジェクトからの価格検索ではオプションの場合があります。前述の必須パラメータのリストを比較すると、履歴から価格検索を実行する場合、LookupConstants.FLAG_ALL_PRICE_SCENARIOS パラメータはオプションです。オプションのパラメータは省略するか、パラメータを null に設定できます。

履歴またはソーシング・プロジェクトからの価格検索の設定

履歴またはソーシング・プロジェクトからの検索には、次のように LookupConstants.FIELD_LOOKUP_TYPE を設定します。

- 価格履歴からの検索 - LookupConstants.OPTION_LOOKUP_FROM_PRICE
- 既存のプロジェクトからの検索 - LookupConstants.OPTION_LOOKUP_FROM_PROJECT

価格検索での数量割引の設定

次のように LookupConstants.FIELD_SELECT_RESPONSE_BY を設定することで、コスト、日付またはリード・タイムで、価格検索に数量割引を設定できます。

- コストによる割引適用の場合 - LookupConstants.OPTION_LOWEST_PRICE
- 日付による割引適用の場合 - LookupConstants.OPTION_MOST_RECENT_RESPONSE
- リード・タイムによる割引適用の場合 - LookupConstants.OPTION_SHORTEST_LEAD_TIME

不適切なパラメータ設定の影響

次のパラメータが未設定の場合または不適切に設定されている場合は、次のアクションが実行されます。

- LookupConstants.FIELD_LOOKUP_TYPE は、価格履歴からの検索に対応する LookupConstants.OPTION_LOOKUP_FROM_PRICE にデフォルト設定されます。

- `LookupConstants.FLAG_IGNORE_QUANTITY` または `LookupConstants.FLAG_IGNORE_DATE_RANGE` は、`true` にデフォルト設定されます。
- `LookupConstants.FIELD_SELECT_RESPONSE_BY` は、コストによる割引適用に対応する `LookupConstants.OPTION_LOWEST_PRICE` にデフォルト設定されます。
- `LookupConstants.FLAG_IGNORE_ITEM_REVISION` または `LookupConstants.FLAG_CONSIDER_BEST_PRICES` は、`false` にデフォルト設定されます。
- `LookupConstants.LookupConstants.FLAG_ALL_PRICE_SCENARIOS` が設定されていない場合は、`true` にデフォルト設定されます。
- 必須パラメータが欠落している場合は、`ExceptionConstants.APDM_ADMIN_MISSINGREQUIREDFIELD` 例外が発生します。
- データ・タイプまたはパラメータの値が不適切に設定されている場合は、`ExceptionConstants.API_INVALID_PARAM` 例外が発生します。

見積依頼検索の場合:

この設定は、価格履歴からのソーシング・プロジェクト検索に類似しています。次にコード・サンプルを示します。

```
HashMap map = new HashMap();
map.put(PriceConstants.ATT_GENERAL_INFORMATION_PRICE_TYPE, priceTypes);
map.put(ProjectConstants.ATT_GENERAL_INFORMATION_CUSTOMER, customers);
map.put(ProjectConstants.ATT_GENERAL_INFORMATION_PROGRAM, programs);
map.put(ProjectConstants.ATT_GENERAL_INFORMATION_SHIP_TO_LOCATION, shipTo
);
map.put(ProjectConstants.ATT_ITEMS_NUMBER, itemMap);
map.put(ProjectConstants.ATT_ITEMS_AML, mpnMap);
map.put(LookupConstants.FLAG_ALL_PRICE_SCENARIOS, allPriceScenarios);
map.put(LookupConstants.FIELD_PRICE_SCENARIO, priceScenario);
map.put(LookupConstants.FLAG_IGNORE_QUANTITY, ignoreQtyRange);
map.put(LookupConstants.FIELD_QUANTITY_RANGE, qtyPercentRange);
map.put(LookupConstants.FLAG_IGNORE_DATE_RANGE, ignoreDateRange);
map.put(LookupConstants.FIELD_DATE_RANGE, dateRange);
map.put(LookupConstants.FIELD_SELECT_RESPONSE_BY, LookupConstants.OPTION_
LOWEST_PRICE);
map.put(LookupConstants.FLAG_EXCLUDE_AUTH_SUPPLIER, excludeAuthSupplier);
rfq.lookupPrices(map);
```

次のリストは、見積依頼検索の必須パラメータの一覧です。

```
PriceConstants.ATT_GENERAL_INFORMATION_PRICE_TYPE
ProjectConstants.ATT_ANALYSIS_SUPPLIER, suppliers
ProjectConstants.ATT_ITEMS_NUMBER or ProjectConstants.ATT_ITEMS_AML
LookupConstants.FIELD_QUANTITY_RANGE if
LookupConstants.FLAG_IGNORE_QUANTITY is 'false'
LookupConstants.FIELD_DATE_RANGE if
LookupConstants.FLAG_IGNORE_DATE_RANGE is 'false'
LookupConstants.FIELD_PRICE_SCENARIO if
LookupConstants.FLAG_ALL_PRICE_SCENARIOS is 'false'
```

注意 `LookupConstants.FLAG_EXCLUDE_AUTH_SUPPLIER` が設定されていない場合は、`false` にデフォルト設定されます。

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、見積依頼の完了ソーシング・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は、ソーシング・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 split2 =  
    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）に割り当てることができますが、製造元部品番号（MPN）に割り当てられません。アイテムがソーシング・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。

特定のパートナのデータを取得するには、次に示すように、「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).toString();
}
```

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、273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](#)」にある例と同じです。

例: 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_RESPONSE,
            "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_RESPONSE, "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。見積依頼はソーシング・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、最初にソーシング・プロジェクトを作成し、次にそのソーシング・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。

単一のソーシング・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、サプライヤとの一对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。

- 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」、「ページ 2」および「見積依頼回答」テーブルへのアクセスと変更
- 見積依頼のソーシング・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」、「ページ 2」および「見積依頼回答」テーブル内のネスト・テーブルの読取りと更新
- 「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、284 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「ページ 3」がなく、「ページ 3」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

SDKでは、見積依頼に対して次のAPIをサポートしています。これらのインタフェースの詳細は、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。ファイルはSDK_samples (ZIPファイル) のHTMLフォルダにあります。このファイルにアクセスするには、2ページの「[クライアント側のコンポーネント](#)」の注意を参照してください。

- `IAgileSession.createObject(Object, Object)`
- `IAgileSession.createObject(int, Object)`
- `IAgileSession.getObject(Object, Object)`
- `IAgileSession.getObject(int, Object)`
- `IRequestForQuote.getName()`
- `IRequestForQuote.assignSupplier(Object)`
- `IRequestForQuote.getTable(Object)`
- `IRequestForQuote.lookupPrices(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、特定のソーシング・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 (284ページの「[ソーシング・プロジェクトに対する見積依頼の作成](#)」を参照) と同様に、`getObject`、`getTable`、`getValue`、`setValue`などの標準呼び出しで `IDataObject` メソッドを使用すると、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て変更できます。

- 「ページ1」 (「カバー・ページ」)、「ページ2」テーブルの読取り
- 「ページ1」 (「カバー・ページ」)、「ページ2」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、ソーシング・プロジェクトを開く必要があります。ただし、ソーシング・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。272ページの「[ソーシング・プロジェクトのステータスへのアクセスと変更](#)」のコード例を参照してください。

ソーシング・プロジェクト番号のみを指定して見積依頼は作成できません。関連ソーシング・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: ソーシング・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、またはソーシング・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;
}

```

ソーシング・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、ソーシング・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の選択もできます。

例: ソーシング・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	読取り/書き込みモード
カバー・ページ	TABLE_COVERPAGE	読取り/書き込み
ページ 2	TABLE_PAGETWO	読取り/書き込み
回答	TABLE_RESPONSES	読取り/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」テーブルに新規行は追加できます。

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	読取り/書込みモード
TABLE_RESPONSES	ATT_RESPONSES_AML	読取り/書込み
TABLE_RESPONSES	ATT_RESPONSES_PRICING	読取り/書込み

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。272ページの「[ソーシング・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と変更](#)」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

```
ITable subtab1 =
    (ITable) row.getValue(RequestForQuoteConstants.ATT_RESPONSES_PRICING);
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。

見積依頼での価格検索の実行

275ページの「[ソーシング・プロジェクトでの価格検索の実行](#)」と同様に、見積依頼について指定の期間と数量に関する価格算出ケースが存在していることを確認できます。価格算出ケースがある場合は、指定のアイテムについて見積依頼を作成する必要はありません。アイテムの価格情報を使用することも、価格情報を変更して再見積のためにサプライヤに見積依頼を送信することもできます。次のコード・サンプルで例を示します。

例: 履歴および別のソーシング・プロジェクトからの価格検索

```

HashMap map = new HashMap();
map.put(PricingConstants.ATT_GENERAL_INFORMATION_PRICE_TYPE, priceTypes);
map.put(PricingConstants.ATT_ANALYSIS_SUPPLIER, suppliers);
map.put(PricingConstants.ATT_GENERAL_INFORMATION_CUSTOMER, customers);
map.put(PricingConstants.ATT_GENERAL_INFORMATION_PROGRAM, programs);
map.put(PricingConstants.ATT_GENERAL_INFORMATION_SHIP_TO_LOCATION, shipTo);
map.put(PricingConstants.ATT_ITEMS_NUMBER, itemMap);
map.put(PricingConstants.ATT_ITEMS_AML, mpnMap);
map.put(PricingConstants.FLAG_ALL_PRICE_SCENARIOS, allPriceScenarios);
map.put(PricingConstants.FIELD_PRICE_SCENARIO, priceScenario);
map.put(PricingConstants.FLAG_IGNORE_QUANTITY, ignoreQtyRange);
map.put(PricingConstants.FIELD_QUANTITY_RANGE, qtyPercentRange);
map.put(PricingConstants.FLAG_IGNORE_DATE_RANGE, ignoreDateRange);
map.put(PricingConstants.FIELD_DATE_RANGE, dateRange);
map.put(PricingConstants.FIELD_SELECT_RESPONSE_BY, PricingConstants.OPTION_LOWEST_PRICE);
map.put(PricingConstants.FLAG_EXCLUDE_AUTH_SUPPLIER, excludeAuthSupplier);
rfq.lookupPrices(map);

```

見積依頼価格検索の必須パラメータ

```

PricingConstants.ATT_GENERAL_INFORMATION_PRICE_TYPE
PricingConstants.ATT_ANALYSIS_SUPPLIER, suppliers
PricingConstants.ATT_ITEMS_NUMBER or PricingConstants.ATT_ITEMS_AML
PricingConstants.FIELD_QUANTITY_RANGE if
PricingConstants.FLAG_IGNORE_QUANTITY is 'false'
PricingConstants.FIELD_DATE_RANGE if
PricingConstants.FLAG_IGNORE_DATE_RANGE is 'false'
PricingConstants.FIELD_PRICE_SCENARIO if
PricingConstants.FLAG_ALL_PRICE_SCENARIOS is 'false'

```

不適切なパラメータ設定の影響

PricingConstants.FLAG_EXCLUDE_AUTH_SUPPLIER が設定されていない場合は、false にデフォルト設定されます。

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、次の操作をサポートしています。

- 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次のメソッドが用意されています。

```
public void assignSupplier(Object supplierParams)
    throws APIException, RemoteException, Exception;
```

製造元部品番号 (mpn) やサプライヤ名などのサプライヤ・データは、文字列またはオブジェクトとして見積依頼回答に割り当てることができます。

例: サプライヤ・データのリテラルとしての追加

```
IRequestForQuote dObj =
    (IRequestForQuote)m_session.getObject(RequestForQuoteConstants.CLASS_R
        FQ, number);
ITable tab =
    dObj.getTable(RequestForQuoteConstants.TABLE_RESPONSES);
Map mp = new HashMap();
mp.put(ProjectConstants.ATT_RESPONSES_NUMBER, "P00007");
mp.put(ProjectConstants.ATT_RESPONSES_SUPPLIER, "SDKSUP");
dObj.assignSupplier(mp);
```

例: サプライヤ・データのオブジェクトとしての追加

アイテムは、次に示すように IItem オブジェクトとして追加することもできます。

```
mp.put(ProjectConstants.ATT_RESPONSES_NUMBER, itemObject);
```

サプライヤを mpn に割り当てる場合は、その mpn を IManufacturerPart オブジェクトとして、またはオブジェクトのペアとして指定する必要があります。ペアとして指定する場合、1 つは mpn 名、もう 1 つは mfr 名です。

```
mp.put(RequestForQuoteConstants.ATT_RESPONSES_NUMBER, mpnObject);
```

または

```
mp.put(RequestForQuoteConstants.ATT_RESPONSES_NUMBER, mpnName);
mp.put(RequestForQuoteConstants.ATT_RESPONSES_MANUFACTURER, mfrName);
```

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 を介して「見積依頼回答」テーブルをサポートしています。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、サーバーでは、材料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。

製品の規制および適合性の管理

この章のトピック

▪ Agile Product Governance & Complianceについて	291
▪ Agile PG&Cのインタフェースとクラス	292
▪ Agile PG&Cの役割	292
▪ デklarレーション、含有基準およびサブスタンスの作成	293
▪ デklar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	297
▪ デklarレーションへのサブスタンスの追加	298
▪ 含有基準へのサブスタンスの追加	304
▪ デklarレーションへの含有基準の追加	305
▪ デklarレーションの送信	306
▪ デklarレーションの完成	307
▪ 適合性管理者へのデklarレーションの提出	308
▪ デklarレーションの公表	308
▪ 重量値の取得および設定	309
▪ 製造元部品のサブスタンス組成の追加	310
▪ 適合性データのロールアップ	312

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klar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、『Product Governance & Compliance ユーザー・ガイド』を参照してください。

Agile PG&Cのインタフェースとクラス

次の表に、Agile PG&C に関連するインタフェースとクラスを示します。

オブジェクト	インタフェース	定数クラス
デklar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klar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klar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klar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klar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klar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klar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。

注意 「適合性管理者」役割には「ディスカバリ(変更)」権限マスクは含まれていません。「適合性管理者」役割のみが付与されている場合、デklarレーションでAPIを使用して部品の適合性判定値を設定し、変更番号をSDK呼出しに渡すには、権限が不十分です。変更番号をSDK呼出しに渡すためには、変更指示クラスの該当するオブジェクトに対する「ディスカバリ(変更)」権限マスクが必要です。詳細は、316ページの「[デklarレーション・オブジェクトに対する「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。

デklarレーション、含有基準およびサブスタンスの作成

次に、これらの PG&C の各クラスを指定および管理する定義および手順を示します。

デklarレーションの作成

デklar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klar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klar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klarレーション・サブクラスは次の 7 つです。

- 均質材のデklar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klarレーション。
- IPC 1752-1 デklar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klarレーション。
- IPC 1752-2 デklar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klarレーション。
- JGPSSI デklar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klarレーション。
- 部品のデklar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klarレーション。
- サブスタンスのデklar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klarレーション。
- 適合のサプライヤ・デklar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klarレーションで使用できます。

デklarレーションを作成する手順は、すべてのデklarレーション・サブクラスについて同じです。デklar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klar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klar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klarレーションも作成できます。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klarレーションを作成する方法を示しています。

例: JGPSSI デklar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klar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がある必須属性は「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klarেশ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 subClass =
        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 subClass =
    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、指定する必要がある属性は、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がある属性は、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も指定できます。

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klar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klar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klar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klar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リビジョンが使用されます。アイテムにリリース済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klarレーションに追加する方法を示しています。

例: デklar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_NUMB
ER, "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klarレーションへのサブスタンスの追加

サブスタンスは、デklar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klarレーションをリリースします。デklar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klar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klar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は作成できません。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klar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klar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klar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。

Agile SDK の場合、デklar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。

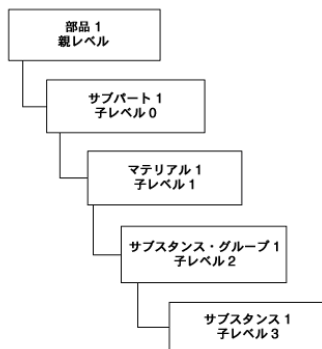
「組成」テーブル	必須属性	DeclarationConstants
アイテム組成	アイテム行 サブスタンス名	ATT_PARENT_ROW ATT_ITEM_COMPOSITION_SUBSTANCE_NAME
製造元部品の組成	製造元部品行 サブスタンス名	ATT_PARENT_ROW ATT_MANUFACTURER_PART_COMPOSITION_SUBSTANCE_NAME
部品グループの組成	部品グループ行 サブスタンス名	ATT_PARENT_ROW ATT_PART_GROUP_COMPOSITION_SUBSTANCE_NAME

BOS（サブスタンス構成表）の構造

サブスタンスをデklar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klarレーションのタイプによって異なります。

- 均質材のデklar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klarレーション/JGPSSI デklar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klarレーション/適合のサプライヤ・デklarレーション - これらのデklar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klarেশ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klarেশ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klarেশ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) を渡すことをお勧めします。

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.CLAS
            S_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 Homogeneous 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_NUMB
            ER, "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 Compounds");
        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klar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klar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(I AgileSession 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.CLAS
            S_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_NUMB
        ER, "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_G
            ROUP, "Lead Compounds");
    params.clear();
    params.put(DeclarationConstants.ATT_PARENT_ROW, rowMfrParts);
    params.put(DeclarationConstants.ATT_MANUFACTURER_PART_COMPOSITION_SU
        BSTANCE_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klarレーションへの含有基準の追加

デklarレーションの「含有基準」テーブルは、デklar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klar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klarレーションでは含有基準はオプションです。デklar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klarレーションに追加する場合は、デklar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klarレーションに対する含有基準の要件を示します。

デklar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klarレーション	均質材レベル
IPC 1752-1 デklarレーション	部品レベル
IPC 1752-2 デklarレーション	均質材レベル
JGPSSI デklarレーション	部品レベル
部品のデklar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klarレーション	部品レベル
適合のサプライヤ・デklar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klar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klar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klar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。

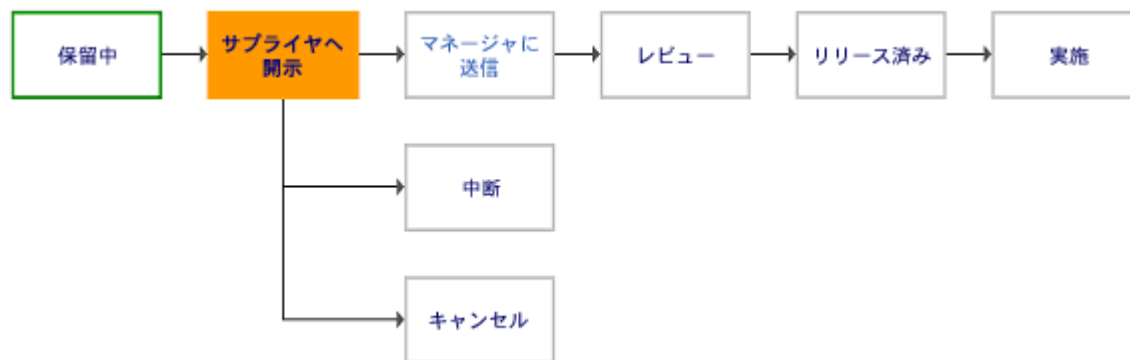
例: 「含有基準」テーブルへの含有基準の追加

```
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klarレーションの送信

「デフォルト・デklarレーション」ワークフローは、次の図に示すように、簡単な操作で処理できます。

図18: 「デフォルト・デklarレーション」ワークフロー



次の表に、「デフォルト・デklar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klar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klar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klar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klar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 サーバーによって、含有基準にリストされているサブスタンスがデklar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klar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klar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	適合性管理者はデklar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。
実施済	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klarレーションを実施し、ワークフローを完了します。

デklarレーションをルートするには、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。

- カバー・ページ, 適合性管理者
- カバー・ページ, ワークフロー
- カバー・ページ, 締切日

技術的には、デklarレーションのルートに必要なのは、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_MANAGER,
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klar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klar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klar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klarレーションは読取り専用になります。次の例は、適合性管理者がデklar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klar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・プロセスをサポートしているAgile APIの詳細は、207ページの「[ワークフローの管理](#)」を参照してください。

デklarレーションの完成

デklar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klar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klar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klar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klar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klar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klar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klarレーションの提出

サプライヤがデklar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」に変更するときは、デklar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klar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klar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klarレーションの公表

Agile API には、マテリアル・デklar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klar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デklar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klar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は定義できません。

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 uommm =
    (IUnitOfMeasureManager)m_session.getManager(
        IUnitOfMeasureManager.class);
    uom = uommm.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。サーバーは、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klar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klar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、299ページの[「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」テーブルに行は追加できません。

「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_P
            ART_NUMBER, "Widget");
        params.put(ManufacturerPartConstants.ATT_GENERAL_INFO_MANUFACTURER_N
            AME, "ACME");
        IManufacturerPart mfrPart =
            (IManufacturerPart)
            m_session.createObject(ManufacturerPartConstants.CLASS_MANUFACTURER_
                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_SPECIFICATION,
            spec.getName());
        params.put(ManufacturerPartConstants.ATT_COMPOSITIONS_COMPOSITION_TY
            PE,
            "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_G
            ROUP, "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klar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klar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、サーバー側での PG&C ロールアップ機能の呼び出しをサポートしています。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 インタフェースでこの機能がサポートされています。

IPGCRollupインタフェースの理解

IPGCRollup インタフェースには、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
    Item2
Assembly 2
    Item1
    Item3
```

IPGCRollupインタフェースの使用

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 ECO または MCO）
- MPN（最新のリリース済 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 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")+"]."+SDKWrapper
        .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 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).toString();
        try {
            itemRollup(itemStr);
        } catch (APIException e) {
            int error =
                ((Integer)e.getErrorCode()).intValue();
        }
    }
    return;
}

```

アイテム・オブジェクトに対する「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 Calculated Compliance 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: アイテム・オブジェクトに対する「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).toString()
    );
    item.setCalculatedComplianceForPartSpec(spec_num, COMPLIANT);

```

デklarレーション・オブジェクトに対する「適合性判定値」フィールドの値の設定

これは、アイテム・オブジェクトに対する「適合性判定値」フィールドの設定を可能にする前述の API と類似しています。この API を使用すると、デklarレーション・オブジェクトに対する「アイテム」および「製造元部品」テーブルの「適合性判定値」フィールドに値を設定できます。

```

Public void setCalculatedComplianceForMDOPartSpec (Object partName, Object
partClassName, Object changeNumber, Object specName, Object
complianceEntryValue)) throws APIException

```

システムでは、SDK クライアントによるこの値の設定が確認され、ロールアップ時には新しい設定が後続の応答で使用されます。この API のパラメータ `changeNumber` はオプションです。デklarレーション・オブジェクトにアイテムのリビジョンが 1 つのみ存在している場合は、`changeNumber` の値を `null` に設定できます。デklarレーション・オブジェクトにアイテムのリビジョンが複数存在している場合は、API を適切に実行するために `changeNumber` の値を設定する必要があります。

前述の API と同様に、SDK クライアントによって設定されたこの値は、デklarレーションでのロールアップ時に上書きされることはありません。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。

注意 `changeNumber` フィールドを `setCalculatedComplianceForDeclarationPartSpec()` に渡すことを意図している SDK 開発者には、その変更に対する「ディスカバリ(変更)」権限マスクが必要です。

例: デklarレーション・オブジェクトに対する「適合性判定値」フィールドの値の設定

```

// complianceValue -- This is the customized calculated compliance
// value and shows if the part is compliant to a Spec
String ComplianceValue = "Compliant";
// partName is the Item/Mfr Part name in Declaration's Item/MfrPart
// table. If it is a mfr part, it should be like "MfrName::MfrPartName"
String partName = "P00001"; String partClassName = "Parts";
// If the added part in Declaration is an Item, the changeName should
// be the
// Change number corresponding to the Item's revision.
// If the added part in Declaration is a Mfr Part, the
// changeName should be "null"
String changeName = "C00001";
// spec_num is the Specification Name in Declaration object's
// Specification Table
String specName = "Rohs";
Declaration.setCalculatedComplianceForDeclarationPartSpec
(partName, partClassName, changeName, specName, complianceValue);

```

例外の処理

この章のトピック

■ 例外について	317
■ 例外定数	318
■ エラーコードの取得	318
■ バルクAPIでのエラー・コードの無効化と有効化	318
■ エラー・メッセージの取得	319
■ 警告メッセージの無効化および有効化	320
■ APIExceptionがエラーではなく警告であることの確認	321
■ 有効または無効にした警告の状態の保存および復元	321
■ Agile APIで自動的に無効にした警告の削除	322

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・サーバー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・サーバー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、API リファレンス・ファイル (<http://edelivery.oracle.com/>) 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。コードで使用しない Agile PLM 警告メッセージは、無効にできます。詳細は、320 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_WAR
NING))
        JOptionPane.showMessageDialog(null, ex.getMessage(), "Warning",
JOptionPane.YES_NO_OPTION);
    }
}
```

バルク API でのエラー・コードの無効化と有効化

SDK では、すべてのエラー・コードまたは指定した一連のエラー・コードを無効/有効にするために、IAgileSession で次の一括操作をサポートしています。

- ▣ IAgileSession.enableWarnings(Integer[])
- ▣ IAgileSession.disableWarnings(Integer[])
- ▣ IAgileSession.enableAllWarnings()
- ▣ IAgileSession.disableAllWarnings()

プロセスは、前述の例と類似しています。次の例は、これらのバルク API を使用して、変更のリリース時に警告を抑制する方法を示しています。

例: 一括モードでのエラー・コードの無効化と有効化

```

public static void releaseECO(I AgileSession session, IChange change)
    throws Exception {
    // Set workflow
    IWorkflow workflow = change.getWorkflows()[0];
    change.setWorkflow(workflow);

    IStatus submit = getStatus(workflow, StatusConstants.TYPE_SUBMIT);
    IStatus ccb = getStatus(workflow, StatusConstants.TYPE_REVIEW);
    IStatus released = getStatus(workflow,
        StatusConstants.TYPE_RELEASED);
    session.disableWarnings(new Integer[] {
        ExceptionConstants.APDM_WFL_ATLEASTONECHGANALYST_WARNING,
        ExceptionConstants.APDM_MISSINGFIELDS_WARNING });
    // instead you can use session.disableAllWarnings()
    // route to SUBMIT
    change.changeStatus(submit, false, null, false, false, new
        Object[] {}, new Object[] {}, new Object[] {}, false);
    // Change status to CCB
    change.changeStatus(ccb, false, null, false, false, new Object[] {},
        new Object[] {}, new Object[] {}, false);
    // route from CCB to release
    change.changeStatus(released, false, "release", false, false, new
        Object[] {}, new Object[] {}, new Object[] {}, false);
    session.enableWarnings(new Integer[] {
        ExceptionConstants.APDM_WFL_ATLEASTONECHGANALYST_WARNING,
        ExceptionConstants.APDM_MISSINGFIELDS_WARNING });
    // instead you can use session.enableAllWarnings()
}

public static IStatus getStatus(IWorkflow workflow, StatusConstants
    status)
    throws Exception {
    IStatus[] states = workflow.getStates(status);
    IStatus state = states[0];
    return state;
}

