HP Service Manager

サポート対象 Windows® および UNIX® オペレーティングシステム向け

ソフトウェアバージョン: 9.30

文書エンジンガイド

ドキュメントリリース日 : 2011 年 7 月 (英語版)

ソフトウェアリリース日 : 2011 年 7 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

商標

Adobe™は、Adobe Systems Incorporatedの登録商標です。

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

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

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

UNIX®は、The Open Groupの登録商標です。

ドキュメントの更新情報

このガイドの表紙には、次の識別情報が記載されています。

- ソフトウェアのバージョン番号は、ソフトウェアのバージョンを示します。
- ドキュメント リリース日は、ドキュメントが更新されるたびに変更されます。
- 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。

最新の更新のチェック、またはご使用のドキュメントが最新版かどうかのご確認には、次のサイトをご利用ください。

http://support.openview.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 の取得登録は、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(英語サイト)

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，最新版をご入手いただけます。詳細については、HP の営業担当にお問い合わせください。

サポート

HP ソフトウェア サポート オンライン Web サイトを参照してください。

http://support.openview.hp.com

HP ソフトウェアが提供する製品、サービス、サポートに関する詳細情報をご覧いただけます。

HP ソフトウェアサポートオンラインでは、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率的にアクセスいただけます。HP ソフトウェア サポート Web サイトのサポート範囲は次のとおりです。

- 関心のある技術情報の検索
- サポートケースとエンハンスメント要求の登録とトラッキング
- 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 Passport 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。HP Passport ID の取得登録は、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(英語サイト)

アクセスレベルに関する詳細は、以下の Webサイトにアクセスしてください。

http://support.openview.hp.com/access_level.jsp

目次

第 1 章

文書エンジンの概要

文書エンジンとは、RAD の変更を必要とせずにシステムをカスタマイズできるカスタマイズツールのことです。文書エンジンによって、リスト、表示、および検索などの標準的なアクションの権限と動作を、集中的に設定することができます。文書エンジンにより、モジュール間の整合性が向上します。文書エンジンの 3 つの主要なコンポーネントは、オブジェクト、ステータス、およびプロセスです。プロセスを再使用することで構造化プログラミングをさらに深めることができ、統合がシームレスになるため開発時間を節約できます。

文書エンジンとは?

文書エンジンは、Service Manager ワークフローを開発して変更するためのツールと方法の集まりから構成されます。文書エンジンの当初の目的は、Service Manager 内部で複数のモジュールをサポートし、モジュール間でのユーザインタフェースの整合性を改善し、新しいモジュールに必要となるコード量を減らす、基盤となる基本機能の集合を開発することにありました。

文書エンジンは、簡単でより拡張性に富んだアクション、特に複数のアプリケーションコールを伴うアクションで、表示アプリケーション機能を拡張します。さらに、文書エンジンは、結合テーブルとマスターフォーマット制御コールの使用をサポートします。文書エンジンは、設定済みの状態でほとんどの顧客の要求を満たすように設計されている一方で、柔軟性も備えています。オブジェクト、ステータス、およびプロセス間の関係は階層的です。

文書エンジンはオブジェクトによって動作を制御します。オブジェクトは、フォームを開いたときに常に参照されます。オブジェクトにより、そのフォームのステータスに適した動作 (オープン、リスト、検索など) が決定します。オブジェクトは、テーブルの動作全体を定義します。オブジェクト内では、ステータスはレコードがそのライフサイクルのどの位置にある(オープン、リスト、検索など) のかを記述します。ステータス内では、レコードに対してユーザが行ったアクションに応じて、さまざまなプロセスが実行されます。ステータスは、レコードの表示方法、および特定の期間や環境でどのオプション(アクション) が利用できるのかも定義します。例えば、ユーザのアクセス権限を指定すると、ステータスは保存などのアクションを判断できます。

プロセスはユーザのアクション時にステータスからコールされます。プロセスは RAD 式、JavaScript および既存 RAD アプリケーションへのコールを使用して、現在のレコードに対してアクションを実行します。

モジュールアプローチの利点

モジュール設計の利点としては、開発での整合性、開発期間の短縮、および柔軟性が挙げられます。

整合性

エンジンを用いることにより、あらゆるアプリケーションが同じベース RAD アプリケーションを使用して動作することが可能となるため、Service Manager アプリケーションスイートに整合性がもたらされます。ロック、アラート、承認、およびレコードリスト機能の使用などのコア機能はいずれも同じコードベースを使用することから、任意のモジュールで同じように機能します。

開発期間の短縮

文書エンジンを用いることで、既存コードとプロセスを再使用できます。

柔軟性

文書エンジンは、Service Manager アプリケーションの内部でのモジュールの動作を変更するための仕組みとして、プロセスレコードを使用します。システム付属の元のプロセスを変更したり削除する必要なく、ベースシステムとは異なる動作の新しいプロセスを作成できます。さらに、システムのベースプロセスをユーザ自身によるプロセスによってオーバーライドすることができるため、組織の特定の要件を満たすよう Service Manager をカスタマイズする場合に、システム開発者は柔軟に処理できます。

文書エンジンへのアクセス

文書エンジンにアクセスするには:

1. Service Manager クライアントを起動して、管理者としてログインします。
2. Service Manager のシステムナビゲータで**[カスタマイズ]**をクリックします。
3. **[文書エンジン]**をクリックします。ここから、オブジェクト、プロセス、またはステータスを定義するための主要な 3 領域にアクセスできます。オブジェクトからコールされるアラートと承認を設定したり、オブジェクトが使用する検索設定レコードを作成することもできます。

第 2 章

オブジェクト

オブジェクトとは、レコードの動作を決定し、定義と統制ルールを設定する定義の基本セットのことです。オブジェクトは、Service Manager 内のデータベースディクショナリ(dbdict) レコードと1 対 1 に対応します。テーブルに専用のオブジェクトレコードが存在しない場合、文書エンジンは DEFAULT オブジェクト内の設定を適用します。すべてのオブジェクトが定義されたリストとデフォルトのステータスを必要とします。指定がなければ、デフォルトのステータスは db.browse、db.list、db.search、および db.view になります。

注: DEFAULT オブジェクトレコードを**変更**したり**削除**しないでください。予想できない結果が生じる場合があります。

オブジェクトレコードは、文書エンジン内でテーブルの動作の定義と統制ルールを設定します。設定する項目の例を以下に挙げます。

- このユーザがテーブルの任意のレコードに対して実行できるアクションを決定するこのオブジェクト内で、ユーザプロファイルを作成するのに使用されるアプリケーション。
- 特定の環境内で使用されるステータスレコード(詳細については、「ステータス」を参照してください) 。
- オブジェクトのカテゴリ、フェーズ、ページのファイル名。
- このオブジェクトに使用される番号レコードの名前。
- このオブジェクトで使用されるロック方法。
- テーブルにあるレコードのリビジョンの設定。
- このオブジェクトに対して実行されるプロセスに利用可能な変数。
- このオブジェクト使用時に常に利用可能となるグローバルリスト。
- activity レコードの使用。
- このオブジェクトに対するアラートの処理方法。
- このオブジェクトの承認の処理方法。
- 作業キューの設定。
- パーソナルビューやグローバルビューとデフォルトテンプレートを設定する機能。
- このオブジェクト内のレコードの追加/更新/削除時の通知。
- このオブジェクトに対する追加の検索選択を設定する機能。

設定済みオブジェクトのリストを表示するには、オブジェクト定義フォームで**[検索]**をクリックします。

オブジェクト定義のフィールドとフィールド説明のリストを表示するには、「[オブジェクト定義]フォームとフィールド」(11 ページ)を参照してください。

オブジェクトの作成と更新

オブジェクトを作成するには:

1. 文書エンジンにアクセスします。手順については、「文書エンジンへのアクセス」(9 ページ)を参照してください。
2. **[オブジェクト]**をダブルクリックします。[オブジェクト]フォームがオープンします。
3. [オブジェクト]フォーム上のタブを使用して、目的の機能を実行するオブジェクトの作成に必要なフィールドに入力します。詳細については、フィールドの説明を参照してください。

オブジェクト定義を更新するには:

1. 文書エンジンにアクセスします。手順については、「文書エンジンへのアクセス」(9 ページ)を参照してください。
2. [オブジェクト]フィールドに更新するオブジェクトの名前を入力するか、**[検索]**をクリックして、オブジェクトを検索します。

[オブジェクト定義]フォームとフィールド

以下は、[オブジェクト定義]フォームのフィールドの説明です。

フィールド名	説明
ファイル名 *file.name*	オブジェクトのデータベースディクショナリ名を入力します。このオブジェクトの名前に対応するデータベースディクショナリ名を使用します(必須)。
共通名 *message*	システムは、データポリシからこの情報をフィルします。これは、オブジェクトの共通名です。共通名は「作業指示」などの簡単な名前にできます。
固有キー *unique.field*	システムは、データベースディクショナリからこの情報をフィルします。これは、オブジェクトの固有キーです。

このフォームには、以下のタブも含まれます。これらのタブ上のフィールドは、それぞれのタブのフィールド説明で説明します。

- 「[オブジェクト情報]タブのフィールドの説明」(12 ページ) - オブジェクトの全般的なプロパティと動作を指定します。
- 「[ロック]タブのフィールドの説明」(14 ページ) - オブジェクトのロックの動作を決定します。
- 「[改定]タブのフィールドの説明」(14 ページ) - オブジェクトの改定を追跡します。
- 「[変数/グローバルリスト]タブのフィールドの説明」(15 ページ) - オブジェクトによって使用されるローカル変数とグローバル変数を記述します。
- 「[アクティビティ]タブのフィールドの説明」(15 ページ) - ログへの記録(アクティビティ)を定義します。
- 「[アラート]タブのフィールドの説明」(16 ページ) - アラートを設定する場所と、アラートを生成する条件を定義します。
- 「[承認]タブのフィールドの説明」(17 ページ) - オブジェクトの承認オプションを設定します。

- 「[キューの管理]タブのフィールドの説明」(19 ページ) - キューとスレッドの表示方法、およびビュー(受信トレイ)を作成できるユーザを制御します。
- 「[ビュー/テンプレート]タブのフィールドの説明」(20 ページ) - ユーザが、オブジェクトのグローバルビューとパーソナルビュー、およびテンプレートを作成できるかどうかを定義します。
- 「[通知]タブのフィールドの説明」(21 ページ) - オブジェクトの任意のレコードの追加、削除、または更新アクティビティに関して自動的に送信される通知を識別します。
- 「[検索設定]タブのフィールドの説明」(21 ページ) - 検索画面の[詳細選択]タブで利用できるオプションを制御します。

