HP ALM Best Practices Series

ALM 実践者向け

ワークフローの
ベスト・プラクティス

ドキュメントリリース日：2015 年 6 月（英語版）

ご注意

保証

HP 製品，またはサービスの保証は，当該製品、およびサービスに付随する明示的な保証文によってのみ規定されるものとします。ここでの記載で追加保証を意図するものは一切ありません。ここに含まれる技術的，編集上の誤り，または欠如について，HP はいかなる責任も負いません。

ここに記載する情報は，予告なしに変更されることがあります。

権利の制限

機密性のあるコンピュータ・ソフトウェアです。これらを所有，使用，または複製するには，HP からの有効な使用許諾が必要です。商用コンピュータ・ソフトウェア，コンピュータ・ソフトウェアに関する文書類，および商用アイテムの技術データは，FAR 12.211 および 12.212 の規定に従い，ベンダーの標準商用ライセンスに基づいて米国政府に使用許諾が付与されます。

著作権について

商標について

Microsoft®，Windows® は，Microsoft Corporation の米国登録商標です。

Oracle® は，Oracle Corporation およびその関連会社の登録商標です。

ドキュメントの更新情報

このマニュアルの表紙には，以下の識別情報が記載されています。

- ソフトウェアバージョンの番号は，ソフトウェアのバージョンを示します。
  - ピリオドの前にある番号は，メジャーリリース番号です。
  - ピリオドのすぐ後にある番号は，マイナーリリース番号です。
  - ピリオドの後にある 2 番目の番号は，マイナーマイナーリリース番号です。
- ドキュメントリリース日は，ドキュメントが更新されるたびに変更されます。
- ソフトウェアリリース日は，このバージョンのソフトウェアのリリース期日を表します。

更新状況，およびご使用のドキュメントが最新版かどうかは，次のサイトで確認できます。

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

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

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

または，HP Passport のログインページの［New users - please register］リンクをクリックします。

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

サポート

HP ソフトウェア・サポート Web サイトを参照してください。

www.hp.com/go/hpsoftwaresupport

この Web サイトでは，HP ソフトウェアが提供する製品，サービス，サポートに関して，問い合わせ先などの情報をご覧いただけます。

HP ソフトウェアオンラインではセルフソルブ機能を提供しています。お客様の業務の管理に必要な対話型の技術支援ツールに素早く効果的にアクセスいただけます。HP ソフトウェアサポートの Web サイトでは，次のようなことができます。

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

一部を除き，サポートのご利用には，HP パスポート・ユーザとしてご登録の上，ログインしていただく必要があります。また，多くのサポートのご利用には，サポート契約が必要です。アクセス・レベルの詳細については，次の Web サイトをご覧ください。

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

HP Passport ID を登録するには，次の Web サイトにアクセスしてください。

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

目次

はじめに

HP ワークフロー・ベスト・プラクティス・ガイドへようこそ。

本ガイドでは, さまざまな組織でワークフローを実装する際のベストプラクティスの概念, ガイドライン，および事例を紹介しています。

ワークフローについて

新しいテクノロジー, アーキテクチャ，ビジネスのトレンド, エンド・ユーザの期待といった要因によって，アプリケーションの性質に変化が起き始めています。その結果，アプリケーション自体が変化しつつあります。新しいツールやアーキテクチャの登場により，複合型アプリケーション，リッチなインターネット・アプリケーション，インタラクティブな Web 2.0 サービスの開発と提供はますます高速かつ容易になっています。適応性のあるアプリケーションの迅速な作成を容易にする目的で，アジャイル開発などの新しいプロセスが導入され始めています。

HP Application Lifecycle Management (ALM) は，ソフトウェアがその存在の全期間にわたってたどるすべてのフェーズを対象とした単一の完備したソリューションです。ALM の手順とソリューションを利用することで，ソフトウェア・チームは，ビジネスの高い期待と要求に応えることができます。HP ALM スイートは，業界分野，企業の目標とプロセス，アプリケーションの数量とそのタイプといった要因に基づいて，さまざまな企業が固有のニーズを達成するためのお手伝いをします。同じ業界で境遇が似通っていても，企業のビジネスのやり方はそれぞれに異なります。HP ALM プロジェクトは，組織のビジネス・プロセスのニーズに応じてさまざまな手段でカスタマイズ可能なので，それぞれの実装において重要な役割を担います。

HP ALM で提供されている最も強力なツールの 1 つは，内蔵のスクリプト機能です。これは，プロジェクト内で実行されるビジネス・フローの定義，制御，管理に用いられます。ALM プロジェクト管理者は，ワークフロー・スクリプトを記述することにより，HP ALM ユーザ・インタフェースをカスタマイズし，ユーザが実行できる操作を制御できます。

本書の目的は，HP ALM ユーザが現在のカスタマイズの手順を評価し，HP ALM で提供されている高度な機能を利用した効率的なワークフロー・スクリプトを作成して維持できるようにすることです。このプロセスのすべての側面は，さまざまなソースから得られたベスト・プラクティス・データと専門知識を利用して調査されています。ソースとしては，HP のオペレーティング・システム管理者，HP のプロフェッショナル・サービス組織，技術文書，業界の専門家による書籍，多くのカスタマ・テスト組織の個人的経験などがあります。このようなガイドラインによって，最初の作成時間を短縮し，HP ALM の運用で最大限の価値を実現することができます。

対象読者

本書は次のような方を対象としています。

- プロジェクト管理者
- テンプレート管理者
- カスタマイズ専門家

前提条件

本書を利用するには，ソフトウェア開発ライフサイクル（SDLC）の主要なフェーズに関する十分な知識が必要です。また，実際の IT 組織のビジネス・プロセスについても知っている必要があります。

ここに示すベスト・プラクティスの実装には，HP ALM の運用に関する知識と管理者特権が不可欠です。

構成

本書は次のように構成されています。

- ワークフローの概要
- ワークフローのカスタマイズのガイドライン
- ワークフロー・イベント
- 結論

フィードバック

質問, コメント, あるいは共有したい貴重なベスト・プラクティス情報をお持ちの場合は, 次の電子メール・アドレスまでメッセージをお送りください。

alm_cust_feedback@hp.com

第 1 章 ワークフローの概要

ワークフローの重要性

すべての企業は独自であり，ビジネス・プロセス，業界における提携関係，開発方法，使用している新旧のテクノロジーといった要因によって，HP ALM の固有の実装を必要としています。実際の IT 組織では，1 つのやり方ですべてに対応するという考え方は通用しません。このため，HP ALM のすべてのお客様は，ワークフロー・スクリプトによって実現される柔軟性をいずれ利用することになります。

ただし，ワークフロー・スクリプトは，プロジェクトとサイト全体のパフォーマンスに大きな影響を与えます。したがって，ワークフローのコードは，論理的で整理された方法で開発することがきわめて重要です。また，ワークフロー・コードの開発と維持のための強固なプロセスを実装することも非常に重要です。

正しいワークフロー・ステップの詳細を以下に示します。

一般的なステップ

プロジェクト・ニーズの理解

ワークフロー・コードを作成または変更する前に，プロジェクトの構造，プロジェクト作業の方法，組織のプロセス，関与するさまざまな担当者について理解しておくことが重要です。

ワークフロー・コードを成功させるには，最初に各グループまたは担当者の要件を理解し，すべてのグループを考慮したワークフローを決定します。共通因子を発見して，全体のニーズを満たす複合プロセスを作成します。

ワークフロー要件ドキュメントの作成

コードの作成に進む前に，要件ドキュメントを作成します。これには，計画しているワークフローのカスタマイズを記述します。このドキュメントの目的は，初期段階でのカスタマイズを定義することです。ワークフローの運用が始まったら，このドキュメントは実装されたすべての変更を反映して定期的に更新する必要があります。

このドキュメントに必要な内容は次のとおりです。

- ワークフロー・プロセス全体
- ワークフローに必要な機能

  次にいくつかの例を示します。

  - 要件レビュー・プロセス - 組織では，各要件をテストにリンクする前にレビューして承認することが要求されている場合があります。
  - ユーザまたはグループがそれぞれのアクセス許可に応じて実行するアクション
  - 特定のフィールド変更が行われたときの電子メール送信
- レイアウトとフォーマット