```

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は、API リファレンス・ファイル (<http://edelivery.oracle.com/>) 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、操作の停止または続行を選択できるオプションが提供されます。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_W
                        ARNING);
                    // Release the Change
                    chgObj.changeStatus(nextStatus, false, "", true, true, null, null,
                        null, false);
                    // Enable all warnings
                    m_session.enableWarning(ExceptionConstants.APDM_UNRESPONDEDCHANGE_WA
                        RNING);
                } 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;
    }
}
```

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_MFRNAMECHANGE_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();
}
```

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、「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。

管理タスクの実行

この章のトピック

▪ Agile PLM管理について	323
▪ Agile PLMの管理に必要な権限	324
▪ 管理インタフェース	324
▪ IAdminインスタンスの取得	325
▪ ノードの使用	325
▪ Agile PLMクラスの管理	330
▪ 属性の使用	334
▪ 管理ノードのプロパティの使用	339
▪ ユーザーの管理	340
▪ ユーザー・グループの管理	345

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・サーバー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取り/書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は作成できません。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・サーバーを管理するには、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、サーバーで利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 管理者に問い合せてください。

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。

管理インタフェース

次の表に、Agile PLM 管理機能に関連するインタフェースを示します。

インタフェース	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	シングルの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。
IUserGroup	ユーザー・グループ。ユーザー・グループを使用して、プロジェクト・チーム、拠点関連のグループ、部署またはグローバル・グループを定義します。

インタフェース	説明
IWorkflow	ワークフロー・ノード。

IAdmin インスタンスの取得

IAdmin インタフェースは、Agile アプリケーション・サーバー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・サーバーにログインして 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。

- サーバー・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・サーバー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

Agile Java クライアントのノード	対応する Agile API
Product Cost Management	
出荷先の場所	サポートされていません
Projects Execution	
プロジェクト・ヘルス	サポートされていません
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
サーバー設定	
サーバーの場所	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・プロフィール	「ユーザー一般情報」 ページ
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・サブライヤ・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
サブライヤ・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は一致しません。

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・サーバーのトップレベル・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の取得もできます。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も指定できます。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ドは、`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勧めします。「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、330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は作成できませんが、既存のサブクラスの読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、オラクル社との Agile PLM 契約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	変更指示	ECO
	変更要求	ECR
	期限付き変更指示	期限付き変更指示
	製造元依頼	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klarレーション	均質材のデklarレーション	均質材のデklarレーション
	IPC 1752-1 デklarレーション	IPC 1752-1 デklarレーション
	IPC 1752-2 デklarレーション	IPC 1752-2 デklarレーション
	JGPSSI デklarレーション	グリーン調達調査共通化協議会のデklarレーション
	部品のデklarレーション	部品のデklarレーション
	サブスタンスのデklarレーション	サブスタンスのデklarレーション
	適合のサプライヤ・デklarレーション	適合のサプライヤ・デklarレーション

基本クラス	クラス	事前定義済のサブクラス
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
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
プロジェクト	アクティビティ	フェーズ
		プロジェクト
		タスク
	ゲート	ゲート
品質変更要求	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

基本クラス	クラス	事前定義済のサブクラス
	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
		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
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、設計変更指示クラスなど）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 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・タイプを使用すると、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た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、93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勧めします。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「ページ 2」属性と「ページ 3」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: 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。

属性の参照 - 9.3

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ります。ID番号や定数による属性の参照は識別が容易ではなく忘れやすいため、APIName フィールドの使用をお勧めします。詳細および手順は、125ページの「[フィールドを使用したPLM内部データへのアクセス](#)」を参照してください。このマニュアルの多くの例は、このフィールドが導入される前に作成されているため、名前で属性を参照しています。

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「ページ 2」属性と「ページ 3」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: 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、「ページ 2」および「ページ 3」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も記述できます。

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 のプロパティ値の表示もできます。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;
}
```

シングルの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は直接変更できません。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。シングルの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、98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・サーバー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・サーバー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・プロフィール」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、343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。パスワードの有効期限が切
れると、ユーザーは Agile アプリケーション・サーバーにログインできなくなります。ただし、ログイン・パ
スワードが無効になった場合、ユーザーは、Agile API プログラムを使用して新しいパスワードを指定できます。

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。

```
HashMap params = new HashMap();
params.put(AgileSessionFactory.USERNAME, user.getName());
params.put(AgileSessionFactory.PASSWORD, "agile");
params.put(AgileSessionFactory.URL, "http://localhost/Agile");
// Pass one element array to get the login exception
APIException[] exception = new APIException[1];
params.put(AgileSessionFactory.STATUS, exception);
I AgileSession session=AgileSessionFactory.createSessionEx(params);
System.out.println(exception[0].getErrorCode());
// If The error code is ExceptionConstants.APDM_PWDNOCHANGE
```

```

        If(exception[0].getErrorCode() ==
            ExceptionConstants.APDM_PWDNOCHANGE) {
// Prompt for new password and change the password
        Session.getUser().changeLoginPassword (oldPassword, newPassword);
    }
}

```

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「ページ 2」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「ページ 2」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** - ユーザーは、いつでもサーバーにログインでき、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** - ユーザーは、同時接続ユーザー・ライセンスがある場合のみ、サーバーにログインできます。

注意 ライセンスは、オラクル社との契約に従って管理されます。

- **制限付き** - ユーザーには、Agile PLM システムへのアクセスに制限があります。サブライヤ・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、サブライヤ・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は作成できません。Agile PLM ユーザーの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: 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勧めします。

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 `null` 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 `null` 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・サーバーでは、ユーザーを完全には削除できません。

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は送信できます。

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_USE
            RS) + ", " +
            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・サーバー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・グループを作成した後は、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_GROUP,
        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);
}
```

ユーザーの「ユーザー・グループ」テーブルへのユーザー・グループの追加

ユーザーの「ユーザー・グループ」テーブルにユーザー・グループを追加するために、IUserGroup を createRow() に渡すことはできません。次の例のように、Map を使用する必要があります。

例: Map を使用した「ユーザー・グループ」テーブルへのユーザー・グループの追加

```
ITable ugTable = user.getTable(UserConstants.TABLE_USERGROUP);
Map map = new HashMap();
map.put(UserConstants.ATT_USER_GROUP_GROUP_NAME, ug.getName());
ugTable.createRow(map);
```

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));
    }
}
```


プロセス拡張の開発

この章のトピック

■ プロセス拡張について	349
■ カスタム自動採番ソースの開発	350
■ カスタム・アクションの開発	353
■ URLベースのプロセス拡張の定義と配置	359
■ 外部レポートの作成	365
■ クラスタ環境でのプロセス拡張の配置	366
■ サード・パーティJARファイルをコピーするためのベスト・プラクティス	366
■ プロセス拡張に関するよくある質問	369

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、サーバー側の拡張（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・サーバーで起動されます。

注意 PX フレームワークで開発できるサーバー側の機能に加え、Agile PLM のイベント・フレームワークでも、Java および Groovy スクリプト（それぞれ Java PX およびスクリプト PX と呼ばれる）を使用した拡張の開発をサポートしています。PX フレームワークで開発したカスタム・アクションの一部はイベント・フレームワークに移行できますが、2つのフレームワークにはそれぞれ固有のインターフェースがあり、それらは同じではありません。

プロセス拡張によって、Agile PLM サーバーと 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アプリケーションを調整できます。

プロセス拡張は、Agile アプリケーション・サーバーに配置されている 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）への依頼の作成もできます。

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、次の領域から起動できます。

- 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も指定できます。

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・サーバーでは、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ー側 API である ICustomAutoNumber インタ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 サーバーは、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 your 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・サーバー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・サーバーがある場合は、クラスタ内の各サーバーにプロセス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。


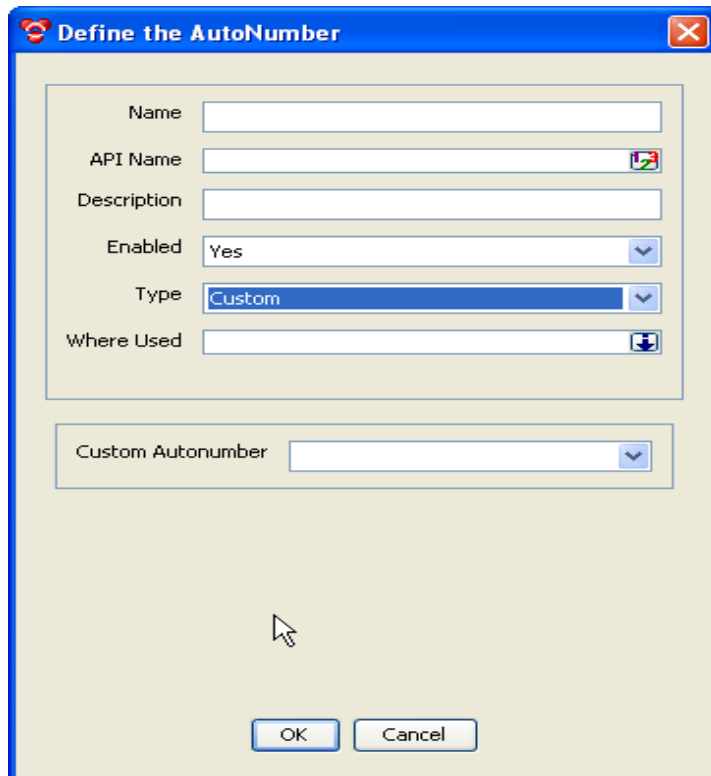
1. 管理者として Agile Java クライアントにログインします。
2. 「管理」タブをクリックします。
3. 「設定」>「データ設定」>「自動採番」の順に進みます。
4. 「自動採番」ノードをクリックします。
5. 「自動採番」ウィンドウでをクリックします。「自動採番の定義」ダイアログ・ボックスが表示されます。

図19:「自動採番の定義」ダイアログ・ボックス



The dialog box titled "Define the AutoNumber" contains the following fields and controls:



- Name: Text input field.
- API Name: Text input field with a small icon to its right.
- Description: Text input field.
- Enabled: Dropdown menu with "Yes" selected.
- Type: Dropdown menu with "Custom" selected.
- Where Used: Text input field with a small icon to its right.
- Custom Autonumber: Text input field with a dropdown arrow on the right.
- Buttons: "OK" and "Cancel" buttons at the bottom.

6. 次の情報を指定します。
 - 名前 - 自動採番ソースの名前を入力します。
 - API名 - 「名前」フィールドへの入力を完了すると、自動的に入力されます。125ページの「[APIName フィールドを使用したPLMメタデータへのアクセス](#)」を参照してください。
 - 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 クライアントにログインします。
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」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ー側 API である ICustomAction インタ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 サーバーは、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"が変更指示の「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");
    }
}
```

この例にある HelloWorld クラスは有用な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。

PLMクライアントでの改行の書式設定

Web クライアントおよび Java クライアントでの ActionResult 出力には、「¥n」文字を使用して改行を設定できます。

注意 SDK では、Web クライアントおよび Java クライアントでの改行に「¥n」文字のみをサポートしています。

例: SDK 文字列出力への改行の書式設定

1. 次の ActionResult 文字列を使用して挿入するプロセス拡張を作成します。

```
return ActionResult(ActionResult.STRING, "Hello ¥n This example tests
formatting new lines ¥n in the SDK");
```

2. Web クライアントおよび Java クライアントで、「アクション」メニューおよび「ツール」メニューからプロセス拡張を実行します。

次のように出力されます。

```
Hello
This example tests formatting new lines
in the SDK
```

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。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。

URL ベースのプロセス拡張では、Agile PLM サーバー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、361 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・サーバー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・サーバーがある場合は、クラスタ内の各サーバー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も設定できます。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、サーバーでは、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。

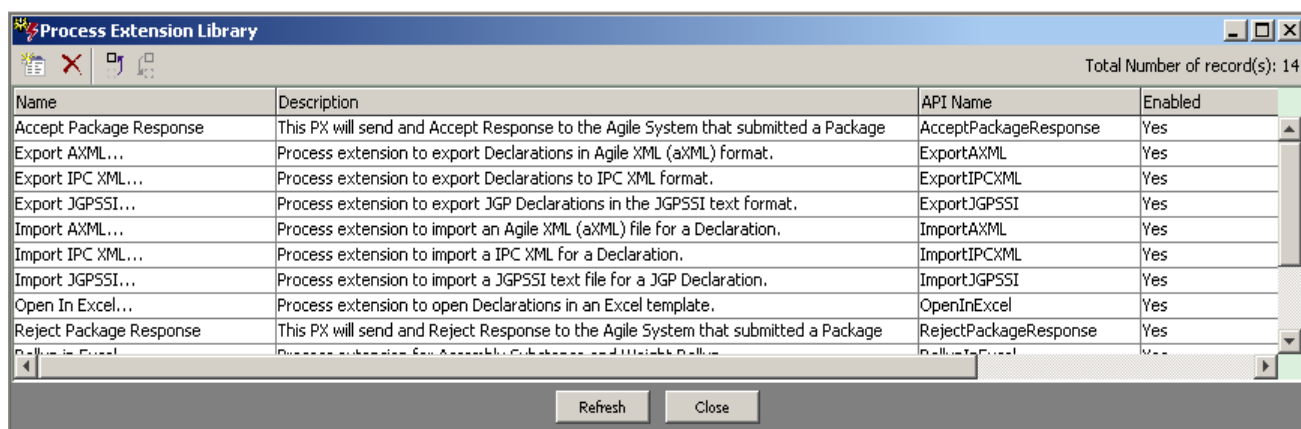
プロセスの拡張ライブラリの使用

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 クライアントにログインします。
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20: 拡張ライブラリ



The screenshot shows a window titled 'Process Extension Library' with a table of process extensions. The table has four columns: Name, Description, API Name, and Enabled. There are 14 records in total. The 'Enabled' column for all records is set to 'Yes'. At the bottom of the window, there are 'Refresh' and 'Close' buttons.

Name	Description	API Name	Enabled
Accept Package Response	This PX will send and Accept Response to the Agile System that submitted a Package	AcceptPackageResponse	Yes
Export AXML...	Process extension to export Declarations in Agile XML (aXML) format.	ExportAXML	Yes
Export IPC XML...	Process extension to export Declarations to IPC XML format.	ExportIPCXML	Yes
Export JGPSSI...	Process extension to export JGP Declarations in the JGPSSI text format.	ExportJGPSSI	Yes
Import AXML...	Process extension to import an Agile XML (aXML) file for a Declaration.	ImportAXML	Yes
Import IPC XML...	Process extension to import a IPC XML for a Declaration.	ImportIPCXML	Yes
Import JGPSSI...	Process extension to import a JGPSSI text file for a JGP Declaration.	ImportJGPSSI	Yes
Open In Excel...	Process extension to open Declarations in an Excel template.	OpenInExcel	Yes
Reject Package Response	This PX will send and Reject Response to the Agile System that submitted a Package	RejectPackageResponse	Yes
Rollback Excel...	Process extension for Assembly Subprocess and Unlink Rollback	RollbackExcel	Yes


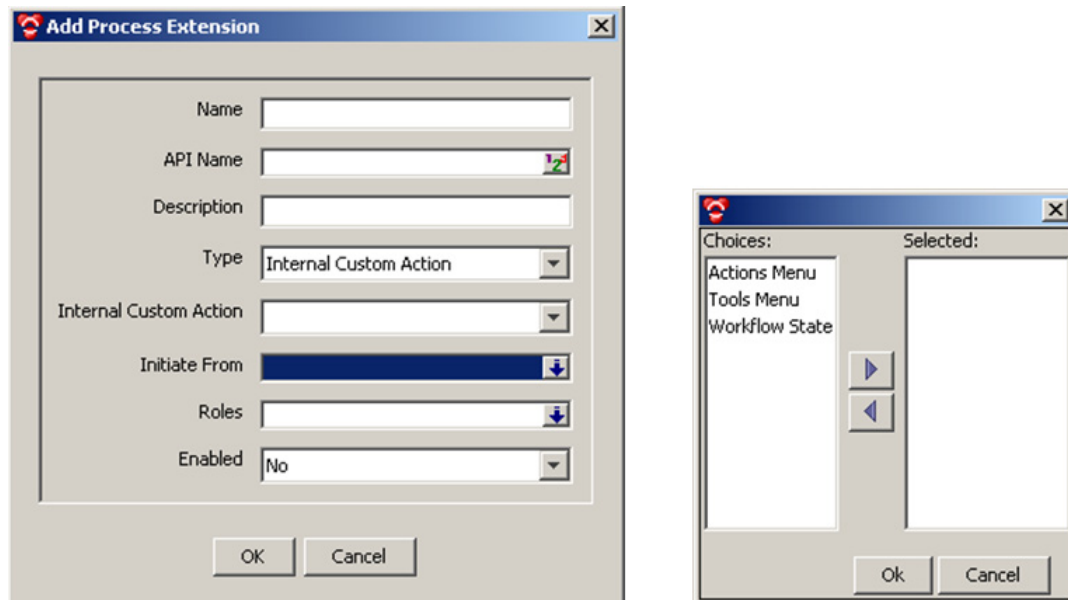
5. 「拡張ライブラリ」ウィンドウで、「プロセス拡張の追加」ボタンをクリックして「プロセス拡張の追加」ダイアログ・ボックスを開きます。

図21: 「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。
 - **API名** - 「名前」フィールドへの指定を完了すると、自動的に入力されます。125ページの「[APIName フィールドを使用したPLMメタデータへのアクセス](#)」を参照してください。
 - **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」に割り当ててす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 クライアントにログインします。
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」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。

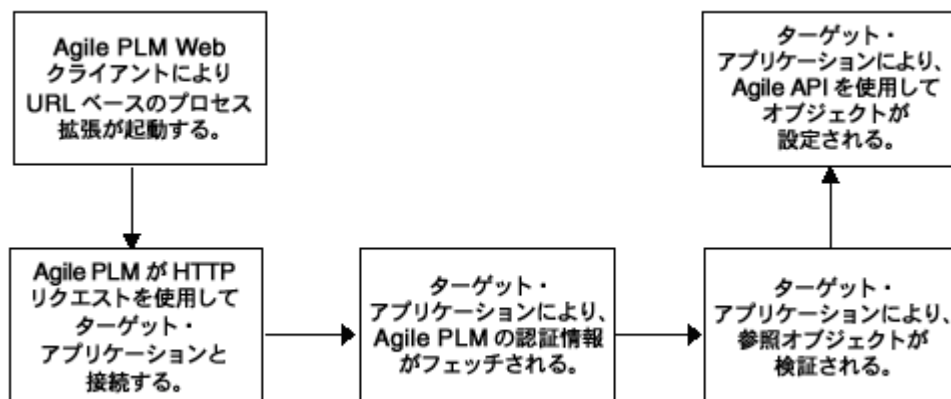
1. 管理者として Agile Java クライアントにログインします。
2. 「管理」タブをクリックします。
3. 「ワークフロー設定」フォルダを開きます。
4. 「ワークフロー」ノードを開きます。
5. 「ワークフロー」ウィンドウで、ワークフローをダブルクリックします。
6. 「ステータス」タブをクリックします。
7. 「保留中」以外のステータスを選択します。ステータス・テーブルの下に表示される「ワークフロー条件」プロパティ・テーブルが、選択したステータスによって更新されます。
8. 「ワークフロー条件」プロパティ・テーブルで、選択したステータスをダブルクリックします。
9. 「プロセスの拡張」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」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と配置

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22: 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、365ページの「[外部レポートの作成](#)」を参照してください。

注意 Agile Java クライアントは URL PX をサポートしていません。


URLベースのプロセス拡張を作成する前の注意

URL ベースのプロセス拡張を作成する場合は、次の要件および制約に注意してください。

- ステータスの変更をトリガーする Agile PLM クライアントがアクティブでない可能性があるため、ワークフローでの変更によって URL ベースのプロセス拡張を開始できない場合があります。
- 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の定義

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** - 393ページの「[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します。
 - 「ツール」メニュー - 「ツール」メニューから Web ページを選択するために使用します。

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・サーバーを介するシングル・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 SDK で再利用できます。外部アプリケーション・サーバー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=10141
&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ア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 サーバー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・アプリケーションでは 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ー・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。

注意 これらのクッキーは、Agile PLM 管理者が設定した期間を経過すると失効となります。

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・システムでは、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。

詳細は、http://wp.netscape.com/newsref/std/cookie_spec.htmlを参照してください。

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.20000 04284	タイトル・ブロック.名前
デklarेशन	agile.classID	選択したオブジェクトのクラス ID
	agile.agile.20000 02615	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

クラス	パラメータ	説明
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
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

クラス	パラメータ	説明
	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・オブジェクトは取得できません。

外部レポートの作成

Agile Webクライアントでは、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には、その前に、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、359ページの「[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・サーバー・クラスタ内のサーバーで実行されていない場合は、そのサーバーに、/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. クラスタ内のすべてのアプリケーション・サーバー上に、/agile_home/integration/sdk/extensions フォルダを作成します（存在しない場合）。
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ーの /agile_home/integration/sdk/extensions フォルダに配置します。

サード・パーティ JAR ファイルをコピーするためのベスト・プラクティス

PX コードにサード・パーティ JAR ファイル（axis.jar など）を使用する場合は、次の手順を使用してファイルを共有ライブラリにコピーし、classpath に追加できます。

次の手順を使用して共有ライブラリを設定します。

Oracle Application Serverに共有ライブラリを設定する手順は、次のとおりです。

1. たとえば、D:\¥commonLib などのフォルダにサード・パーティ JAR ファイルを格納します。
2. Agile サーバーを停止します。
3. <OAS_HOME>\¥j2ee¥home¥application-deployments¥Agile に移動します。
4. orion-application.xml をテキスト・エディタで開きます。
5. ライブラリ・パスを指定する行を追加します。

```
<library path="ABSOLUTE_PATH_TO_YOUR_LIBRARY" />
```

<!-- 必要な JAR ファイルが格納されているフォルダへの絶対パスを指定します。)

たとえば、次の 2 行の後に<library path="D:\¥commonLib" />を追加します。

```
<library path=" ../APP-INF/lib" />
```

```
<library path=" ../APP-INF/classes" />
```

6. Agile サーバーを起動します。

このセクションの手順は、次の依存 JAR ファイルを配置する方法を示しています。

- Agile JAR ファイルに依存しないサード・パーティ JAR ファイルからの依存 JAR ファイル

- Agile JAR ファイルに依存または非依存のサード・パーティ JAR ファイルからの依存 JAR ファイル

サード・パーティ JAR ファイルを WebLogic に配置する手順は、次のとおりです。

1. ドメイン内のすべてのサーバーを停止します。
2. ドメイン・ディレクトリの lib サブディレクトリに共有 JAR ファイルをコピーします。

例: `cp c:\¥3rdpartyjars¥utility.jar agile_home¥agileDomain¥lib`

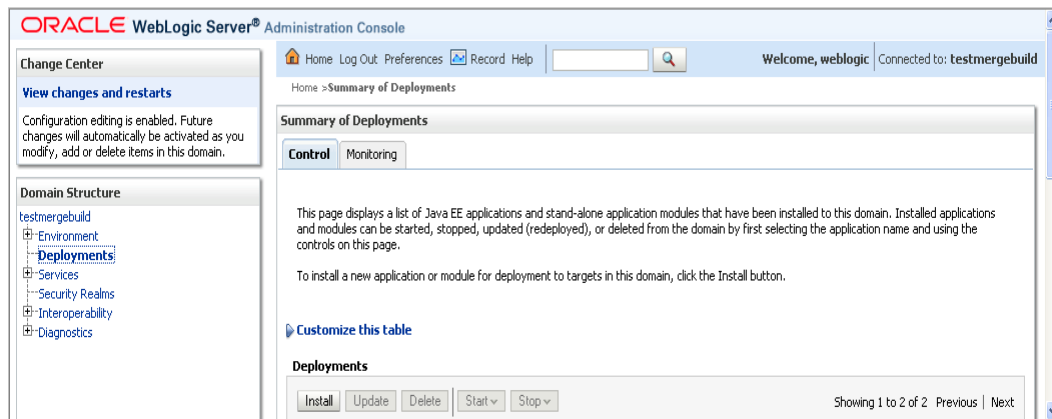
注意 WebLogic Server には、起動時に lib ディレクトリへの読取り権限が必要です。

注意 管理サーバーでは、lib ディレクトリ内のファイルをリモート・マシンの管理対象サーバーに自動的にコピーしません。物理的に同じドメイン・ディレクトリを管理サーバーと共有しない管理対象サーバーがある場合は、管理対象サーバーの domain_name/lib ディレクトリに JAR ファイルを手動でコピーする必要があります。

3. ドメイン内の管理サーバーと管理対象サーバーを起動します。
ドメイン内の管理サーバーとすべての管理対象サーバーが起動されます。WebLogic Server によって、lib ディレクトリ内の JAR ファイルがシステム・クラスパスに追加されます。複数のファイルがアルファベット順に追加されます。

WebLogic 管理コンソールから依存 JAR ファイルを配置する手順は、次のとおりです。

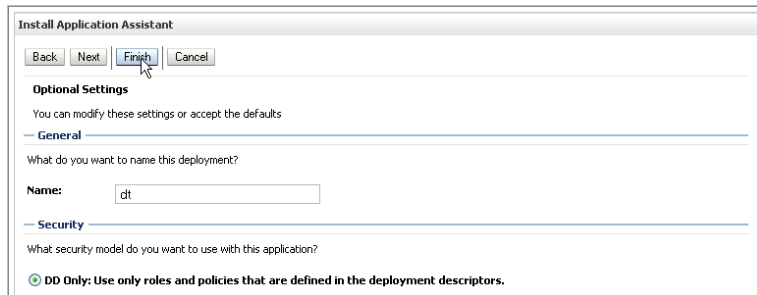
1. WLS 管理コンソールで、「デプロイメント」>「制御」>「インストール」の順に選択します。



2. JAR ファイルを選択し、「次へ」を選択します。



3. 「アプリケーション・インストール・アシスタント」で、デフォルトの設定を受け入れて「終了」をクリックします。



4. ライブラリが Agile サーバー/クラスタを指し示していることを確認します。
5. weblogic-application.xml というファイルを次の内容で作成します。

```
<?xml version="1.0" encoding="UTF-8"?>
  <wls:weblogic-application
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:wls=http://www.bea.com/ns/weblogic/weblogic-
      application
    xsi:schemaLocation="http://www.bea.com/ns/weblogic/web
      logic-application
      http://www.bea.com/ns/weblogic/weblogic-application.
      xsd http://java.sun.com/xml/ns/j2ee
      http://java.sun.com/xml/ns/j2ee/j2ee_1_4.xsd">
    <wls:library-ref>
      <wls:library-name><Shared Library Jar file
        name></wls:library-name>
    </wls:library-ref>
  </wls:weblogic-application>
```