[オブジェクト情報]タブのフィールドの説明

このタブは、オブジェクトの全般的なプロパティと動作を指定します。

以下は、[オブジェクト情報]タブのフィールドの説明です。

フィールド名	説明
説明フィールド *desc.field*	オブジェクトの簡潔な説明を指定します。
プロファイルアプリケーション *profile.appl*	追加と削除などの特定の関数をユーザが実行できるかどうかを決定するプロファイルを作成する RAD アプリケーションを指定します。例えば db.environment などです(必須)。
プロファイル変数 *profile.variable*	このオブジェクトがコールされるときに、環境レコードにアクセスすることなく常にアクセスできる変数を指定します。例えば $L.env などです(必須)。
番号レコード名 *number.record*	プロセス、フォーマットコントロール、または RAD コールのいずれかを経由して、getnumb アプリケーションのコールからアクセスできるオブジェクトの番号クラスを定義します。レコードの固有キーとなるシーケンシャル番号の取得にも使用できます。例えば EXWorkOrder などです。
カテゴリテーブル名 *category.file.name*	このオブジェクトのカテゴリ値に関連付けられたカテゴリテーブルにリンクするテーブル名を指定します。このタイプのレコードを表示するときに「カテゴリ」という名前のフィールドが存在すると、オブジェクトはカテゴリテーブル名と対応する名前が付いたレコードを検索します。存在する場合、カテゴリファイル名は変数 $L.category として格納されます。
フェーズテーブル名 *phase.file.name*	このオブジェクトに関連付けられたフェーズテーブルにリンクするテーブル名を指定します(適用可能な場合)。このタイプのレコードを表示するときに「フェーズ」という名前のフィールドが存在すると、オブジェクトはフェーズテーブル名を検索し、対応する名前が付いたレコードを選択します。存在する場合、フェーズテーブル名は変数 $L.phase として格納されます。
ページングテーブル名 *paging.file*	ページの格納に使用するテーブルの名前を指定します。ページは、最新の更新が適用される前に、現在のレコードの完全コピーとして作成されます。これは、レコードが更新されるたびに行われ、ここから詳細な監査証跡が作成されます。

フィールド名	説明
マスタフォーマットコントロール *master.fc*	マスタフォーマットコントロールレコードの名前を指定します(レコードオブジェクトのマスタフォーマットコントロールレコードが存在する場合)。マスタフォーマットコントロールにより、1 つの領域内のすべてのフェーズに適用されるフォーマットコントロールステートメントを 1 レコードで定義できます。変更管理の依頼フェーズなどです。通常、マスタフォーマットコントロールの名前は、インシデント管理、問題管理、またはサービスデスク内のデータベースディクショナリ、またはすべてのカテゴリの名前です。
結合定義 *joindef*	複数のテーブルの結合に使用する結合定義の名前(joincomputer など)、または有効な結合定義名に評価される式(joindef in $L.category など)を指定します。
ステータスフィールド *statusField*	レコードのステータス情報を含むフィールド名を指定します。
担当者フィールド *assignedToFields*	このオブジェクトの担当者名フィールドを含むフィールド名を指定します。このフィールドは、ログインしているオペレータにレコードが割り当てられていることを、フォルダ資格が確認するときに参照されます。
ワークグループフィールド *workgroupFields*	このオブジェクトの担当グループフィールドを含むフィールド名を指定します。このフィールドは、ログインしているオペレータのワークグループのいずれかにレコードが割り当てられていることを、フォルダ資格が確認するときに参照されます。
オープンステータス *open.state*	新規レコードをオープンするときに使用するステータス定義レコードを指定します。
クローズステータス *close.state*	既存のレコードのクローズ処理に使用するステータス定義レコードを指定します。
リストステータス *list.state*	レコードのリストを表示するのに使用するステータス定義レコードを指定します。
デフォルトステータス *default.state*	オブジェクトのデフォルトとして使用されるステータスの名前を指定します。このオブジェクトのレコードの編集にデフォルトのステータスが使用されます。
検索ステータス *search.state*	検索に使用するステータス定義レコードを指定します。
ブラウズステータス *browse.state*	レコードがロックを使用するときに使用するステータス定義レコードを指定します。基本的に、このフィールドは別のユーザによってレコードが現在ロックされているときの読み取り専用ステータスを定義します。

フィールド名	説明
手動ステータス *manual.states*	オープン、クローズ、リスト、表示、検索、または参照のライフサイクルステータス以外の、このオブジェクトに使用できるステータスの配列を指定します。

[ロック]タブのフィールドの説明

[ロック]タブは、オブジェクトのロックの動作を決定します。これは、誰かがレコードを更新しているとき、システムが別のユーザが同じレコードを更新できなくすることを意味します。

以下は、[ロック]タブのフィールドの説明です。

フィールド	説明
ロックを使用 *use.locking*	ロックを有効にするには、このチェックボックスをオンにします。
表示する時にロック *lock.on.display*	true または true に評価される条件の場合に、Service Manager はレコードが表示されると、直ちにレコードのロックを試みます。このフィールドを使用するには、[ロックを使用]がオンである必要があります。
上位レコードをロック *lock.parent*	現在のレコードと上位のレコードをロックします。例えば、このフィールドがオンである場合に、誰かが変更タスクを更新すると、そのタスクの変更依頼もロックされます。
上位 ID フィールド *parent.id*	ロックする上位の ID を含む現在のレコードのフィールド名を入力します。
上位ファイル名/オブジェクト *parent.object*	上位レコードを含むテーブルの名前。
監視変数 *watch.variables*	監視変数は、レコードが変更されたかどうかを文書エンジンが確認する場合に使用されます。レコードを最初に表示するときは、監視変数は NULL である必要があります。

[改定]タブのフィールドの説明

[改定]タブは、オブジェクトの改定の動作を決定します。

以下は、このタブのフィールドの説明です。

フィールド名	説明
改定テーブル名 *revision.file*	このオブジェクトのレコードに対する改定を保存するテーブルの名前を指定します。

フィールド名	説明
改定の最大数 *max.revisions*	このオブジェクトに許される改定の合計数を指定します。空欄にすると、上限はなくなります。

[変数/グローバルリスト]タブのフィールドの説明

[変数/グローバルリスト]タブは、オブジェクトによって使用されるローカル変数とグローバル変数を記述します。

注：グローバル変数は、メモリ内に作成され格納されます。

以下は、[変数/グローバルリスト]タブのフィールドの説明です。

フィールド名	説明
ローカル変数 *local.variables*	プロセス内で使用できるローカル変数のリストを入力します。ローカル変数はアプリケーション内で定義します。ローカル変数はユーザが作成するオブジェクトに割り当てられ、そのオブジェクトに関連付けられたすべてのプロセスとステータスで利用できます。他のオブジェクトは、ローカル変数を利用できません。
グローバルリスト *global.lists*	グローバルリストは、作成するとすべてのプロセスで利用できます。グローバルリストが起動リストに追加されている場合は、ログイン時に構築することができます。グローバルリストは、オブジェクトにアクセスするたびに利用できます。

[アクティビティ]タブのフィールドの説明

[アクティビティ]タブは、オブジェクトの更新(アクティビティ)ログを定義します。

以下は、[アクティビティ]タブのフィールドの説明です。

フィールド名	説明
アクティビティログテーブル *activitylog.file.name*	オブジェクトのアクティビティログエントリを保持するテーブルの名前。
選択リスト変数 *activity.selection.var*	特定オブジェクトのレコードの更新を実行するときに、オペレータが選択できるアクティビティのタイプを表示するのに更新フォームで使用する変数。
通知リンク *activity.post.link*	情報の通知に使用するリンクの名前。
アクティビティレコードが生成されない場合に更新が必	アクティビティの更新が必要な場合にオンにするチェックボックス。

フィールド名	説明
要 *activity.mandatory*	
更新するフィールド *update.field.var*	フォーム上のアクティビティの更新を含むフィールド、または変数。このフィールドは、**[アクティビティレコードが生成されない場合に更新が必要]**がオンである場合にのみ表示されます。
表示メッセージ *activity.mandatory.msg*	アクティビティの更新が必要なことを示すメッセージ。このフィールドは、**[アクティビティレコードが生成されない場合に更新が必要]**がオンである場合にのみ表示されます。

[アラート]タブのフィールドの説明

[アラート]タブは、アラートを設定する場所と、アラートを生成する条件を定義します。一意のキーがある任意のオブジェクトが、これらのアラートを使用できます。

以下は、[アラート]タブのフィールドの説明です。

フィールド名	説明
アラートの場所 *alert.location*	[アラートの場所]は、実行するアラートの定義の名前が存在する(格納された)場所を定義します。アラートの定義が格納された場所を指定します。 レコード:レコード自体にアラートを格納します。 カテゴリ:[オブジェクト情報]タブで定義されたカテゴリファイルにアラートを格納します。 フェーズ:[オブジェクト情報]タブで定義されたフェーズレコードにアラートを格納します。 オブジェクト:オブジェクトレコードにアラートを格納します。これを選択すると、アラート配列テーブルが表示され、使用するアラートをフィルできます。 注:オブジェクトのテーブルは、アラートに関連付ける一意のキーが必要です。一意のキーではなく、Null キーがないテーブルはアラートを使用できません。
アラート条件 *alert.condition*	アラートを処理するかどうかを判断するための条件を指定します。例えば、open in $L.file~=false などです。
アラートフィールド名 *alert.field.name*	アラートの場所で定義された実際のアラート名を含むフィールド名を指定します。
アラートステータスフィールド *alert.status.field*	アラートの処理後にアラートステータスを書き込む、現在のレコード内のフィールドを指定します。

フィールド名	説明
アラート更新処理 *alert.update.process*	アラートの実行後、システムが実行する追加関数のプロセスレコードの名前を指定します。
アラートのログを記録 *log.alerts*	オンにすると、アラート履歴を維持するため、処理後にアラートはアラートログファイルに移動します。
上位に対してアラートを処理 *alerts.against.parent*	[ロック/改定]タブで[上位レコードをロック]フィールドを選択した状態で、このチェックボックスをオンにすると、アラートはアクティブになると上位レコードにも登録されます。
アラートを再計算する条件 *alert.recalc*	既存のアラートの条件を再計算するかどうかを決定する条件を指定します。
アラートをリセットする条件 *alert.reset*	既存のアラートをすべて削除し、すべての条件を再計算する場合を決定します。