  たとえば，新規不具合の作成時に使用可能なフィールド，ユーザ・グループごとの異なるフィールド・セットの定義，各タブでのフィールドの位置などを決定します。

このドキュメントはすべての関係者の承認を受ける必要があります。

ワークフロー・コードの作成

ワークフロー・カスタマイズ・ドキュメントが承認されたら，テスト環境でワークフロー・コードの作成を開始します。テスト環境の詳細については，「テスト環境」の項を参照してください。

ワークフロー・コードのテスト

エンド・ユーザをテスト環境に招待して，変更を検証してもらいます。ワークフロー実装がエンド・ユーザのニーズを満たすことを確認します。

ワークフローの運用環境への移行

ワークフローはクライアント側で実施されます。ログイン時に，カスタマイズとワークフローのファイルが，ローカル・クライアント・マシンの次のディレクトリにダウンロードされます。`%temp%\TD_80`

ワークフロー・コードが運用環境に移行されたら，カスタマイズとワークフローの最新の変更にアクセスするために，いったんログアウトしてからもう一度ログインする必要があります。

ワークフロー要求の管理

特に複数のユーザが関わっている場合，ワークフロー・コードに対する制御を維持するために，新規ワークフロー要求を管理するためのシステムを定義します。

このシステムは，要求をトレースし，要求の背後にあるビジネス・ニーズ，変更の影響，その重要性，要求の範囲（機能を必要とする人数）などを理解するために役立ちます。このようなシステムは，要求の進行状況に関する通知とステータスを送信するためにも使用できます。

1 つのソリューションとしては，新規ワークフロー要求の管理専用の ALM プロジェクトを定義することが挙げられます。

第 2 章 ワークフローのカスタマイズのガイドライン

プロジェクト・カスタマイズ・オプション

HP ALM ワークフローのスクリプト機能は，次に示すいくつかのカスタマイズ・セクションに基づいています。コードを作成する前に，プロジェクト・カスタマイズの他のすべてのカスタマイズ・ニーズを確認します。これらはワークフローの実装に用いられます。

アクセス許可グループ

プロジェクトを不正なアクセスから保護するために，ALM では，各ユーザを 1 つまたは複数のユーザ・グループに割り当てることができます。ALM には，標準設定の権限を持つグループがあらかじめ定義されています。各グループは，特定の ALM 機能に対するアクセス権を持っています。

既存のグループの権限に基づいて，新しいグループを作成できます。作成する新しいユーザ・グループと類似したアクセス権限を持つ既存のグループを選択することで，必要なカスタマイズのレベルを最小化できます。

ユーザ・グループに基づくアクセス許可の設定は，アクセス権の管理だけでなく，メール・アクションや通知などにも使用できます。

HP では，複数のユーザ・グループにユーザを割り当てることは**推奨していません**。

プロジェクト・リスト

ALM プロジェクトには，標準設定のプロジェクト・カスタマイズに使用するための定義済みのリストのセットが付属しています。たとえば，不具合ステータスや はい-いいえリストなどです。これらのリストの一部は，組織で使用する個々のプロセスをサポートするためにカスタマイズできます。一部のリストは，ALM が内部のシステム・ロジックでリストの値に依存しているため，カスタマイズできません。ユーザ定義リストを作成することもできます。ユーザ定義リストの値は，ルックアップ・リスト・フィールドに入力できます。

プロジェクトのエンティティ

エンティティは，ALM プロジェクトの構成ブロックです。エンティティには，特定のアプリケーション管理プロセスについてユーザが入力したデータが表形式で格納されます。エンティティは，要件，テスト，デザイン・ステップ，添付，不具合など，任意の作業オブジェクトです。

プロジェクト・カスタマイズを使用すると，ALM エンティティの属性やプロパティを設定できます。たとえば，必須フィールド，読み取り専用，値の検証などです。各エンティティには，システム・フィールドと呼ばれる ALM の標準設定フィールドが含まれます。エンティティには，ユーザが作成できるユーザ・フィールドが含まれる場合もあります。ユーザ・フィールドのタイプは次のいずれかです。

- ユーザ・リスト（プロジェクト内の全ユーザのリスト）
- リスト
- 数値
- 日付
- 文字列

プロジェクト・カスタマイズでは，各プロジェクト・エンティティに対してプロパティを定義できます。たとえば，ユーザが入力する必要があるフィールドはどれか，データ履歴が記録されるフィールドはどれかといったことです。これらのプロパティの一部は，ワークフローを使用して設定することもできます。推奨される方法は，標準設定の動作をプロジェクト・カスタマイズで設定し，特別な場合のみワークフロー・スクリプトで変更することです。

1 つのエンティティのユーザ定義フィールドの数は 99 個までに制限されています。このため，HP では，関係者全員と協力して，大半の関係者のニーズに合致し，短期間で冗長になることがないフィールドだけを含めることを**推奨**します。

要件タイプ

プロジェクトに対して要件タイプを定義することができます。要件タイプは，オプションのフィールドと，利用可能なユーザ定義フィールドを定義します。これにより，特定のタイプの要件でのみ利用可能なユーザ定義フィールドを作成できます。

ワークフローの一般的ルール

ワークフロー・コードを使用することで，プロジェクトをさらにカスタマイズできます。次のような設定を定義できます。

- 表示されるフィールドと必須フィールド
- ダイアログ・ボックスでフィールドが表示される順序
- ダイアログ・ボックスの各タブに表示されるフィールド
- 特定のフィールドに割り当てるリスト
- 特定のフィールドの標準設定値
- フィールド値の間の依存関係

これらの設定はユーザ・グループに基づいて定義できます。

重要な注：

- ワークフロー・コードは，プロジェクト・カスタマイズの特定のカスタマイズ・カテゴリに定義された設定をオーバーライドします。
- 一部のカスタマイズ，たとえばユーザ・グループの遷移ルールの定義や要件タイプのフィールド・プロパティの設定は，特定のプロジェクト・カスタマイズ・ページとワークフロー・コードのどちらでも実行できます。HP では，特定のカスタマイズに関しては 1 つの方法を使用することに決め，2 つの方法を混在させないことを**推奨**します。
- ALM の自動メールを使用すると，指定された不具合フィールドに変更が生じるたびに，電子メールを通じてユーザに自動的に通知できます。ワークフローで `SendMail_AfterPost` 関数を使用することで，すべてのプロジェクト・エンティティに対する自動通知を定義したり，複雑な条件を追加したり，特定のユーザまたはユーザ・グループに対して使用したりすることができます。自動メールとワークフロー関数の間に重なりが生じないようにすることを**推奨**します。
- ワークフロー・スクリプトを使用すると，モジュールの開始と終了のアクションを制御し，モジュールへのアクセスを制限することができます。HP では，ユーザ・グループによる特定のモジュールへのアクセスを禁止するには，プロジェクト・カスタマイズの［モジュール アクセス］ページを使用することを**推奨**します。ワークフロー・スクリプトを使用してモジュールへのアクセスをブロックすることは，モジュール・アクセス機能と競合するため避けてください。

テスト環境

HP では，運用環境でワークフローのカスタマイズを実装する前に，固有の構成を反映したテスト環境でカスタム機能を検証することを**推奨**します。

テスト環境とは

テスト環境は，運用環境と別に存在し，運用環境を正確に反映した環境です。この環境は，運用システムにインストールされている，データベース・サーバ，ソフトウェア，運用プロジェクトなどの構成とアプリケーションをシミュレートします。テスト環境でワークフローをテストすることで，達成できる結果をより正確に予測し，運用環境への悪影響を発見して予防することができます。

テスト環境を設定する理由

ワークフローは，プロジェクトの機能に重要な影響を与えます。HP では，次の理由でテスト環境をセットアップすることを**推奨**しています。

- 実稼働前にワークフローをテストすることが望まれます。
- テスト環境は運用環境から独立しているので，ワークフローが失敗しても実際の損害はありません。運用環境への損害としては，データの損失，ワークフローのエラーによる機能の停止などが考えられます。
- 問題を早期に発見して検出できます。
- 計画中のプロセスを関係者が検証できます。