6. weblogic-application.xml ファイルを管理サーバーの agile_home¥agiledomain¥applications¥application.ear¥META-INF フォルダに配置します。
7. 管理サーバーおよびすべての管理対象サーバーを再起動します。

プロパティ・ファイルをWebLogicに配置する手順は、次のとおりです。

1. ドメイン内のすべてのサーバーを停止します。
2. プロパティ・ファイルを各サーバーの agile_home/pxConfig ディレクトリにコピーします。

注意 WebLogic Server には、起動時にディレクトリへの読取り権限が必要です。

注意 管理サーバーでは、ディレクトリ内のファイルをリモート・マシンの管理対象サーバーに自動的にコピーしません。物理的に同じドメイン・ディレクトリを管理サーバーと共有しない管理対象サーバーがある場合は、管理対象サーバーの agile_home/pxConfig ディレクトリにプロパティ・ファイルを手動でコピーする必要があります。

3. 次のようにして、プロパティ・ファイルが格納されているディレクトリ（例: agile_home/pxConfig）を WebLogic CLASSPATH に追加します。
 - a. すべての管理対象サーバーの agile_home/agileDomain/bin/setEnv.bat を編集します。
 - b. **agile_home/pxConfig** フォルダを CLASSPATH に追加します。


```
set CLASSPATH=%JAVA_HOME%/lib/tools.jar
set CLASSPATH=%CLASSPATH%;%LIB_HOME%/ojdbc14.jar
set CLASSPATH=%CLASSPATH%;%WLS_HOME%/server/lib/
weblogic_sp.jar;%WLS_HOME%/server/lib/weblogic.jar
set CLASSPATH=%CLASSPATH%;%LIB_HOME%/agbase.jar
set CLASSPATH=%CLASSPATH%;%LIB_HOME%/wlsauth.jar
set CLASSPATH=%CLASSPATH%;%LIB_HOME%/crypto.jar
set CLASSPATH=%CLASSPATH%;%LIB_HOME%/xercesImpl.jar
set CLASSPATH=%CLASSPATH%;%LIB_HOME%/jdom.jar;
set CLASSPATH=%CLASSPATH%;%LIB_HOME%/log4j.jar;
set CLASSPATH=%CLASSPATH%;%LIB_HOME%/jobaccess.jar;
set CLASSPATH=%CLASSPATH%;%LIB_HOME%/colt.jar
set CLASSPATH=%CLASSPATH%;%LIB_HOME%/jms.jar
set CLASSPATH=%CLASSPATH%;%LIB_HOME%/jndi.jar
set CLASSPATH=%CLASSPATH%;%LIB_HOME%/tibjms.jar
set CLASSPATH=%CLASSPATH%;%LIB_HOME%/oc4jclient.jar
set CLASSPATH=%CLASSPATH%;%LIB_HOME%/oc4j.jar
set CLASSPATH=%CLASSPATH%;../ldaplib/ldaputil.jar
set CLASSPATH=%CLASSPATH%;D:\Agile931b28/agileDomain/config
set CLASSPATH=%CLASSPATH%;agile_home/pxConfig
```
4. ドメイン内の管理サーバーとすべての管理対象サーバーを起動します。

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 サーバーと 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・サーバー上にあるため、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・サーバーの 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・サーバーに配置した後は、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・サーバー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・サーバー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ー側 API である ICustomAutoNumber インタ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てののですか。

「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てすることもできます。

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ー側 API である ICustomAction インタ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、¥oas¥j2ee¥home¥applications¥Agile¥APP-INF¥classes¥ディレクトリに配置できます。

Web サービス拡張の開発

この章のトピック

| | |
|--|-----|
| ■ Webサービス拡張について | 373 |
| ■ Webサービスについて | 374 |
| ■ Webサービスの開発および配置 | 377 |
| ■ Webサービスの使用 | 379 |
| ■ ユーザーの認証 | 379 |
| ■ クライアント/サーバー・アクセスでのシングル・サインオン・クッキーの使用 | 380 |
| ■ MyFirstWebServiceサンプルの環境の準備 | 382 |
| ■ MyFirstWebServiceサンプルの作成 | 383 |
| ■ Webサービス・クライアントについて | 384 |
| ■ MyFirstClientの作成 | 385 |
| ■ Microsoft .NETの相互運用性 | 388 |
| ■ Webサービス拡張に関するよくある質問 | 388 |

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の簡素化もできます。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・サーバーのデータを、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、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 は、単独のライセンス製品です。AIS の詳細は、『Agile Integration Services Developer Guide』を参照してください。

主な機能

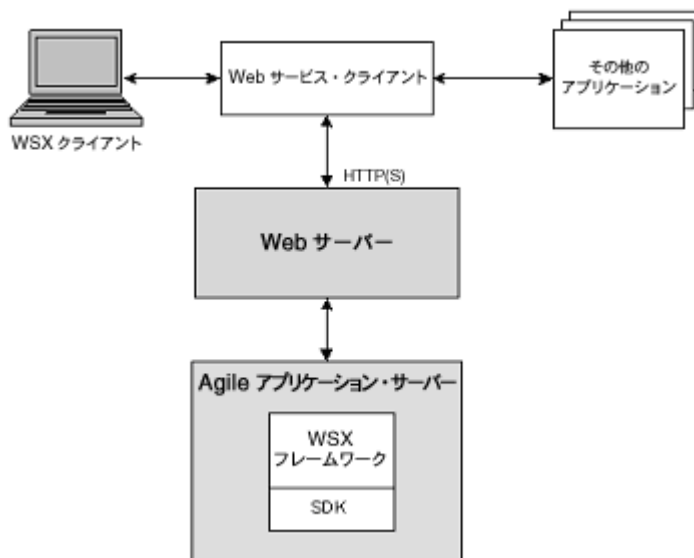
WSX の主な機能は、次のとおりです。

- **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 サービスの開発もできます。
- **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23: 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 サービスで提供される主なメリットは、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には、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・コレクション、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 サーバーを介した) 通信は、セキュア・ソケット・レイヤー (SSL) とサーバー側の証明書を介して暗号化できるため、認証、プライバシー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 (<http://xml.apache.org/axis>) を参照してください。

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://forums.sun.com/thread.jspa?threadID=5020700>
- **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・ライブラリも活用できます。
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・サーバー・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・サーバーがある場合は、クラスタ内の各サーバーに、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・サーバーによって、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://ws.apache.org/axis/java/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://ws.apache.org/axis/java/reference.html>
「Deployment (WSDD) Reference」のセクションを参照してください。

注意 これらのサイトは定期的に変更されることがあります。その場合は、検索エンジンを使用してこれらのドキュメントを検索してください。

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://ws.apache.org/axis/java/apiDocs/org/apache/axis/encoding/Serializer.html> および <http://ws.apache.org/axis/java/apiDocs/org/apache/axis/encoding/Deserializer.html> を参照してください。これらのサイトは定期的に変更されます。その場合は検索エンジンを起動して、これらのインタフェースに関する最新情報を検索してください。

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・サーバー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、次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/サーバー・アクセスでのシングル・サインオン・クッキーの使用

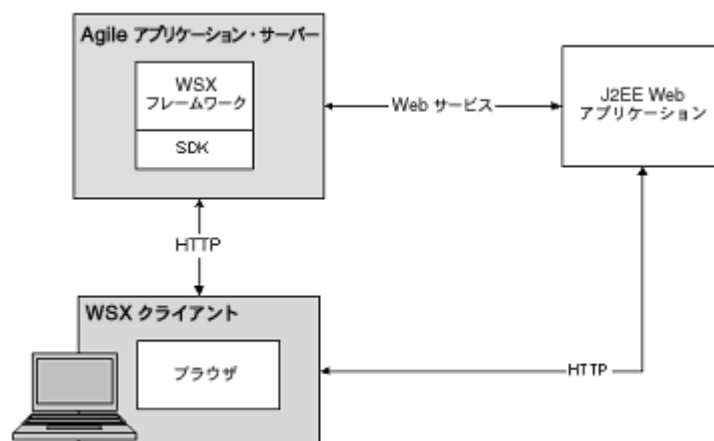
WSX クライアントのユーザーが、サード・パーティのシングル・サインオン製品で保護されている Agile 9.X サーバーで認証されると、ブラウザにはシングル・サインオン・クッキーが割り当てられます。このクッキーは、カスタム j2ee Web アプリケーションが Agile 9.X サーバーと同じ DNS ドメインにある場合、そのアプリケーションに送信されます。これで、Agile 9.X サーバーに配置されている Web サービスを起動する場合は、ユーザー名とパスワードのかわりにシングル・サインオン・クッキーを有効なアカウント情報として渡すことができます。

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 サーバーと 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24: 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` オブジェクト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. 次の行を SOAP スタブ・クラスに追加します。

```
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:\j2sdk150）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_5_0-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>/j2sdk150）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ー上で、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
```

サーバー・プラットフォームでWSXサンプルを作成する手順は、次のとおりです。

1. SDK_samples (ZIP) ファイルをコピーします。このファイルにアクセスする方法は、2ページの「[クライアント側のコンポーネント](#)」の注意を参照してください。
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。その場合は、次のようにします。