[承認]タブのフィールドの説明

[承認]タブは、オブジェクトの承認オプションと、承認オプションに関連付けられた通知を設定します。承認は ApprovalDef ファイルで定義されます。

以下は、[承認]タブのフィールドの説明です。

フィールド名	説明
承認条件 *approval.condition*	承認条件が true に評価されると、オブジェクトのレコードに承認が使用されます。
承認場所 *approval.location*	承認情報の格納先を、レコード、フェーズ、オブジェクト、またはカテゴリから指定します。
承認フィールド名 *approval.field.name*	フィールド名には、承認場所で定義されたテーブル内の実際の承認名が含まれます。
承認ステータスフィールド *approval.status.field*	承認ステータスを格納する現在のレコード内のフィールド。
承認グループ *approval.groups*	このオブジェクトの承認を発行するために、現在のユーザが所属する必要があるグループを含む変数を格納します。
承認タイプ *appr.cond.type*	承認には 4 つの定義済みのタイプがあります。 全員の承認が必要 :システムがレコードのステータスを承認に設定する前に、承認定義で定義されたすべてのグループ/オペレータが承認を発行する必要があります。一部の(すべてではない) グルー

フィールド名	説明
	プ/オペレータしか承認を発行していない場合、ステータスは承認待ちとなります。 1人の承認が必要:承認するグループ/オペレータのメンバーのうち、1 人 (グループ) の承認によってレコードが承認されます。 定足数:承認グループの過半数のメンバが承認を行った場合にレコードが承認されます。 全員の承認が必要 - 即時否認:すべてのグループ/オペレータがレコードを承認する必要があります。最初の否認によって、ステータスが Deny に変わります。その他すべての承認者はアクションを実行する必要はありません。
承認通知 *single.notify.approval*	依頼が承認されたときに実行する通知を選択します。
否認通知 *single.notify.denial*	1 人の承認者が依頼を否認したときに実行する通知を選択します。
撤回通知 *single.notify.retraction*	以前のアクションを撤回したときに実行する通知を選択します。
最終承認通知 *final.notify.approval*	最終的な承認が与えられたときに送信する通知を選択します。
最終否認通知 *final.notify.denial*	依頼が否認されたときに送信する通知を選択します。
承認 FC *appr.fc*	承認時に実行するフォーマットコントロールレコードの名前を指定します。
承認プロセス *approval.process*	レコードが承認されたときに実行するプロセスを選択します。
否認プロセス *denial.process*	レコードが否認されたときに実行するプロセスを選択します。
オープン時に予備承認 *preapprove.cond*	レコードを自動的に承認するべきかどうかを決定します。 条件が true であり、ユーザが承認待ちのグループのうちの 1 つに所属する場合、承認は自動的に処理されます。ユーザが承認待ちのグループの 1 つに所属しない場合、承認は自動的には行われず、通常の承認プロセスを経由して実行する必要があります。デフォルト値は true です。

フィールド名	説明
承認をログ *log.approvals*	ApprovalLog テーブルに承認の履歴をログ記録するには、このチェックボックスをオンにします。
承認コメントが必要 *approval.comments*	オンにすると、承認者からの承認コメントが必要になります。
承認を集計する *aggregate.approvals*	オンにすると、承認は累積的になります。
承認を再計算する条件 *approval.recalc*	既存の承認の条件を再計算するかどうかを決定する条件を指定します。
承認をリセットする条件 *approval.reset*	既存の承認をすべて削除し、すべての条件を再計算する場合を決定します。

[キューの管理]タブのフィールドの説明

[キューの管理]タブは、キューとビュー、およびスレッドの表示方法を制御します。関連するオブジェクトレコードのないファイルのデータポリシでこれらと同じフィールドが利用できます。また、オブジェクトレコードが存在する場合は、オブジェクトレコードから datadict レコードにこれらのフィールドが仮想的に結合されます。

以下は、[キューの管理]タブのフィールドの説明です。

フィールド名	説明
管理条件 *scm.condition*	特定のユーザだけにこのオブジェクトのレコードを表示するキューの表示を許可する条件を指定します。例えば browse in $G.pm.environment などです。
管理表示フォーマット *scm.manage.screen*	ビューの表示に使用するフォーマット。設定済み Service Manager には、デフォルトの表示フォーマット sc.manage.generic があり、その他のフォームが選択されていない場合に使用されます。HP は、sc.manage.generic フォーマットを変更しないことを推奨します。
管理デフォルトビュー *scm.inbox*	このキューのデフォルトビューを選択します。特定のユーザ用にユーザビューを指定し、HP Service Manager キューの管理用に特定のビューのリストを設定できます。ユーザに特定のビューが定義されていない場合、デフォルトのユーザビューが使用されます。
管理デフォルトクエリ *scm.query*	デフォルトビューが選択されていない場合に実行するデフォルトクエリを指定します。
デフォルトクエリの説明 *scm.query.name*	上記フィールドの名前を指定します。このフィールドにメッセージを関連付けられます。例えば scmsg(491, "us") などです。

フィールド名	説明
スレッドビュー -> 検索 *scm.thread.list.edit*	true または true に評価される式を指定すると、検索実行時に新規スレッドが開きます。
検索フォーマット (必要な場合) *scm.search.format*	デフォルトの検索フォーマットを選択します。
スレッド検索 -> リスト *scm.thread.search.list*	true または true に評価される式を指定すると、ユーザが表示するレコードのリストを検索するときに新規スレッドが開きます。
スレッドリスト -> 編集 *scm.thread.list.edit*	true または true に評価される式を指定すると、ユーザがレコードのリストから表示するレコードを選択するときに新規スレッドが開きます。
スレッドビュー -> 編集 *scm.thread.inbox.edit*	true または true に評価される式を指定すると、ユーザがキューから既存レコードを表示するときに新規スレッドが開きます。
追加を許可する条件 *scm.add.condition*	オペレータがレコードを追加できるかどうかを評価する式を指定します。
追加/オープンアプリケーション *scm.add.appl*	レコードを追加、またはオープンするときにコールするアプリケーションの名前を指定します。
パラメータ名 *scm.add.names*	[追加/オープンアプリケーション]フィールドで指定されたアプリケーションに渡すパラメータ名を指定します。
パラメータ値 *scm.add.values*	[追加/オープンアプリケーション]フィールドで指定されたアプリケーションに渡すパラメータ値を指定します。

[ビュー/テンプレート]タブのフィールドの説明

[ビュー/テンプレート]タブは、ユーザがグローバルビューとパーソナルビュー、およびテンプレートサポートを作成できるかどうかを定義します。

以下は、このタブのフィールドの説明です。

フィールド名	説明
パーソナルビューを作成可能 *personal.inbox*	true または false に評価される条件を指定します。true で、ユーザはパーソナルビューを作成できます。
システムビューを作成可能 *global.inbox*	ユーザがグローバルビューを作成できるかどうかを決定するための、true または false に評価される条件を指定します。

フィールド名	説明
デフォルトテンプレート *default.template*	このテーブルのレコードに使用するデフォルトテンプレートの名前を指定します。
テンプレートをサポートする *supportTemplates*	オブジェクトのテンプレートのサポートを有効にするには、このチェックボックスをオンにします。

[通知]タブのフィールドの説明

[通知]タブは、オブジェクトの追加、更新、または削除アクティビティに対して自動的に送信される通知を識別します。

以下は、[通知]タブのフィールドの説明です。

フィールド名	説明
追加 *notification.add*	テーブルにレコードが追加されると自動的に送信される通知を選択します。
更新 *notification.update*	テーブルが更新されると自動的に送信される通知を選択します。
削除 *notification.delete*	テーブルからレコードが削除されると自動的に送信される通知を選択します。

[検索設定]タブのフィールドの説明

[検索設定]タブは、検索画面の[詳細選択]タブで利用できるオプションを制御します。

以下は、[検索設定]タブのフィールドの説明です。

フィールド名	説明
テーブル名 *tablename*	クエリ対象のテーブル名。
検索フォーマット *searchFormat*	[詳細選択]タブで使用されるサブフォーマットの名前。
初期化プロセス *init.process*	検索フォームを表示する前に実行するプロセス名。
詳細検索を許可 *allowAdvAccess*	詳細検索を許可するかどうかを決定する条件を定義します。

[定義されたクエリ]タブ

[定義されたクエリ]タブは、検索画面の[詳細選択]タブ([インシデントの検索]フォームの[詳細選択]タブなど)で使用するクエリステートメントとラベルを定義します。フィールドは SearchConfig テーブルで定義されます。

以下は、[定義されたクエリ]タブのフィールドの説明です。

フィールド名	説明
ID *id*	クエリの一意の ID を入力します。スペースを含む特殊文字を含めることはできません。
クエリ *query*	システム言語の構文を使用するクエリ式。`assignee.name=operator()` などです。
説明 *description*	[詳細選択]タブ上のチェックボックスのラベルを入力します。

[範囲]タブ

[範囲]タブで、開始日と終了日の間の範囲の検索を簡単に設定できます。これには、**[設定の修正]**リンクをクリックして、フォームの入力として使用する開始日と終了日を表す変数を定義します。[変数 1]列にこの変数を入力し、[フィールド]と[演算子 1]列を使用して、実行するクエリを定義します。[設定の修正]では、[定義されたクエリ]タブの変更も行われます。

以下は、[範囲]タブのフィールドの説明です。

フィールド名	説明
フィールド *fieldName*	クエリ内で使用するテーブル内のフィールドを指定します。
演算子 1 *operator1*	クエリ内の比較演算子です(例 : >=)
変数 1 *variable1*	フォーム上で入力として使用する変数を指定します。
特殊タイプ *specialType*	現時点では使用されません。

第 3 章

ステータス

ステータスはオブジェクトによってコールされ、プロセスによって定義されます。ステータスレコードには、ある特定の期間におけるレコードの表示および動作の方法に関する情報が含まれます。

ステータスレコードの定義には、以下の項目を含められます。

- ステータス名。
- レコードの表示に使用する表示画面。
- リストが最初に表示される場合の初期プロセス。
- データの表示に使用するフォーマット。
- ユーザがレコードを変更できるかどうかを示す入力条件。
- ユーザが特定表示オプション(表示アクション)をトリガした場合に実行するプロセス。

使用されるステータスレコードは、ユーザが表示しているレコードの数によって異なります。レコードの検索、リストの表示、単一レコードの表示、レコードの参照を行うための設定済みステータスレコードが存在します。

検索

現在のファイル変数にレコードが存在しない場合、ユーザが検索モードにあることが想定されます。使用されるステータスは、テーブルのオブジェクトレコードで定義された検索ステータスです。デフォルトのステータスは db.search です。