ワークフロー・コードのデバッグ

ワークフロー・コードのデバッグには，いくつかの方法があります。

オプション	ツールの機能	用途
ALM ワークフローを通じた Msgbox の追加	これは ALM ワークフロー・コードの内蔵機能です。 ワークフローの任意の場所に `Msgbox`（メッセージ・ボックス）を追加して，フィールド値，アクション名，コード内の場所などを表示できます。 この方法を使用する場合，コードのエラー処理行 `On Error Resume Next` をコメント化することを推奨します。	コード内の特定の場所のデバッグ
Dbgview	このフリーウェアは，ローカル・システムのデバッグ出力をモニタして，すべてのアプリケーション・イベントを出力します。 ワークフロー・イベントだけを表示するには，‘workflow’ という単語でフィルタします。 注：このオプションで観察できるのは組み込みイベントだけで，ユーザが追加した機能は観察できません。 ベンダー：Microsoft（SysInternals） URL： http://technet.microsoft.com/en-us/sysinternals/bb896647	ALM から呼び出されたワークフロー・プロシージャの表示
Microsoft Visual Studio	これは市販製品であり，ブレークポイント，変数のインスペクトなどを使用してコードを検証できます。 Visual Studio にワークフロー・コードをアタッチするには，Visual Studio をアタッチしたい点でワークフロー・コードからエラーを起こします。そのための方法としては，存在しないサブルーチンの呼び出しが挙げられます。Visual Studio がインストールされていると，スクリプトでそのようなエラーが発生した場合，ダイアログがポップアップ表示され，スクリプトにアタッチすることができ	• コード内の特定の場所のデバッグ • 作成したコードのフローの表示 • ランタイムでのコードの観察

	ます。 このオプションを使用するには，ワークフロー・コード内の `On Error Resume Next` をコメント化する必要があります。	

上記のツールに加えて，独自のロガーを実装することもできます。HP では，独自のロガーを使用する場合，パフォーマンスへの悪影響を防ぐため，必要に応じてロガーを有効または無効にするオプションを実装することを**推奨**します。

ワークフローに関する推奨手順と非推奨手順

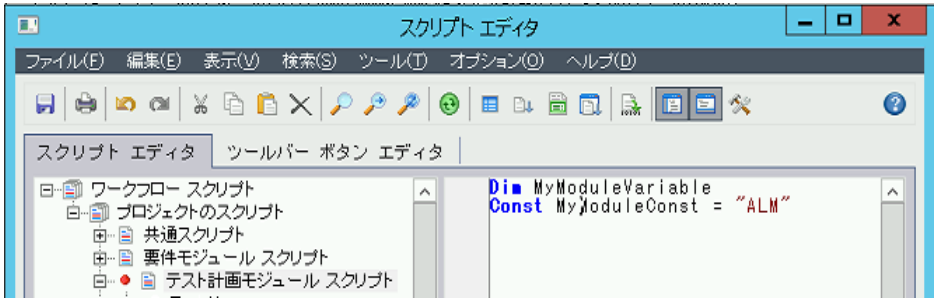
ワークフローに熟練するために役立つ**推奨**手順をいくつか次に示します。

推奨手順

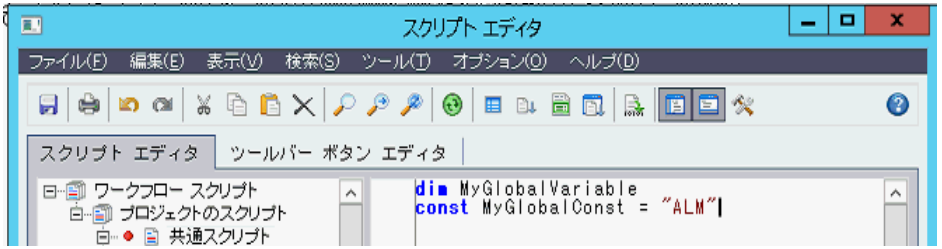
グローバルを使用する

異なるイベントの間で値を受け渡すには，グローバル変数またはグローバル定数を使用する必要があります。

変数はモジュール・レベルに存在することができます。変数はモジュール・イベントで使用できます。



異なるモジュール・イベントの間で値を受け渡すには，グローバル変数を共通モジュールで定義する必要があります。



コードを最適化する

VBScript でプログラミングする場合，同じコードを何度もプログラムすることが必要な場合があります。通常，このような場合は，関数を使用してコードの繰り返しを避けるべきです。

- 冗長なコードの代わりに，プロシージャや関数を使用します。

  次の一般的なタスクは，別に関数またはプロシージャを定義した方がよい場合の例です。

  - フィールドのプロパティの設定
  - フォームのフィールドのセットアップ
  - リストの依存関係の設定
  - OTA API を操作する手順

- ElseIf ステートメントの繰り返しの代わりに，Switch ステートメントを使用します。

  経験則としては，ElseIf 条件が 2 個以上ある場合，Switch ステートメントを使用します。

コードの可読性を改善する

関数には，繰り返しコードの再利用以外に，コードの可読性を高める効果もあります。コードの可読性を改善するために役立つ**推奨**手順を次に示します。

- 関連するコードのブロックを空行で論理的に区切ります。
- 導入（ヘッダ）コメントを使用して，最初の変数宣言から最後の変数宣言までの部分とコード自体を区別します。
- すべてのコメントの前に空行を置きます。

- プロシージャ内のコードとコメントは，スペース 2 個または 4 個のタブ・ストップを使用してインデントします（Visual Basic Editor では標準設定でスペース 4 個のタブ・ストップが使用されます）。スペースと同様，インデントはコードを論理的に整理し，見やすくする役割を果たします。次に示すのは，コードの可読性とメンテナンス性を高めるための正しいインデントの使い方に関する一般的なガイドラインです。
  - プロシージャ内のすべてのコードとコメントは，少なくとも 1 個のタブ・ストップでインデントします。インデントされないコード行は，プロシージャの最初と最後，およびエラー・ハンドラとの関連で使用する行ラベルだけです。
  - 改行を使用してプロシージャの引数リストをフォーマットする場合，タブを使用して引数とそのデータ・タイプ宣言をインデントし，リスト内の最初の引数と揃えるようにします。
  - 変数宣言はタブ・ストップ 1 個分インデントします。1 行に 1 つの変数だけを宣言します。
  - 制御構造は少なくとも 1 個のタブ・ストップでインデントします。1 つの制御構造が別の制御構造の中に埋め込まれている場合，埋め込まれた構造をタブ・ストップ 1 個分インデントします。制御構造内部のコードはさらにタブ・ストップ 1 個分インデントします。
  - 行継続文字を使用してコード行を分割する場合，新しい行はタブ・ストップ 1 個分追加でインデントします。これは，複数行が 1 つにまとめられることを視覚的に明確にするためです。継続行の後の行のインデントが継続行と一致する場合は，次の行と区別するために継続行をさらにタブ・ステップ 1 個分追加でインデントします。
  - コメントは，参照先のコードと同じレベルにインデントします。

コードにコメントを付加する

次のようなコメント・テンプレートを使用して，コードをドキュメント化します。

```
'###############################################################
'#日付：
'#設計者：
'#目的：
'###############################################################
```

名前でアクセスする

特定のエンティティのフィールドにアクセスすることは，一般的な作業の 1 つです。

- フィールドに名前でアクセスするには，**＜エンティティ＞**_Fields.Field(”**＜フィールド名＞**”）というステートメントを使用します。
- コレクションのすべてのフィールドにアクセスするには，**＜エンティティ＞**_Fields.FieldById(i) をループで使用します。これはたとえば，フィールドの順序をリセットするために使用できます。

どちらの方法も，現在のエンティティのフィールドを対象とします。

現在のエンティティは，次の方法で定義できます。

- 現在のエンティティ

  現在のイベントがトリガされるエンティティ。ほぼすべてのイベントは，フィールドが取得できるエンティティ・タイプを指しています。たとえば，Defects_Bug_ イベントでは，Bug_Fields だけが操作できます。TestPlan_DesignStep_ イベントでは，DesignStep_Fields だけにアクセスできます。

- フォーカスのある項目

  イベントで定義されるコレクションのすべてのエンティティの中で，現在フォーカスのあるエンティティのフィールドだけが取得できます。たとえば，カーソルが置かれているテストや，マニュアル・ランナーの現在の実行です。