3. サーバーの\$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、サーバーのagile.propertiesで設定できます。<http://hostname:port/virtualPath/services/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/services -
Dusername=<username> -Dpassword=<password>
 - **Solaris/Linux の場合** - \$ANT_HOME/bin/ant -
Dwsx.url=http://webserver/virtualPath/services -
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 (<http://ws.apache.org/axis/>)
- **Java Web Services Developer Pack (JWS DP)** - Sun社のSOAPプロトコルのJava実装 (<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 (<http://msdn.microsoft.com/net>)
- **SOAP::Lite for Perl** - SOAPプロトコルのPerl実装 (<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-wsd12java output="${built.dir}/src"
    url="${wsx.url}/MyFirstWebService?wsdl">
    <mapping namespace="http://www.agile.com/ws/SampleWsx"
      package="client"/>
  </axis-wsd12java>
</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ーに示して、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 箇所では 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 サーバーでは、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: 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);
```

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 ファミリー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・コレクション) 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・サーバー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 サーバー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。

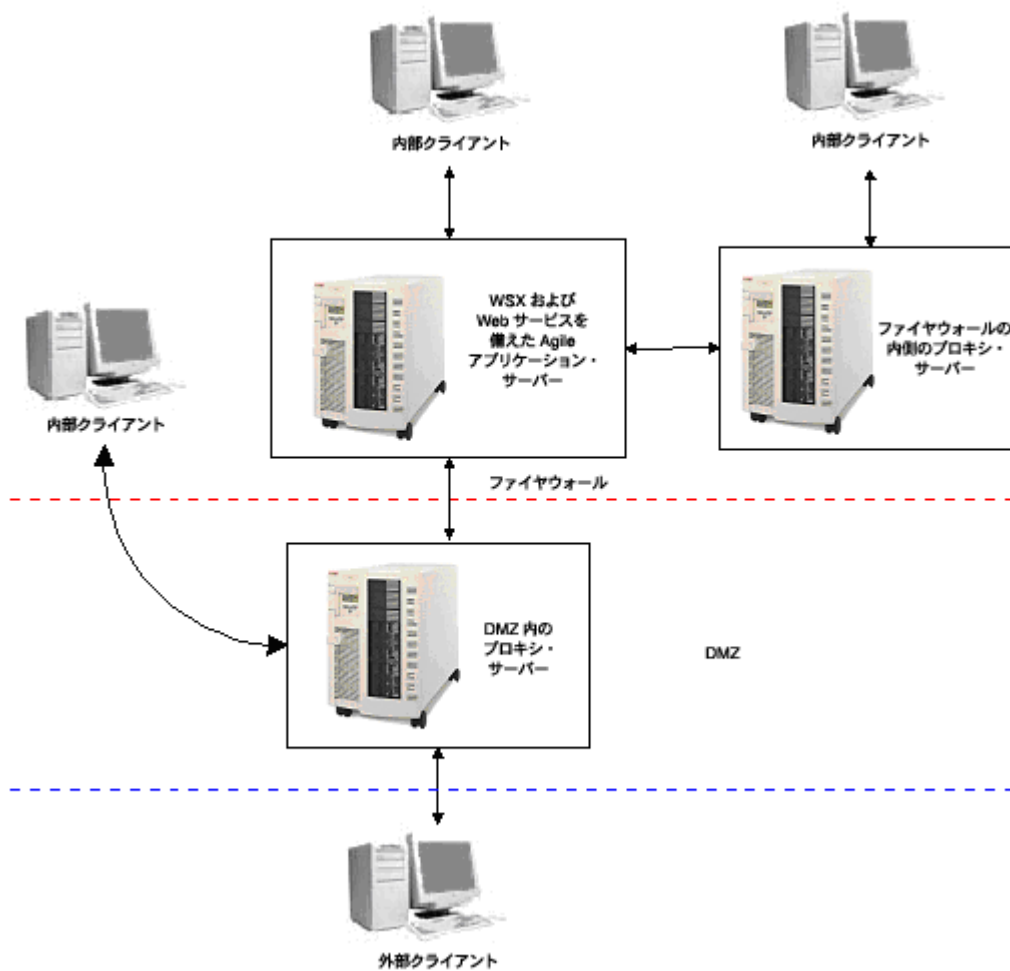
Agile の Java API のみでは不可能で、WSX では可能な操作とは、どのようなものですか。

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・サーバー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・サーバー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・サーバーに対して実行された後、アプリケーション・サーバーに転送されます。プロキシ・サーバーは 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・サーバー、DMZ に設置されていない別のプロキシ・サーバー、または直接アプリケーション・サーバーに対して実行できます。

図25: Web サービス・クライアントが Agile PLM サーバーに接続する方法



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・サーバー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、379 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 添付ファイル进行处理しますか。

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ー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。<http://ws.apache.org/axis/faq.html>の「Axis FAQ」から参照を開始できます。

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ー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・サーバー・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립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・デスク립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・デスク립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、<http://ws.apache.org/axis/>にある『Axis Reference Guide』を参照してください。

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・サーバー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・サーバー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・サーバー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を確認されます。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・サーバーを再起動せずに 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 サーバー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>

ダッシュボード管理拡張の開発

この章のトピック

| | |
|----------------------------------|-----|
| ■ ダッシュボード管理拡張について | 393 |
| ■ カスタム・チャート・ダッシュボード管理拡張の開発 | 394 |
| ■ カスタム・テーブル・ダッシュボード管理拡張の開発 | 398 |
| ■ カスタム (URL) 拡張の定義 | 403 |

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 サーバー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 サーバーが、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の取得もできます。

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「ダッシュボード」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』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: 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 displays a comparison chart between
 * Morning and Evening Temperatures for each day of the
 * week with 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・サーバー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・サーバーがある場合は、クラスタ内の各サーバー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、Java クライアントに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 |
| API 名 | 入力した「名前」フィールドが AgilePLM によってキャメルケース命名規則に変換されます。 | 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」 |

| ダッシュボード・テーブル | 説明 | 可能な設定 |
|--------------|---------------------|---------|
| 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) |

- 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 のサーバー 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 Type</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
        // are set
        HashMap hmlImage = new HashMap();
        hmlImage.put(CustomTableConstants.URL,baseUrl+"/images/action_noshad.gif");
        // Tool Tip hmlImage.put(CustomTableConstants.LABEL,"Action_Noshad");
        row1.put("myImage",hmlImage);
        // For Money Type. The Currency and value properties are set
        HashMap hmlMoney = new HashMap();
        hmlMoney.put(CustomTableConstants.MONEY_CURRENCY_CODE,"USD");
        hmlMoney.put(CustomTableConstants.MONEY_VALUE,new Integer(3000));
        row1.put("myMoney",hmlMoney);
        // 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 Tippam");

// 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/services/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・サーバー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・サーバーがある場合は、クラスタ内の各サーバー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 の追加もできます。

タブにテーブルを追加する手順は、次のとおりです。

1. 新しいタブを定義します。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、**Tables** 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、必要な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。

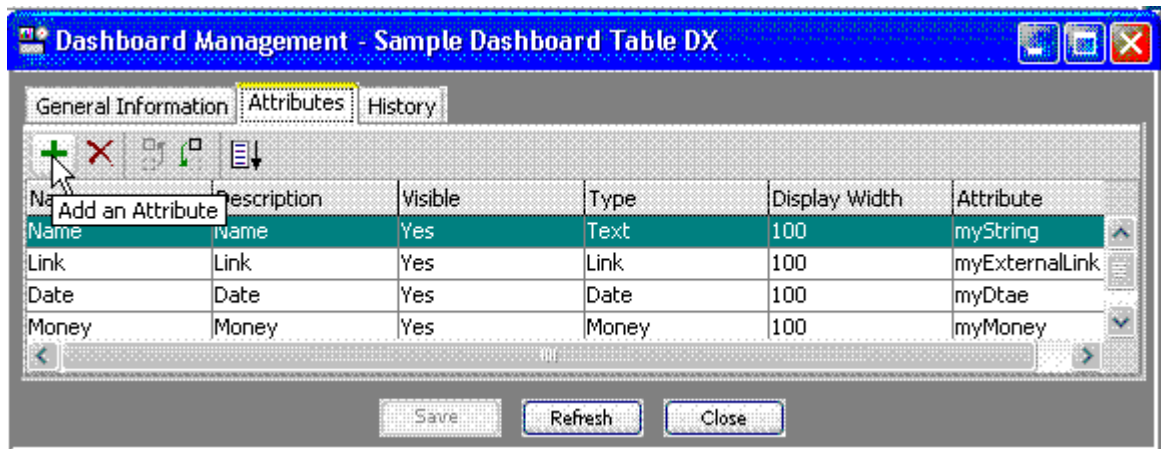
| ダッシュボード・テーブル | 説明 | 可能な設定 |
|--------------|---|--|
| 名前 | テーブルの名前を入力します。 | 文字列 |
| API 名 | 前に入力した「名前」フィールドが AgilePLM によってキャメルケース命名規則に変換されます。 | 文字列 |
| 説明 | テーブルの説明を入力します。 | 文字列 |
| リスト・タイプの表示 | テーブルのタイプがリストされます。「 テーブル 」を選択します。 | 「チャート」、「テーブル」、「カスタム」、「詳細検索」 |
| ダッシュボード拡張機能 | テーブル・タイプの表示用に作成されたすべてのプロセス拡張がリストされます。 | これらは、355ページの「 カスタム・アクションのパッケージ化および配置 」で定義した属性です。 |
| 変数 | Web クライアントで使え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. 402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、401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。

図26: サンプル・ダッシュボード・テーブル DX



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』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。「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。

Tables

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 クライアントで使えるようにするかどうかを選択します。 | 「はい」または「いいえ」 |

Agile PLM のイベントおよびイベント・フレームワーク

この章のトピック

- Agile PLMのイベントおよびイベント・フレームワークの理解 405
- Agile PLMイベントの主要なコンポーネント 405
- イベントの生成、処理および実行 409

Agile PLMのイベントおよびイベント・フレームワークの理解

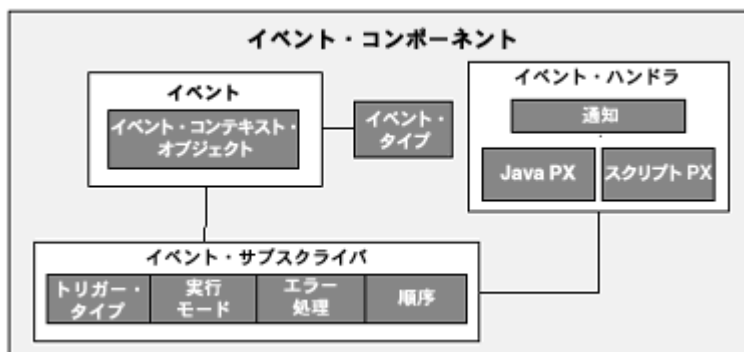
Agile PLM のイベントは、PLM アプリケーション内で自動化アクションを生成するためのトリガー・ポイントとして機能します。すべてのイベントは、Agile PLM アプリケーション内でソースから生成されます。ソースは、ユーザー、UI アクションまたはシステムで開始されたソース（タイマーなど）によってトリガーされるビジネス・アクションです。これらのソースは、何か（イベント）がアプリケーション内で発生し、それがアクションを必要とする可能性があることを他の PLM モジュールに合図します。必要なアクションは、ユーザーまたは PLM モジュールによって実行されます。イベント駆動型アプリケーションによって、イベント・ベースの統合の管理や、リアル・タイム・モードでの複雑なイベント分析が大幅に容易になります。

Agile SDK 環境では、イベント・フレームワークによって PX フレームワークが拡張されるため、Agile PLM アプリケーションの自動化と拡張が容易になりました。イベント・フレームワークは、イベント駆動型アプリケーションの迅速な開発と配置のための柔軟な環境を提供します。この環境をサポートするために、イベント・フレームワークには、様々なタイプの Agile PLM イベントを作成、設定および実行するための包括的なパラメータが用意されています。

Agile PLMイベントの主要なコンポーネント

イベント・フレームワークは、ユーザーによるイベントおよびイベント確認通知受信者の設定をサポートします。次にイベント・フレームワークの基本コンポーネントを示し、後続の段落で説明します。

図27: イベントのコンポーネント



イベント・タイプ

Agile PLM でのイベント・タイプは、「オブジェクトの作成」、「オブジェクトの削除」、「ワークフローの検証」などの特定のアクションを指します。Agile PLM には、イベントが発生する可能性のある、事前定義のイベント・タイプのリストが用意されています。

各イベント・タイプには、そのハンドラ・タイプに従って、それぞれのイベント・コンテキスト・オブジェクトに対応する Java およびスクリプト・インタフェースがあります。たとえば、「オブジェクトの作成」には、`ICreateEventInfo` Java インタフェースと `ICreateScriptObj` スクリプト・インタフェースがそれぞれ用意されています。

イベントを作成するときは、Java クライアントの「イベントの作成」ダイアログのドロップダウン・リストからイベント・タイプを選択します。このダイアログにアクセスする方法は、『Agile PLM 管理者ガイド』を参照してください。次の図に、デフォルトのイベント・タイプとその説明を示します。

図28: イベント・タイプとその説明

| Name | Description |
|--|---|
| Approve for Workflow | Workflow, Approve |
| Audit for Workflow | Workflow, Audit |
| Cancel Check Out Files | Files, Cancel Check Out |
| Change Approvers or Observers for Workflow | Workflow, Change Approvers or Observers |
| Change Status for Sourcing Object | Sourcing Object, Change Status |
| Change Status for Workflow | Workflow, Change Status |
| Check In Files | Files, Check In |
| Check Out Files | Files, Check Out |
| Comment for Workflow | Workflow, Comment |
| Compliance Rollup on Object | Object, Compliance Rollup |
| Create Object | Object, Create |
| Delete Object | Object, Delete |
| Escalation for Workflow | Workflow, Escalation |
| Export Object | Object, Export |
| Extend Actions Menu | Actions Menu, Extend |
| Extend Tools Menu | Tools Menu, Extend |
| Get File | File, Get |
| Incorporate Item | Item, Incorporate |
| Promotion Failure for Workflow | Workflow, Promotion Failure |
| Purge Version Files | Files, Purge Version |
| Reject for Workflow | Workflow, Reject |
| Reminder for Workflow | Workflow, Reminder |
| Save As Object | Object, Save As |
| Scheduled Event | Event, Scheduled |
| Transfer Authority | Authority, Transfer |
| Unincorporate Item | Item, Unincorporate |
| Update Relationship | Relationship, Update |
| Update Table | Table, Update |
| Update Title Block | Title Block, Update |

注意 イベント・タイプは新規に作成することも、削除することもできません。

イベント・ハンドラおよびハンドラ・タイプ

イベント・ハンドラは、イベントが発生すると呼び出されるカスタム・アクションを表します。イベント・ハンドラは、イベントのトリガー時にユーザー、インタフェースまたはシステムによって実行されるアクションの機能を拡張します。イベントに関する情報は、イベントからハンドラにイベント・コンテキスト・オブジェクトによって渡されます。ハンドラはイベント・トリガーによって起動されます。

SDK およびイベント・フレームワークは、次のイベント・ハンドラ・タイプをサポートしています。

- **Java PX** - `com.agile.px` パッケージの `IEventAction` インタフェースを実装し、コンパイルされた Java コードをトリガーする Java プロセス拡張です。「イベント情報オブジェクト - Java PX」を参照してください。
- **スクリプト PX - Groovy** スクリプト言語に基づいたスクリプト・プロセス拡張です。
Groovy スクリプト・コードは、Agile PLM データベースに直接格納されます。開発するイベント・スクリプト・オブジェクト（イベント・スクリプト PX ハンドラ）はテキスト・ファイルです。このファイルは、Java クライアントのイベント管理ノードを使用して Agile PLM サーバーに配置されます。テキスト・ファイルの送信の詳細は、438 ページの「[Agile PLM 管理者との作業](#)」を参照してください。イベント・フレームワークおよびイベント・スクリプト PX 開発プロセスでの Groovy の実装の詳細は、469 ページの「[フレームワーク環境での Groovy の実装](#)」を参照してください。
- **通知** - Agile PLM には、ユーザーがアクションを実行する必要がある場合、または発生したアクションについてユーザーに通知する必要がある場合に、通知をユーザーに送信する機能があります。通知は SDK およびスクリプトからトリガーされます。イベント通知の詳細は、『Agile PLM 管理者ガイド』に記載されています。SDK を使用して通知をプログラムで送信する方法は、140 ページの「[Agile SDK を使用した通知の送信](#)」を参照してください。

注意 SDK 開発者は、Java PX およびスクリプト PX を使用して、Agile PLM の機能を拡張します。通知テンプレートは、Agile PLM 管理者がアクションや情報に関する通知を定義および設定するために使用します。

イベント確認通知受信者

イベント確認通知受信者は、イベント・ハンドラを特定のイベントにリンクします。したがって、特定のイベントが発生すると、イベントに関連付けられているイベント・ハンドラが、イベント確認通知受信者を介して、指定された順に従って開始されます。ハンドラ・アクションには、Java プロセス拡張の起動、スクリプト・プロセス拡張の起動または通知の送信が含まれます。

イベント・トリガーおよびトリガー・タイプ

イベント・トリガーは、イベント拡張の起動時期を判断するために使用されます。イベント・フレームワークには、問題が発生したときのアクションを自動化するためのフックが用意されています。たとえば、ステータスの変更には CEO の承認が必要とします。この場合、イベント・トリガーはアクションの発生を合図します。このアクションは、続いて該当するイベントのすべてのイベント確認通知受信者に通知します。ほとんどの Agile PLM イベントは、ビジネス・アクションに関連しています。次に例を示します。

- 「**オブジェクトの作成**」 イベント - オブジェクトの作成アクションに関連しています。
- 「**ワークフローのステータスの変更**」 イベント - ワークフロー・ステータス変更アクションに関連しています。

イベント・トリガー・タイプ

- **事前** - このトリガー・タイプは、アクション発生前の時点を合図します。一般的に「事前」トリガーは、データの検証が必要なイベントや、次のアクションに他のパラメータが必要なイベントに使用されます。イベント・ハンドラは、「事前」トリガー・ポイントから同期して起動されます。
- **事後** - このトリガー・タイプは、アクションによる変更がデータベースに対してコミットされた直後の時点を合図します。一般的に「事後」トリガーは、完了したアクションに関連する追加のタスク、通知および外部システムとの統合タスクを実行するイベントに使用されます。イベント・ハンドラは、「事後」トリガー・ポイントから同期または非同期で起動されます。

同期および非同期の実行モード

イベント・ハンドラは、同期または非同期で起動されます。通常、同期とは、現在のアクションでアプリケーションの一部として実行されることを意味します。同期操作は操作が完了するまでプロセスをブロックし、非同期動作はプロセスをブロックせず単に操作を開始します。

Agile PLM での 2 つのモードの相違は、次のとおりです。

- **同期** - このモードでは、イベント・ハンドラは、イベント（たとえば、ワークフロー・ステータスの変更）をトリガーする Agile PLM スレッドと同じスレッドで（実行アクションの一部として）実行されます。元の Agile PLM アクションは、ハンドラ・アクションの完了後に再開されます。
- **非同期** - このモードでは、イベント・ハンドラにそれ自体の（実行アクションとは別に実行される）スレッドがあり、1 度開始すると停止できません。このトランザクションは、それ自体のステータスに基づいてコミットまたはロールバックされます。イベントをトリガーする Agile PLM スレッドは、ハンドラ・アクションが完了しているかどうかに関係なく（ブロックされずに）、引き続き単独で実行されます。通知は、常に非同期で処理されます。

イベント・エラー処理ルール

イベント・エラー処理ルールは、同期して実行するイベント・ハンドラで使用されます。「続行」と「停止」のオプションがあります。選択したオプションによって、イベント・ハンドラの実行中にエラーが発生した場合の Agile PLM の動作が決まります。エラー処理ルールの詳細は、『Agile PLM 管理者ガイド』を参照してください。

- **続行** - 同期実行モードでのハンドラの実行中にエラーが発生した場合、エラーは無視されます。
- **停止** - 同期実行モードでのハンドラの実行中にエラーが発生した場合、元のアクションおよびイベント確認通知は中止されます。

イベントの順序

イベントの順序は、イベント・ハンドラが起動される順序を決定する正の整数です。順序を使用すると、同一の Agile オブジェクトの同じイベント・タイプに複数のイベント確認通知受信者がある場合に、イベント・ハンドラの実行順序を制御できます。

注意 カスタム PX と Java 同期 PX の両方を 1 つのワークフロー・ステータス変更アクションに設定した場合は、常に Java PX がカスタム PX の前に実行されます。

イベントの生成、処理および実行

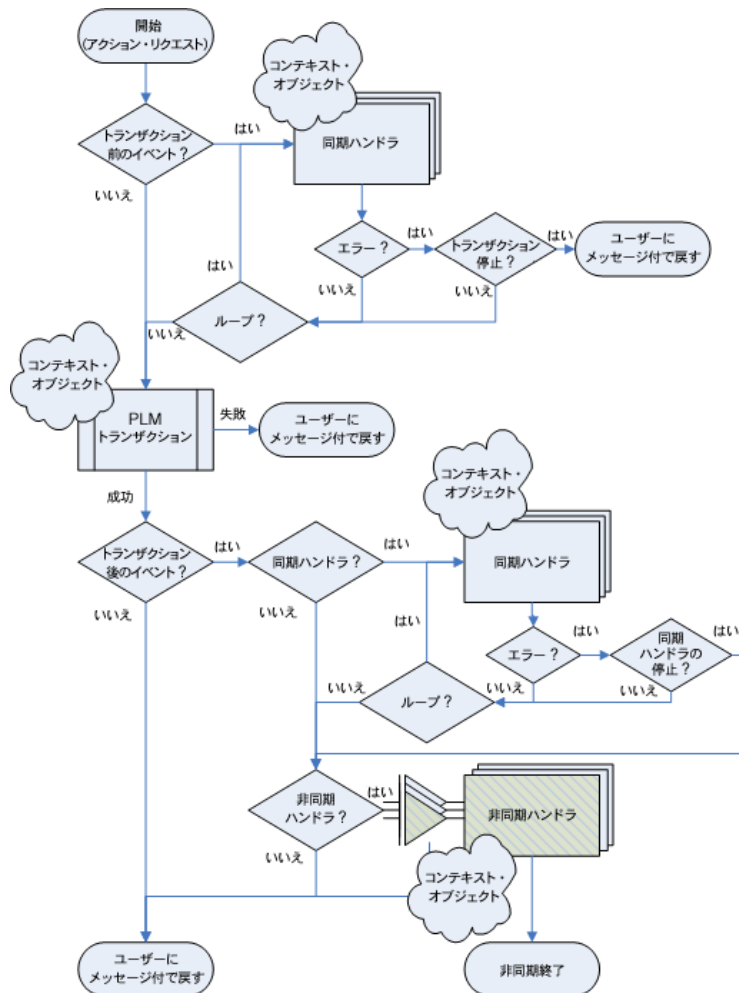
次の図は、PLM クライアントからイベントがトリガーされる時点から、イベントが終了するまでの Agile PLM トランザクションの一般的なフローを示しています。異なる条件に対するアクションと、フロー終了時のアクションが要約されています。

クライアントによってアクションが実行されると、Agile PLM は、要求されたアクションについて、確認通知受信者を介して事前イベントが設定されているかどうかをチェックします。確認通知受信者がある場合、Agile PLM は、その確認通知受信者を介して設定されたイベント・ハンドラを適切な順序で起動します。ハンドラがそれぞれのタスクを完了した後、Agile PLM は、確認通知受信者エラー処理ルールの設定を使用して、エラーがあった場合に停止するか続行するかを判断します。Agile PLM は、ハンドラがそれぞれのタスクをすべて完了するまで、残りの次のイベント・ハンドラを起動します。コンテキスト・オブジェクトは、各イベント・ハンドラ内で実行された更新内容を保持します。すべての事前イベント・ハンドラが完了すると、Agile PLM は、更新内容を保持しているコンテキスト・オブジェクトを使用して、ユーザーが要求した元のアクションを開始します。

元のアクションが実行された後は、同様のプロセスによって、同期モードのすべての事後イベント・ハンドラが起動されます。コンテキスト・オブジェクト情報は、このプロセスの最初から最後まで保持されます。同期モードの事後イベント・ハンドラでは、元のアクションはロールバックできません（すでに実行されているため）。同期モードの事後イベント・ハンドラがすべて正常に完了すると、Agile PLM は、非同期モードの事後イベント・ハンドラ（ある場合）を、定義された順序がない独立したプロセスで開始します。

同期モードの事前イベント・ハンドラの実行時にエラーが発生したときに、エラー処理ルールが「停止」の場合、Agile PLM はエラー・メッセージをユーザーに返します。

図29: Agile PLM イベントのトリガーおよび処理



イベント・コンテキスト・オブジェクトの使用

この章のトピック

- イベント・コンテキスト・オブジェクトの理解..... 411
- イベント情報オブジェクトおよびイベント・スクリプト・オブジェクトの使用..... 415
- イベントに関するよくある質問..... 442

イベント・コンテキスト・オブジェクトの理解

コンテキスト・オブジェクトは、情報をイベントからハンドラに渡し、ハンドラ間で渡します。イベントが発生すると、イベント・コンテキスト・オブジェクトが作成されます。コンテキスト・オブジェクトが保持する情報には、イベント・タイプ、事前イベントと事後イベントのトリガー、およびイベントが発生した Agile オブジェクトなど、特定のイベントに関するビジネス関連データが含まれます。ビジネス関連データは、イベント・タイプによって様々です。

異なるイベント・タイプごとに異なるコンテキスト・オブジェクトがあります。コンテキスト・オブジェクトのインタフェースについては、412ページの「[イベント情報オブジェクト](#)」および414ページの「[イベント・スクリプト・オブジェクト](#)」を参照してください。これらのインタフェースは、Javadoc HTMLファイルのイベント情報オブジェクト（Javaインタフェース）およびイベント・スクリプト・オブジェクト（スクリプト・インタフェース）に記載されています。Javadoc HTMLファイルは、Oracle® E-Delivery Webサイト (<http://edelivery.oracle.com/>) のOracle Agile Product Lifecycle Managementメディア・パック内のSDK_samples.zipフォルダにあります。

永続データと一時データ

イベント・フレームワークでは、コンテキスト・オブジェクトによってイベント・ハンドラ（Java PX またはスクリプト PX）に渡されるオブジェクトに、永続データと一時データがあります。

- **永続データ** - PLM データベースにすでに存在しているデータ。このタイプのデータは、すべての Agile SDK API、および Web 拡張や URL 拡張も含めたプロセス拡張によって処理されます。getDataObject() を使用して IDataObject オブジェクトの値を取得するとき、取得するデータはデータベースにすでに存在しています。したがってそのデータは永続的です。

注意 永続データを返すメソッドは getDataObject() のみです。他のコンテキスト・オブジェクト・データ取得メソッドは、Javadoc API 定義に記述されていないかぎり、一時データを返します。

- **一時データ** - PLM データベースにまだ存在していない、変化するデータ。一時データには、イベントをトリガーしたアクションに対するユーザー要求に関する情報が含まれています。

イベントがトリガーされると、Agile PLM では、イベント・コンテキスト・オブジェクトに一時データが作成されます。この同じ一時データは、Javadoc API 定義で指定されていないかぎり、コンテキスト・オブジェクトによって事前および事後のすべてのハンドラに渡されます。

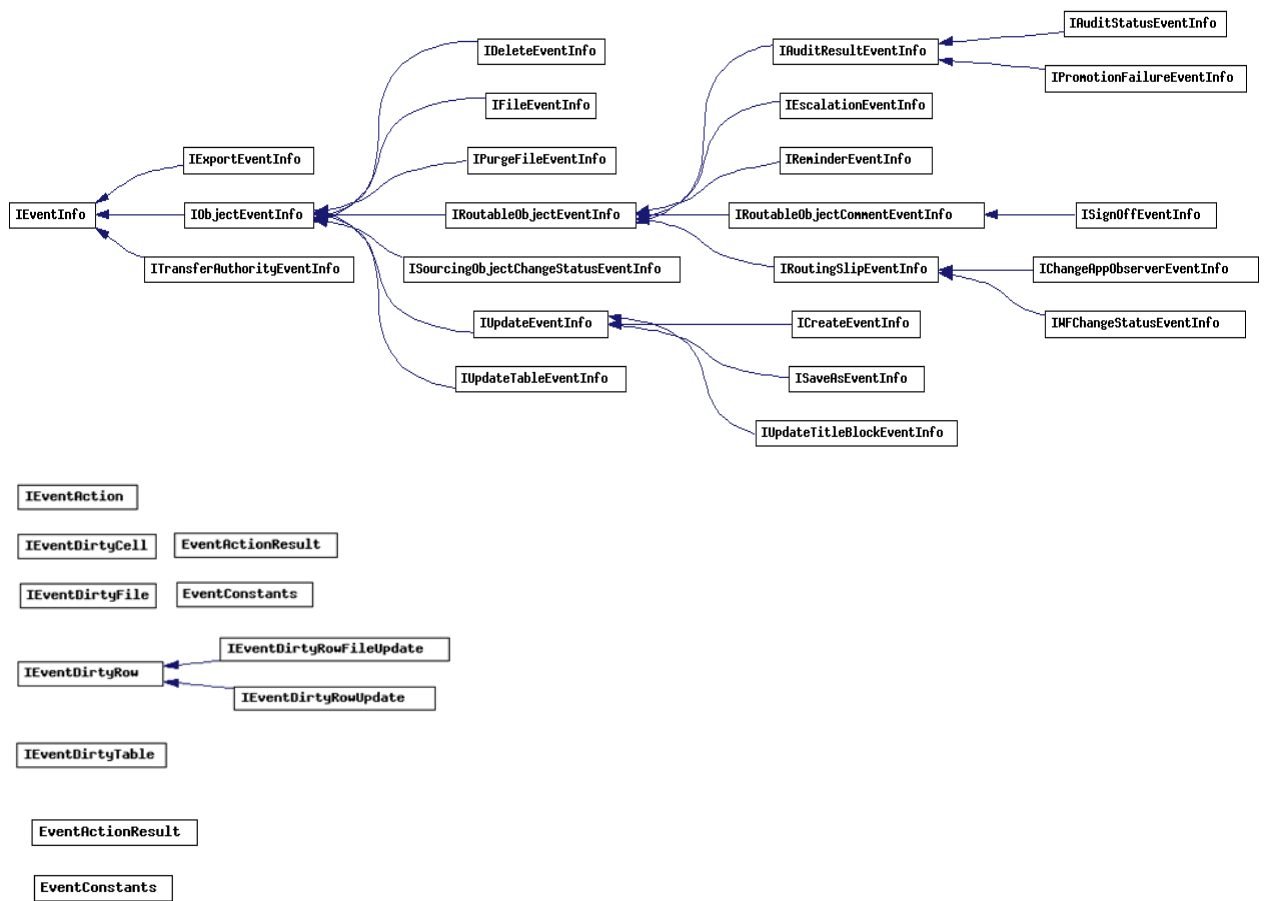
注意 事後イベント・トリガー・インスタンス内の一時データは変更しないでください。変更した場合は、例外が発生します。

イベント情報オブジェクト

次に、Java PX のインタフェースを示します。

| イベント・タイプ | イベント情報オブジェクト |
|--|--------------------------------------|
| ワークフローの承認およびワークフローの却下 | ISignOffEventInfo |
| ワークフローの検証 | IAuditStatusEventInfo |
| ワークフローの承認者またはオブザーバの変更 | IChangeAppObserverEventInfo |
| ソーシング・オブジェクトのステータスの変更 | ISourcingObjectChangeStatusEventInfo |
| ワークフローのステータスの変更 | IWFChangeStatusEventInfo |
| ワークフローのコメント | IEscalationEventInfo |
| オブジェクトの作成 | IRoutableObjectComment12:30EventInfo |
| オブジェクトの削除 | IDeleteEventInfo |
| ワークフローのエスカレーション | IEscalationEventInfo |
| オブジェクトのエクスポート | IExportEventInfo |
| ツール・メニューの拡張、スケジュール済イベント、およびすべてのイベント情報オブジェクトの基本インタフェース | IEventInfo |
| ファイル取出し、ファイルのチェックイン、ファイルのチェックアウトおよびファイルのチェックアウトのキャンセル | IFileEventInfo |
| 確定(アイテム)、未確定(アイテム)、「アクション」メニューの拡張およびオブジェクトについての適合性ロールアップ | IObjectEventInfo |
| ワークフローの昇格失敗 | IPromotionFailureEventInfo |
| バージョン・ファイルのページ | IPurgeFileEventInfo |
| ワークフローの督促 | IReminderEventInfo |
| オブジェクトに名前を付けて保存 | ISaveAsEventInfo |
| 権限委譲 | ITransferAuthorityEventInfo |
| テーブルの更新および関係の更新 | IUpdateTableEventInfo |
| タイトル・ブロックの更新 | IUpdateTitleBlockEventInfo |

図30: イベント情報オブジェクトのクラス階層

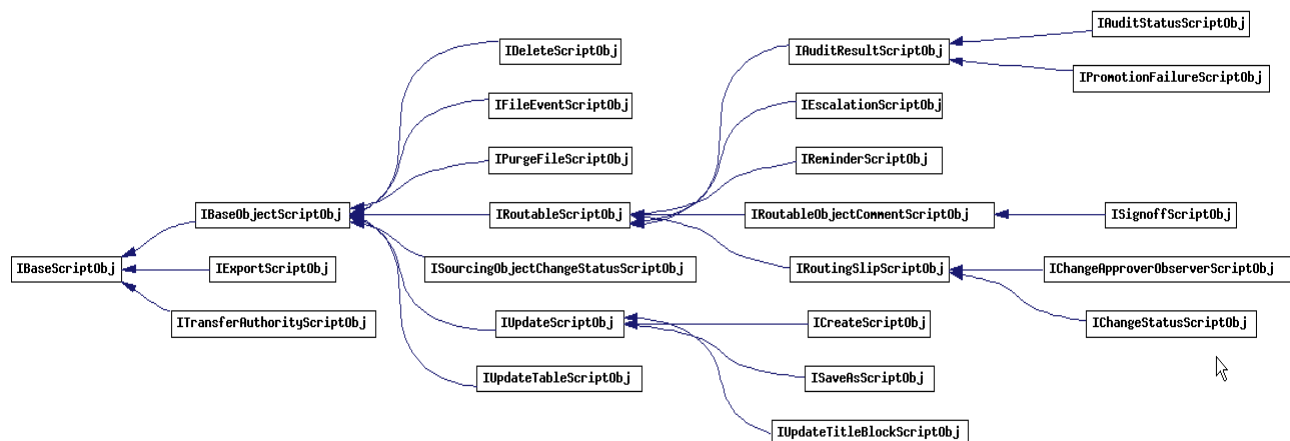


イベント・スクリプト・オブジェクト

次に、スクリプト PX のインタフェースを示します。

| イベント・タイプ | スクリプト・オブジェクト・インタフェース |
|--|--------------------------------------|
| ワークフローの承認およびワークフローの却下 | ISignoffScriptObj |
| ワークフローの検証 | IAuditStatusScriptObj |
| ワークフローの承認者またはオブザーバの変更 | IChangeApproverObserverScriptObj |
| ソーシング・オブジェクトのステータスの変更 | ISourcingObjectChangeStatusScriptObj |
| ワークフローのステータスの変更 | IChangeStatusScriptObj |
| ワークフローのコメント | IRoutableObjectCommentScriptObj |
| オブジェクトの作成 | ICreateScriptObj |
| オブジェクトの削除 | IDeleteScriptObj |
| ワークフローのエスカレーション | IEscalationScriptObj |
| オブジェクトのエクスポート | IExportScriptObj |
| ツール・メニューの拡張、スケジュール済イベント・イベント・スクリプト・オブジェクト、およびすべてのイベント・スクリプト・オブジェクトの基本インタフェース | IBaseScriptObj |
| ファイル取出し、ファイルのチェックイン、ファイルのチェックアウトおよびファイルのチェックアウトのキャンセル | IFileEventScriptObj |
| 確定(アイテム)、未確定(アイテム)、「アクション」メニューの拡張およびオブジェクトについての適合性ロールアップ | IBaseObjectScriptObj |
| ワークフローの昇格失敗 | IPromotionFailureScriptObj |
| バージョン・ファイル・フォルダのパージ | IPurgeFileScriptObj |
| ワークフローの督促 | IReminderScriptObj |
| オブジェクトに名前を付けて保存 | ISaveAsScriptObj |
| 権限委譲 | ITransferAuthorityScriptObj |
| テーブルの更新および関係の更新 | IUpdateTableScriptObj |
| タイトル・ブロックの更新 | IUpdateTitleBlockScriptObj |

図31: イベント・スクリプト・オブジェクトのクラス階層



注意 スクリプト PX では、すべての SDK または Java コールを起動できます。また、サード・パーティの Java または Groovy ライブラリが SDK 拡張ディレクトリに配置されている場合は、そのライブラリにもアクセスできます。

イベント情報オブジェクトおよびイベント・スクリプト・オブジェクトの使用

このセクションでは、イベント情報オブジェクトおよびイベント・スクリプト・オブジェクトを使用して、イベント関連アクションに対するPXハンドラを開発する方法を説明します。イベント関連アクションには、たとえば、「オブジェクトの作成」や「オブジェクトの削除」などの一般的なオブジェクト・アクション、「ワークフローのステータスの変更」などのワークフロー・アクションがあります。

次に、イベント情報オブジェクトおよびイベント・スクリプト・オブジェクトの説明とサンプルを示します。その他に、これらのオブジェクトを使用したコード・サンプル、Agile PLM でイベントの特別なインスタンスを正しく処理するためのガイドラインについても説明します。

基本イベント・アクションの使用

基本イベント情報オブジェクトは、他のすべてのイベント情報オブジェクトで使用される **Java** インタフェースです。イベント情報オブジェクトのクラス階層の図を参照してください。

基本イベント情報オブジェクト - Java PX

基本イベント情報オブジェクトは、IEventInfoです。

- 目的と機能 - 「ツール・メニューの拡張」および「スケジュール済イベント」イベント情報オブジェクトのインタフェース。また、IEventInfoは、すべてのイベント情報オブジェクトの継承インタフェースです。イベント・タイプ、イベント・トリガー・タイプ、イベント名、イベント確認通知受信者名、イベント・ハンドラ名およびユーザー定義マップに関する情報を保持します。

ユーザー定義マップは、すべてのユーザー定義データのプレースホルダとして機能し、確認通知受信者間の通信チャネルを提供します。これらのユーザー定義マップは、Synchronous Java PX で設定され、後続の Synchronous Java PX および Asynchronous Java PX によって読み取られます。Asynchronous PX 内で設定されたマップは、他の Java PX では使用できません。Java PX が失敗した場合でも、マップは Agile PLM によって受け入れられて次

の Java PX に渡されますが、イベント・コンテキスト・オブジェクトの他の変更はすべて破棄されます。

IObjectEventInfo は、オブジェクト関連イベントの基本イベント情報オブジェクトです。

- 目的と機能 - IObjectEventInfo には、イベントがトリガーされた Agile オブジェクトが含まれています。また、「確定(アイテム)」、「未確定(アイテム)」、「「アクション」メニューの拡張」および「オブジェクトについての適合性ロールアップ」イベント情報オブジェクトのインタフェースであり、IEventInfo から継承します。
- 継承インタフェース: IEventInfo。

次の例は、IEventInfo および IObjectEventInfo の使用方法を示しています。

例: IEventInfo の使用方法

```
private void testIEventInfo(IEventInfo req)throws APIException{
    // getEventType()
    int evttype = req.getEventType();

    // getEventTriggerType()
    int evtTriggerType = req.getEventTriggerType();

    // getEventName()
    String eventName = req.getEventName();

    // getEventSubscriberName()
    String subscriberName = req.getEventSubscriberName();
    // getEventHandlerName()
    String handlerName = req.getEventHandlerName();
    // setUserDefinedMap()
    Map map = new HashMap();
    map.put("METHOD", "setUserDefinedMap()");
    req.setUserDefinedMap(map);

    // getUserDefinedMap()
    Map map2 = new HashMap();
    map2 = req.getUserDefinedMap();
    String mapValue = map2.get("METHOD").toString();
}
```

例: IObjectEventInfo によるコール

```
private void testIObjectEventInfo(IEventInfo req)throws APIException{
    String objNumber = "";
    IObjectEventInfo info = (IObjectEventInfo)req;

    // getDataObject()
    IDataObject obj = info.getDataObject();
    if (obj==null)
        objNumber = "NULL";
    else
        objNumber = obj.getName();
}
```

基本イベント・スクリプト・オブジェクト - スクリプトPX

基本イベント・スクリプト・オブジェクトは、IBaseScriptObj です。

- 目的と機能 - 「ツール・メニューの拡張」および「スケジュール済イベント」イベント・スクリプト・オブジェクトのインタフェース。IBaseScriptObj は、すべてのイベント・スクリプト・オブジェクトの継承インタフェースです。IBaseScriptObj は、イベント・タイプ、イベント・トリガー・タイプ、イベント名、イベント確認通知受信者名、イベント・ハンドラ名およびユーザー定義マップに関する情報を保持します。

注意 このインタフェースには、Agile PLM 通知を送信するためのスクリプト・メソッドが用意されています。

IBaseObjectScriptObj は、オブジェクト関連イベントの基本イベント情報オブジェクトです。

- 目的と機能 - IBaseObjectScriptObj には、イベントがトリガーされた Agile オブジェクトが含まれています。また、「確定(アイテム)」、「未確定(アイテム)」、「「アクション」メニューの拡張」および「オブジェクトについての適合性ロールアップ」イベント情報オブジェクトのインタフェースであり、IBaseScriptObj から継承します。
- 継承インタフェース: IBaseScriptObj。

次の例は、IBaseScriptObj および IBaseObjectScriptObj の使用方法を示しています。

例: IBaseScriptObj の使用方法

This example opens a SDK session and retrieves information pertaining to the Event.