レコードリスト

レコードのリストを表示するときには、テーブルオブジェクトレコードで定義されたリストステータスが使用されます。デフォルトのリストは db.list です。

注: Service Manager の一部の旧バージョンでは、レコードリストのことを QBE リストと呼びます。

単一レコードの表示

単一レコードを表示する場合、レコードが最初に確認され、データベースディクショナリの[ステータス]フィールドが含まれるかどうかが確認されます。フィールドが存在して入力されている場合、そのフィールドの内容が、レコードの現在のステータスとして使用されます。このフィールドが存在しないか、NULL の場合、テーブルのオブジェクトレコードで定義されているデフォルトステータスが使用されます。デフォルトステータスのデフォルト値は db.view です。

レコードの参照

読み取り専用モードでレコードを参照するときには、テーブルのオブジェクトレコードで定義された参照ステータスが使用されます。ロックを使用するファイルのみが参照ステータスを必要とします。更新権限なしにレコードを表示すると参照画面に似ていますが、この画面はデフォルトステータスを使用していて、参照ステータスは使用していません。デフォルトの参照ステータスはありません。

表示アプリケーションの統合のヒント

表示画面に関連付けられた表示イベントが存在しない場合、OnFormModified のときに se.lock.object を実行する se.default 表示イベントが自動的に使用されます。表示オプションレコードの[レコードの修正]チェックボックスがオンであると、ボタンが押されたときにレコードはロックされます。

ステータスの作成と更新

新しいステータスを作成するには:

1. 文書エンジンにアクセスします。「文書エンジンへのアクセス」(9 ページ)を参照してください。
2. [ステータスの定義]フォーム上のタブを使用して、目的の機能を実行するステータスの作成に必要なフィールドに入力します。「ステータスの定義のフィールドの説明」(25 ページ)のフィールドの説明を参照してください。

既存ステータスを変更するには:

1. 文書エンジンにアクセスします。「文書エンジンへのアクセス」(9 ページ)を参照してください。
2. [ステータス]フィールドに更新するステータスの名前を入力するか、**[検索]**をクリックして、ステータスを検索します。

ステータスの定義のフィールドの説明

ステータスの定義レコードは、ある特定の期間におけるレコードの動作と表示の仕方を定義します。

ステータスレコードでは、非ベース方法を定義したり、ベース方法の動作を変更できます。ベース方法は、「se.search.engine の基本関数」、「se.view.engine の基本関数」、および「se.list.engine の基本関数」で説明されています。これらには、保存、検索、フィル、OK、キャンセル、および検索などの関数が含まれます。

以下は、[ステータスの定義]フォームのフィールドの説明です。

フィールド名	説明
ステータス *state*	ステータスの名前を指定します(必須)。
表示アプリケーション *display.screen*	ステータスに関連付ける表示アプリケーション。
初期化プロセス *init.process*	ステータスに入る前に実行するプロセスの名前。
フォーマット *format.name*	ステータスレコードをオープンするフォーマットを指定します。このフォームはレコードに格納され、変数にしたり、またはハードコードすることができます。フォーマットの名前はハードコードしたり、変数に含めたり、システム言語の式を経由してレコードから取得できま

フィールド名	説明
	す。
入力条件 *input.condition*	ステータスを表示するための入力条件を評価し、レコードが読み取り専用であるかどうかを決定します。読み取り専用には false を、編集可能には true を指定します。true または false に評価される式を入力することもできます。それ以外の場合、「true」を入力するか true または false に評価される式を入力します。
非ベース方法	
表示アクション *process.label*	ユーザからの入力後に、表示アクションで指定されるアクションパラメータを指定します。表示アクションは、ユーザがクリックするボタンやメニューアイテムの表示オプションにあるアクションフィールドで指定します。
プロセス名 *valid.process*	アクションの結果として実行されるプロセスの名前を指定します。
条件 *process.condition*	true に評価されると [プロセス名] で指定されたプロセスがコールされる式を指定します。
最初に保存 *run.save.before*	このプロセスを実行する前にプロセスの保存を実行する場合に選択します。最初に保存するには true を、それ以外の場合は false を入力します。デフォルトは false です。

第 4 章

プロセス

プロセスとは、文書エンジンが利用できる作業の最小独立単位のことであり、このレベルでデータが操作されます。ユーザは、独自のプロセスを作成したり、Service Manager 付属の 700 を超えるプロセスのいずれかを使用できます。プロセスの定義は、[初期式]、[初期 JavaScript]、[RAD]、[最終式]、[最終 JavaScript]と[次のプロセス]に入力される初期式、RAD と最終式、およびオプションの次のプロセスのコールから構成されます。式は RAD 式、JavaScript のいずれかまたは両方を使用して記述します。

プロセスの作成と変更

プロセスを作成するには:

1. 文書エンジンにアクセスします。「文書エンジンへのアクセス」(9 ページ)を参照してください。
2. プロセスパネル上のタブを使用して、目的の機能を実行するプロセスの作成に必要なフィールドに入力します。詳細については、「[プロセス定義]フォームとフィールドの説明」(28 ページ)を参照してください。

プロセスを変更するには:

1. 文書エンジンにアクセスします。「文書エンジンへのアクセス」(9 ページ)を参照してください。
2. [プロセス名]フィールドに変更するプロセスの名前を入力するか、**[検索]**をクリックして、プロセスを検索します。

[プロセス定義]フォームとフィールドの説明

[プロセス定義]フォームで、新規プロセスを定義したり、既存プロセスを編集します。プロセスはコードや式を実行し、ユーザが選択したアクションを実行します。

以下は、[プロセス定義]フォームのフィールドの説明です。

フィールド名	説明
プロセス名 *process*	プロセスの名前を指定します(必須)。
カーソル位置を保存 *save.cursor.position*	アクション後に同じカーソル位置に戻る場合に、このボックスをオンにします(例えば、フィルの場合など)。
完了時に標準プロセスを起動 *run.standard*	オンにすると、現在のアクションやプロセスの完了後に標準プロセスが実行されます。標準プロセスには、保存などがあります。ユーザが保存プロセスを作成していて、定義したこの保存プロセスの完了後に文書エンジン付属の保存プロセスを実行する場合、この

フィールド名	説明
	ボックスをオンにします。標準プロセスは、このマニュアルの基本関数のセクションで定義されています。
ウィンドウで実行 *run.in.window*	オンにすると、プロセスは別のウィンドウで実行されます。
ウィンドウタイトル *window.name*	**[ウィンドウで実行]**チェックボックスがオンの場合に、ウィンドウのタイトルを指定します。scmsg 式を使用して、ローカライズされたウィンドウタイトルを使用できます(scmsg(1980, "us") など) 。

上記のフィールドに加え、このフォームにはプロセスをさらに定義することができるタブもあります。これらのタブには、以下が含まれています。

- 「[初期式]タブ」(29 ページ)
- 「[初期 JavaScript]タブ」(29 ページ)
- 「[RAD]タブ」(29 ページ)
- 「[最終式]タブ」(30 ページ)
- 「[最終 JavaScript]タブ」(30 ページ)
- 「[次のプロセス]タブ」(30 ページ)

[初期式]タブ

[初期式]タブは、初期 JavaScript と[RAD]タブで定義された RAD コールの前に実行される初期式を定義します。初期式は、標準的な Service Manager の式を使用して記述します。

[初期 JavaScript]タブ

[初期 JavaScript]タブは、[RAD]タブで定義された RAD の前に実行される初期 JavaScript 式を定義します。

[RAD]タブ

[RAD]タブでは、プロセスの途中に実行されるプレ RAD 式、RAD コール、ポスト RAD 式を定義します。

以下は、[RAD]タブのフィールドの説明です。

注: これらのフィールドは、このタブで定義されている各 RAD アプリケーションに繰り返し現れます。

フィールド名	説明
RAD 呼び出しの前に評価される式 *pre.rad.expressions*	[RAD アプリケーション]フィールドで定義された RAD アプリケーションの前に実行する式を指定します。RAD アプリケーションに渡されるすべてのパラメータ値は、変数、または式の形式である必要があります。プレ RAD 式で変数に値が割り当てられている必要があります。例えば $L.value.name="Test" などです。
RAD アプリケーション *application*	実行する RAD アプリケーションの名前を指定します。
条件 *rad.condition*	RAD アプリケーションを実行すべき状況(条件が true に評価される場合)、スキップすべき状況(条件が false に評価される場合)を指定します。
パラメータ名 *names*	RAD アプリケーションに渡されるパラメータ名を指定します。
パラメータ値 *values*	RAD アプリケーションに渡されるパラメータ値を指定します。これらの値は、変数、または式の形式である必要があります。二重引用符で囲む("Wizard Name")と文字列値を渡せます。
ポスト RAD 式 *post.rad.expressions*	RAD アプリケーション完了後に実行する RAD 式を指定します。

[最終式]タブ

[最終式]タブは、[RAD]タブの処理が完了した後に実行する最終式を定義します。最終式は、標準的な Service Manager の式を使用して記述します。

[最終 JavaScript]タブ

このタブで定義される JavaScript は、最終式と[RAD]タブ上の RAD アプリケーションの実行後に実行されます。

[次のプロセス]タブ

このタブは、現在のプロセスが完了すると実行する次のプロセスを指定します。

以下は、[次のプロセス]タブのフィールドの説明です。

フィールド名	説明
次のプロセス *next.process*	実行する次のプロセスの名前を指定します。
条件 *process.condition*	[次のプロセス]フィールドのプロセスに関連付ける、true または false に評価される条件を指定します。sm.close プロセスの true などです。

第 5 章

文書エンジンのリソース

文書エンジンには、任意のオブジェクトによって使用できるローカル変数と基本関数が含まれています。基本関数は、ユーザが利用できる標準アクションを実行するようにあらかじめ定義されています。例えば、ユーザがレコードを更新した後に[保存]ボタンをクリックすると、文書エンジンはこの依頼を処理して、データベースに変更を書き込みます。基本関数は、別のプロセスを実行するために、ステータスレコードで上書きできます。

DEFAULT オブジェクト

データベースマネージャによってアクセスされるファイルは、対応するオブジェクト向けに定義されたルールとプロセスを自動的に使用します。ファイルに定義されたオブジェクトがない場合、DEFAULT オブジェクトが使用されます。DEFAULT オブジェクトはデータベースマネージャの以前のバージョンの機能を複製します。