同一または他のオブジェクト・タイプの別のオブジェクトのフィールドを取得するには，OTA API を使用します。

新規レイアウトの設定前にリセットを行う

PageNo や ViewOrder といった個別フィールドのレイアウトを設定する前に，すべてのフィールドのレイアウトを必ずリセットしてください。

フィールドには標準設定の定義済みの順序があるため，新しいカスタム順序を定義する前に，この順序をリセットすることが重要です。そうしないと，目的のフィールド以外のフィールドに同じ順序が存在し，残りのすべてのフィールドの順序が不明になる可能性があります。

この後の「ワークフローの例」を参照してください。

コードをバックアップする

HP では，ワークフロー・コードを定期的にバックアップすることを**推奨**します。ワークフロー・スクリプトの全部または一部をコピーし，外部のテキスト・ファイルに貼り付けてファイル・システムに保存することができます。

フィールド名でなくグローバル定数を使用する

コードの可読性を高めるため，フィールド名の代わりにグローバル定数を使用します。

各フィールド名に対してグローバル定数を宣言する必要があります。上記の「グローバルを使用する」の項を参照してください。

例：

```
If Bug_Fields.Field(Bug_Status).Value="Closed" then
      Bug_Fields.Field(Bug_Closed_In_Version).isRequired = true
End if
```

オブジェクトをクリーンアップする

オブジェクトはスコープの最後にクリーンアップするようにします。各オブジェクト・インスタンスに対して，未使用のオブジェクトをクリーンアップすることが重要です。これにより，ALM のパフォーマンスが向上し，エラーを防ぐことができます。

```
set myTDConnection = TDConnection
set myTDConnection = nothing
```

標準化する

HP では，すべてのプロジェクトに対して標準化を適用することを**推奨**します。複数のプロジェクトに対して責任を負う ALM プロジェクト管理者は，すべてのプロジェクトで共通の規則を使用するようにします。これにより，コードの可読性とメンテナンス性が高まり，プロジェクトにまたがる機能が有効になります。

エラー処理を行う

ワークフロー・スクリプトの品質に影響する最も重要な要因の 1 つは，エラー処理の正しい実装です。

アプリケーションの動作の制御を改善するためのエラー処理の簡単な**推奨**手順の一般的な例をいくつか以下に示します。

- 各プロシージャおよび関数の先頭で On Error Resume Next ステートメントを使用する
- 各プロシージャおよび関数の末尾で On Error GoTo 0 を使用する
- 何らかの標準化したメッセージ・ボックスでユーザにエラーを通知する

コードは，各ワークフロー・スクリプト（不具合，テスト計画など）に 1 回追加します。次に示す PrintError 関数の例を参照してください。

そのためには，実行時エラーに関する情報を記録した Visual Basic の Err オブジェクトを使用します。

```
Sub GetBug1
  On Error Resume Next
  Set Bug1 = TDConnection.BugFactory.item(1)
  PrintError("GetBug1")
End Sub
Sub PrintError(strFunctionName)
  If Err.Number <> 0 Then
    MsgBox "Error #" & Err.Number & ":" & Err.Description, _
    vbOKOnly, "Workflow Error in Function " & strFunctionName
  End If
End Sub
```

非推奨手順

重複させない

Entity_CanChange イベントと EntityChange イベントの間でコードが重複しないようにします。

フィールドの変更に基づくコードは，Entity_CanChange イベントまたは Entity_Change イベントに作成することができます。これらのイベントの間の違いを理解して，次に示す単純なルールに基づいてワークフロー・コードを作成してください。

- アクセス許可（ステータス変更の許可）を処理するコードは Entity_CanChange イベントに作成
- 依存関係の値または依存関係リストを処理するコードは EntityChange イベントに作成

コードを詰め込みすぎない

CanLogin イベントや EnterModule イベントのコードが多すぎると，パフォーマンスが低下します。よくあるエラーとして，CanLogin イベントの間にエンティティを更新することが挙げられます。

表示設定の前に他のプロパティを設定しない

フィールドの IsVisible プロパティの設定は，IsRequired プロパティや IsReadOnly プロパティの設定よりも前に行ってください。

画面に表示されていないフィールドを必須あるいは読み取り専用にしても無意味なので，ALM はこれらの設定を無視します。したがって，これらのプロパティを設定する前にフィールドを表示しておくことが重要です。

この後の「ワークフローの例」を参照してください。

乱雑なワークフロー・コードを書かない

ALM プロジェクトが使用される期間は数年程度ですが，それでもワークフロー・コードは明快で拡張可能である必要があります。

- If よりも Select を使用する
- 関数を使用する

パラメータを更新しない

ワークフロー関数でパラメータ値を更新しないでください。特に，Entity_CanChange イベントで NewValue パラメータを更新しないでください。

新規作成中に変更しない

新規エンティティ・イベントでアクションを変更することは**推奨されません**。新規エンティティ・イベントが呼び出されるのはエンティティが作成される時点であり，新規エンティティ・ダイアログ・ボックスが開かれる時点ではないからです。

一般的な使用例としては，ユーザが 2 回目に新規エンティティ・ダイアログ・ボックスを開く場合が挙げられます。ダイアログ・ボックスが 1 回目に開かれたときにエンティティはすでに作成されているので，新規エンティティ・イベントは呼び出されません。

MoveTo で代入しない

MoveTo イベント内でフィールドに値を代入しないでください。これがよくないのは，MoveTo イベントによってエンティティがロックされてしまうからです。

After_Post で変更しない

After_Post イベントではオブジェクトの変更を行わないでください。

多数の API 呼び出しを使用しない

多数の API 呼び出しは使用しないでください。API を呼び出すと，サーバとデータベースの間の通信のレベルが上がるからです。API を呼び出すたびにネットワーク通信が発生し，スクリプトの実行時間が長くなります。

たとえば，100 個のエンティティを処理する場合，それぞれを別々に取得するのではなく，1 個のフィルタで全部を取得するようにしてください。

ワークフローでの API の使用

変更のための ALM API の使用

ALM API には，ユーザ・インタフェース（またはそれを使用するアプリケーション）とサーバ・ロジックの間の分離層が用意されています。HP では，API 呼び出しを使用する際に次のルールに従うことを**推奨**します。

- 現在のセッションを取得するには，定義済みの `TDConnection` オブジェクトを使用する

  Visual Basic や Excel などの外部アプリケーションから OTA API を使用する場合，OTA を使用するすべてのアプリケーションにとっての最初のステップは，`TDConnection` オブジェクトのインスタンスを作成し，サーバへの接続を初期化し，データベースに接続することです。ただし，ワークフローには定義済みの `TDConnection` オブジェクト（この場合，`TDConnection` は単にクラス名であるだけでなく，`TDConnection` のインスタンスを含むグローバル変数の名前でもあります）があります。これは現在のユーザが作業しているのと同じセッションを指しています。このため，すべての ALM コレクションおよびオブジェクトへのアクセスが，ワークフローの任意の場所からいつでも可能です。

- `Command` オブジェクトを使用したデータベースの直接更新は，次のような問題の可能性があるため避ける
  - サーバ・メカニズムをバイパスすることによる次の問題の発生
    - エンティティのロック
    - 履歴の消失
    - その他の不要な機能（メールのセットアップ）
  - クエリのメンテナンスの増加
  - データの破損や不整合の発生

- カスタム・メールをユーザに送信するには，OTA のメール・メソッドを使用する

  OTA では，ALM のメール機能にアクセスして，次のことを実行できます。

  — ALM の自動通知システムでは実装できないカスタム条件の作成

  — 電子メールの件名またはテキストの変更

  — 特定の ALM グループまたはユーザへの電子メールの送信

  — 自動メール通知で使用される “admin” 以外の特定ユーザからの電子メールの送信

  メール・メソッドは，不具合，テストなど任意の ALM オブジェクトから，または TDConnection オブジェクトから直接使用できます。TestDirector オブジェクトから Mail メソッドを使用すると，そのオブジェクトとカスタム件名およびテキストを含む電子メールを送信できます。

例 – SQL ステートメントの代わりにワークフロー・オブジェクトを使用

次のコマンドは使用しないでください。

```
Com.CommandText = "UPDATE TESTCYCL SET TC_TESTER_NAME =
'" & Cstr(ASSIGNED_TESTER) & "' " &_
"Where TC_CYCLE_ID = " & iTestSetId & " and TC_TESTER_NAME is NULL"
Set UpdateRecSet = Com.Execute
```