```
void invokeScript(IBaseScriptObj obj)
{
    // getEventType()
    int evttype = obj.getEventType();

    // getEventTriggerType()
    int evtTriggerType = obj.getEventTriggerType();

    // getEventName()
    String eventName = obj.getEventName();

    // getEventSubscriberName()
    String subscriberName = obj.getEventSubscriberName();
    // getEventHandlerName()
    String handlerName = obj.getEventHandlerName();

    //logMonitor ()
    obj.logMonitor("Status is Passed");

    //getAgileSDKSession()
    IAgileSession session = obj.getAgileSDKSession();

    //getPXEventInfo ()
    IEventInfo req = obj.getPXEventInfo();

    //sendNotification()
    obj.sendNotification("Test", true, [ "admin"], " passed from
BaseScriptObj" + eventName);
    // setUserDefinedMap()
    obj.setUserDefinedMap ( [ 'Agile 93' : 'PLM Product',
'Scripting': 'Is Fun Tool']);
    // getUserDefinedMap()
    Map myMap = obj.getUserDefinedMap ();
}
```

例: IBaseObjectScriptObj の使用方法

```
IBaseObjectScriptObj retrieves Class ID and object number and sets P1,
P2, and P3 attribute values of the object
void testIBaseObjectScriptObj(IBaseScriptObj obj)
{
    String objNumber = "";
    //Disable all the warning exceptions raise
    obj.disableAllWarnings();

    //get class ID
    int classID = obj.getClassId();

    //get object number
    objNumber = obj.getObjectNumber();

    //set attributes
    obj.setValueByAttId(CommonConstants.ATT_PAGE_TWO_TEXT01,"Text
Value");
    // setValueByAttId one from each attribute type
    /*
    * Date:      Page Two.Date01
    * Text:      page Two.Text01
    * MultiText: Page Three.MultiText10
    * List:      Page Two.List01
    * MultiList: Page Three.MultiList01
    * Numeric:   Page Two.Numeric 01
    * Money:     Page Three.Money01
    */

    // get attribute value
    obj.getValueByAttId(CommonConstants.ATT_PAGE_TWO_TEXT01);
    //log information to handler monitor
    obj.logMonitor("Object Number:" + objNumber);
    obj.logMonitor("Class Id :" + classID);
}
```

一般的なオブジェクト・アクションの使用

一般的なオブジェクト・アクションは、作成、削除、名前を付けて保存、タイトル・ブロックの更新などのアクションです。これらのイベントに対する情報オブジェクトおよびスクリプト・イベント・オブジェクトは、その継承インタフェースに応じてグループ化して説明します。

一般的なオブジェクト・アクション - Java PX

オブジェクトの作成

このイベントの情報オブジェクトは、ICreateEventInfo です。

- **目的と機能** - ICreateEventInfo は、要求した新規オブジェクトの番号とサブクラス識別子を取得します。また、クライアントが設定した番号とサブクラスを上書きすることもできます。
- **継承インタフェース** - IUpdateEventInfo、IObjectEventInfo、IEventInfo。
- **継承インタフェースの目的と機能:**
 - IObjectEventInfo。
 - IUpdateEventInfo は、ダーティ属性の配列を取得します。ユーザーは新しい属性および値を上書きまたは設定できます。

次の例は、ICreateEventInfo および IUpdateEventInfo の使用方法を示しています。

例: ICreateEventInfo の使用方法

```
private void testICreateEventInfo(IAgileSession session, IEventInfo
req) throws APIException {
    ICreateEventInfo info = (ICreateEventInfo)req;
    String number = "";
    Integer subclass = null;
    String newNumber = getNewNumber(info); // user defined method
    Integer newSubclassId = getNewSubclassId(session, req); // user
defined method
    Integer newSubclass = null;
    // getNewNumber()
    number = info.getNewNumber();

    // getNewSubclassId()
    subclass = info.getNewSubclassId()

    // setNewNumber()
    info.setNewNumber(newNumber);

    // setNewSubclassId()
    info.setNewSubclassId(newSubclassId);
    newSubclass = info.getNewSubclassId();
}
```

例: IUpdateEventInfo の使用方法

```
private void testIUpdateEventInfo(IAgileSession session, IEventInfo
req) throws Exception {
    // Interface methods
    IUpdateEventInfo info = (IUpdateEventInfo)req;

    //getCells()
    IEventDirtyCell[] cells = info.getCells();
    // getAttributeIds()
    Integer[] attrs = info.getAttributeIds();

    // setCell()
    // Get class specific P1 attribute
    Integer plattrId = getPlAttribute(session, info);

    // Override client value
    info.setCell(plattrId, "set desc from CO");

    // Add new Dirty value
    info.setCeldirty1(CommonConstants.ATT_PAGE_TWO_TEXT02, "setCell()");
    String value2 =
    info.getValue(CommonConstants.ATT_PAGE_TWO_TEXT02).toString();
    // setCell() one from each attribute type
    * Date:      Page Two.Date01
    * Text:      already cover
    * MultiText: Page Three.MultiText10
    * List:      Page Two.List01
    * MultiList: Page Three.MultiList01
    * Numeric:   Page Two.Numeric 01
    * Money:     Page Three.Money01
    */
```

```

Integer subClassId = getSubclassId(info);
IAttribute attr1 =
    session.getAdminInstance().getAgileClass(subClassId).getAttribute
        (CommonConstants.ATT_PAGE_TWO_LIST01);
IAttribute attr2 =
    session.getAdminInstance().getAgileClass(subClassId).getAttribute
        (CommonConstants.ATT_PAGE_THREE_MULTILIST01);
IAgileList list1 =
    (IAgileList)attr1.getAvailableValues();
list1.setSelection(new Object[]{"b"});
IAgileList list2 = null;
if (attr2!=null){
    list2 = (IAgileList)attr2.getAvailableValues();
    list2.setSelection(new Object[]{"a", "b", "e"});
}
SimpleDateFormat df = new SimpleDateFormat("MM/dd/yyyy");
String d = "1/31/2009";
Date date = df.parse(d);
info.setCell(CommonConstants.ATT_PAGE_TWO_DATE01, date);
String multitext = "set multitext field in CO";
info.setCell(CommonConstants.ATT_PAGE_TWO_LIST01, list1);
info.setCell(CommonConstants.ATT_PAGE_TWO_LIST02, list1); // To test
IEventDirtyCell
info.setCell(CommonConstants.ATT_PAGE_TWO_NUMERIC01, 888.66);
if (attr2!=null){
    info.setCell(CommonConstants.ATT_PAGE_THREE_MULTITEXT10,
        multitext);
    info.setCell(CommonConstants.ATT_PAGE_THREE_MULTILIST01, list2);
}
Money money = new Money (new Integer(100), "USD");
// removeCell()
info.removeCell(CommonConstants.ATT_PAGE_THREE_TEXT01);
}

```

タイトル・ブロックの更新

このイベントの情報オブジェクトは、IUpdateTitleBlockEventInfo です。

- **目的と機能** - IUpdateTitleBlockEventInfo は、「タイトル・ブロックの更新」イベント情報オブジェクトのインタフェースです。この更新が、IItem オブジェクトのタイトル・ブロックに対するレッドライン更新またはレッドライン取消し更新であるかを確認します。この IItem オブジェクトは、Change オブジェクトの「対象アイテム」テーブルから取得されます。
- **継承インタフェース** - IUpdateEventInfo、IObjectEventInfo、IEventInfo。

オブジェクトに名前を付けて保存

このイベントの情報オブジェクトは、ISaveAsEventInfo です。

- **目的と機能** - ISaveAsEventInfo は、次のタスクを実行します。
 - 新たに保存したオブジェクトの番号とサブクラスを取得します。
 - PLM クライアントが設定した番号とサブクラスを上書きします。
- **継承インタフェース** - IUpdateEventInfo、IObjectEventInfo、IEventInfo。

オブジェクトの削除

このイベントの情報オブジェクトは、IDeleteEventInfo です。

- **目的と機能** - IDeleteEventInfo は、イベントの削除の情報オブジェクトのインタフェースです。次のタスクを実行します。
 - 新たに削除したオブジェクトの番号とサブクラスを取得します。
 - 実行したアクションがソフト削除か、ハード削除であるかを確認します。
- **継承インタフェース** - IObjectEventInfo、IEventInfo。

オブジェクトのエクスポート

このイベントの情報オブジェクトは、IExportEventInfo です。

- **目的と機能** - IExportEventInfo は、次のタスクを実行します。
 - エクスポート・ファイルの形式を取得します。
 - エクスポートされたオブジェクトの配列を返します。
 - オブジェクトをエクスポートするためのテーブルを返します。
- **継承インタフェース** - IEventInfo。

一般的な基本イベント・スクリプト・オブジェクトの使用

オブジェクトの作成

このイベントのスクリプト・イベント・オブジェクトは、ICreateScriptObj です。

- **目的と機能** - ICreateEventInfo と同じです。
- **継承インタフェース** - IUpdateScriptObj、IBaseObjectScriptObj、IBaseScriptObj。
- **継承インタフェースの目的と機能** - IUpdateEventInfo、IObjectEventInfo および IEventInfo と同じです。

次の例は、ICreateScriptObj および IUpdateScriptObj の使用方法を示しています。

例: ICreateScriptObj の使用方法

```
// In the example, ICreateScriptObj modifies the number and class ID of
// the new object.
void testICreateScriptObj (IBaseScriptObj obj)
{
    String origNumber = "";
    String newNumber = "";
    int newSubclassId =
        ItemConstraints.CLASS_DOCUMENT ; // new subclass ID of choice
    int newSubclass ;
    int subclass;
    // getNewNumber()
    origNumber = obj.getNewNumber();
    newNumber = origNumber + "new";

    // setNewNumber()
    obj.setNewNumber(newNumber);
    newNumber = obj.getNewNumber();
}
```

```
// getNewSubclassId()
subclass = obj.getNewSubclassId();

// setNewSubclassId()
obj.setNewSubclassId(newSubclassId);
newSubclass = obj.getNewSubclassId();

// log new object number and new subclass ID in to handler
monitor
obj.logMonitor("new object number is:" + newNumber);
obj.logMonitor("new subclass ID is:" + newSubclass);

}
```

例: IUpdateScriptObj の使用方法

```
// In this example, IUpdateScriptObj gets the ID and value of Dirty
attributes, sets Dirty attribute values from context object, and removes
the value of Dirty attributes.
void testIUpdateScriptObj (IBaseScriptObj obj)
{
    String dirtyAttr = "";
    String dirtyValue = "";

    // get Attribute Ids
    int[] attrs = obj.getAttrIds();
    attrs.each {attr->

        // get Attribute value
        dirtyAttr = obj.getDirtyAttr(attr);

        // log attribute Id and value in to handler monitor
        obj.logMonitor("Dirty Attr Id :" + attr);
        obj.logMonitor(" dirty Attr value:" + dirtyAttr);
    }

    // Overwrite client value
    obj.setDirtyAttrValue(CommonConstants.ATT_PAGE_TWO_TEXT02, "set text
value from CO");
    //Remove Dirty Attribute value
    obj.removeDirtyAttr(CommonConstants.ATT_PAGE_TWO_TEXT02);
    // get dirty attribute value after removing value from CO
    dirtyValue =
    obj.getDirtyAttrValue(CommonConstants.ATT_PAGE_TWO_TEXT02);
    //log original attribute value in to handler monitor
    obj.logMonitor("Attribute value after remove:" + dirtyValue);
}
```

タイトル・ブロックの更新

このイベントのスクリプト・イベント・オブジェクトは、IUpdateTitleBlockScriptObj です。

- **目的と機能** - IUpdateTitleBlockScriptObj は、この更新が、IItem オブジェクトのタイトル・ブロックに対するレッドライン更新またはレッドライン取消し更新であるかを確認します。この IItem オブジェクトは、Change オブジェクトの「対象アイテム」テーブルから取得されます。
- **継承インタフェース** - IUpdateScriptObj、IBaseObjectScriptObj、IBaseScriptObj。

オブジェクトに名前を付けて保存

このイベントのスクリプト・イベント・オブジェクトは、ISaveAsScriptObj です。

- **目的と機能** - ISaveAsScriptObj は、新規オブジェクトの名前または番号を取得します。
- **継承インタフェース** - IUpdateScriptObj、IBaseObjectScriptObj、IBaseScriptObj。

オブジェクトの削除

このイベントの情報オブジェクトは、IDeleteScriptObj です。

- **目的と機能** - IDeleteScriptObj は、イベント・スクリプト・オブジェクトを削除します。
- **継承インタフェース** - IBaseObjectScriptObj、IBaseScriptObj。

オブジェクトのエクスポート

このイベントのスクリプト・イベント・オブジェクトは、IExportScriptObj です。

- **目的と機能** - IExportScriptObj は、エクスポート・オブジェクトの形式を取得します。
- **継承インタフェース** - IBaseScriptObj。

例: IExportScriptObj の使用方法

```
In this example, IExportScriptObj retrieves information about the
object that is exported.
void testIExportScriptObj (IBaseScriptObj obj)
{
    int format;
    int[] selectedTables =
    String objects = "";
    String user = "";
    String objectNumber =
        "P00002" ; // the object number being exported
    // get current user
    obj.getCurrentUser();
    // get file format for the export.
    format = obj.getExtractedFormat();

    // get list of Object Names being exported
    objects = obj.getExtractedObjects();

    // get tables selected for export
    selectedTables = obj.getSelectedTables(objectNumber);
    // log current user, extracted format, object names and tables which
    exported in to handler monitor
    obj.logMonitor("current user is:" + user);
    obj.logMonitor("file format is:" + format);
    obj.logMonitor("Object Names are:" + objects);
    obj.logMonitor("Selected Tables are:" + selectedTables);
}
```

テーブル・アクションと関係アクションの使用

これらのアクションには、特定のビジネス・オブジェクトに対してサポートされているオブジェクト・テーブルの更新があります。

テーブル・アクションと関係アクション - Java PX

テーブルの更新

このイベントの情報オブジェクトは、IUpdateTableEventInfo です。

- **目的と機能** - IUpdateTableEventInfo は、「テーブルの更新」および「関係の更新」のインタフェースです。対象オブジェクトのダーティ・テーブルを取得します。
- **継承インタフェース** - IObjectEventInfo および IEventInfo。415 ページの「[基本イベント情報オブジェクト - Java PX](#)」を参照してください。
- **関連インタフェース**
 - IEventDirtyFile は、IEventDirtyRowFileUpdate または IFileEventInfo に関連付けられているダーティ・ファイルのインタフェースです。ファイル・テーブル内の単一の行を表し、ダーティ・ファイルのチェックアウト日を取得します。
 - IEventDirtyTable は、IUpdateTableEventInfo または IEventDirtyRow に関連付けられているダーティ・テーブルのインタフェースです。ダーティ・テーブルには、変更した行のコレクションが含まれています。変更したテーブルの一時テーブル情報へのアクセスを提供します。
 - IEventDirtyCell は、IEventDirtyRowUpdate または IUpdateEventInfo に関連付けられているダーティ・セルのインタフェースです。行内の単一のセルを表し、このダーティ・セルに対応する属性識別子を返します。
 - IEventDirtyRowFileUpdate は、ダーティ・ファイルを取得します。ダーティ行アクション（ファイルの追加、ファイルの置換）の実行に使用されるダーティ行のインタフェースです。
 - IEventDirtyRowUpdate は、更新が発生するダーティ行を取得します。このインタフェースは、「添付ファイル」テーブル以外のすべてのテーブルに対する更新で使用されます。

注意 ダーティ・オブジェクトの詳細は、Oracle® E-Delivery Web サイト (<http://edelivery.oracle.com/>) にある SDK サンプルの Documentation フォルダを参照してください。

次の例は、IEventDirtyTable、および「アイテム BOM」テーブルに対する IEventDirtyRowUpdate の使用方法を示しています。

例: IEventDirtyTable の使用方法

```
private void testIEventDirtyTable(IAgileSession session,
IEventDirtyTable table, IDataObject obj, int evtTriggerType) throws
Exception {
    //getTableId()
    String tableName = getTableName(obj, table);

    // size()
    int size = table.size();

    //iterator()
    Iterator it = table.iterator();
    while (it.hasNext()){
        IEventDirtyRow row = (IEventDirtyRow)it.next();
        if(row.getAction() != EventConstants.DIRTY_ROW_ACTION_ADD_FILE &
        &row.getAction() != EventConstants.DIRTY_ROW_ACTION_REPLACE_FILE)
            //user defined method//
            testIEventDirtyRowUpdateCommon(session, row, obj,
            evtTriggerType);
        else
    }
```

```

        testIEventDirtyRowFileUpdate(row); // user defined method
    }
}

```

例: 「アイテム BOM」テーブルに対する IEventDirtyRowUpdate

```

private void
testIEventDirtyRowUpdate_ItemBOM_Update(IEventDirtyRowUpdate row,
IDataObject obj) throws Exception {
IEventDirtyCell[] cells1 = row.getCells();
readCells(cells1); // user defined method

/* setCell() - Override
 * List01 ==> c
 * MultiText30 ==> setCell() on update
 * Text01 ==> setCell() on update
 * Numeric01 ==> 888.66
 * BOM Notes ==> setCell() from CO
 */
//List01
IAttribute attrList =
obj.getAgileClass().getAttribute(ItemConstants.ATT_BOM_BOM_LIST01);

IAgileList list =
(IAgileList)attrList.getAvailableValues();

list.setSelection(new Object[]{"c"});
row.setCell(ItemConstants.ATT_BOM_BOM_LIST01, list);
//MultiText30
row.setCell(ItemConstants.ATT_BOM_BOM_MULTITEXT30, "setCell() MT30
update");
//Text01
row.setCell(ItemConstants.ATT_BOM_BOM_TEXT01, "setCell() T01
update");
// Numeric01 row.setCell(ItemConstants.ATT_BOM_BOM_NUMERIC01,
888.66);
// BOM Notes row.setCell(ItemConstants.ATT_BOM_BOM_NOTES, "setCell()
Notes update");

/*
 * setCell() - New
 * List02 ==> a
 * Text02 ==> setCell() on update
 */

// List02
row.setCell(ItemConstants.ATT_BOM_BOM_LIST02, list);
//Text02
row.setCell(ItemConstants.ATT_BOM_BOM_TEXT02, "setCell() T02
update");
// Qty row.setCell(ItemConstants.ATT_BOM_QTY, new Integer(2));
//removeCell() ==> Date01 & Ref Des
row.removeCell(ItemConstants.ATT_BOM_BOM_DATE01);
row.removeCell(ItemConstants.ATT_BOM_FIND_NUM);

```

```
// getCell()
IEventDirtyCell cell =
row.getCell(ItemConstants.ATT_BOM_BOM_LIST02);
}
```

関係の更新

「テーブルの更新」を参照してください。

テーブル・アクションと関係アクション - スクリプトPX

テーブルの更新

このイベントのスクリプト・イベント・オブジェクトは、IUpdateTableScriptObj です。

- **目的と機能** - IUpdateTableScriptObj は、「テーブルの更新」および「関係の更新」のインタフェースです。対象オブジェクトのダーティ・テーブルを取得します。
- **継承インタフェース** - IBaseObjectScriptObj。416ページの「[基本イベント・スクリプト・オブジェクト - スクリプトPX](#)」を参照してください。
- **関連インタフェース**
 - IEventDirtyRow は、IUpdateTableScriptObj に関連付けられているダーティ行の基本インタフェースです。テーブル内の単一の行を表します。行に関連付けられているアクションがあります。
 - IEventDirtyRowFileUpdate は、ダーティ・ファイルの行を取得します。ダーティ・ファイルの行アクション（ファイルの追加、ファイルの置換）の実行に使用されるダーティ・ファイルの行のインタフェースです。
 - IEventDirtyRowUpdate は、ダーティ行を取得します。ダーティ行アクション（追加、削除、更新、レッドライン追加、レッドライン削除、レッドライン更新、レッドライン取消し）の実行に使用されるダーティ行のインタフェースです。
 - IEventDirtyFile は、IEventDirtyRowFileUpdate または IFileEventInfo に関連付けられているダーティ・ファイルのインタフェースです。ファイル・テーブル内の単一の行を表し、ダーティ・ファイルのチェックアウト日を取得します。

次の例は、IEventDirtyTable、および「アイテム BOM」テーブルに対する IEventDirtyRowUpdate の使用方法を示しています。

例: IEventDirtyTable の使用方法

```
// IUpdateTableScriptObj: get table id and Dirty row ids
void testIUpdateTableScriptObj(IBaseScriptObj obj)
{
    //getTableId()
    int table_id= obj.getTableId();
    obj.logMonitor("Table Id" + table_id);

    // getDirtyRowIds()
    rowIDs =obj.getDirtyRowIds();
    rowIDs.each{rID-> //loop through Dirty rows
    obj.logMonitor("rowID is " + rID);

    //getDirtyRow()
    row = obj.getDirtyRow(rID);
```



```

// getAction
action = row.getAction();
obj.logMonitor ("Action" + action);

if ((action == EventConstants.DIRTY_ROW_ACTION_ADD_FILE) || (action
== EventConstants.DIRTY_ROW_ACTION_REPLACE_FILE))
    testIEventDirtyRowFileUpdate(obj,row,rID); // user method
else
    testIEventDirtyRowUpdate(obj,row,rID); // user method
}

```

例: IEventDirtyRowUpdate の使用方法

```

//This Item BOM table example gets the object number and action for a
single row associate with IUpdateTableScriptObj
void testIEventDirtyRowUpdate(IBaseScriptObj obj,IEventDirtyRow row,
int rID )
{

    //getRowId()
    int rid = row.getRowId();

    // getObjectNumber()
    int rnumber = row.getObjectNumber();
    obj.logMonitor("row object" + rnumber);

    // getDirtyRow
    dirtyRow = obj.getDirtyRow(rID);

    //from client, update following attributes in Bom table
    // find number,Bom notes, Multi Text01,List01,date01, Text01,
    numeric0
    //getDirtyAttrIds()
    dirtyAttrs = dirtyRow.getDirtyAttrIds();
    def sort_attrs = dirtyAttrs as List;
    sort_attrs.sort(); //sort attribute Ids

    // getDirtyAttrValue()
    sort_attrs.each {dirtyAttr->
        attrValue = dirtyRow.getDirtyAttrValue(dirtyAttr);
        obj.logMonitor('***'+ "attribute value" +'=' + attrValue);
    }

    // array of vlues to overwrite attributes from Context object
    //[find number,Bom notes, Multi Text01,List01,date01, Text01,
    numeric01]
    set_dirty_value = ["5","Bom Note","Multi text 03","e","2009-01-30
    08:00:00","Text 01","224466"];
    //setDirtyAttrValue , overwrite attribute values
    int indexy =0;
    sort_attrs.each{att->
        dirtyRow.setDirtyAttrValue(att , set_dirty_value[indexy++]);
    }

    // getDirtyAttrValue() after overwrite in CO
    int indexB=0;
    sort_attrs.each {dirtyAttr->
        attrValue2 = dirtyRow.getDirtyAttrValue(dirtyAttr);
    }
}

```

```

        all_attrValue2[indexB++] = attrValue2;
    }

    // removetDirtyAttr() remove dirty attributes and rollback changes

    sort_attrs.each {dirtyAttr->
        dirtyRow.removetDirtyAttr(dirtyAttr);
    }
}

```

関係の更新

「テーブルの更新」を参照してください。

ワークフロー・オブジェクト・アクションの使用

ワークフロー・オブジェクト・アクションは、「ワークフローのステータスの変更」、「ワークフローの承認者またはオブザーバの変更」、および Product Cost Management のソーシング・プロジェクトのステータスの変更など、ワークフロー関連のアクションです。

ワークフロー・オブジェクト・アクション - Java PX

ワークフローのステータスの変更

このイベントの情報オブジェクトは、IWFCChangeStatusEventInfo です。

- **目的と機能** - IWFCChangeStatusEventInfo は、オブジェクトのワークフロー・ステータスの変更および割り当てられた通知者を取得し、「自動昇格」フラグが設定されているかどうかを確認します。
- **継承インタフェース** - IRoutingSlipEventInfo、IRoutableObjectEventInfo、IObjectEventInfo、IEventInfo。
- **継承インタフェースの目的と機能:**
 - IRoutingSlipEventInfo - すべてのイベント情報オブジェクトの**継承**インタフェースです。送信スリップ・オブジェクトが含まれ、承認者、オブザーバ、コメントおよび緊急の各フラグを設定/取得するためのメソッドを提供します。
 - IRoutableObjectEventInfo - ワークフロー・アクションに関連するすべてのイベント情報オブジェクトの**継承**インタフェースです。承認者またはオブザーバの取得/設定などのメソッドを提供します。

次の例は、IWFCChangeStatusEventInfo、IRoutingSlipEventInfo および IRoutableObjectEventInfo の使用方法を示しています。

例: IWFCChangeStatusEventInfo の使用方法

```

private void testIWFCChangeStatusEventInfo(IAgileSession session,
IEventInfo req) throws APIException {
    IWFCChangeStatusEventInfo info = (IWFCChangeStatusEventInfo)req;
    //getNotifiers()
    IDataObject[] notifiers = info.getNotifiers();

    // isAutoPromote()
    boolean isAutoPromote = info.isAutoPromote();
    // setNotifiers()
    IUser user = getUser(session, "yvonnec");
    IUserGroup ug = getUserGroup(session, "SOA");
}

```

```

Collection col = new ArrayList();
col.add(user);
col.add(ug);
info.setNotifiers(col);

// getFromStatus()
IStatus fromStatus = info.getFromStatus();

// getToStatus()
IStatus toStatus = info.getToStatus();
}

```

例: IRoutingSlipEventInfo の使用方法

```

private void testIRoutingSlipEventInfo(IAgileSession session,
IEventInfo req) throws APIException {
    IRoutingSlipEventInfo info = (IRoutingSlipEventInfo) req;
    Collection colAppvrs = new ArrayList();
    Collection colObsvrs = new ArrayList();
    Collection colNewAppvrs = new ArrayList();
    Collection colNewObsvrs = new ArrayList();
    String newComments = "Override in context object.";

    //getApprovers()
    IDataObject[] approvers = info.getApprovers();
    String approverList = arrayToString(approvers);

    //getObservers()
    IDataObject[] observers = info.getObservers();

    //getComments()
    String comments = info.getComments();

    //isUrgent()
    boolean isUrgent = info.isUrgent();

    //setApprovers()
    IUser user1 = getUser(session, "badriv");
    IUser user2 = getUser(session, "albert1");
    IUser user3 = getUser(session, "brucec");
    IUserGroup ug1 = getUserGroup(session, "SOA");

    colAppvrs.add(user1);
    colAppvrs.add(user2);
    colObsvrs.add(user3);
    colObsvrs.add(ug1);

    info.setApprovers(colAppvrs);
    IDataObject[] newApprovers = info.getApprovers();

    //setObservers()
    info.setObservers(colObsvrs);
    IDataObject[] newObservers = info.getObservers();

    //setComments()
    info.setComments(newComments);
    String latestComments = info.getComments();

    //setUrgent()
    boolean newUrgent = getOppositeBoolean(isUrgent);
}

```

```
info.setUrgent(newUrgent);  
boolean latestUrgent = info.isUrgent();  
}
```

例: IRoutableObjectEventInfo の使用方法

```
private void testIRoutableObjectEventInfo(IEventInfo req) throws  
APIException {  
    IRoutableObjectEventInfo info = (IRoutableObjectEventInfo) req;  
    //getWorkflow()  
    IWorkflow wf = info.getWorkflow();  
}
```

ワークフローの承認

このイベントの情報オブジェクトは、ISignOffEventInfo です。

- **目的と機能** - ISignOffEventInfo は、「ワークフローの承認」および「ワークフローの却下」のインタフェースです。ステータス、承認者（ユーザー）、承認者グループに関する情報を返して上書きし、サインオフ・フラグのステータスを確認します。
- **継承インタフェース** - IRoutableObjectCommentEventInfo、IRoutableObjectEventInfo、IObjectEventInfo、IEventInfo。
- **継承インタフェースの目的と機能** - IRoutableObjectCommentEventInfo は、「ワークフローのコメント」のインタフェースです。「ワークフローのコメント」、「ワークフローの承認」および「ワークフローの却下」に入力されたデータを取得して上書きします。

ワークフローの却下

「ワークフローの承認」を参照してください。

ワークフローのエスカレーション

このイベントの情報オブジェクトは、IEscalationEventInfo です。

- **目的と機能** - IEscalationEventInfo は、「ワークフローのエスカレーション」イベントに関する次の情報を取得します。
 - サインオフ・ユーザー、エスカレート先ユーザー、エスカレーション期間およびワークフローのステータス
- **継承インタフェース** - IRoutableObjectEventInfo、IObjectEventInfo、IEventInfo。
- **継承インタフェースの目的と機能** - 「ワークフローの承認」を参照してください。

ワークフローの督促

このイベントの情報オブジェクトは、IReminderEventInfo です。

- **目的と機能** - IReminderEventInfo は、ワークフローの催促に割り当てられている通知者を返します。
- **継承インタフェース** - IRoutableObjectEventInfo、IObjectEventInfo、IEventInfo。

ワークフローの検証

このイベントの情報オブジェクトは、IAuditStatusEventInfo です。

- **目的と機能** - IAuditStatusEventInfo は、実行された「ワークフロー検証」アクションのタイプを取得します。

- **継承インタフェース** - `IAuditResultEventInfo`、`IRoutableObjectEventInfo`、`IObjectEventInfo`、`IEventInfo`。
- **継承インタフェースの目的と機能** - `IAuditResultEventInfo` は、「ワークフローの検証」および「ワークフローの昇格失敗」のインタフェースです。検証または昇格失敗に対するエラー・メッセージを取得します。

ワークフローの昇格失敗

「ワークフローの検証」を参照してください。

ワークフローのコメント

このイベントの情報オブジェクトは、`IRoutableObjectCommentEventInfo` です。「ワークフローの承認」を参照してください。

ワークフローの承認者またはオブザーバの変更

このイベントの情報オブジェクトは、`ICheckAppObserverEventInfo` です。

- **目的と機能** - `ICheckAppObserverEventInfo` は、アクション・タイプ、ステータス、および「ワークフローの承認者またはオブザーバの変更」の適用先ステータスを取得します。
- ワークフロー・ステータスに対する承認者またはオブザーバによる変更アクション。
- **継承インタフェース** - 「`IRoutingSlipEventInfo`」を参照してください。

ワークフロー・オブジェクト・アクション・スクリプトPX

ワークフローのステータスの変更

このイベントの情報オブジェクトは、`ICheckStatusScriptObj` です。

- **目的と機能** - `ICheckStatusScriptObj` は、オブジェクトのワークフロー・ステータスの変更および割り当てられた通知者を取得し、「自動昇格」フラグが設定されているかどうかを確認します。
- **継承インタフェース** - `IRoutingSlipScriptObj`、`IRoutableScriptObj`、`IBaseObjectScriptObj`、`IBaseScriptObj`。
- **継承インタフェースの目的と機能:**
 - `IRoutingSlipScriptObj` - 送信スリップ・オブジェクトが含まれるすべてのイベント・スクリプト・オブジェクトの**継承**インタフェースです。
 - `IRoutableScriptObj` - ワークフロー・アクションに関連するすべてのイベント・スクリプト・オブジェクトの**継承**インタフェースです。

次の例は、`ICheckStatusScriptObj`、`IRoutingSlipScriptObj` および `IRoutableObjectScriptObj` の使用方法を示しています。

例: `ICheckStatusScriptObj` の使用方法