システム管理者は、[管理モード]チェックボックスがオンであれば、データベースマネージャからファイルにアクセスすることで、対応オブジェクトのあるファイルに DEFAULT オブジェクトからアクセスすることもできます。このチェックボックスは、システム管理者のみが利用できます。ケイパビリティワード "AlwaysAdmin"のあるユーザは、Service Manager 内で情報にアクセスするとき、常に DEFAULT オブジェクトを使用します。ユーザにこのケイパビリティワードを割り当てることは推奨されません。

文書エンジンは環境プロファイルを使用します。従来のバージョンのシステムでは、システム管理者にはフォーマットコントロール設定にかかわらず、すべての権限(追加、更新、削除)が与えられていました。文書エンジンでは、与えられる権限はフォーマットコントロールレコードで定義された権限と一致します。従来の方法での権限付与を希望する管理者は、DEFAULT オブジェクトレコードの[アプリケーションプロファイル]設定を db.environment から db.environment.sysadmin に変更できます。

RAD アプリケーション

データベース駆動型アプリケーションにより、ユーザは以下の 3 つの基本レコードセットを操作できます。

- レコードなし(情報を検索する場合)。
- 複数レコード(情報のリストを検索する場合)。
- 1 レコード(単一のレコードを変更する場合)。

データ操作時、ユーザの観点からは以下の作業を実行します。

- 検索情報を入力(または新規レコードを作成)できる空のフォームを表示する。
- または、単一レコードの更新/表示を実行する。
- または、検索を実行して、リストとして表示される複数レコードを得る。

文書エンジンは、この概念を実現するのに 3 つの主要な RAD アプリケーションを使用します使用される RAD ルーチンは、いつでも表示されているレコード数によって決定されます。

se.search.engine

検索エンジンは、表示されているレコードが存在しない場合に使用されます。このルーチンの主な目的は、クエリを公式化して、正しいテーブルからレコードを選択することにあります。このルーチンは、データベースに新しいレコードを追加する目的で、空のレコードに情報を最初に入力する場合にも使用できます。デフォルトのステータスは db.search です。

se.list.engine

リストエンジンは、複数レコードを表示します。リストエンジンを使用することで、ユーザはリストから特定レコードを選択したり、レコードの全体のリストに対しアクションを実行できます。デフォルトのステータスは db.list です。

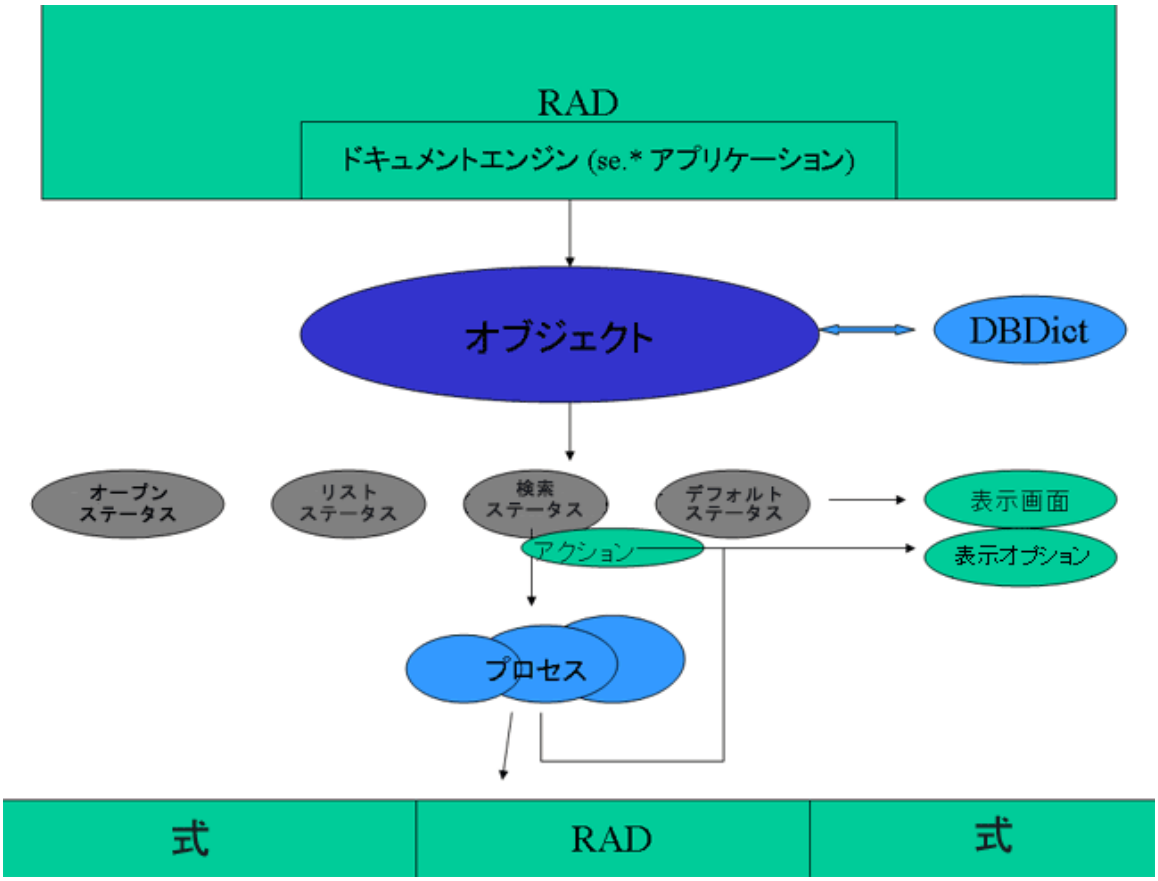
se.view.engine

表示エンジンは単一レコードを表示します。このアプリケーションは、更新や削除など、特定の 1 レコードに対してアクションを実行するのに使用されます。デフォルトのステータスは db.view です。

注：Service Manager のレコードリスト機能を使用する場合には、表示エンジンはリストと単一レコードの両方の情報に使用されます。

RAD アプリケーションフロー

次の図は、他のカスタマイズツールが文書エンジンと対話する仕組みの概要です。



注：文書エンジンに関連する RAD アプリケーションは、se.view.engine のようにすべて "se" が先頭に付きます。文書エンジンは、各レコードをユーザが利用できる文書のように扱います。

文書エンジンを使用してファイルを表示すると、文書エンジンは本マニュアル内で前述した 3 つのアプリケーションのいずれか 1 つを使用します。表示画面、表示オプション、およびフォーマットは、レコードの現在のステータスによって決定します。表示オプションがトリガーされると、標準表示機能が実行された後、エンジンは現在のステータスレコードで定義された利用できるプロセスと照らし合わせてそのオプションのアクションを確認します。true に評価される条件が存在するプロセスが定義されていれば、エンジンは現在のレコード (またはレコードセット) に対してそのプロセスを実行します。ステータスレコードにアクションが定義されていない場合、エンジンは、そのアクションが現在のアプリケーション内の基本プロセスとして定義されているかどうかを確認します。定義されていれば、レコードに対してその基本プロセスが実行されます。定義されていなければ、アクションは実行されません。

表示アプリケーションの統合のヒント

表示アプリケーションに関連付けられた表示イベントが存在しない場合、OnFormModified のときに se.lock.object を実行する se.default 表示イベントが自動的に使用されます。

表示オプションレコードの [レコードの修正] チェックボックスがオンであると、ボタンが押されたときにレコードはロックされます。

se.search.engine の基本関数

表示アクション	説明
back	レコードを終了します。
find	フィールドにリンクされているレコードからの詳細情報を表示します。
fill	ターゲットレコードにリンクされたレコードからの情報をフィルします。ターゲットフィールドにソースレコードからのデータを入力します。
advanced	豊富な検索オプションを備えた詳細検索プロセスを起動します。通常、これはシステム管理者のみが利用できますが、このオプションをその他のユーザが利用できるようにすることが可能です。
clear	現在の画面をクリアします。
openinbox / inbox	現在のファイルに関連付けられたビューを開くようにユーザに求めます。これは、ビューとして結果を表示する、定義済みのクエリです。
search	入力された情報を使用して検索クエリを作成することで標準クエリを実行し、レコードリストに結果を表示します。
add	add フォーマットコントロールを実行し、データベースへのレコードの追加を試みます。
restore	画面のコンテンツを復元します (clear の実行後)。
irquery	IR クエリテキスト検索を実行します。
validitylookup	Service Manager 妥当性検証を実行します。妥当性検証は、妥

表示アクション	説明
	当性テーブルのエントリから、ある特定の値が定義されたフィールドに関して有効であることを確認します。妥当性検証を使用することで、システムはユーザが入力した値が検証ルールに準拠することを確認します。
expandarray	Expand array は、行の追加や削除などの配列を編集するための追加機能を提供します。
reset	現在のファイルをリセットします。このアクションは、データベース内にあるテーブルの**すべての**レコードを削除します。
regen	現在のファイルのインデックスを再生成します。このアクションは、IR キーのあるテーブルにのみ適用されます。IR 再生成のみ実行します。
export/unload	レコードなしの export/unload プロセスを起動します。データのない空の DBDICT 定義のみをアンロードする場合に使用します。
views	選択すると、現在のレコードを表示するための代替フォームのリストが表示されます。
findrevision	このオブジェクトのレコードのリビジョンを表示します。
initrevision	このオブジェクトのレコードのリビジョンを作成します。
newsite	
newview	
newTable	
addFilter	顧客が標準検索画面で詳細フィルタ検索を実行するとトリガされます。
editFilter	顧客が標準検索画面で詳細フィルタ検索を実行するとトリガされます。
addCompound	顧客が標準検索画面で詳細フィルタ検索を実行するとトリガされます。
removeSelection	顧客が標準検索画面で詳細フィルタ検索を実行するとトリガされます。

se.view.engine の基本関数

表示アクション	説明
save	フォーマットコントロールで更新を実行し、更新が有効であれば

表示アクション	説明
	データベース内のレコードを更新します。
add	フォーマットコントロールで追加を実行し、新規レコードが有効であればデータベースに新規レコードを追加します。
ok	レコードが変更されている場合、保存を実行します。それ以外の場合は終了します。
reselect	データベースから現在のレコードを再度選択します(変更された場合)。
fill	ターゲットレコードにリンクされたレコードからの情報をフィルします。ターゲットフィールドにソースレコードからの情報を入力します。
find	フィールドにリンクされているレコードからの詳細情報を表示します。
next	リスト内の次のレコードに移動します(変更の確認後)。
previous	リスト内の前のレコードに移動します(変更の確認後)。
back	リスト、または検索フォームに戻ります。
menu	コールしたメニューに戻ります。
delete	フォーマットコントロールの削除を実行し、依頼の検証後にデータベースからのレコードの削除を試みます。
views	選択すると、現在のレコードを表示するための代替フォームのリストが表示されます。
print	現在のレコードを印刷します。
printlist*	現在のレコードのリストを印刷します。
validitylookup	Service Manager 妥当性検証を実行します。妥当性検証は、妥当性テーブルのエントリから、ある特定の値が定義されたフィールドに関して有効であることを確認します。妥当性検証を使用することで、システムはユーザが入力した値が検証ルールに準拠することを確認します。
export/unload	標準的な Service Manager エクスポート/アンロード機能(複数レコード)。
massunload*	レコードのリストのアンロード機能を提供します。
massadd*	既存のレコードのセットを正確に複製する新しいレコードのセットを追加します。ただし、新しいレコードのセットは、新しいデータで更新された一意のキー値を持ちます。処理が終了すると、テーブルのレコード数が2倍になります。
massupdate*	レコードのリスト内の指定されたフィールドのセットに同じ更新を行います。
massdelete*	テーブルから指定されたレコードのセットを削除します。