代わりに次のコード・スニペットを使用してください。

```
Set tstestF = currentTestSet.tstestFactory
Set tsFilter = tstestF.Filter
tsFilter("TC_TESTER_NAME")= ""
Set tsTestList = filter.newList
For each tsTest in tsTestList
    tsTest.Field("TC_TESTER_NAME") = "admin"
Next
```

クライアントのアクティビティの最小化

サーバからデータを取得する際には，クライアント側でなくサーバ側で情報をフィルタすることを**推奨**します。クライアント側でのフィルタ処理は，パフォーマンスに与える影響が非常に大きいからです。ロードするレコードの数が多すぎる場合も，サーバのパフォーマンスに影響があります。

例 – 履歴処理でのフィルタの使用

Command オブジェクトを使用して HISTORY テーブルを処理する場合，SQL で WHERE 条件を実装してフィルタを作成し，クライアントにすべてのレコードセットが取得されないようにします。

例 – デザイン・ステップの計算

デザイン・ステップに，ステップの継続時間を保持するユーザ定義のフィールドがあります。ここでの目的は, 継続時間が 30 分より長いデザイン・ステップの数を求めることです。

次のコードはよくない例です。

```
For Each Test In TestLists
    Set DesStepF = Test.DesignStepFactory
    Set DSList = DesStepF.NewList("")
    For Each DStep In DSList
        If DStep.Field(“DS_USER_01”)>30 Then
          HowManyFound = HowManyFound + 1
        End If
    Next
Next
```

代わりに次のコード・ブロックを使用してください。

```
Set TestF = TDConnection.TestFactory
Set TestList = TestF.NewList("")
For Each Test In TestList
  Set DesStepF = Test.DesignStepFactory
  Set DSList = DesStepF.NewList("select * from DESSTEPS WHERE
DS_USER_01>30 ")
  HowManyFound = HowManyFound + DSList.count
Next
```

第 3 章 ワークフロー・イベント

ALM ユーザ・セッション中に，ユーザがさまざまな操作を行うと，ALM によってイベント・プロシージャがトリガされます。これらのプロシージャにコードを記述することで，関連するユーザ操作の実行をカスタマイズできます。イベント・プロシージャは，関数，サブルーチンのいずれかにすることができます。

一般的な内容

次に示すのは，イベント関数およびサブルーチンに関する一般的な情報と，HP ALM で使用される命名規則です。

イベント関数

これらのプロシージャは ALM によってトリガされ，ユーザ操作を実行すべきかどうかを確認する役割を果たします。これらの関数にコードを記述することで，ALM がユーザの要求を実行するかどうかを決定できます。コードが false の値を返した場合，ALM は操作を実行しません。

たとえば，ユーザが［不具合の追加］ダイアログ・ボックスで［送信］ボタンを押すと，ALM はサーバ上のデータベースに不具合を送信する前に，関数 `Bug_CanPost` を呼び出します。`Bug_CanPost` 関数にコードを追加することで，ALM が不具合を送信するかどうかを制御できます。たとえば，ユーザがコメントを追加しないと不具合を却下できないようにすることができます。

イベント・サブルーチン

これらのプロシージャは，イベントが発生したときに，何らかの操作を実行するためにトリガされます。

たとえば，ユーザが［不具合の追加］ダイアログ・ボックスを開くと，ALM はサブルーチン `Bug_New` を呼び出します。`Bug_New` サブルーチンにコードを追加することで，ユーザがダイアログ・ボックスを開いたときに実行する必要がある操作を実行できます。たとえば，ユーザが `QA` **テスト担当者**ユーザ・グループに属していない場合，［変更検出モード］フィールドの値を `BTW` に変更できます。

命名規則

HP ALM でのイベント・プロシージャの命名規則は次のとおりです。

<エンティティ>_<イベント>

注：一部のイベント・プロシージャ名（例：`GetDetailsPageName`）には，エンティティ名が含まれません。

エンティティ

エンティティ	説明
AnalysisItem	レポートとグラフ・データ
AnalysisItemFolder	レポートとグラフ・フォルダ・データ
Bug	不具合データ
BusinessModel	ビジネス・モデル・データ
BusinessModelActivity	ビジネス・モデル・アクティビティ・データ
BusinessModelFolder	ビジネス・モデル・フォルダ・データ
BusinessModelPath	ビジネス・モデル・パス・データ
Component	ビジネス・コンポーネント・データ
ComponentFolder	ビジネス・コンポーネント・フォルダ・データ
ComponentStep	ビジネス・コンポーネント・ステップ・データ
DashboardFolder	ダッシュボード・フォルダ・データ
DashboardPage	ダッシュボード・ページ・データ
DesignStep	設計ステップ・データ
Resource	テスト・リソース・データ
Resource Folder	テスト・リソース・フォルダ・データ
Run	テスト実行データ
Step	テスト実行ステップ・データ
TestSet	テスト・セット・データ
TestSetTests	テスト・インスタンス・データ

ワークフローではいくつかの拡張機能がサポートされる場合もあります。

共通モジュール

CanLogin

このイベントは，指定されたユーザが指定されたプロジェクトにログインできるかどうかを確認するためにトリガされます。これは，プロジェクトのログインを許可または禁止するために使用します。このイベントは，ユーザを更新するために使用できます。

トピック	説明
構文	CanLogin(DomainName, ProjectName, UserName) DomainName はドメイン名，ProjectName はプロジェクト名，UserName はユーザ名です。
タイプ	関数
戻り値	True または False
利用方法	CanLogin（すべてのモジュール）

例 - ログイン時にユーザに通知

```
Function CanLogin(DomainName, ProjectName, UserName)
CanLogin = false
Call MsgBox("Hi " & User.UserName & ",” _
            & vbCrLf & "" _
            & vbCrLf & "Your project " & TDConnection.ProjectName
& " was upgraded to ALM 11.0" _
            & vbCrLf & "" _
            & vbCrLf & "The Project was moved to the server :
http://ALM:port/qcbin" _
            & vbCrLf & "" _
            & vbCrLf & "QC Admin" _
            , vbExclamation, "Important Message")
Exit function
End function
```

CanLogout

このイベントは，現在のユーザが現在のプロジェクトからログアウトできるかどうかを確認するためにトリガされます。

トピック	説明
構文	CanLogout
タイプ	関数
戻り値	True または False
利用方法	CanLogout（すべてのモジュール）

例 - ログアウト前にユーザに通知

```
Function CanLogout

Call MsgBox("Hi " & User.UserName & ",” _

            & vbCrLf & "" _

            & vbCrLf & "Your project " & TDConnection.ProjectName
& " will be upgraded to ALM 11.0  on 01/01" _

            & vbCrLf & "" _

            & vbCrLf & "The Project will be moved to the server:
http://ALM:port/qcbin" _

            & vbCrLf & "" _

            & vbCrLf & "QC Admin" _

            , vbExclamation, "Important Message")

End Function
```

ActionCanExecute

このイベントは，ユーザが開始した操作を ALM が実行する前に，操作が実行可能かどうかを確認するためにトリガされます。このイベント・プロシージャにコードを追加することで，ユーザが特定の操作を開始したときに操作を実行したり，または特定の場合で操作の実行を防ぐことができます。

トピック	説明
構文	ActionCanExecute(ActionName) ActionName は，ユーザが開始した操作です。 操作は context.action の形式で表されます。 ユーザ定義操作の名前には，先頭にプレフィックス「UserDefinedActions」が付記されます。
タイプ	関数
戻り値	True または False
利用方法	ActionCanExecute（すべてのモジュール）

例 - 不具合削除の防止

```
Function ActionCanExecute(ActionName)
On Error Resume Next
if ActionName= "Defects.DeleteDefect" then
       if Bug_Fields.Field("BG_STATUS").value  =”Closed” then
              ActionCanExecute = true
       Else
Msgbox “You don’t have enough credentials to perform Delete”
              ActionCanExecute = false
              Exit function
       End if
End if
`……
End function
```

例 - アクション名の検索

```
Function ActionCanExecute(ActionName)
On Error Resume Next
if user.Username=”Project_admin” then
       MsgBox actionname
```

```
End if
End function
```