```
ICheckStatusScriptObj: Change Status for Workflow Event script object.
void testICheckStatusScriptObj (IBaseScriptObj obj)
{
    // getFromStatus()
    String current_status = obj.getFromStatus();
}
```

```
// getToStatus()
String next_status = obj.getToStatus();

// getNotifiers()
String notifiers =obj.getNotifiers();

// setNotifiers()
obj.setNotifiers(["admin", "demo1","weiz", "demo5"]);
// isAutoPromote()
boolean auto = obj.isAutoPromote();
}
```

例: IRoutingSlipScriptObj の使用方法

```
void testIRoutingSlipScriptObj(IBaseScriptObj obj)
{
    // getCurrentUser()
    String user= obj.getCurrentUser();

    // getObservers()
    def observers = obj.getObservers();

    // getApprovers()
    def approvers = obj.getApprovers();

    // getComments()
    String comments = obj.getComments();

    // isUrgent()
    boolean flag = obj.isUrgent();

    // setApprovers()
    obj.setApprovers(["admin", "demo1"]);

    // setObservers()
    obj.setObservers(["demo6", "demo5"]);

    // setComments()
    obj.setComments(" new_comments");
    boolean new_urgent= true;

    // setUrgent()
    obj.setUrgent(new_urgent);
}
```

例: IRoutableScriptObj の使用方法

```
IRoutableScriptObj : get workflow
void testIRoutableScriptObj(IBaseScriptObj obj)
{

    // getWorkflow()
    String wf = obj.getWorkflow();
}
```

ワークフローの承認

このイベントの情報オブジェクトは、ISignOffScriptObj です。

- **目的と機能** - ISignOffScriptObj は、「ワークフローの承認」および「ワークフローの却下」のインタフェースです。ステータス、承認者（ユーザー）、承認者グループに関する情報を返して、サインオフ・フラグのステータスを確認します。

- **継承インタフェース** - IRoutableObjectCommentScriptObj、IRoutableScriptObj、IBaseObjectScriptObj、IBaseScriptObj。
- **継承インタフェースの目的と機能** - IRoutableObjectCommentScriptObj - 「ワークフローのコメント」のインタフェースです。ルーティング可能なオブジェクトに入力されたコメントを取得します。

ワークフローの却下

「ワークフローの承認」を参照してください。

ワークフローのエスカレーション

このイベントの情報オブジェクトは、IEscalationScriptObj です。

- **目的と機能** - IEscalationScriptObj は、ワークフローのエスカレート先のユーザー名を取得します。
- **継承インタフェース** - IRoutableScriptObj、IBaseObjectScriptObj、IBaseScriptObj。
- **継承インタフェースの目的と機能** - 「ワークフローの承認」を参照してください。

ワークフローの督促

このイベントの情報オブジェクトは、IReminderScriptObj です。

- **目的と機能** - IReminderScriptObj は、督促期間およびワークフローの催促に割り当てられている通知者を返します。
- **継承インタフェース** - IRoutableScriptObj、IBaseObjectScriptObj、IBaseScriptObj。

ワークフローの検証

このイベントの情報オブジェクトは、IAuditStatusScriptObj です。

- **目的と機能** - IAuditStatusScriptObj は、実行された「ワークフロー検証」アクションのタイプを取得します。
- **継承インタフェース** - IAuditResultScriptObj、IRoutableScriptObj、IObjectScriptObj、IBaseScriptObj。
- **継承インタフェースの目的と機能** - IAuditResultEventInfo は、「ワークフローの検証」および「ワークフローの昇格失敗」のインタフェースです。検証または昇格失敗に対するエラーまたは警告メッセージ、およびワークフロー・ステータス情報を取得します。

ワークフローの昇格失敗

「ワークフローの検証」を参照してください。

ワークフローのコメント

このイベントの情報オブジェクトは、IRoutableObjectCommentScriptObj です。「ワークフローの承認」を参照してください。

ワークフローの承認者またはオブザーバの変更

このイベントの情報オブジェクトは、IChangeAppObserverScriptObj です。

- **目的と機能** - IChangeAppObserverScriptObj は、アクション・タイプ、ステータス、および「ワークフローの承認者またはオブザーバの変更」の適用先ステータスを取得します。
- **継承インタフェース** - 「IRoutingSlipScriptObj」を参照してください。

特定のオブジェクトベース・アクションの使用

これらのアクションでは、PLM の Production Collaboration ソリューションのアイテム・オブジェクトの確定と未確定、およびソーシング・プロジェクトのステータスの変更がサポートされます。

特定のオブジェクトベース・アクション - Java PX

確定(アイテム)および未確定(アイテム)

このイベントの情報オブジェクトは、IObjectEventInfo です。このインタフェースおよび適用可能な継承インタフェースについては、418ページの「[一般的なオブジェクト・アクション - Java PX](#)」に記載されています。

ソーシング・オブジェクトのステータスの変更

このイベントの情報オブジェクトは、ISourcingObjectChangeStatusEventInfo です。

- **目的と機能** - ISourcingObjectChangeStatusEventInfo は、ソーシング・オブジェクトで実行されたアクションのタイプを取得します。
- **継承インタフェース** - IObjectEventInfo および IEventInfo。

特定のオブジェクトベース・アクション - スクリプトPX

確定(アイテム)および未確定(アイテム)

このイベントのスクリプト・イベント・オブジェクトは、IBaseObjectScriptObj です。このインタフェースおよび適用可能な継承インタフェースについては、421ページの「[一般的なオブジェクト・アクション - スクリプトPX](#)」に記載されています。

ソーシング・オブジェクトのステータスの変更

このイベントの情報オブジェクトは、ISourcingObjectChangeStatusScriptObj です。

- **目的と機能** - ISourcingObjectChangeStatusScriptObj は、ソーシング・オブジェクトで実行されたアクションのタイプを取得します。
- **継承インタフェース** - IBaseObjectScriptObj および IBaseScriptObj。

ファイル・アクションおよび添付ファイル・オブジェクト・アクションの使用

これらは、ファイルのチェックアウト、ファイルのチェックイン、およびバージョン・ファイルのバージョンなど、ファイル関連のアクションです。

ファイル・アクションおよび添付ファイル・オブジェクト・アクション - Java PX

ファイル取出し、ファイルのチェックアウト、ファイルのチェックイン、ファイルのチェックアウトのキャンセル

これらのイベントの情報オブジェクトは、IFileEventInfo です。

- **目的と機能** - IFileEventInfo は、選択した添付ファイルに対するダーティ・ファイルを取得します。
- **継承インタフェース** - IObjectEventInfo および IEventInfo。
- **関連インタフェース** - IEventDirtyRowFileUpdate または IFileEventInfo のインタフェースである IEventDirtyFile は、ファイル・テーブル内の単一の行を表します。

次の例は、IFileEventInfo の使用方法を示しています。

例: IFileEventInfo の使用方法

```
private void testIFileEventInfo(IEventInfo req) throws APIException {
    IFileEventInfo info = (IFileEventInfo)req;
    StringBuffer buffer = new StringBuffer();

    // getFiles()
    IEventDirtyFile[] files = info.getFiles();
    if(files!=null){
        int length = files.length;
        if (length!=0){
            for (int i=0;i<length;i++){
                String fileName = files[i].getFilename();
                String msg = i + ": " + fileName;
                System.out.println(msg);
            }
        }
    }else { //rows == null
        throw new APIException (new Exception("getFiles() returns
        null"));
    }

    // Test IEventDirtyFile
    testIEventDirtyFile(files);
}
```

ファイル・バージョンのパージ

このイベントの情報オブジェクトは、IPurgeFileEventInfo です。

- **目的と機能** - IPurgeFileEventInfo は、パージされたファイルのバージョンを取得します。
- **継承インタフェース** - IObjectEventInfo および IEventInfo。

ファイル・アクションおよび添付ファイル・オブジェクト・アクション - スクリプトPX

ファイル取出し、ファイルのチェックアウト、ファイルのチェックイン、ファイルのチェックアウトのキャンセル

これらのイベントのイベント・スクリプト・オブジェクトは、IFileEventScriptObj です。

- **目的と機能** - IFileEventScriptObj は、選択した添付ファイルに対するダーティ・ファイルを取得します。

- 継承インタフェース - IBaseObjectScriptObj および IBaseScriptObj。

ファイル・バージョンのパージ

このイベントのイベント・スクリプト・オブジェクトは、IPurgeFileScriptObj です。

- 目的と機能 - IPurgeFileScriptObj は、パージされたファイルのバージョンを取得します。
- 継承インタフェース - IBaseObjectScriptObj および IBaseScriptObj。

次の例は、IFileEventScriptObj の使用方法を示しています。

例: IFileEventScriptObj の使用方法

```
// FileEventScriptObj: get file for file actions related events
void testIFileEventScriptObj (IBaseScriptObj obj, IEventDirtyRow row)
{
    // getFile()
    file = row.getFile();
    testIEventDirtyFile(obj, file);
}
```

例: IEventDirtyFile の使用方法

```
IEventDirtyFile: get file information for Dirty file associated with
IEventDirtyRowFileUpdate
void testIEventDirtyFile (IBaseScriptObj obj, IEventDirtyfile file)
{

    // getFilename()
    String file_name = file.getFilename();

    // getSize()
    int file_size = file.getSize();

    // getType()
    file_type = file.getType();

    // getFileFolderNumber()
    file_folder_num = file.getFileFolderNumber();

    // getFileFolderVersionNumber()
    file_folder_ver = file.getFileFolderVersionNumber();
    //getCheckoutUser()
    user = file.getCheckoutUser();

    // getCheckoutDate()
    date = file.getCheckoutDate();

    obj.logMonitor("file name is:" + file_name);
    obj.logMonitor("file size is:" + file_size);
    obj.logMonitor("file type is:" + file_type);
    obj.logMonitor("file folder number is:" + file_folder_num);
    obj.logMonitor("file folder version number is:" +
        file_folder_ver);
    obj.logMonitor("checkout user is:" + user);
    obj.logMonitor("checkout date is:" + date);
}
```

Product Governance & Complianceアクションの使用

このアクションでは、PG&C オブジェクトの適合性の確認がサポートされます。

Product Governance & Complianceアクション - Java PX

オブジェクトについての適合性ロールアップ

このイベントの情報オブジェクトは、IObjectEventInfoです。このインタフェースおよび適用可能な継承インタフェースについては、418ページの「[一般的なオブジェクト・アクション - Java PX](#)」に記載されています。

Product Governance & Complianceアクション - スクリプトPX

オブジェクトについての適合性ロールアップ

このイベントの情報オブジェクトは、IBaseScriptObjです。このインタフェースおよび適用可能な継承インタフェースについては、421ページの「[一般的なオブジェクト・アクション - スクリプトPX](#)」に記載されています。

その他のオブジェクト・アクションの使用

これらは、ファイルのチェックアウト、ファイルのチェックイン、およびバージョン・ファイルのバージョンなど、ファイル関連のアクションです。

その他のオブジェクト・アクション - Java PX

権限委譲

このイベントの情報オブジェクトは、ITransferAuthorityEventInfo です。

- **目的と機能** - ITransferAuthorityEventInfo には、「権限委譲」アクションに属するデータが含まれています。「権限委譲」アクションに対して PLM クライアントが設定したデータを取得して上書きします。
- **継承インタフェース** - 「IExportEventInfo」を参照してください。

その他のオブジェクト・アクション - スクリプトPX

権限委譲

このイベントの情報オブジェクトは、ITransferAuthorityScriptObj です。

- **目的と機能** - ITransferAuthorityScriptObj は、「権限委譲」アクションに対して PLM クライアントが設定したデータを取得して上書きします。
- **継承インタフェース** - IBaseScriptObj。

PLMクライアントでのイベント統合ポイントの使用

イベントは、Agile PLM クライアントの「「アクション」メニューの拡張」、「ツール・メニューの拡張」および「スケジュール済イベント」から起動できます。

イベント統合ポイント - Java PX

「アクション」メニューの拡張

このイベントの情報オブジェクトは、IObjectEventInfoです。このインタフェースおよび適用可能な継承インタフェースについては、418ページの「[一般的なオブジェクト・アクション - Java PX](#)」に記載されています。

ツール・メニューの拡張

このイベントの情報オブジェクトは、IEventInfoです。このインタフェースおよび適用可能な継承インタフェースについては、418ページの「[一般的なオブジェクト・アクション - Java PX](#)」に記載されています。

スケジュール済イベント

このイベントの情報オブジェクトは、IEventInfoです。このインタフェースおよび適用可能な継承インタフェースについては、418ページの「[一般的なオブジェクト・アクション - Java PX](#)」に記載されています。

イベント統合ポイント - スクリプトPX

「アクション」メニューの拡張

このイベントの情報オブジェクトは、IBaseScriptObjです。このインタフェースおよび適用可能な継承インタフェースについては、421ページの「[一般的なオブジェクト・アクション - スクリプトPX](#)」に記載されています。

ツール・メニューの拡張

このイベントの情報オブジェクトは、IBaseObjectScriptObjです。このインタフェースおよび適用可能な継承インタフェースについては、421ページの「[一般的なオブジェクト・アクション - スクリプトPX](#)」に記載されています。

スケジュール済イベント

このイベントの情報オブジェクトは、IBaseScriptObjです。このインタフェースおよび適用可能な継承インタフェースについては、421ページの「[一般的なオブジェクト・アクション - スクリプトPX](#)」に記載されています。

Java PXハンドラとスクリプトPXハンドラに関するガイドライン

次の注釈と推奨事項は、Java PX とスクリプト PX の適切な開発と実装を目的としています。次の情報が記載されています。

- ハンドラをコーディングするために Agile PLM 管理者から入手する必要がある情報、およびイベントを設定して実装するために管理者に伝達する必要がある情報
- イベントを適切に処理するための情報
- PX をテストするための情報

Agile PLM管理者との作業

Agile PLM 管理者は、イベント・タイプ、実行モードおよびトリガー・タイプを判断するために必要な情報を伝達します。Java PX ハンドラまたはスクリプト PX ハンドラを開発した後は、イベント確認通知受信者に関する情報を管理者に通知する必要があります。この情報には、イベント・タイプ、実行モード、トリガー・タイプ、順序および適用可能なエラー処理ルールなどがあります。

イベント確認通知受信者に関する情報の他に、管理者には、イベント・ハンドラを設定するために次の特定の情報がが必要です。

- **Java PXハンドラ** - Java PXはサーバーに配置されます。手順については、355ページの「[カスタム・アクションのパッケージ化および配置](#)」を参照してください。配置後、管理者は、Javaクライアントで「管理」>「設定」>「システム設定」>「イベント管理」>「イベント・ハンドラ」>「新規作成」を選択してJava PXハンドラを検索します。「イベント・ハンドラの作成」ダイアログ・ボックスが表示されます。「イベント・ハンドラ・タイプ」フィールドで、「Java PX」>「イベント・アクション」の順に選択します。管理者は、「イベント・アクション」フィールドに表示される名前を認識して、ハンドラの設定を続行する必要があります。

図32: 「イベント・ハンドラの作成」ダイアログ

The 'Create Event Handler' dialog box contains the following fields and values:

- Event Handler Type: Java PX
- Name: Migration Example - E. Handler
- API Name: MigrationExampleEHandler
- Description: Handler for Migration Example -Selects Event action
- Enabled: Yes
- Role: (empty)
- Event Action: com.oracle.px.IEventAction.HelloWorld

- **スクリプト PX ハンドラ** - スクリプト PX のコードはテキスト・ファイルです。開発者は、スクリプトのテキスト・ファイルを PLM 管理者に送信する必要があります。管理者は、このテキストを「イベント・ハンドラの作成」ダイアログの「スクリプト」フィールドに貼り付け、確認通知受信者の設定を続行します。このフィールドにアクセスするには、Java クライアントから「管理者」>「イベント管理」>「イベント・ハンドラ」>「イベント・ハンドラの作成」>「スクリプト PX」>「スクリプト」の順に選択します。

図33: スクリプト PX の作成

The 'Create Event Handler' dialog box for Script PX contains the following fields and values:

- Event Handler Type: Script PX
- Name: (empty)
- API Name: (empty)
- Description: (empty)
- Enabled: Yes
- Role: (empty)
- Script:




```
import com.agile.agileDSL.ScriptObj.IBaseScriptObj
// add other import statements here
void invokeScript(IBaseScriptObj obj) {
//script body starts here.
}
```

イベントJava PXおよびイベント・スクリプトPXのテスト

開発したハンドラを Agile PLM 管理者が使用してイベント確認通知受信者を設定する場合、開発者は、PX のテストについて管理者と調整する必要があります。一方、開発者自身が設定する場合は、この目的のために次の情報を使用します。

新しい Java PX またはスクリプト PX を起動して、ハンドラ（コード）で指定したアクションが適切に実行されることを確認します。

PX を「ツール」メニューからのユーザー・アクションで起動するように設定している場合は、Web クライアントまたは Java クライアントで次のようにテストできます。

- Web クライアントのツールバーで、 ボタン>「<Event_name>」の順に選択します。
- Java クライアントのツールバーで、「プロセス拡張」 ボタンを選択するか、「ツール」>「プロセス拡張」>「<Event_name>」の順に選択します。

Java PX、スクリプトPXおよび通知ハンドラのトリガーに関するガイドライン

ハンドラの開発時には、次のガイドラインを使用して、これらのイベント・タイプの特別なインスタンスが適切にトリガーされるようにしてください。

一般的なオブジェクト・アクション

「オブジェクトの作成」イベントおよび「名前を付けて保存」イベント

- **Web クライアントと Java クライアントの動作** - このアクションは、1 つの「オブジェクトの作成」イベント（または「名前を付けて保存」イベント）をトリガーします。そのコンテキスト・オブジェクトには、「ページ 1」、「ページ 2」および「ページ 3」に対するすべてのダーティ属性が含まれます。
- **SDK の動作** - このアクションは、1 つの「オブジェクトの作成」イベント（または「名前を付けて保存」イベント）をトリガーします。さらに、更新されたテーブル属性（「ページ 1」、「ページ 2」および「ページ 3」）に応じて、1~3 つの「タイトル・ブロックの更新」イベントがトリガーされます。コンテキスト・オブジェクトには、そのテーブルに属するすべてのダーティ属性が含まれます。

「タイトル・ブロックの更新」イベント

Web クライアント、Java クライアントおよび SDK の動作 - 更新されたテーブル属性（「ページ 1」、「ページ 2」および「ページ 3」）に応じて、テーブルごとに 1 つずつ、1~3 つの「タイトル・ブロックの更新」イベントがトリガーされます。コンテキスト・オブジェクトには、そのテーブルに属するすべてのダーティ属性が含まれます。

「テーブルの更新」イベント

複数の行を追加、変更または削除する場合は、このアクションによって、次のイベント・トリガー動作が発生します。

- 1 つのイベントがトリガーされます。コンテキスト・オブジェクトには、適用可能なすべての行が含まれます。Java クライアントのみ例外で、次のように動作します。
 - 「関係」テーブル - 適用可能な行ごとに 1 つのイベントがトリガーされます。コンテキスト・オブジェクトには、その適用可能な行が含まれます。
 - 「添付ファイル」テーブル - 更新および削除アクションの場合、適用可能な行ごとに 1 つのイベントがトリガーされます。コンテキスト・オブジェクトには、その適用可能な行が含まれます。

ワークフロー・アクション

「ワークフローの昇格失敗」イベント

このイベントは、次のアクションによってトリガーされます。

- 関係によってトリガーされた「ステータスの変更」の失敗。
- 自動昇格の失敗 - 自動昇格の失敗は、次のオブジェクト・アクションの完了後に自動昇格の条件が満たされない場合に、これらのアクションによって起動されます。また、トリガーされた各イベントは、オブジェクト履歴で追跡されます。
 - ステータスの変更
 - 変更のサインオフ
 - 承認者の削除
 - 「カバー・ページ」、「ページ 2」および「ページ 3」の属性の更新
 - 「対象アイテム」テーブル、「関係」、「添付ファイル」タブの編集

注意 自動昇格がトリガーされるのは、編集の場合のみです。

- 添付ファイルのチェックアウトのキャンセル
- 添付ファイルのチェックイン

「自動転送オブジェクト (ATO) の作成」アクション

ATO 作成は、ECO ステータスが「保留中」から「提出済」に変更されると有効化されます。ATO の作成時にトリガーされるイベントの順序は、次のとおりです。

1. 次の内容について、事前イベントおよび事後イベントの確認通知受信者が作成されます。
 - ECO 変更ステータス
 - ATO 作成
 - ATO 変更ステータス
2. ECO が作成されます。
3. ワークフローが割り当てられ、ステータスが「提出済」に変更されます。

ファイル・アクションおよび添付ファイル・アクション

「ファイルのチェックイン」イベント

このアクションは、次のイベントをトリガーします。

- ファイル・フォルダ・オブジェクトに対する「ファイルのチェックイン」イベント - このイベントは、ファイル・フォルダ・オブジェクトに対して「ファイルのチェックイン」アクションが実行されるとトリガーされます。
- ビジネス・オブジェクトに対する「ファイルのチェックイン」イベントおよび「テーブルの更新」イベント - これらのイベントは、ビジネス・オブジェクトに対して「ファイルのチェックイン」アクションが実行されるとトリガーされます。

注意 ビジネス・オブジェクトに対する「テーブルの更新」イベントは、「ファイルのチェックイン」イベントのトリガー後にトリガーされます。また、イベントがトリガーされるのは、添付ファイルのフォルダ・バージョンが**最新**に設定されていない場合のみです。Web クライアントでは、選択したすべての行に対して 1 つのイベントがトリガーされます。Java クライアントでは、選択した行ごとに 1 つのイベントがトリガーされます。

「ファイルのチェックアウト」アクション

このアクションは、次のイベントをトリガーします。

- ファイル・フォルダ・オブジェクトに対する「ファイルのチェックアウト」イベントおよび「ファイル取出し」イベント - これらのイベントは、ファイル・フォルダ・オブジェクトに対して「ファイルのチェックアウト」アクションが実行されるとトリガーされます。
- ビジネス・オブジェクトに対する「ファイルのチェックアウト」イベントおよび「ファイル取出し」イベント - これらのイベントは、Java クライアントでビジネス・オブジェクトに対して「ファイルのチェックアウト」アクションが実行されるとトリガーされます。
 - 「ファイルを1つのZIPファイルとしてダウンロードする」オプションを選択している場合は、1つの「ファイル取出し」イベントがトリガーされ、行ごとに「チェックアウト」イベントがトリガーされます。
 - 「ファイルを1つのZIPファイルとしてダウンロードする」オプションを選択していない場合は、行ごとに1つの「チェックアウト」イベントがトリガーされ、ファイルごとに「ファイル取出し」イベントがトリガーされます。

注意 Web クライアントでは、選択したすべての行に対して1つのイベントがトリガーされます。Java クライアントでは、選択した行ごとに1つのイベントがトリガーされます。

「ファイルのチェックアウトのキャンセル」イベント

このアクションは、次のイベントをトリガーします。

- ファイル・フォルダ・オブジェクトに対する「ファイルのチェックアウトのキャンセル」イベント - このイベントは、ファイル・フォルダ・オブジェクトに対して「ファイルのチェックアウトのキャンセル」アクションが実行されるとトリガーされます。
- ビジネス・オブジェクトに対する「ファイルのチェックアウトのキャンセル」イベント - このイベントは、ビジネス・オブジェクトに対して「ファイルのチェックアウトのキャンセル」アクションが実行されるとトリガーされます。

注意 Web クライアントでは、選択したすべての行に対して1つのイベントがトリガーされます。Java クライアントでは、選択した行ごとに1つのイベントがトリガーされます。

「ファイル取出し」イベント

このイベントは、次の操作によってトリガーされます。

- ビジネス・オブジェクトの「添付ファイル」テーブルまたはファイル・フォルダ・オブジェクトの「ファイル」テーブルにある「**取出し**」ボタンのクリック
- オブジェクトの「添付ファイル」テーブルにある「**ファイル名**」のクリック

注意 表示可能なファイルの場合、Agile Viewer が起動され、イベントはトリガーされません。

- 「ファイルのチェックアウト」アクション

注意 Java クライアントでは、ユーザーがZIP ファイルをダウンロードするかわりに、ファイルを個々にダウンロードすることを選択した場合、選択した各ファイルごとに1つのイベントがトリガーされます。

イベントに関するよくある質問

このセクションでは、イベント・フレームワーク、および Java プロセス拡張とスクリプト・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。

カスタム・プロセス拡張と Java プロセス拡張 (Java PX) にはどのような相違点がありますか。

カスタム・プロセス拡張と同様に、**Java PX** もカスタム・アクションを介して Agile PLM クライアントの機能を拡張します。実行するには、com.agile.px の IEventAction インタフェースをイベント・フレームワークに実装します。プロセス拡張を使用すると、Agile PLM サーバーと Agile PLM ユーザーは外部システムに接続できます。また、Java PX には、カスタム PX よりも詳細な情報を提供するイベント・コンテキスト・オブジェクトが含まれています。

Agile の Java API を Java 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 はカスタム PX で使用する場合と同様に使用できます。

Java PX には、特別なセキュリティ要件がありますか。

いいえ。カスタム PX と同じです。

Java PX/スクリプト PX 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 (ハンドラ) 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、ハンドラに必要な役割を Java クライアントに指定できます。ハンドラが実行されると、ハンドラに指定した役割と権限は、現在のユーザーの役割と権限より優先されます。そのハンドラが完了した時点で、クライアントはユーザーの元の役割と権限に戻ります。

ユーザーに割り当てられた役割は、設定時に Java PX またはスクリプト PX に割り当てられた役割より優先されますか。

いいえ。Java PX またはスクリプト PX に割り当てられた役割がユーザーの元の役割より優先されます。したがって、PX ハンドラ内で発生したすべてのアクションに、これらの役割に基づいた権限が適用されます。ただし、イベント・コンテキスト・オブジェクトのアクセスには、getXXX および setXXX メソッド・コールを含めて、権限チェックは必要ありません。

Java プロセス拡張は、どのように設定して配置するのですか。

カスタム PX と同様に、アプリケーション・サーバーの 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.IEventAction 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、イベント・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。

Java PX をアプリケーション・サーバーに配置した後は、どのようにその Java PX を有効にするのですか。

配置した後の Java PX コードは、Java クライアントの Agile PLM 内で使用するように設定できます。設定するには、「管理」>「設定」>「システム設定」>「イベント管理」>「イベント・ハンドラ」>「新規作成」>「イベント・ハンドラの作成」>「Java PX」>「イベント・アクション」の順に選択します。

Java PX ハンドラの JAR ファイルを配置した後に、アプリケーション・サーバー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・サーバーのクラスパスを拡張します。

イベント・フレームワークに移行できるカスタム PX はどれですか。

イベント・フレームワークで移行がサポートされているのは、カスタム・アクション PX のみです。これらは、「アクション」メニュー、「ツール・メニュー」および「ワークフローのステータスの変更」から開始される PX です。

対応するイベント・タイプは、「「アクション」メニューの拡張」、「ツール・メニューの拡張」および「ワークフローのステータスの変更」です。

開発者/ユーザーはどのようなエラー処理ルールを指定する必要がありますか。

同期ハンドラの場合は、この確認通知の処理中にエラーが発生した場合に、システムがどのように反応するかを判断するエラー処理ルールを指定する必要があります。エラー処理ルールは、同期ハンドラにのみ適用されます。「停止」と「続行」の2つの選択肢がサポートされています。

Agile PLM は、後続のイベント処理をすべて停止した後、イベントの発生元である作成者に処理を戻します。残りのすべての同期確認通知受信者は呼び出されません。

事前イベントの場合、作成者は、確認通知受信者からのエラー受信時に、アクションを開始するクライアントにエラーをスローするのみで、元のアクションは実行されません。システムによって、ハンドラによる変更がロールバックされる場合もあります。ただし、トランザクションをロールバックできるかどうかは、ハンドラ・タイプによって異なります。Java PX ハンドラの場合、ハンドラは独自のトランザクションがある SDK プログラムのため、トランザクション・ロールバックは実行されません。

スクリプト PX を配置する必要はありますか。

いいえ。スクリプトはスクリプト・ハンドラでエディタに貼り付けられ、Agile PLM によってコードがデータベースに格納されます。したがって、プログラムはオブジェクト・コードではなくプレーン・テキスト・ファイルで配信されます。

イベント・ハンドラを使用して通知を送信できますか。

はい。Java PX およびスクリプト PX から通知を送信できます。

どのような場合にスクリプトを使用する必要がありますか。

スクリプトは、プロトタイプ作成、簡単な操作およびテスト主導型の開発に使用します。

スクリプト・コードをコンパイルする必要はありますか。

いいえ。スクリプト・コードは、ハンドラへの保存時に構文エラーが検証され、イベントのトリガー時にコンパイルされます。

ダーティ・ファイル、および関連メソッドとインタフェースとはどのようなものですか。

これらについては、Oracle® E-Delivery Web サイト (<http://edelivery.oracle.com/>) の SDK サンプル・フォルダ内にある、Javadoc で生成された SDK ドキュメントの `IEventDirtyRowFileUpdate` に記載されています。

単一のアクションで複数のイベントをトリガーできますか。

はい。複数の添付ファイルの行の更新など、一部のアクションでは、単一のアクションで複数のイベントがトリガーされます。また、事前および事後のトリガー・タイプに対する確認通知受信者がいる場合、確認通知受信者の起動順は、PLM クライアント、アクションおよびオブジェクト・タイプによって異なる場合があります。たとえば、事前トリガーに対して 1 人の確認通知受信者、事後トリガーに対して 1 人の確認通知受信者が指定されている「添付ファイル」テーブルから、3 行を削除する場合、PLM クライアントの動作は次のようになります。

- **Web クライアント** - 単一の「テーブルの更新」イベントがトリガーされます。
- **Java クライアント** - 次に示すように、3 つの「テーブルの更新」イベントがトリガーされ、削除された各行に対して 1 つずつイベントがトリガーされます。

- 変更、アイテム、転送依頼、製造元部品、サプライヤ、拠点、顧客およびパッケージ・オブジェクトの場合、起動されるイベント確認通知受信者の順序は、次のとおりです。
 - 事前（事前トリガー確認通知受信者からのハンドラ - 1 行目用）
 - 事前（事前トリガー確認通知受信者からのハンドラ - 2 行目用）
 - 事前（事前トリガー確認通知受信者からのハンドラ - 3 行目用）
 - 事後（事後トリガー確認通知受信者からのハンドラ - 1 行目用）
 - 事後（事後トリガー確認通知受信者からのハンドラ - 2 行目用）
 - 事後（事後トリガー確認通知受信者からのハンドラ - 3 行目用）
- 製品サービス依頼、品質変更要求、製造元、ユーザー、ユーザー・グループの場合の順序は、事前 - 事後 - 事前 - 事後 - 事前 - 事後です。

次の操作でのベスト・プラクティスはどのようなものですか。

- 事前イベントのハンドラで回避する必要があることはありますか。

事前イベントは主に検証に使用してください。コンテキスト・オブジェクトを変更できますが、SDK コールを使用したオブジェクトの直接更新は回避してください（事前イベントで SDK コールを使用すると、オブジェクト・バージョンの不一致が生じる可能性があります）。

- 確認通知受信者はどのように順序付けすればよいですか。

場合によっては、同じイベント・タイプに対して、基本クラス・レベル、クラス・レベルおよびサブクラス・レベルでイベント確認通知受信者を設定できます。たとえば、「オブジェクトの作成」イベント・タイプに対して、アイテム基本クラス・レベルに 1 人、部品クラス・レベルに 1 人、部品サブクラス・レベルに 1 人など、複数のイベント確認通知受信者を設定できます。クラス階層に基づいて確認通知受信者を実行する場合は、基本クラス、各クラスおよび各サブクラスに対して順序の範囲を割り当てることをお勧めします。たとえば、異なる階層レベルに対して次の範囲を割り当てることができます。

基本クラス 0～99、クラス 100～199、サブクラス 200～299

- Agile SDK コールとスクリプト PX コールをスクリプト PX 内で併用できますか。

同じハンドラ・コード内で、SDK コールとスクリプト PX コールの両方を使用して Agile オブジェクトを更新していないかぎり、両方を併用できます。Agile オブジェクトを更新する場合は、SDK コール、またはスクリプト・コンテキスト・オブジェクトでサポートされている更新 API のいずれかを使用できますが、両方は使用できません。手順については、「スクリプトを使用した SDK へのアクセス」を参照してください。

Agile PLMクライアント機能とAgile APIとのマッピング

この章のトピック

| | |
|------------------|-----|
| ■ ログイン機能 | 447 |
| ■ 一般機能 | 448 |
| ■ 検索機能 | 448 |
| ■ 添付ファイル機能 | 449 |
| ■ ワークフロー機能 | 449 |
| ■ 製造拠点機能 | 450 |
| ■ フォルダ機能 | 450 |
| ■ プロジェクト機能 | 451 |
| ■ 管理機能 | 451 |

ログイン機能

次の表に、Agile アプリケーション・サーバー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・サーバー・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・サーバーにログインする | AgileSessionFactory.createSession() |
| セッションを閉じて、Agile アプリケーション・サーバー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 | <code>IAgileSession.createObject()</code> |
| 既存のオブジェクトをロードする | <code>IAgileSession.getObject()</code> |
| オブジェクトを別のオブジェクトとして保存する | <code>IDataObject.saveAs()</code> |
| オブジェクトを削除する | <code>IDataObject.delete()</code>
<code>IFolder.delete()</code>
<code>IQuery.delete()</code> |
| オブジェクトの削除を取り消す | <code>IDataObject.undelete()</code> |
| オブジェクトのセル値を取得する | <code>IDataObject.getValue()</code> |
| オブジェクトにセル値を設定する | <code>IDataObject.setValue()</code> |
| オブジェクトのテーブルを取得する | <code>IDataObject.getTable()</code> |
| テーブルに行を追加する | <code>ITable.createRow()</code> |
| テーブルから行を削除する | <code>ITable.removeRow()</code> |
| オブジェクトに対する確認通知を取得する | <code>ISubscribable.getSubscriptions()</code> |
| 確認通知イベントを有効にする | <code>ISubscription.enable()</code> |
| オブジェクトに対する確認通知を変更する | <code>ISubscribable.modifySubscriptions()</code> |

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 | <code>IQuery.setName()</code> |
| 検索をパブリックまたはプライベートにする | <code>IQuery.setQueryType()</code> |
| 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 | <code>IQuery.setSearchType()</code> |
| 検索条件を設定および取得する | <code>IQuery.setCriteria()</code>
<code>IQuery.getCriteria()</code> |
| 検索を実行する | <code>IQuery.execute()</code> |
| 検索で大文字と小文字を区別する | <code>IQuery.setCaseSensitive()</code> |
| 検索を削除する | <code>IQuery.delete()</code> |
| 検索を別の検索として保存する | <code>IQuery.saveAs()</code> |

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 | <code>IFileFolder.getFile()</code> |
| 「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 | <code>IAttachmentFile.getFile()</code> |
| ファイル・フォルダをチェックアウトする | <code>IFileFolder.checkOut()</code> |
| ファイル・フォルダをチェックインする | <code>IFileFolder.checkIn()</code> |
| チェックアウトをキャンセルする | <code>IFileFolder.cancelCheckOut()</code> |
| 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 | <code>IAttachmentContainer.setIncorporated()</code> |

ワークフロー機能

次の表に、Agile PLM のルーティング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 |
|---------------------------------------|---|
| ルーティング可能なオブジェクトを検証する | <code>IRoutable.audit()</code> |
| ルーティング可能なオブジェクトのステータスを変更する | <code>IRoutable.changeStatus()</code> |
| オブジェクトを他の Agile PLM ユーザーに送信する | <code>IDataObject.send()</code> |
| ルーティング可能なオブジェクトを承認する | <code>IRoutable.approve()</code> |
| ルーティング可能なオブジェクトを却下する | <code>IRoutable.reject()</code> |
| ルーティング可能なオブジェクトにコメントする | <code>IRoutable.comment()</code> |
| ルーティング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 | <code>IRoutable.addApprovers()</code>
<code>IRoutable.removeApprovers()</code> |

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 | <code>IManufacturingSiteSelectable.getManufacturingSite()</code> |
| アイテムに対するすべての製造拠点を取得する | <code>IManufacturingSiteSelectable.getManufacturingSites()</code> |
| 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 | <code>IManufacturingSiteSelectable.setManufacturingSite(ManufacturingSiteConstants.ALL_SITES)</code> |
| 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 | <code>IManufacturingSiteSelectable.setManufacturingSite(ManufacturingSiteConstants.COMMON_SITE)</code> |
| 特定の製造拠点を使用するようにアイテムを設定する | <code>IManufacturingSiteSelectable.setManufacturingSite(site)</code> |

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 | <code>IFolder.addChild()</code> |
| 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 | <code>IFolder.setFolderType()</code> |
| フォルダ名を設定する | <code>IFolder.setName()</code> |
| 現在のユーザーのフォルダを取得する | <code>IUser.getFolder()</code> |
| フォルダからアイテムを削除する | <code>IFolder.removeChild()</code> |
| フォルダからすべてのオブジェクトをクリアする | <code>IFolder.clear()</code> |
| フォルダを削除する | <code>IFolder.delete()</code> |

プロジェクト機能

次の表に、プロジェクト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 |
|---------------------------------|---|
| プロジェクトを別のプロジェクトまたはテンプレートとして保存する | <code>IProgram.saveAs()</code> |
| プロジェクトを再スケジュールする | <code>IProgram.reschedule()</code> |
| リソース・プールからユーザーを割り当てる | <code>IProgram.assignUsersFromPool()</code> |
| プロジェクトの所有権を別のユーザーに委譲する | <code>IProgram.delegateOwnership()</code> |
| プロジェクト・リソースを入れ替える | <code>IProgram.substituteResource()</code> |
| 基準を作成する | <code>IProgram.createBaseline()</code> |
| プロジェクトの基準表示を選択する | <code>IProgram.selectBaseline()</code> |
| プロジェクトをロックまたはロック解除する | <code>IProgram.setLock()</code> |
| ディスカッションに返信する | <code>IMessage.reply()</code> |

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 | <code>IAdmin.getNode()</code> |
| 管理ノードのすべてのサブノード（子ノード）を取得する | <code>ITreeNode.getChildNodes()</code> |
| 管理ノードのすべてのプロパティを取得する | <code>INode.getProperties()</code> |
| 管理ノードのプロパティに対する値を取得する | <code>IProperty.getValue()</code> |
| リスト・フィールドに対する可能な値を取得する | <code>IProperty.getAvailableValues()</code> |
| すべての Agile PLM クラスを取得する | <code>IAdmin.getAgileClasses(ALL)</code> |
| トップレベルのすべての Agile PLM クラスを取得する | <code>IAdmin.getAgileClasses(TOP)</code> |
| インスタンス化可能なすべての Agile PLM クラスを取得する | <code>IAdmin.getAgileClasses(CONCRETE)</code> |
| 特定のクラスに対するサブクラスのリストを取得する | <code>IAgileClass.getSubclasses()</code> |
| サブクラスの自動採番ソースを取得する | <code>IAgileClass.getAutoNumberSources()</code> |
| テーブルに対する属性の配列を取得する | <code>IAgileClass.getTableAttributes()</code> |
| テーブルのメタデータを取得する | <code>IAgileClass.getTableDescriptor()</code> |
| Agile PLM リスト・ライブラリを取得する | <code>IAdmin.getListLibrary()</code> |

| 機能 | 対応するメソッド |
|-------------------------------|---|
| 新しい Agile PLM リストを作成する | <code>IListLibrary.createAdminList()</code> |
| Agile PLM リストを取得する | <code>IListLibrary.getAdminList()</code> |
| 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 | <code>IAgileSession.createObject()</code> |
| 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 | <code>IProperty.setValue()</code> |
| ユーザー・パスワードを変更する | <code>IUser.changeApprovalPassword()</code>
<code>IUser.changeLoginPassword()</code> |

リリース 9.2.1 以前のテーブル定数の リリース 9.2.2 以降への移行

この章のトピック

- 9.2.2 テーブル定数にマップされたプレリリース 9.2.2 のテーブル定数 453
- 削除されたプレリリース 9.2.2 のテーブル定数 456

「関係」テーブルのマージおよび置換に関する情報は、71ページの「[新規およびマージされた「関係」テーブルへのアクセス](#)」で先に説明しました。この付録の各表では、リリース 9.2.2 のテーブル定数と、単一のテーブル定数にマージおよびマップされるか、または新規テーブル定数にマップされるテーブル定数を示します。

9.2.2 テーブル定数にマップされたプレリリース 9.2.2 の テーブル定数

次の表に、プレリリース 9.2.2 のテーブル定数と、リリース 9.2.2 にマージおよびマップされるか、SDK の以降のリリースにマップされる新規テーブル定数を一覧表示します。

| プレ 9.2.2 のテーブル定数 | 9.2.2 のテーブル定数 |
|--|----------------------------------|
| <ul style="list-style-type: none"> TABLE_RELATIONSHIPS_AFFECTED_BY TABLE_RELATIONSHIPS_AFFECTS TABLE_REFERENCES | TABLE_RELATIONSHIPS |
| <ul style="list-style-type: none"> ATT_RELATIONSHIPS_AFFECTED_BY_CRITERIA_MET ATT_RELATIONSHIPS_AFFECTS_CRITERIA_MET | ATT_RELATIONSHIPS_CRITERIA_MET |
| <ul style="list-style-type: none"> ATT_RELATIONSHIPS_AFFECTED_BY_CURRENT_STATUS ATT_RELATIONSHIPS_AFFECTS_CURRENT_STATUS | ATT_RELATIONSHIPS_CURRENT_STATUS |
| <ul style="list-style-type: none"> ATT_RELATIONSHIPS_AFFECTED_BY_DATE01 ATT_RELATIONSHIPS_AFFECTS_DATE01 | ATT_RELATIONSHIPS_DATE01 |
| <ul style="list-style-type: none"> ATT_RELATIONSHIPS_AFFECTED_BY_DATE02 ATT_RELATIONSHIPS_AFFECTS_DATE02 | ATT_RELATIONSHIPS_DATE02 |
| <ul style="list-style-type: none"> ATT_RELATIONSHIPS_AFFECTED_BY_DATE03 ATT_RELATIONSHIPS_AFFECTS_DATE03 | ATT_RELATIONSHIPS_DATE03 |
| <ul style="list-style-type: none"> ATT_RELATIONSHIPS_AFFECTED_BY_DATE04 ATT_RELATIONSHIPS_AFFECTS_DATE04 | ATT_RELATIONSHIPS_DATE04 |
| <ul style="list-style-type: none"> ATT_RELATIONSHIPS_AFFECTED_BY_DATE05 ATT_RELATIONSHIPS_AFFECTS_DATE05 | ATT_RELATIONSHIPS_DATE05 |

| | |
|--|-------------------------------|
| ▫ ATT_REFERENCES_DATE01 | ATT_RELATIONSHIPS_DATE06 |
| ▫ ATT_REFERENCES_DATE02 | ATT_RELATIONSHIPS_DATE07 |
| ▫ ATT_REFERENCES_DATE03 | ATT_RELATIONSHIPS_DATE08 |
| ▫ ATT_REFERENCES_DATE04 | ATT_RELATIONSHIPS_DATE09 |
| ▫ ATT_REFERENCES_DATE05 | ATT_RELATIONSHIPS_DATE10 |
| ▫ ATT_RELATIONSHIPS_AFFECTED_BY_DESCRIPTION
▫ ATT_RELATIONSHIPS_AFFECTS_DESCRIPTION
▫ ATT_REFERENCES_DESCRIPTION | ATT_RELATIONSHIPS_DESCRIPTION |
| ▫ ATT_RELATIONSHIPS_AFFECTED_BY_LIST01、
ATT_RELATIONSHIPS_AFFECTS_LIST01 | ATT_RELATIONSHIPS_LIST01 |
| ▫ ATT_RELATIONSHIPS_AFFECTED_BY_LIST02、
ATT_RELATIONSHIPS_AFFECTS_LIST02 | ATT_RELATIONSHIPS_LIST02 |
| ▫ ATT_RELATIONSHIPS_AFFECTED_BY_LIST03
▫ ATT_RELATIONSHIPS_AFFECTS_LIST03 | ATT_RELATIONSHIPS_LIST03 |
| ▫ ATT_RELATIONSHIPS_AFFECTED_BY_LIST04
▫ ATT_RELATIONSHIPS_AFFECTS_LIST04 | ATT_RELATIONSHIPS_LIST04 |
| ▫ ATT_RELATIONSHIPS_AFFECTED_BY_LIST05
▫ ATT_RELATIONSHIPS_AFFECTS_LIST05 | ATT_RELATIONSHIPS_LIST05 |
| ▫ ATT_REFERENCES_LIST01 | ATT_RELATIONSHIPS_LIST06 |
| ▫ ATT_REFERENCES_LIST02 | ATT_RELATIONSHIPS_LIST07 |
| ▫ ATT_REFERENCES_LIST03 | ATT_RELATIONSHIPS_LIST08 |
| ▫ ATT_REFERENCES_LIST04 | ATT_RELATIONSHIPS_LIST09 |
| ▫ ATT_REFERENCES_LIST05 | ATT_RELATIONSHIPS_LIST10 |
| ▫ ATT_RELATIONSHIPS_AFFECTED_BY_MULTITEXT01
▫ ATT_RELATIONSHIPS_AFFECTS_MULTITEXT01 | ATT_RELATIONSHIPS_MULTITEXT01 |
| ▫ ATT_RELATIONSHIPS_AFFECTED_BY_MULTITEXT02
▫ ATT_RELATIONSHIPS_AFFECTS_MULTITEXT02 | ATT_RELATIONSHIPS_MULTITEXT02 |
| ▫ ATT_RELATIONSHIPS_AFFECTED_BY_MULTITEXT03
▫ ATT_RELATIONSHIPS_AFFECTS_MULTITEXT03 | ATT_RELATIONSHIPS_MULTITEXT03 |
| ▫ ATT_RELATIONSHIPS_AFFECTED_BY_MULTITEXT04
▫ ATT_RELATIONSHIPS_AFFECTS_MULTITEXT04 | ATT_RELATIONSHIPS_MULTITEXT04 |
| ▫ ATT_RELATIONSHIPS_AFFECTED_BY_MULTITEXT05
▫ ATT_RELATIONSHIPS_AFFECTS_MULTITEXT05 | ATT_RELATIONSHIPS_MULTITEXT05 |
| ▫ ATT_REFERENCES_MULTITEXT01 | ATT_RELATIONSHIPS_MULTITEXT06 |
| ▫ ATT_REFERENCES_MULTITEXT02 | ATT_RELATIONSHIPS_MULTITEXT07 |
| ▫ ATT_REFERENCES_MULTITEXT03 | ATT_RELATIONSHIPS_MULTITEXT08 |

| | |
|---|-------------------------------|
| ▫ ATT_REFERENCES_MULTITEXT04 | ATT_RELATIONSHIPS_MULTITEXT09 |
| ▫ ATT_REFERENCES_MULTITEXT05 | ATT_RELATIONSHIPS_MULTITEXT10 |
| ▫ ATT_RELATIONSHIPS_AFFECTED_BY_TEXT01
▫ ATT_RELATIONSHIPS_AFFECTS_TEXT01 | ATT_RELATIONSHIPS_TEXT01 |
| ▫ ATT_RELATIONSHIPS_AFFECTED_BY_TEXT02
▫ ATT_RELATIONSHIPS_AFFECTS_TEXT02 | ATT_RELATIONSHIPS_TEXT02 |
| ▫ ATT_RELATIONSHIPS_AFFECTED_BY_TEXT03、
ATT_RELATIONSHIPS_AFFECTS_TEXT03 | ATT_RELATIONSHIPS_TEXT03 |
| ▫ ATT_RELATIONSHIPS_AFFECTED_BY_TEXT04、
ATT_RELATIONSHIPS_AFFECTS_TEXT04 | ATT_RELATIONSHIPS_TEXT04 |
| ▫ ATT_RELATIONSHIPS_AFFECTED_BY_TEXT05
▫ ATT_RELATIONSHIPS_AFFECTS_TEXT05 | ATT_RELATIONSHIPS_TEXT05 |
| ▫ ATT_REFERENCES_TEXT01 | ATT_RELATIONSHIPS_TEXT06 |
| ▫ ATT_REFERENCES_TEXT02 | ATT_RELATIONSHIPS_TEXT07 |
| ▫ ATT_REFERENCES_TEXT03 | ATT_RELATIONSHIPS_TEXT08 |
| ▫ ATT_REFERENCES_TEXT04 | ATT_RELATIONSHIPS_TEXT09 |
| ▫ ATT_REFERENCES_TEXT05 | ATT_RELATIONSHIPS_TEXT10 |
| ▫ ATT_RELATIONSHIPS_AFFECTED_BY_NOTES
▫ ATT_RELATIONSHIPS_AFFECTS_NOTES | ATT_RELATIONSHIPS_NOTES1 |
| ▫ ATT_REFERENCES_NOTES | ATT_RELATIONSHIPS_NOTES2 |
| ▫ ATT_RELATIONSHIPS_AFFECTED_BY_NUMBER
▫ ATT_RELATIONSHIPS_AFFECTS_NUMBER
▫ ATT_REFERENCES_NUMBER | ATT_RELATIONSHIPS_NAME |

削除されたプレリリース 9.2.2 のテーブル定数

次のプレリリース 9.2.2 のテーブル定数は使用できなくなりました。以降のリリースの SDK では使用しないでください。

- ▣ `ATT_RELATIONSHIPS_AFFECTED_BY_EVENT`
- ▣ `ATTRELATIONSHIPS_AFFECTS_TRIGGER_EVENT`
- ▣ `ATT_RELATIONSHIPS_AFFECTED_BY_TRIGGER_EVENT`
- ▣ `ATT__RELATIONSHIPS_AFFECTS_EVENT`
- ▣ `ATT_RELATIONSHIPS_AFFECTS_RESULT`
- ▣ `MaterialDeclarationConstants.TABLE_RELATIONSHIPSAFFECTEDBY`
- ▣ `MaterialDeclarationConstants.TABLE_RELATIONSHIPSAFFECTS`
- ▣ `MaterialDeclarationConstants.TABLE_REFERENCES`

イベント・フレームワークへのカスタム・プロセス拡張の移行

この章のトピック

- この付録について 457
- カスタムPXとJava PXの理解 457
- 移行タスク・リスト 458

この付録について

この付録には、次の情報が記載されています。

1. 「プロセス拡張の開発」で作成したカスタムPX Java コードを、イベント・フレームワークで使えるように変更する方法
2. 変更したコードがイベント・フレームワークでJava PXとして機能するように設定する方法

カスタムPXとJava PXの理解

次に、これらのPXの相違点について説明し、イベント・フレームワークに移行して、405ページの「[イベントの使用](#)」でJava PXとして設定できるカスタムPXを示します。

PXフレームワークでのカスタムPX

「プロセス拡張の開発」で定義および設定したカスタムPXは、Agileアプリケーション・サーバーに配置されるJavaクラス、またはURLへのリンクのいずれかです。これらのPXは、PXフレームワークで実行されます。3ページの「[SDKのアーキテクチャ](#)」の図を参照してください。JavaクラスのカスタムPXには、次のタイプがあります。

- com.agile.px パッケージのサーバー側 Java API である ICustomAction インタフェースを実装したカスタム・アクション PX
- com.agile.px パッケージのサーバー側 Java API である ICustomAutoNumber インタフェースを実装したカスタム自動採番ソース PX

イベント・フレームワークでのプロセス拡張

405ページの「[イベントの使用](#)」で設定したPXは、次のいずれかのタイプです。

- com.agile.px パッケージのサーバー側 API である IEventAction インタフェースを実装した Java プロセス拡張 (Java PX)

- `com.agile.agile.DSLObj` パッケージのサーバー側 API である `invokeScript (IBaseScriptObj obj)` インタフェースを実装したスクリプト・プロセス拡張 (スクリプト PX)

注意 すべての Java プロセス拡張が正常に動作するように、使用中のアプリケーション・サーバーに同梱されている JDK のバージョンで、既存の Java ベースの PX を再コンパイルすることをお勧めします。

イベント・フレームワークに移行可能なカスタムPX

イベント・フレームワークで移行がサポートされているのは、カスタム・アクション PX のみです。これらは、「アクション」メニュー、「ツール・メニュー」および「ワークフローのステータス」から開始される PX です。「プロセスの拡張ライブラリの使用」を参照してください。

対応するイベント・タイプは、次のとおりです。

| カスタム PX | Java PX |
|--------------|-----------------|
| 「アクション」メニュー | 「アクション」メニューの拡張 |
| ツール・メニュー | ツール・メニューの拡張 |
| ワークフローのステータス | ワークフローのステータスの変更 |

移行タスク・リスト

カスタム PX をイベント・フレームワークに正しく移行し、この環境で Java PX として正常に動作するように、次のタスクを完了してください。

タスク 1: カスタムPXコードの変更

次のコード・サンプルは、既存のカスタム PX コードをイベント・フレームワークの Java PX コードに変更する方法を示しています。Java PX とカスタム PX の主な相違点は、PX で実装する必要があるインタフェースです。Java PX は `IEventAction` を実装し、カスタム PX は `ICustomAction` を実装します。

カスタムPXコード

次のコードは、「カスタム・アクションの定義」で示した既存のカスタムPXの例です。カスタムPXを移行するには、459ページの「[Java PXコード](#)」に示すとおりコードを変更する必要があります。

例: カスタム PX

```
public class HelloWorld implements ICustomAction
{
    public ActionResult doAction(IAgileSession session, INode
        actionNode, IDataObject affectedObject)
    {
        ...

        return new ActionResult(ActionResult.STRING, "Hello World");
    }
}
```


Java PXコード

次の例は、イベント・フレームワークで実行するように変更したカスタム PX です。この移行を可能にしたコードの変更部分を太字で示しています。

例: 変更後のカスタム PX コード

```
public class HelloWorld implements IEventAction
{
    public EventActionResult doAction(IAgileSession session, INode
    actionNode, IEventInfo request)
    {
        IObjectEventInfo objectEventInfo = (IObjectEventInfo) request;
        IDataObject affectedObject = objectEventInfo.getDataObject();

        ...