表示アクション	説明
irquery	IR クエリテキスト検索を実行します。
expandarray	Expand array は、行の追加や削除などの配列を編集するための追加機能を提供します。
count*	レコードリストのレコードをカウントし、合計カウントを表示します。
audithistory	標準オーディット履歴ルーチンをコールします。オーディット履歴は、auditdef テーブルで定義されたオーディット対象のフィールドによって決まります。
inbox.save / inbox*	ビューとして現在のクエリを保存します。
approval.log	現在のレコードの承認履歴を表示します。
alert.log	現在のレコードのアラート履歴を表示します。
alerts	現在のレコードに関する現在のアラートおよびスケジュールされたアラートを表示します。
pagelist / listpages	ページリストがレコードの完全な履歴を表示します。ページを有効にすると、更新が実行されるたびに、ページファイルにレコードのコピーが書き込まれます。ページリストは、ページファイルを基にレコードの完全な履歴を表示します。
clocks	このレコードに関連付けられたクロックレコードを表示します。
xmlfill	サービスカタログ内のユーザオプションなどの XML フィールドを処理します。
createTemplate	現在のレコードからテンプレートレコードを作成します。
applyTemplate	現在のレコードに既存のテンプレートを適用します。

注：* これらのコマンドは、レコードリスト機能使用時にのみ適用されます。

se.list.engine の基本関数

表示アクション	説明
exit(または back)	検索 (コール) フォームに戻ります。
inbox.save / inbox	ユーザに現在のクエリを新規ビューとして保存するように求めます。
count	標準のカウント機能を実行し、合計レコードカウントを表示します。
refresh	現在のクエリを使用して、リストをリフレッシュします。
big.green	現在のモジュールを完全に終了します。「大きな緑色の矢印」
print	表示されているレコードのリストを印刷します。
views	選択すると、現在のレコードを表示するための代替フォームのリストが表示されます。
export/unload	標準的な Service Manager エクスポート/アンロード機能 (複数レコード) 。
massadd	既存のレコードのセットを正確に複製する新しいレコードのセットを追加します。ただし、新しいレコードのセットは、新しいデータで更新された一意のキー値を持ちます。処理が終了すると、テーブルのレコード数が 2 倍になります。
massupdate	レコードのリスト内の指定されたフィールドのセットに同じ更新を行います。
massdelete	テーブルから指定されたレコードのセットを削除します。

ローカル変数

ローカル変数は、$L. で始まり、現在実行している RAD アプリケーション内でのみ存続します。サーバは、RAD アプリケーションを終了する時にローカル変数を自動でクリーンアップします。

文書エンジンで使用される標準変数のリストを以下に挙げます。

$L.action - 表示オプションからの表示アクションの値

$L.bg - バックグラウンドフラグ

$L.category - カテゴリレコード (存在する場合)

$L.env - 現在の環境レコード

$L.exit - 内部終了パラメータ

$L.file - 現在のファイル変数

$L.file.save - レコードの基のステータスでのコピー

$L.format - レコードを表示するのに使用するフォーマットの名前

$irspread - IR 検出オプションを決定します。0=浅い検索、2=深い検索、4=完全一致

$L.mode - 表示されているレコードのモード。通常、新規レコードを作成する場合は add、既存レコードを変更する場合は update、または既存レコードの処理を完了する場合は close です

$L.mult - $L.file 変数内に複数のレコードが存在する場合に true になるフラグ

$L.object - オブジェクトレコード

$L.phase - フェーズレコード (存在する場合)

$L.sql または $L.query - 現在のクエリ

$L.sort - 現在のソート順

$L.state - システムが使用しているステータスレコード (レコードのステータス) 。

表示モード (単一レコードを表示している場合) で利用できる変数。

$L.fc - 詳細 FormatControl レコードのコピー

$L.fc.master - マスター FormatControl レコードのコピー

第 6 章

トラブルシューティングの概要

文書エンジンのトラブルシューティングを正しく行うためには、以下の情報を収集する必要があります。

- どのデータベースディクショナリとオブジェクトが使用されているのか
- レコードはどのステータスにあるのか
- どのプロセスがコールされたか
- 問題を再現するための手順(STR)

文書エンジンを通るアプリケーションパスの調査

文書エンジンのトラブルシューティングでは、Service Manager アプリケーションのトラブルシューティングの場合と同様、Service Manager `sm.ini` ファイルに「**RTM:3**」および「**debugdbquery:999**」と入力して、新規クライアント接続を開始します。このユーザプロセスがデバッグする文書エンジンプロセスを最初に呼び出す場合を除いて、このトレースを実行しても sm.log ファイルにステータスやプロセスレコードの選択が示されないことがありますが、呼び出されたプロセスの特定に役立つ情報が得られます。

使用されるデータベースディクショナリ、またはオブジェクトの特定

使用されるデータベースディクショナリ、またはオブジェクトを特定するには、ログファイルを検索します。

ログのサンプル:

```
1320 07/18/2006 11:00:36 RADTRACE 20 [ 1] se.get.object get.object select CPU( 0 1411 )
```

1320 07/18/2006 11:00:36 (0x0129AC08) DBACCESS - Cache Find against file Object found 1 record, query:file.name="pcsoftware"

```
1320 07/18/2006 11:00:36 RADTRACE 20 [ 1] se.get.object set.access process CPU( 0 1411 )
```

レコードのステータスの特定

次に調べる疑問点は、レコードがどのステータスにあるのかということです。この情報を得るには、トレース sm.log で以下の文字列を検索します。

ログのサンプル:

```
1320 07/18/2006 11:00:48 RADTRACE 10 [ 1] se.get.state select.state select CPU( 0 1491 )
```

1320 07/18/2006 11:00:48 (0x01292FB0) DBACCESS - Cache Find against file States found 1 record, query:state="pcs.list"

```
1320 07/18/2006 11:00:48 RADTRACE 10 [ 1] se.get.state exit.normal process CPU( 0 1491 )
```

プロセスの名前の特定

以下の例にあるようなトレースを含む `sm.log` を検索することで、プロセスの名前も同様に見つけられます。

ログのサンプル:

```
1320 07/18/2006 11:00:50 RADTRACE 20 [ 1] se.call.process select.process select CPU( 0
1542 )
```

1320 07/18/2006 11:00:50 (0x00B56810) DBACCESS - Cache Find against file Process found 1 record, query:process="upgrade.pcs"

```
1320 07/18/2006 11:00:50 RADTRACE 20 [ 1] se.call.process run.pre.exp process CPU( 0 1542
)
```

アプリケーションエラーの調査

プロセスは複数の RAD アプリケーションをコールし、複数の式を実行し、場合によってその後、さらにプロセスを呼び出します。誤った構文や誤った論理が原因でアプリケーションや式がエラー終了した場合、以下の情報が sm.log ファイルに書き込まれます。

ログのサンプル:

```
Process panel run.pre.exp in RAD se.call.process encountered error in line 1
(se.call.process,run.pre.exp)
Cannot evaluate expression (se.call.process,run.pre.exp)
Cannot evaluate expression (se.call.process,run.pre.exp)
Bad arg(2) oper = (se.call.process,run.pre.exp)
Cannot evaluate expression (se.call.process,run.pre.exp)
Cannot evaluate expression (se.call.process,run.pre.exp)
Bad arg(2) oper = (se.call.process,run.pre.exp)
Cannot evaluate expression (se.call.process,run.pre.exp)
Bad arg(2) oper nullsub (se.call.process,run.pre.exp)
Cannot evaluate expression (se.call.process,run.pre.exp)
Bad arg(2) oper in (se.call.process,run.pre.exp)
Cannot evaluate expression (se.call.process,run.pre.exp)
Unrecoverable error in application:se.search.objects on panel call.list.engine
Unrecoverable error in application:se.list.engine on panel call.process.1
Unrecoverable error in application:se.call.process on panel run.pre.exp
```

この例では(設定済みではない)、se.call.process、run.pre.exp 内で、つまりプロセスの初期式を評価中にエラーが発生しました。どのプロセスが問題を引き起こしているのかを突き止めるには、上記の手順に従って、以下の行のプロセスに着目します。

```
DBACCESS - Cache Find against file Process found 1 record, query:process="upgrade.pcs"
```

名前が **upgrade.pcs** のプロセスレコードに移動し、[初期式]タブ上のステートメントを確認します。この例では、式に「nullsub」という語が含まれています。例えば、このテストでの問題の式は以下のようになります。

```
$L.icount=nullsub($L.icount, anynumberIwant)
```

変数 anynumberIwant は有効なフィールド、リテラル、または変数ではないため、この問題を避けるために変更する必要があります。

変数の値や式の結果の印刷

文書エンジンでは通常、ワークフローを通るパスは、フィールドや変数に割り当てられた値によって決定されます。ワークフローに影響を与えるフィールドや変数に割り当てられた値を判断するには、RAD式で `JavaScript print()` 関数、または `log rtecall ($L.void=rtecall(log, $L.rc, message)` を使用します。メッセージは以下のように結合した文字列にすることができます。

```
$L.message="The value of $L.test is " + $L.test
```

ここで `$L.test` は文字列の値が割り当てられた変数です。

第 7 章

作業指示の例の概要

作業指示とは? 作業指示とは、単一の技術者に割り当てられる、インシデントの解決に必要なアクティビティ、またはその他のモジュール(作業指示システムが拡張され、変更、問題や既知のエラーを含む場合) を識別するための特定のタスクのことです。以下の例は Service Manager バージョン 7.11 に基づいており、ユーザがインシデント管理の作業指示の作成、更新、およびクローズを行えるように記述されています。この例は、別の Service Manager アプリケーション用に簡単に変更できます。