EnterModule

このイベントは，ユーザが ALM モジュールに切り替えたときにトリガされます。

このイベント・プロシージャにコードを追加することで，ユーザが指定されたモジュールに切り替えるときに常に操作を実行することができます。

トピック	説明
構文	EnterModule
タイプ	サブルーチン
戻り値	
利用方法	EnterModule（すべてのモジュール）

例 - ボタンを非表示

```
Sub EnterModule
'不具合グリッドの［メールの送信］ボタンを非表示
  On Error Resume Next
    Actions.action("Defects.SendByEmail").Visible= false
  On Error GoTo 0
End Sub

Sub DialogBox(DialogBoxName, IsOpen)
'不具合詳細ダイアログの［メールの送信］ボタンを非表示
'ActiveModule と ActiveDialogName を使用して現在のコンテキストを取得
On Error Resume Next
if (DialogBoxName="actBugDetails" or DialogBoxName="Details" or
DialogBoxName="Bug Details") and IsOpen=true  then
       Actions.action("BugDetails.SendByEmail").Visible= false
End if
  On Error GoTo 0
End Sub
```

ExitModule

このイベントは，ユーザが指定されたモジュールを終了するときにトリガされます。

トピック	説明
構文	ExitModule
タイプ	サブルーチン
戻り値	
利用方法	ExitModule（すべてのモジュール）

DialogBox

このイベントは，ダイアログ・ボックスが開くとき，または閉じるときにトリガされます。

トピック	説明
構文	DialogBox(DialogBoxName, IsOpen) DialogBoxName はダイアログ・ボックスの名前，IsOpen はダイアログ・ボックスが開いているかどうかを示します。
タイプ	サブルーチン
戻り値	
利用方法	DialogBox（すべてのモジュール）

例 - ビュー・タイプの識別

この例は，現在のビュー・タイプがグリッド，詳細，新規エンティティのどれであるかを識別します。タイプは共通モジュールのグローバル変数 DialogIsOpen に保持されます。

```
Sub DialogBox(DialogBoxName, IsOpen)
On error resume next
If DialogBoxName="New Bug" and IsOpen=true then
          DialogIsOpen = "NEW"
              Else
```

```
                    DialogIsOpen =”OTHER” ‘Details Or Grid
End if
 On Error GoTo 0
End sub
```

CanCustomize

このイベントは，ユーザがカスタマイズ・ウィンドウを開こうとしたときに，ユーザが指定されたプロジェクトをカスタマイズ可能であるかどうかを確認するためにトリガされます。

トピック	説明
構文	CanCustomize(DomainName, ProjectName, UserName) DomainName はドメイン名，ProjectName はプロジェクト名，UserName はユーザ名です。
タイプ	関数
戻り値	True または False
利用方法	CanCustomize（すべてのモジュール）

例 - カスタマイズへの進入禁止

この例は，許可されていないユーザがカスタマイズに入ることを禁止します。

```
Function CanCustomize(DomainName, ProjectName, UserName)
on error resume next
if  User.IsInGroup("TDAdmin”) then
  CanCustomize = true
else
    MsgBox User.FullName & vbcrlf & vbcrlf & "You don't have
enough privileges" &vbcrlf & vbcrlf &"Please Open a SR in Project
Center Admin", vbExclamation, "Not Allowed"
   CanCustomize = false
end if
On Error GoTo 0
End Function
```

Attachment_New

このイベントは，ALM に添付ファイルが追加されたときにトリガされます。

トピック	説明
構文	`Attachment_New(Attachment)` `Attachment` は `lAttachment` インタフェースです。
タイプ	サブルーチン
戻り値	
利用方法	Attachment_New（すべてのモジュール）

Attachment_CanOpen

このイベントは，ALM がサーバからの添付ファイルを開く前に，添付ファイルが開けるかどうかを確認するためにトリガされます。

トピック	説明
構文	`Attachment_CanOpen(Attachment)` `Attachment` は `IAttachment` インタフェースです。
タイプ	関数
戻り値	True または False
利用方法	`Attachment_CanOpen`（すべてのモジュール）

Attachment_CanPost

このイベントは，ALM がサーバ上の既存の添付ファイルを更新する前に，添付ファイルが更新可能かどうかを確認するためにトリガされます。

トピック	説明
構文	`Attachment_CanPost(Attachment)` `Attachment` は `IAttachment` インタフェースです。

タイプ	関数
戻り値	True または False
利用方法	`Attachment_CanPost`（すべてのモジュール）

Attachment_CanDelete

このイベントは，ALM がサーバから添付ファイルを削除する前に，添付ファイルが削除可能かどうかを確認するためにトリガされます。

トピック	説明
構文	`Attachment_CanDelete(Attachment)` `Attachment` は `IAttachment` インタフェースです。
タイプ	関数
戻り値	True または False
利用方法	`Attachment_CanDelete`（すべてのモジュール）

GetDetailsPageName

このイベントは ALM によってトリガされ，次のダイアログ・ボックスの `PageNum` で指定されているインデックス番号のページ（タブ）の名前を取得します。

- エンティティの［詳細］ダイアログ・ボックス
- エンティティの［新規＜エンティティ＞］ダイアログ・ボックス

トピック	説明
構文	`GetDetailsPageName(PageName, PageNum)` `PageName` は標準設定のページ名（「ページ 1」など）で，`PageNum` はページ番号です。 注：ダイアログ・ボックスに表示されているその他のページからのページの相対的な位置にかかわらず，ページ番号は絶対ページ番号のことを指します。

タイプ	関数
戻り値	ページ名を含む文字列
利用方法	GetDetailsPageName（すべてのモジュール）

例 - タブの更新

```
Function GetDetailsPageName(PageName,PageNum)
  On Error Resume Next
     Select Case activemodule
        Case "Requirements"
           Select Case PageNum
                  Case 1
                    GetDetailsPageName="Req_Details-First Tab"
                  Case 2
                    GetDetailsPageName="Req_Details-Second Tab"
                  Case 3
                    GetDetailsPageName="Req_Details-Third Tab"
           End select
        Case "Defects"
           Select Case PageNum
                  Case 1
                    GetDetailsPageName="Def_Details-First Tab"
                  Case 2
                    GetDetailsPageName="Def_Details-Second Tab"
                  Case 3
                    GetDetailsPageName="Def_Details-Third Tab"
           End select
     End select
  On Error GoTo 0
End Function
```

```
Function GetNewBugPageName(PageName,PageNum)
On Error Resume Next
  Select Case PageNum
         Case 1
           GetNewBugPageName="Def_Details-First Tab"
         Case 2
           GetNewBugPageName="Def_Details-Second Tab"
         Case 3
           GetNewBugPageName="Def_Details-Third Tab"
  End select
  On Error GoTo 0
End Function
```

エンティティ・モジュール

Entity_New

このイベントは，ALM にオブジェクトが追加されたときにトリガされます。このイベント・プロシージャにコードを追加して，新規オブジェクトが追加されたときに操作を実行できます。

トピック	説明
構文	＜エンティティ＞_New
タイプ	サブルーチン
戻り値	
利用方法	AnalysisItem_New AnalysisItemFolder_New Baseline_New Bug_New BusinessModelFolder_New BusinessModelPath_New Component_New ComponentFolder_New ComponentStep_New

	Cycle_New DashboardFolder_New DashboardPage_New DesignStep_New Library_New LibraryFolder_New Release_New ReleaseFolder_New Req_New Resource_New ResourceFolder_New Step_New Test_New TestConfiguration_New TestFolder_New TestSet_New TestSetFolder_New

Entity_MoveTo

このイベントは，ユーザがあるオブジェクトから別のオブジェクトにフォーカスを移動するときにトリガされます。

このイベント・プロシージャにコードを追加して，ユーザがフォーカスを移動するときに操作を実行できます。

トピック	説明
構文	＜エンティティ＞_MoveTo
タイプ	サブルーチン
戻り値	
利用方法	AnalysisItem_MoveTo AnalysisItemFolder_MoveTo Baseline_MoveTo Bug_MoveTo BusinessModel_MoveTo BusinessModelActivity_MoveTo BusinessModelFolder_MoveTo BusinessModelPath_MoveTo