        ActionResult actionResult = new ActionResult(ActionResult.STRING,
        "Hello World");
        return new EventActionResult(request, actionResult);
    }
}
```

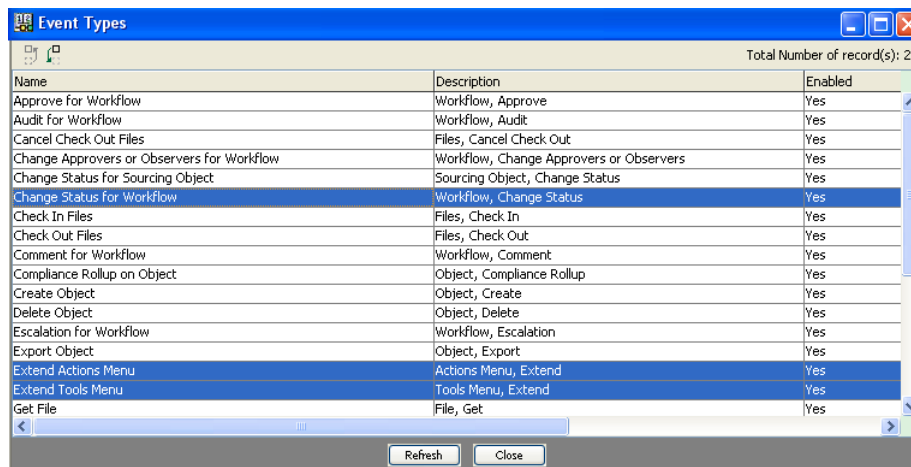
注意 Java PX は、IDataObject の値を IEventInfo から取得します。

タスク 2: 変更したコードのパッケージ化および配置

JARファイルを作成して、変更したJavaコードをイベント・フレームワークでできるようにパッケージ化して配置します。手順については、355ページの「[カスタム・アクションのパッケージ化および配置](#)」を参照してください。このパッケージは、459ページの「[タスク 3: イベント・フレームワークでのイベントの設定](#)」を完了するために使用するアクションです。

タスク 3: イベント・フレームワークでのイベントの設定

「イベント・タイプ」ダイアログには、イベント・フレームワークでサポートされているイベント・タイプが一覧表示されます。移行中のカスタムPXに対してサポートされているタイプは、「ワークフローのステータスの変更」、「「アクション」メニューの拡張」および「ツール・メニューの拡張」です。次に、これらのイベントの作成および設定手順について説明します。詳細は、405ページの「[イベントの使用](#)」および『Agile PLM 管理者ガイド』を参照してください。



| Name | Description | Enabled |
|--|---|---------|
| Approve for Workflow | Workflow, Approve | Yes |
| Audit for Workflow | Workflow, Audit | Yes |
| Cancel Check Out Files | Files, Cancel Check Out | Yes |
| Change Approvers or Observers for Workflow | Workflow, Change Approvers or Observers | Yes |
| Change Status for Sourcing Object | Sourcing Object, Change Status | Yes |
| Change Status for Workflow | Workflow, Change Status | Yes |
| Check In Files | Files, Check In | Yes |
| Check Out Files | Files, Check Out | Yes |
| Comment for Workflow | Workflow, Comment | Yes |
| Compliance Rollup on Object | Object, Compliance Rollup | Yes |
| Create Object | Object, Create | Yes |
| Delete Object | Object, Delete | Yes |
| Escalation for Workflow | Workflow, Escalation | Yes |
| Export Object | Object, Export | Yes |
| Extend Actions Menu | Actions Menu, Extend | Yes |
| Extend Tools Menu | Tools Menu, Extend | Yes |
| Get File | File, Get | Yes |

イベントの作成

イベントを作成する手順は、次のとおりです。


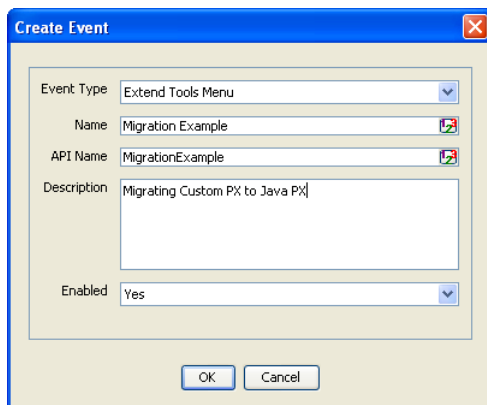
1. 管理者権限で Java クライアントにログインします。
2. 「管理」>「システム設定」>「イベント管理」>「イベント」>「新規作成」ボタン  の順に選択します。「イベントの作成」ダイアログが表示されます。
3. 「イベント・タイプ」で、ドロップダウンの矢印をポイントして、リストからイベントを選択します。

図34: 「ツール・メニューの拡張」イベントの作成



Create Event

Event Type: Extend Tools Menu

Name: Migration Example

API Name: MigrationExample

Description: Migrating Custom PX to Java PX

Enabled: Yes

OK Cancel

図35: ワークフローのステータスの変更

The 'Create Event' dialog box is shown with the following fields:

- Event Type: Change Status for Workflow
- Name: (empty)
- API Name: (empty)
- Description: (empty text area)
- Enabled: Yes
- Workflow: Default Change Orders
- Object Type: Change Orders
- Status - From: (empty)
- Status - To: (empty)

Buttons: OK, Cancel

図36: 「アクション」メニューの拡張

The 'Create Event' dialog box is shown with the following fields:

- Event Type: Extend Actions Menu
- Name: (empty)
- API Name: (empty)
- Description: (empty text area)
- Enabled: Yes
- Object Type: Changes (dropdown menu is open showing a list of options)

Buttons: OK, Cancel

Object Type dropdown menu options:

- Changes
- Change Orders
- ECO
- ECO2
- ECO3
- Change Requests
- ECR
- ECR2

注意 「イベントの作成」ダイアログのフィールドは、すべてのイベント・タイプに対して同じではありません。たとえば、「タイトル・ブロックの更新」ではオブジェクトのクラスを割り当てますが、「ワークフローの承認」ではワークフローのステータスを選択します。オブジェクト・クラスの割当ておよびワークフロー・ステータスの詳細は、「クラスへのプロセス拡張の割当て」および「ワークフロー・ステータスへのプロセス拡張の割当て」を参照してください。

4. イベント・タイプを選択して、必要な情報を指定します。たとえば、次に示すように指定します。このダイアログを完了する方法の詳細は、『Agile PLM 管理者ガイド』を参照してください。

図37: 「ツール・メニューの拡張」イベントの作成

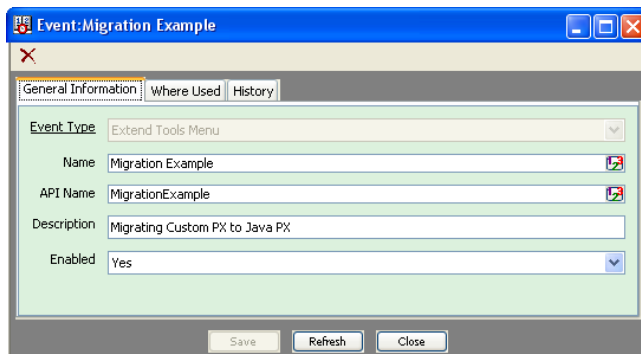
The 'Create Event' dialog box is shown with the following fields:

- Event Type: Extend Tools Menu
- Name: Migration Example
- API Name: MigrationExample
- Description: Migrating Custom PX to Java PX
- Enabled: Yes

Buttons: OK, Cancel

5. 「OK」をクリックします。「イベント:<Event_Name>」ページが表示されます。

図38: イベントの「一般情報」ページ



このダイアログからイベントを変更できます。変更を加えると、「保存」ボタンが有効になります。また、このイベントが「イベント」ビューにリストされます。表示するには、「イベント管理」>「イベント」の順に選択します。次のタスクは、このイベントのハンドラを作成することです。

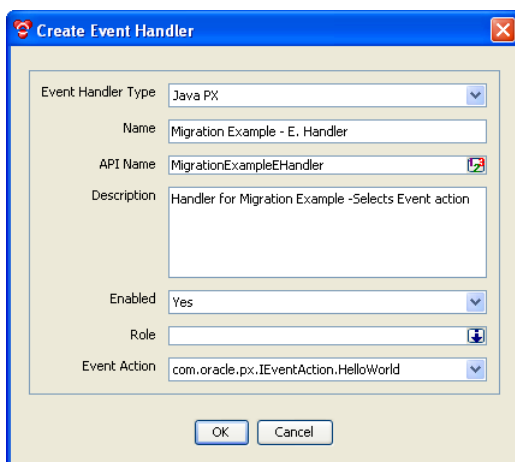
イベント・ハンドラの作成

イベント・ハンドラによって、コンパイルした Java コードの実行が可能になります。次に、この手順について説明します。

イベント・ハンドラを作成する手順は、次のとおりです。

1. 管理者権限で Java クライアントにログインし、「管理」>「システム設定」>「イベント管理」>「イベント・ハンドラ」>「新規作成」ボタン⁵⁴の順に選択します。「イベント・ハンドラの作成」ダイアログが表示されます。

図39: 「イベント・ハンドラの作成」ダイアログ



2. 「イベント・ハンドラの作成」ダイアログで、次の処理を実行します。
 - 「イベント・ハンドラ」ドロップダウン・リストで、「Java PX」を選択します。
 - 「イベント・アクション」フィールドで、459ページの「[タスク 2: 変更したコードのパッケージ化および配置](#)」で作成したイベント・アクションを選択します。
 - 「役割」フィールドが空白の場合、イベント・ハンドラは、現在のユーザーの役割と権限をデフォルト

トで使用します。ただし、カスタム・アクションにはオーバーライド権限も設定できます。詳細および手順については、『Agile PLM 管理者ガイド』を参照してください。

- 残りのフィールドを完成し、「OK」をクリックします。

イベント確認通知受信者の作成

これは、Java PX を特定のイベントにバインドするプロセスです。バインドするには、「イベント確認通知受信者の作成」ダイアログで次の処理を実行します。

- イベントをイベント・ハンドラにバインドします。
- トリガー順（順序）を設定します。
- 実行モードを設定します。

イベント確認通知受信者を作成する手順は、次のとおりです。


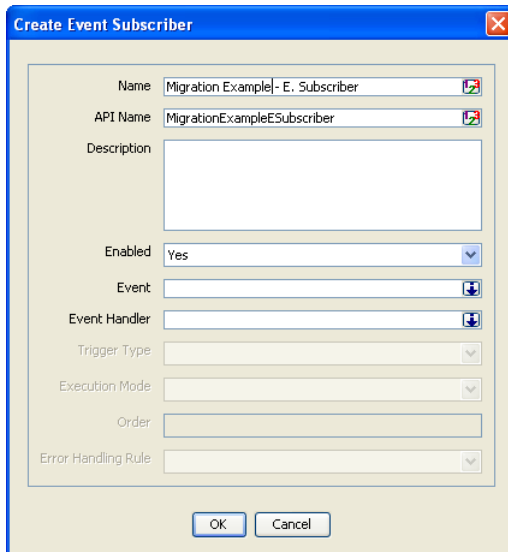

1. 管理者権限で Java クライアントにログインし、「管理」>「システム設定」>「イベント管理」>「イベント確認通知受信者」>「新規作成」ボタン  の順に選択します。「イベント確認通知受信者の作成」ダイアログが表示されます。

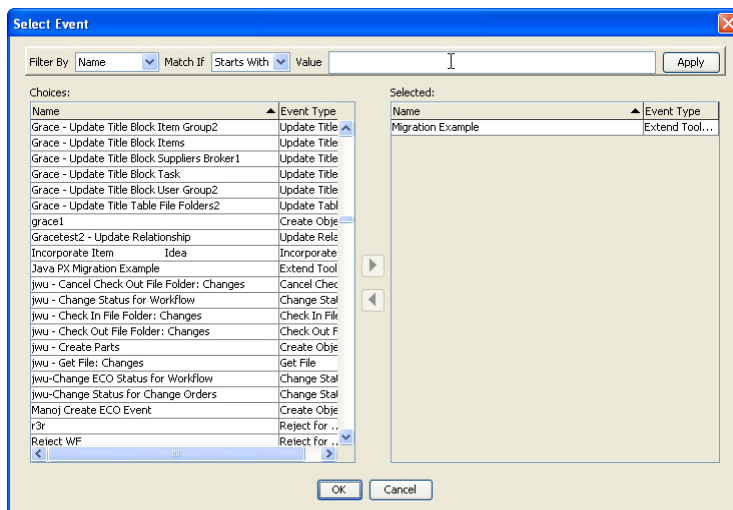
図40: 「イベント確認通知受信者の作成」ダイアログ



2. 「イベント確認通知受信者の作成」ダイアログで、次の処理を実行します。
 - このイベント確認通知受信者のイベントを選択します。「イベント」フィールドのドロップダウン・リスト  をクリックします。「イベントの選択」ダイアログが表示されます。

- 「イベントの選択」ダイアログで、適用可能なイベントを探して選択し、「選択済」列に移動します。

図41: イベント確認通知受信者のイベントの選択




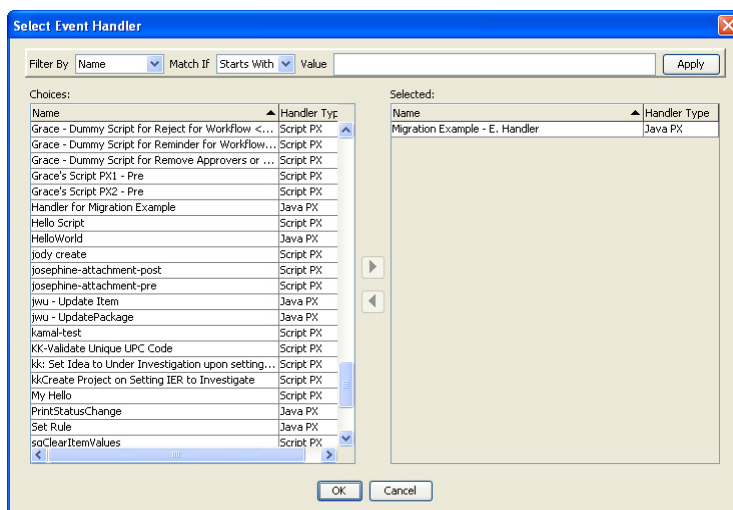
- このイベント確認通知受信者のイベント・ハンドラを選択します。「イベント・ハンドラ」フィールドのドロップダウン・リスト  をクリックします。「イベント・ハンドラの選択」ダイアログが表示されます。「イベント・ハンドラの選択」ダイアログで、適用可能なイベント・ハンドラを探して選択し、「選択済」列に移動します。

図42: 確認通知受信者のイベント・ハンドラの選択



注意 この手順が完了すると、「イベント確認通知受信者の作成」ダイアログで灰色表示されていた4つのフィールドのいくつかが設定可能になります。たとえば、「トリガー・タイプ」と「実行モード」が設定可能になります。

トリガー・タイプ、実行モード、順序およびエラー処理ルールの設定

次に、設定が必要なイベント・トリガー・タイプとイベント実行モード、およびイベントの起動順序とエラー処理ルールの各オプションについて説明します。

「トリガー・タイプ」フィールド

このフィールドには、次の2つのオプションがあります。

- **事前** - このトリガー・タイプは、アクション発生前の時点を含むします。「事前」トリガーは、一般的に、データが必要なイベントや、次のアクションに他のパラメータが必要なイベントに使用されます。このトリガー・タイプを使用して設定されたイベント確認通知受信者は、「同期」実行モードでのみ実行されません。
- **事後** - このトリガー・タイプは、アクションの発生直後の時点を含むします。このトリガーは、前のアクションに基づいて監査タスクを実行するイベントに使用されます。このトリガー・タイプを使用して設定されたイベント確認通知受信者は、「同期」または「非同期」のいずれの実行モードでも実行できます。

注意 移行したカスタム PX の場合は、常に「事後」を選択してください。カスタム PX は常に、アクションの発生後に実行されます。

「実行モード」フィールド

このフィールドには、「同期」オプションと「非同期」オプションがあります。一般的に、同期とは同時に発生することを意味し、非同期とは同時に発生しないことを意味します。同期操作は操作が完了するまでプロセスをブロックし、非同期動作はプロセスをブロックせず単に操作を開始します。

Agile PLM での2つのオプションの相違は、次のとおりです。

- **同期** - このモードでは、イベント・ハンドラは、イベント（たとえば、ワークフロー・ステータスの変更）をトリガーする Agile PLM スレッドと同じスレッドで実行されます。元の Agile PLM アクションは、ハンドラ・アクションの完了後に再開されます（ブロック）。

注意 移行した PX の場合は、常に「同期」を選択してください。

- **非同期** - このモードでは、イベント・ハンドラにそれ自体のスレッドがあり、1度開始すると停止できません。このトランザクションは、それ自体のステータスに基づいてコミットまたはロールバックされます。イベントをトリガーする Agile PLM スレッドは、ハンドラ・アクションが完了しているかどうかに関係なく（ブロックされずに）、引き続き単独で実行されます。

「順序」フィールド

「順序」フィールドは、イベント・ハンドラが起動される順序を決定する正の整数です。このフィールドは、同じ Agile オブジェクトの同じイベント・タイプに対して複数のイベント確認通知受信者がいる場合に役立ちます。

注意 カスタム PX と Java PX の両方を1つのワークフロー・ステータス変更アクションに設定した場合は、常に Java PX がカスタム PX の前に実行されます。

エラー処理ルール

このフィールドは、「同期」実行モードの場合にのみ、ユーザーが設定します。オプションは、「続行」（デフォルト値）と「停止」です。選択したオプションによって、イベント確認通知受信者の処理中にエラーが発生した場合の Agile PLM の動作が決まります。エラー処理ルールの詳細は、『Agile PLM 管理者ガイド』を参照してください。

- **続行** - このオプションを選択すると、エラーは無視され、イベントは残りの確認通知の処理を続行します。

- **停止** - このオプションは、後続のイベント処理を停止し、イベントの発生元である作成者に処理を戻します。

注意 移行した PX の場合は、「**続行**」を選択してください。

例: 移行したPXに対するイベント確認通知受信者の設定

1. Java クライアントで、「**管理**」>「**イベント管理**」>「**イベント確認通知受信者**」>「**新規作成**」ボタン¹⁾の順に選択します。「イベント確認通知受信者」ダイアログが表示されます。
2. 「**トリガー・タイプ**」、「**実行モード**」および「**エラー処理ルール**」の各フィールドのオプションを、次のように設定します。

これらは、移行した PX に対する推奨オプションです。

図43: イベント確認通知受信者

タスク 4: 移行したPXのイベント・フレームワークでのテスト

新しい Java PX を起動して、ハンドラ（コード）で指定したアクションが発生することを確認します。イベント・タイプに応じて、ユーザーは、「**アクション**」メニューの拡張、「**ツール・メニューの拡張**」および「**ワークフローのステータスの変更**」から Java PX を起動できます。移行した PX のイベント・フレームワークでの動作が、PX フレームワークでの動作と同じであることを確認してください。

PX を「**ツール**」メニューからのユーザー・アクションで起動するように設定している場合は、Web クライアントまたは Java クライアントで次のようにテストできます。

- Web クライアントのツールバーで、 ボタン>「<イベント名>」の順に選択します。
- Java クライアントのツールバーで、「**プロセス拡張**」 ボタンを選択するか、「**ツール**」>「**プロセス拡張**」>「<イベント名>」の順に選択します。

タスク 5: プロセス拡張ライブラリからのカスタムPXの削除

移行したカスタム PX は、Agile のプロセス拡張ライブラリから削除することをお勧めします。削除することで、カスタム PX と Java PX が重複して実行されることを防止します。

カスタムPXをPXライブラリから削除する手順は、次のとおりです。

1. カスタム PX に対するすべての参照を削除します。これは、使用中のカスタム PX は削除できないためです。
2. Java クライアントの**プロセス拡張ライブラリ**を開き、PX を選択して削除します。

タスク 6: PLM管理者への通知

459ページの「[タスク 3: イベント・フレームワークでのイベントの設定](#)」に示したように、イベント設定は、管理者ユーザーがJavaクライアントで実行するUIベースの管理者機能です。SDK開発者としての役割に応じて、次のように、PLM管理者に必ず通知してください。

- イベント確認通知受信者の作成と設定がPLM管理者機能の場合は、459ページの「[タスク 2: 変更したコードのパッケージ化および配置](#)」の変更したカスタムPXコードを配置した後で、必ず管轄の管理者に通知してください。これは、新しいイベント・ハンドラとそれを使用するための仕様をPLM管理者に通知して、残りのタスクを完了するためです。
- 一方、自分が Java クライアントで UI ベースの設定を実行する場合は、新しい Java PX、その目的と機能、および新しい Java PX を PLM クライアントで使用するために必要な情報を PLM 管理者に通知します。

イベント・フレームワークでのGroovyの実装

この章のトピック

- この付録について 469
- Groovyとは..... 469
- イベント・フレームワークの実装..... 470

この付録について

この付録では、Groovy スクリプト言語とこのツールの情報源、イベント・フレームワークでの Groovy の実装、Java PX とスクリプト PX の配置の相違点と実装プリファレンス、ユースケースについて説明します。

Groovyとは

Groovy は、Java プラットフォーム用のスクリプト言語として使用できるオブジェクト指向のプログラミング言語です。

スクリプトPXとJava PXの使用目的

スクリプトは、簡単なビジネス・ロジックを伴うアプリケーションの迅速な開発と配置に適しています。スクリプトを使用すると、Agile PLM 管理者およびパワー・ユーザーは、自分の要件に特有の拡張機能を開発し、必要に応じて迅速に変更を加えることができます。スクリプトは、パフォーマンスに影響を与えるデータ構造を使用した複雑なアプリケーションの開発には適していません。

スクリプトは、次の目的で使用します。

- データ検証、通知、フィールド値のデフォルト値設定など、簡単なビジネス・ロジック伴う機能の自動化
- 既存のアプリケーションに対する特有のカスタマイズの実装
- 既存のシステムに対する拡張機能の構築
- 迅速なプロトタイプ作成
- テスト・ユースケースの作成

情報源

World Wide Web には、Groovy に関する情報を提供する多数のサイトへのリンクが用意されています。印刷媒体の出版社やベンダーもこのツールに関する情報を提供しています。次にいくつか例を示します。

- World Wide Web からの情報の入手
 - **Groovyホーム** - ドキュメント、ダウンロード、チュートリアル、ユーザー・ガイド、Eclipseプラグインの例、上級使用ガイド、およびGroovyホーム (<http://groovy.codehaus.org/>) で管理されるその他のサイトへのリンクがあります。

- 出版社およびベンダーからの情報の入手
 - 出版社: **Manning Publications** - 『Groovy in Action』。著者: Dierk Koenig、Andrew Glover、Paul King および Guillaume Laforge。
 - 出版社: **Morgan Kaufmann** - 『Groovy Programming: An Introduction for Java Developers』。著者: Kenneth Barclay および John Savage。

イベント・フレームワークの実装

主な実装ファクトは、次のとおりです。

- スクリプト・エンジンは、Groovy 言語に基づいて、Agile PLM サーバー上の Java 2 Platform Enterprise Edition (J2EE) 内で動作します。
- Groovy は、イベント・フレームワークに完全に埋め込まれます。
- Groovy は、サポートされている唯一のスクリプト言語です。
- スクリプト・コードは現在、Agile PLM データベースの CLOB フィールドに格納されています。
- 開発するイベント・スクリプト・オブジェクト（イベント・スクリプト PX ハンドラ）はテキスト・ファイルです。このファイルは、Java クライアントで「イベント管理」>「イベント・ハンドラ」の順に選択して、イベント管理ノードから配置します。
- スクリプトは、スクリプト API および Agile SDK を呼び出すことができます。
- 『Agile PLM 管理者ガイド』には、イベントを理解して管理するためのバックグラウンド情報と手順が記載されています。

スクリプトの起動

1. `void invokeScript(IBaseScriptObj obj)` を使用してスクリプトを起動します。InvokeScript はスクリプトの実行開始点であり、IBaseScriptObj はすべてのイベント・スクリプト・オブジェクトの基本インタフェースです。
2. `obj` のランタイム・タイプは、このスクリプトを起動したイベントのタイプに基づいて動的に解決されます。たとえば、スクリプトが「イベントの作成」で起動された場合は、ランタイム時に `ICreateScriptObj` が `obj` のタイプとして動的に解決されます。この `obj` のインスタンスでは、`ICreateScriptObj` とそのスーパー・インタフェースに対して定義されたすべてのメソッドを起動できます。
3. `IAgileSession getAgileSDKSession()` を使用して SDK セッションにアクセスし、SDK の機能を起動します。
4. `AgileDSLException` を使用して、スクリプトからの例外情報を返します。

ユースケース

ほとんどのイベント・タイプに対するスクリプトPXハンドラの例は、http://www.oracle.com/technology/sample_code/products/agile/9.3/index.htmlにあります。これらのハンドラでは、事前/事後のトリガー、同期/非同期実行および原因が異なる複数のアクションが利用されています。