この例では、作業指示システムを作成して、文書エンジンの使用方法を説明します。この作業指示システムにより、ユーザはインシデントの作業指示を作成して、これらの作業指示のステータスを表示できます。作業指示システムにより、ユーザはインシデント管理から作業指示を表示して更新することもできます。すべての作業指示はシステム内の特定インシデントに関連付けられ、インシデントに関するすべての作業指示がクローズされるまでそのインシデントをクローズすることはできません。

この例は、システムをカスタマイズするユーザを対象としています。ユーザは、以下のカスタマイズ機能に関する十分な知識が必要となります。

- 新規テーブルを作成するためのデータベースディクショナリ
- 設定済みフォームを変更して、新規フォームを作成するためのフォームデザイナ
- ウィザード作成ツール

作業指示の例では、以下の手順を段階的に説明します。

- データベースディクショナリを使用して、新規データベースディクショナリ(dbdict) を作成します。
- テーブルのキーフィールドを指定します。
- フォームデザイナを使用して、EXWorkOrder テーブル向け EXWorkOrder フォームを作成します。
- 番号管理ファイルを作成します。
- フォームを変更して、ドロップダウンリストを追加します。
- EXWorkOrder フォームのリンクを作成します。
- 作業指示に関する情報の収集に使用するウィザード用フォームを作成します。
- インシデントのクローズ、およびインシデントの更新用フォームを変更します。
- インシデントに作業指示をリンクするためのエイリアスを作成します。
- ステータス定義を作成します。
- オープン、クローズ、および表示のための表示アプリケーション定義を作成します。
- オープン、クローズ、および表示のための追加、キャンセル、フィル、および検索の表示オプション定義を作成します。
- プロセス定義レコード im.set.close を変更します。
- 作業指示の例をテストします。

テーブルの作成

データベースディクショナリユーティリティを使用して、新規テーブルを作成します。この例では、EXWorkOrder というテーブルを作成します。このテーブルを作成する前に、必要なフィールドとそのフィールドの属性を把握しておく必要があります。この例では、テーブルはインシデントに関連付けられた作業指示のデータを格納します。

テーブルを作成するには:

1. システムナビゲータから、**[カスタマイズ]**>**[データベースディクショナリ]**をクリックします。
   [データベースディクショナリ]フォームがオープンします。
2. [ファイル名]に「EXWorkOrder」と入力します。
3. **[新規]**をクリックします。
4. [フィールド]タブで、**[新規フィールド/キー]**をクリックしてから、以下の情報を入力します。

フィールド	説明
ID	番号ファイルからフィルされます。 タイプ: 文字
RelatedID	関連するレコードの固有 ID によって入力されます(例 : インシデント番号)。 タイプ: 文字
status	作業指示のステータスを格納するのに使用されます。 タイプ: 文字
initiator	作業指示を開いたオペレータ。 タイプ: 文字
assignee.name	作業指示に割り当てられたオペレータ。 タイプ: 文字
description	作業指示の作業内容の説明に使用します。 タイプ: 文字の配列
category	関連するレコードのカテゴリから入力されます。 タイプ: 文字
RelatedCIs	作業指示に関連付けられた構成アイテム(CI) のリストに使用されます。 タイプ: 文字の配列
impact	タイプ: 文字
urgency	タイプ: 文字
priority	タイプ: 文字
closure.code	タイプ: 文字

deadline	作業指示がそれまでに完了する必要がある期日。オープン時に入力されます。 タイプ: 日付/時刻
est.finish	担当者による完了予定日時。 タイプ: 日付/時刻
update.action	. タイプ: 文字の配列
closure.comments	. タイプ: 文字の配列

5. テーブルにフィールドを追加した後、テーブルにキーを追加する必要があります。**EXWorkOrder** テーブルにキーを追加し終えるまでは、データベースディクショナリユーティリティを終了しないでください。詳細については、「テーブルへのキーフィールドの追加」を参照してください。

テーブルへのキーフィールドの追加

データベースディクショナリユーティリティを使用して、新規テーブルにキーを追加します。テーブル EXWorkOrder を作成したら、このテーブルにキーを追加する必要があります。キーはインデックス型の検索を可能にし、データの整合性を保証します。

テーブルにキーを追加するには:

1. データベースディクショナリで**[キー]**タブをクリックします。
2. 新規キーフィールドに利用できる最初のエントリを選択します。
3. **[新規フィールド/キー]**をクリックします。
4. 作成、または編集する各キーについて、以下の情報を入力します。

フィールド	説明
ID	タイプ: 固有
RelatedID	タイプ: null なし
RelatedCIs	タイプ: null と重複
assignee.name	タイプ: null と重複

5. **[OK]**をクリックします。

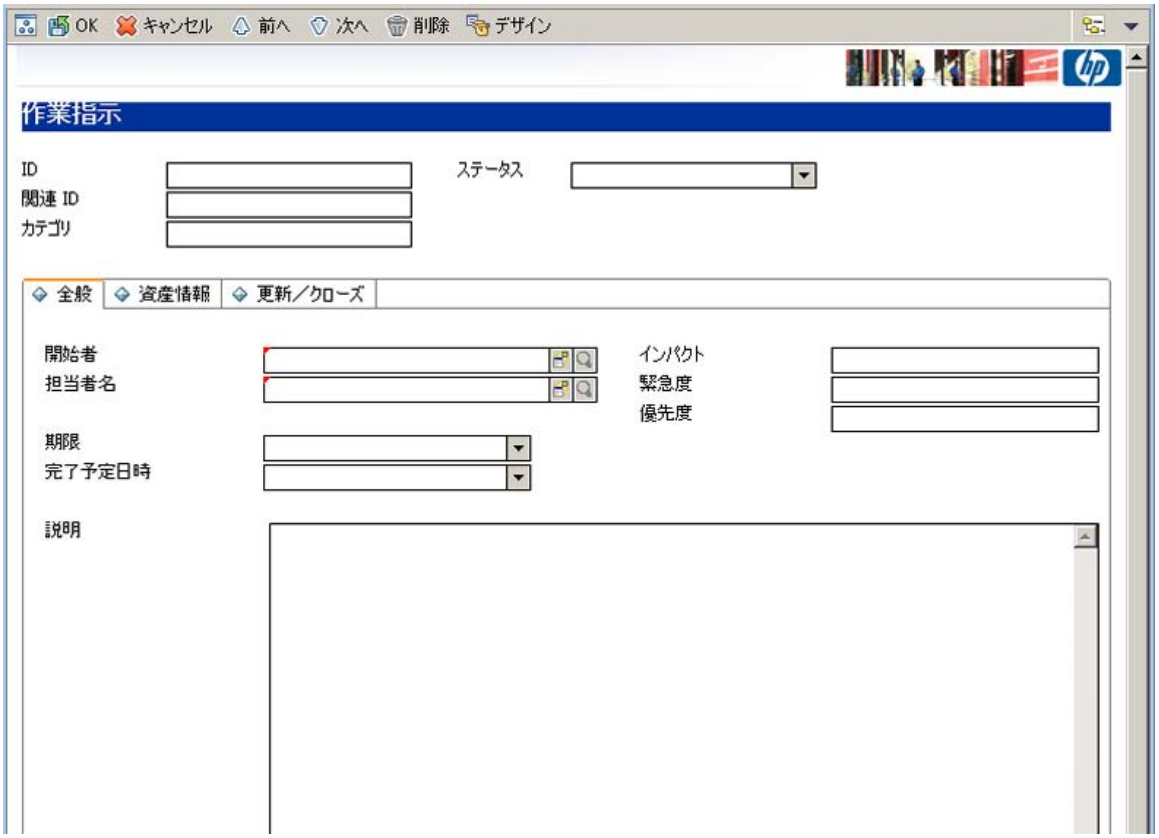
フォームの作成

フォームデザイナに移動し、EXWorkOrder テーブル用フォームを作成します。フォーム作成用のウィザードを使用して、[単一レコードの詳細]を選択して続行します。すべてのフィールドを備え、必要に応じて変更可能な基本フォームが作成されます。以下は EXWorkOrder フォームの一例です。

フォームを作成するには:

1. システムナビゲータから、**[カスタマイズ]**>**[フォームデザイナ]**をクリックします。
   [フォームデザイナ]フォームがオープンします。
2. [フォーム名]に「EXWorkOrder」と入 力します。
3. **[新 規]**をクリックします。
4. **[はい]**をクリックして、フォームウィザードを使 用します。
5. 作 成 するフォームの対 象 となるテーブル名 として、「EXWorkOrder」と入 力します。
6. 作 成 するフォームのタイプとして、**[単 一 レコードの詳 細]**を選 択します。
7. **[OK]**をクリックします。
8. **[続 行]**をクリックして、フィールドのデフォルト値 を受 け入 れ、フォームに含 めます。
9. フォームデザイナツールを使 用して、フォームレイアウトを変 更します。以 下 に **EXWorkOrder** フォームの一 例を挙 げます。
10. 以 下 はステータスの値 です。
    - 新 規 (デフォルトのステータス)
    - オープン
    - 準 備 完 了
    - クローズ済 み
11. 以 下 はクローズコードの値 です。
    - 実 施 済 み
    - キャンセル済 み
    - ロールバック

以 下 は EXWorkOrder フォームの例 です。



フォームのコピーの作成

フォームデザイナで、sc.manage.problem のコピーを作成して sc.manage.WorkOrder を作成します。EXWorkOrder テーブルのフィールドを使用するようにテーブルの列の入力を変更します。

フォームのコピーから新規フォームを作成するには:

1. システムナビゲータから、**[カスタマイズ]**>**[フォームデザイナ]**をクリックします。
   [フォームデザイナ]フォームがオープンします。
2. [フォーム]に「`sc.manage.problem`」と入力します。
3. **[検索]**をクリックします。
4. [sc.manage.problem.g]を選択します。
5. 詳細オプションメニューで**[コピー/名前変更]**をクリックします。
6. [新規の名前]に「`sc.manage.WorkOrder`」と入力します。
7. **[OK]**をクリックします。
8. フォームデザイナで、EXWorkOrder テーブルを使用して列の入力フィールドを更新します。
9. - インシデント ID - ID
   - カテゴリ - category
   - 関連 ID - RelatedID

- ステータス - status
- 担当者 - assignee.name
- 説明 - description,1
- 優先度 - priority
- インパクト - impact
- 緊急度 - urgency