	Component_MoveTo ComponentFolder_MoveTo（旧 MoveToComponentFolder） ComponentStep_MoveTo Cycle_MoveTo DashboardFolder_MoveTo DashboardPage_MoveTo DesignStep_MoveTo Library_MoveTo LibraryFolder_MoveTo Release_MoveTo ReleaseFolder_MoveTo Req_MoveTo Resource_MoveTo ResourceFolder_MoveTo Run_MoveTo Step_MoveTo Test_MoveTo TestConfiguration_MoveTo TestFolder_MoveTo TestSet_MoveTo TestSetFolder_MoveTo TestSetTests_MoveTo

例 - 移動時にセットアップを更新

別のエンティティに移動する際にセットアップを更新します。

```
Sub Bug_MoveTo
        Select Case Bug_Fields.Field("BG_STATUS").value
           Case "New"
               Setup_Status_New
           Case "Open"
               Setup_Status_Open
           Case "Fixed"
               Setup_Status_Fixed
           Case "Closed"
```

```
                    Setup_Status_Closed
            End select
End sub
```

例 - 依存関係リスト

次のコードは，フィールドに関連付けられたリストを，別のフィールドの値に基づいて変更する方法を示します。

SUB_AREA（RQ_USER_01）および TESTING_AREA（RQ_USER_02）という名前のユーザ定義フィールドが要件エンティティに追加されており，各テスト領域に SUB_LIST_＜テスト領域＞ という名前のユーザ定義リストが追加されているとします。

このコードは，**＜エンティティ＞**_MoveTo および **＜エンティティ＞**_FieldChange イベントで呼び出す必要があります。

```
Req_Fields.field("RQ_USER_02").List = Lists("SUB_LIST_" &
Req_Fields.field("RQ_USER_01").value)
```

Entity_ FieldCanChange

このイベントは，ALM がフィールド値を変更する前に，フィールドが変更可能かどうかを判断するためにトリガされます。

このイベント・プロシージャにコードを追加して，特定の場合にフィールドが変更されるのを防ぐことができます。

トピック	説明
構文	**＜エンティティ＞**_FieldCanChange (FieldName, NewValue) FieldName はフィールドの名前，NewValue はフィールド値です。
タイプ	関数
戻り値	True または False
利用方法	AnalysisItem_FieldCanChange AnalysisItemFolder_FieldCanChange Baseline_FieldCanChange Bug_FieldCanChange BusinessModel_FieldCanChange BusinessModelActivity_FieldCanChange BusinessModelFolder_FieldCanChange BusinessModelPath_FieldCanChange

	Component_FieldCanChange ComponentFolder_FieldCanChange ComponentStep_FieldCanChange Cycle_FieldCanChange DashboardFolder_FieldCanChange DashboardPage_FieldCanChange DesignStep_FieldCanChange Library_FieldCanChange LibraryFolder_FieldCanChange Release_FieldCanChange ReleaseFolder_FieldCanChange Req_FieldCanChange Resource_FieldCanChange ResourceFolder_FieldCanChange Run_FieldCanChange Step_FieldCanChange Test_FieldCanChange TestConfiguration_FieldCanChange TestFolder_FieldCanChange TestSet_FieldCanChange TestSetFolder_FieldCanChange TestSetTests_FieldCanChange

例 - 変更の許可または拒否

この関数は，特定のユーザ・グループに対して，不具合のステータス・フィールドを変更するアクセス許可を，フィールドの現在の値と新しい値に基づいて承認または拒否します。

```
Function Bug_FieldCanChange(FieldName, NewValue)
On Error Resume Next
if FieldName = "BG_STATUS" then
       if User.IsInGroup("QATester") then
              if Bug_Fields.Field("BG_STATUS").value ="Fixed" then
                 Select Case NewValue
                    Case "Fixed", "Closed"
                        Bug_FieldCanChange  = true
```

```
                    Case else
                        Bug_FieldCanChange  = false
                        Exit function
                    End select
                End if
            End if
        End if
        On Error GoTo 0
        End Function
```

Entity_ FieldChange

このイベントは，指定されたフィールドの値が変更されるときにトリガされます。フィールドからフォーカスが移動すると，値の変更によってフィールド変更イベントがトリガされます。

このイベント・プロシージャにコードを追加することで，特定フィールドの値が変更されるときにアクションを実行できます。たとえば，ユーザが別のフィールドに入力する値に応じて，あるフィールドの表示，非表示を切り替えることができます。

トピック	説明
構文	＜エンティティ＞_FieldChange (FieldName) FieldName はフィールドの名前です。
タイプ	サブルーチン
戻り値	
利用方法	AnalysisItem_FieldChange AnalysisItemFolder_FieldChange Baseline_FieldChange Bug_FieldChange BusinessModel_FieldChange BusinessModelActivity_FieldChange BusinessModelFolder_FieldChange BusinessModelPath_FieldChange Component_FieldChange ComponentFolder_FieldChange ComponentStep_FieldChange

	Cycle_FieldChange DashboardFolder_FieldChange DashboardPage_FieldChange DesignStep_FieldChange Library_FieldChange LibraryFolder_FieldChange Release_FieldChange ReleaseFolder_FieldChange Req_FieldChange Resource_FieldChange ResourceFolder_FieldChange Run_FieldChange Step_FieldChange Test_FieldChange TestConfiguration_FieldChange TestFolder_FieldChange TestSet_FieldChange TestSetFolder_FieldChange TestSetTests_FieldChange

例 - 依存関係の値

テスト・ステータスの値を「To Automate」に変更する際に，テンプレート記述が追加されます。

```
Sub Test_FieldChange(FieldName)

On Error Resume Next

  if Test_Fields.Field("TS_STATUS").Value="To Automate" then

     if Test_Fields.Field("TS_DESCRIPTION").value="" then

       myComments="<html><body><b>TO AUTOMATE-" & Now & "/ Checked
by " & user.UserName & "</b><br></body></html>"

       Test_Fields.Field("TS_DESCRIPTION").value= myComments

     Else

       myComments="<br><b>TO AUTOMATE-" & Now & "/ Checked by "&
       user.UserName & "</b><br>"

       Test_Fields.Field("TS_DESCRIPTION").value =
       Test_Fields.Field("TS_DESCRIPTION").value & "<br> "&
       myComments

     End if
```

```
  End if
 On Error GoTo 0
End Sub
```

例 - 変更時にセットアップを更新

不具合ステータスなどのフィールドが変化したときに，セットアップを更新します。

```
Sub Bug_FieldChange(FieldName)
On Error Resume Next
If FieldName=”BG_STATUS” then
   Select Case Bug_Fields.Field("BG_STATUS").value
          Case “New”
              Setup_Status_New
          Case “Open”
              Setup_Status_Open
          Case “Fixed”
              Setup_Status_Fixed
          Case “Closed”
              Setup_Status_Closed
  End select
End if
 On Error GoTo 0
End Sub
```

「依存関係リスト」の例も参照してください。

Entity_ CanPost

このイベントは，ALM がサーバにオブジェクトを送信する前に，オブジェクトが送信可能かどうかを確認するためにトリガされます。

このイベント・プロシージャにコードを追加して，特定の場合にオブジェクトの送信を防ぐことができます。

トピック	説明
構文	＜エンティティ＞_CanPost
タイプ	関数

戻り値	True または False
利用方法	AnalysisItem_CanPost AnalysisItemFolder_CanPost Baseline_CanPost Bug_CanPost BusinessModel_CanPost BusinessModelFolder_CanPost BusinessModelPath_CanPost Component_CanPost ComponentFolder_CanPost Cycle_CanPost DashboardFolder_CanPost DashboardPage_CanPost Library_CanPost LibraryFolder_CanPost Release_CanPost ReleaseFolder_CanPost Req_CanPost Resource_CanPost ResourceFolder_CanPost Run_CanPost Step_CanPost Test_CanPost TestConfiguration_CanPost TestFolder_CanPost TestSet_CanPost TestSetFolder_CanPost

例 - 更新の無効化

要件がコメントなしで完了した場合，ユーザは要件の送信を許可されません。

```
Function Req_CanPost
On Error Resume Next
if Req_Fields.Field("RQ_REQ_PRIORITY").IsModified then
   if Req_Fields.Field("RQ_DEV_COMMENTS").IsModified=false then
      Req_CanPost=false
      MsgBox "The priority was updated, you have to add a comment"
```

```
        Exit function
    End if
End if
On Error GoTo 0
End Function
```

Entity_CanDelete

このイベントは，ALM がサーバからオブジェクトを削除する前に，オブジェクトが削除可能かどうかを確認するためにトリガされます。

トピック	説明
構文	＜エンティティ＞_CanDelete
タイプ	関数
戻り値	True または False
利用方法	AnalysisItem_CanDelete AnalysisItemFolder_CanDelete Baseline_CanDelete Bug_CanDelete BusinessModel_CanDelete BusinessModelFolder_CanDelete BusinessModelPath_CanDelete Component_CanDelete ComponentFolder_CanDelete Cycle_CanDelete DashboardFolder_CanDelete DashboardPage_CanDelete Library_CanDelete LibraryFolder_CanDelete Release_CanDelete ReleaseFolder_CanDelete Req_CanDelete Resource_CanDelete ResourceFolder_CanDelete Test_CanDelete TestConfiguration_CanDelete TestFolder_CanDelete

	TestSet_CanDelete TestSetFolder_CanDelete

Entity_AfterPost

このイベントは，サーバにオブジェクトが送信された後にトリガされます。送信後，プロジェクト・フィールドは変更できません。データベースに新しい値が格納されないためです。

トピック	説明
構文	＜エンティティ＞_AfterPost
タイプ	サブルーチン
戻り値	
利用方法	AnalysisItem_AfterPost AnalysisItemFolder_AfterPost Baseline_AfterPost Bug_AfterPost BusinessModel_AfterPost BusinessModelFolder_AfterPost BusinessModelPath_AfterPost Component_AfterPost ComponentFolder_AfterPost Cycle_AfterPost DashboardFolder_AfterPost DashboardPage_AfterPost Library_AfterPost LibraryFolder_AfterPost Release_AfterPost ReleaseFolder_AfterPost Req_AfterPost Resource_AfterPost ResourceFolder_AfterPost Run_AfterPost Step_AfterPost Test_AfterPost TestConfiguration_AfterPost TestFolder_AfterPost

	TestSet_AfterPost TestSetFolder_AfterPost

例 - メールの送信

ターゲット・リリース・フィールドが更新された場合に，要件の作成者に通知メールが送信されます。メールを送信するために，sendreqmail という名前のカスタマイズ関数を追加する必要があります。

```
Sub Req_AfterPost

If Req_Fields.field("RQ_TARGET_RCYC").IsModified Then

   Sendreqmail Req_Fields.field("RQ_REQ_ID").Value,
   Req_Fields.field("RQ_REQ_AUTHOR").Value, "", "Target Cycle has
   changed", "Please Review"

End if

End sub

Sub sendreqmail(ReqId,Mto,cc,msubject,mcomment)

Dim tdc, bgf, bg

       Set tdc = TDConnection

       Set rf =  tdc.ReqFactory

       Set req = rf.Item(ReqId)

       req.Mail  mto , cc, 2, mSubject, mComment


       Set req = Nothing

       Set rf  = Nothing

       Set tdc = Nothing

End sub
```

ワークフローの例 - セットアップの定義

このワークフロー・コードは，フィールドのプロパティを更新します。更新されるプロパティは，可視性，読み取り専用，順序です。

これらの関数は不具合モジュール・ノードに追加します。

ユーザの確認

ユーザが特定のグループに属するかどうかを確認して，次の操作を決定します。

```
If User.IsInGroup("Developer") then
       Mygroup=”DEV”
End if
```

フィールドの外観の設定

このサブルーチンは，フィールドの外観を設定します。可視性，必須，ステータス，ページ番号，画面上での順序が設定されます。

```
Sub SetFieldApp(FieldName, Vis, Req, PNo, VOrder)
    With Bug_Fields(FieldName)
           .IsVisible = Vis
           .IsRequired = Req
           .PageNo = PNo
       .ViewOrder = VOrder
    End With
End Sub
```

初期状態にリセット

すべての不具合フィールドを非表示にするには，次のサブルーチンを追加します。

```
Sub ResetMetadata
       For i=0 to Bug_Fields.Count
           Bug_Fields.FieldById(i).IsVisible = false
       Next
End sub
```

ステータスの設定

次のサブルーチンは，ユーザのアクセス許可に応じて新規ステータスを設定します。このサブルーチンは各ステータスに対して作成する必要があります。

```
Sub Setup_Status_New
If User.IsInGroup("Developer") then
   Mygroup=”DEV”
ElseIf User.IsInGroup("QATester") then
   Mygroup=”QA”
ElseIf User.IsInGroup("Documentation") then
   Mygroup=”DOC”
End if

Call ResetMetadata ‘set to initial status
Select case Mygroup
  Case ”DEV”
     SetFieldApp "BG_ACTUAL_FIX_TIME", True, False, 0, 0
     SetFieldApp "BG_CLOSING_DATE", True, False, 0, 1
     SetFieldApp "BG_CLOSING_VERSION", True, False, 0, 2
     SetFieldApp "BG_DETECTED_BY", True, True, 0, 3
     SetFieldApp "BG_DETECTED_IN_RCYC", True, False, 0, 4
     SetFieldApp "BG_DETECTED_IN_REL", True, False, 0, 5
     SetFieldApp "BG_DETECTION_DATE", True, True, 0, 6
     SetFieldApp "BG_DETECTION_VERSION", True, False, 0, 7
     SetFieldApp "BG_ESTIMATED_FIX_TIME", True, False, 0, 8
     SetFieldApp "BG_PLANNED_CLOSING_VER", True, False, 0, 8
     SetFieldApp "BG_PRIORITY", True, False, 0, 10
Case ” QA”
     SetFieldApp "BG_ACTUAL_FIX_TIME", True, False, 0, 0
     SetFieldApp "BG_CLOSING_DATE", True, False, 0, 1
     SetFieldApp "BG_CLOSING_VERSION", True, False, 0, 2
     SetFieldApp "BG_DETECTED_BY", True, True, 0, 3
     SetFieldApp "BG_DETECTED_IN_RCYC", True, False, 0, 4
```

```
            SetFieldApp "BG_DETECTED_IN_REL", True, False, 0, 5
            SetFieldApp "BG_DETECTION_DATE", True, True, 0, 6
            SetFieldApp "BG_DETECTION_VERSION", True, False, 0, 7
      Case ”DOC”
            SetFieldApp "BG_ACTUAL_FIX_TIME", True, False, 0, 0
            SetFieldApp "BG_CLOSING_DATE", True, False, 0, 1
            SetFieldApp "BG_CLOSING_VERSION", True, False, 0, 2
            SetFieldApp "BG_DETECTED_BY", True, True, 0, 3
           SetFieldApp "BG_DETECTED_IN_RCYC", True, False, 0, 4
   End Select
End sub
```

第 4 章 結論

必要な機能を備えたパフォーマンスの高いソフトウェアの必要性は，ビジネスのイノベーションと成功に密接に関わっています。ビジネスにおけるソフトウェアの重要性の上昇と，仮想化やクラウドといった複雑で大きな変化を伴うトレンドのために，プロセス改善のニーズは高まり続けています。

HP ALM は，最新のアプリケーション・ライフサイクルのニーズに応えるために，戦略および計画チームへの統合を含むチーム間の整合性の改善，イノベーションを促進し，戦術的な遅れを防ぐベストプラクティスの提供，オペレーションの組織化の最も重要な最終段階への橋渡しといった機能を提供します。HP ALM は拡張可能で動的であり，ALM の動的性質に容易に適応できます。製薬から自動車製造までのさまざまな産業，従来型のウォーターフォール開発から最新のアジャイル開発までのさまざまな開発手法，フラットな組織から階層的な組織，さらにはマトリクス型の組織までのさまざまな組織構造といった，多種多様な適応を可能にする高い柔軟性を備えています。

多くの意味で，これは製品に組み込まれている多くのカスタマイズ機能の結果です。これにより，ALM を採用した各組織に固有のビジネス・プロセスを差別化するツールが得られます。ワークフロー・スクリプトを使うことで，プロジェクト管理者は，標準の手順や画面をプロジェクト固有のニーズに合わせて調整することができます。本書では，さまざまな使用方法について解説し，各種コーディング手法の利点と欠点を示し，最も有用なイベントの詳細を説明し，コードの作成に役立つ実用的な例を豊富に紹介しています。

本書に記載されたベスト・プラクティスは，組織における HP ALM ワークフローの適切な採用に大きな効果があるはずです。