10. **[OK]**をクリックします。

作業指示フォームのリンクの作成

リレーショナルデータベースの特長の 1 つは、冗長な情報が排除されていることです。このために、特定の主題に関する情報は 1 つの場所またはテーブルに記録され、他の主題にリンクされます。リンクはデータとリンク定義の組み合わせであり、リンク定義とはリンクされた情報に対する関係を含む条件の集合です。

作業指示フォームのリンクを作成するには:

1. システムナビゲータから、**[カスタマイズ]**>**[カスタマイズツール]**>**[リンク]**をクリックします。
   [リンクファイル]フォームがオープンします。
2. [名前]に「`EXWorkOrder`」と入力します。
3. [説明]に説明を追加します。
4. **[新規]**をクリックします。
5. 以下の情報を入力します。

ソースフィールド名	ターゲットファイル名	ターゲットフィールド名
initiator	operator	name
assignee.name	operator	name
RelatedCIs	device	logical.name

6. initiator の行を選択(強調表示)して、**[initiator]**を右クリックし、**[行の選択]**を選択します。
7. link.structure.g フォームに、initiator に関する以下の情報を入力します。次に **assignee.name** と **RelatedCIs** にも、それぞれのフィールドに関する以下の情報を使用して手順 6 を繰り返します。

ソースフィールド (フィル先/通知元)	ターゲットフィールド名 (フィル元/通知先)
initiator	name
assignee.name	name

RelatedCIs	logical.name

8. **[保存]**をクリックします。
9. **[OK]**をクリックします。

番号管理ファイルの作成

EXWorkOrder テーブル内のレコードのシーケンス番号を生成するための番号管理ファイルを作成します。

番号管理ファイルを作成するには:

1. システムナビゲータから、**[カスタマイズ]**>**[カスタマイズツール]**>**[シーケンス番号]**をクリックします。[番号管理ファイル]フォームがオープンします。
2. [クラス]に「EXWorkOrder」と入力します。
3. [最後の番号]に「1」と入力します。
4. [説明]に「WorkOrder の番号」と入力します。
5. [長さ]に「5」と入力します。
6. [プリフィックス]に「WO」と入力します。
7. **[追加]**をクリックします。

オブジェクト定義の作成

このオブジェクトの目的は EXWorkOrder オブジェクトの特性と動作を定義することにあります。この EXWorkOrder オブジェクトは、作業指示レコードに含める必要のあるデータ、システムによる作業指示の処理方法を決定します。

注:オブジェクト定義のフィールドの説明については、「[オブジェクト定義]フォームとフィールド」(11 ページ)を参照してください。

オブジェクト定義を作成するには:

1. システムナビゲータから、**[カスタマイズ]**>**[文書エンジン]**>**[オブジェクト]**をクリックします。[オブジェクト定義]フォームがオープンします。
2. [ファイル名]に「*EXWorkOrder*」と入力します。
3. **[追加]**をクリックしてオブジェクトレコードを作成します。
4. [オブジェクト情報]タブに以下の情報を入力します。

フィールド	値
固有キー	自動的に入力されます(ID)
共通名	自動的に入力されます(EXWorkOrder)
説明フィールド	いずれかのモジュールに関連付けること

	が可能な作業指示を保持するテーブル
プロファイルアプリケーション	db.environment
プロファイル変数	$L.env
番号レコード名	EXWorkOrder
カテゴリテーブル名	category
マスタフォーマットコントロール	EXWorkOrder
ステータスフィールド	status
担当者フィールド	assignee.name
オープンステータス	EXWorkOrder.open
クローズステータス	EXWorkOrder.close
デフォルトステータス	EXWorkOrder.view
検索ステータス	EXWorkOrder.search

5. **[保存]**をクリックします。
6. **[キューの管理]**タブを選択します。
7. [キューの管理]タブに以下の情報を入力します。

フィールド	値
管理条件	true
管理表示フォーマット	sc.manage.WorkOrder
管理デフォルトクエリ	assignee.name=operator()
追加を許可する条件	false
デフォルトクエリの説明	自分の作業指示

8. **[保存]**をクリックします。
9. **[OK]**をクリックします。

初期化プロセス定義の作成

作業指示の例のプロセス定義は、ユーザが作業指示レコードを開くときに使用する初期式とRADアプリケーションを指定します。

初期化のためのプロセス定義を作成するには:

1. システムナビゲータから、**[カスタマイズ]**>**[文書エンジン]**>**[プロセス]**をクリックします。
   [プロセス定義]フォームがオープンします。
2. [プロセス名]に「`EXWorkOrder.open.initial`」と入力します。
3. **[追加]**をクリックします。
4. [初期式]タブに「`$L.format="EXWorkOrder" `」と入力します。
5. **[保存]**をクリックします。
6. [RAD]タブに以下の情報を入力します。

フィールド	値
RAD 呼び出しの前に評価される式	$L.number.record="EXWorkOrder";$L.number.type="string"
RAD アプリケーション	getnumb
パラメータ名	**パラメータ値**
name	$L.number.record
index	ID in $L.file
text	$L.number.type

7. **[保存]**をクリックします。

表示アプリケーション定義の作成

作業指示フォームのオープン、クローズ、および表示画面のための表示アプリケーション定義を作成します。ここから、ユーザは作業指示レコードのオープン、クローズ、および表示を行えます。

オープン、クローズ、および表示のための画面アプリケーション定義を作成するには:

1. システムナビゲータから、**[カスタマイズ]**>**[カスタマイズツール]**>**[表示アプリケーション]**をクリックします。
   [表示アプリケーション定義]フォームがオープンします。
2. [画面 ID]に「`EXWorkOrder.open`」と入力します。
3. 以下の情報を入力します。

フィールド	値
画面 ID	EXWorkOrder.open
タイトル	新規作業指示のオープン
フォーマット	$L.format

I/O (RIO の場合)	true
オプション 0 のアクション:	画面の再描画
言語	ENG

4. **[追加]**をクリックし、**[OK]**をクリックします。

この例では、クローズ画面の表示アプリケーション定義も作成する必要があります。

1. システムナビゲータから、**[カスタマイズ]**>**[カスタマイズツール]**>**[表示アプリケーション]**をクリックします。
[表示アプリケーション定義]フォームがオープンします。
2. 以下の情報を使用します。

フィールド	値
画面 ID	EXWorkOrder.close
タイトル	作業指示のクローズ
フォーマット	$L.format
I/O (RIO の場合)	true
オプション 0 のアクション:	画面の再描画
言語	ENG

3. **[追加]**をクリックし、**[OK]**をクリックします。

この例では、表示画面の表示アプリケーション定義も作成する必要があります。

1. システムナビゲータから、**[カスタマイズ]**>**[カスタマイズツール]**>**[表示アプリケーション]**をクリックします。
[表示アプリケーション定義]フォームがオープンします。
2. 以下の情報を使用します。

フィールド	値
画面 ID	EXWorkOrder.view
タイトル	新規作業指示の表示
フォーマット	$L.format
I/O (RIO の場合)	true
オプション 0 のアクション:	画面の再描画
言語	ENG

3. **[追加]**をクリックし、**[OK]**をクリックします。

表示オプション定義の作成

この手順では、以下の WorkOrder 定義に関する追加、キャンセル、フィル、および検索の表示オプション定義の作成を段階的に説明します。

- オープン
- クローズ
- 表示

表示オプション定義の作成では、各 WorkOrder 画面定義(オープン、クローズ、表示)について同じ手順を 4 回繰り返します。ただし、毎回、[表示オプション定義]フォームのフィールドに別の値を入力します。手順の下にある表は、表示オプションの設定に必要な値を示しています。

表示オプションの詳細については、HP Service Manager オンラインヘルプサーバの表示アプリケーションに関するオンラインヘルプトピックを参照してください。

オープン、クローズ、および表示のための表示アプリケーション定義を作成した後、オープン、クローズ、および表示画面のための表示オプションを作成する必要があります。

表示オプションの定義を作成するには:

1. システムナビゲータから、**[カスタマイズ]**>**[カスタマイズツール]**>**[表示オプション]**をクリックします。[表示アプリケーション定義]フォームがオープンします。
2. [画面 ID]に「`EXWorkOrder.open`」と入力します。
3. 以下の情報を入力します。

フィールド	値
固有 ID	自動生成(EXWorkOrder.open_add)
アクション	add
GUI オプション	4
テキストオプション	4
バンク	1
条件	true
デフォルトのラベル	追加

4. **[追加]**をクリックします。
5. **[OK]**をクリックします。
6. 表の値を使用して、以下の表示オプション定義のそれぞれについて手順 1 から 6 を繰り返します。

注: [アクション]フィールドに必要な値がドロップダウンリストにない場合でも、フィールドに適用可能な値を入力できます。

オープン - キャンセル

フィールド	値
画面 ID	EXWorkOrder.open
固有 ID	システムによる生成 (EXWorkOrder.open_cancel)
アクション	back
GUI オプション	3
テキストオプション	3
バンク	1
条件	true
デフォルトのラベル	キャンセル

オープン - フィル

フィールド	値
画面 ID	EXWorkOrder.open
固有 ID	システムによる生成 (EXWorkOrder.open_fill)
アクション	fill
GUI オプション	9
テキストオプション	9
バンク	1
条件	true
デフォルトのラベル	フィル

オープン - 参照

フィールド	値
画面 ID	EXWorkOrder.open
固有 ID	システムによる生成 (EXWorkOrder.open_find)
アクション	find
GUI オプション	8

テキストオプション	8
バンク	1
条件	true
デフォルトのラベル	参照

クローズ - キャンセル

フィールド	値
画面 ID	EXWorkOrder.close
固有 ID	システムによる生成 (EXWorkOrder.close_ cancel)
アクション	back
GUI オプション	3
テキストオプション	3
バンク	1
条件	true
デフォルトのラベル	キャンセル

クローズ - クローズ

フィールド	値
画面 ID	EXWorkOrder.close
固有 ID	システムによる生成 (EXWorkOrder.close_ close)
アクション	close
GUI オプション	5
テキストオプション	5
バンク	1
条件	true
デフォルトのラベル	クローズ

クローズ - フィル

フィールド	値
画面 ID	EXWorkOrder.close
固有 ID	システムによる生成 (EXWorkOrder.close_fill)
アクション	fill
GUI オプション	9
テキストオプション	9
バンク	1
条件	true
デフォルトのラベル	フィル

クローズ - 参照

フィールド	値
画面 ID	EXWorkOrder.close
固有 ID	システムによる生成 (EXWorkOrder.close_find)
アクション	find
GUI オプション	8
テキストオプション	8
バンク	1
条件	true
デフォルトのラベル	参照

表示 - キャンセル

フィールド	値
画面 ID	EXWorkOrder.view
固有 ID	システムによる生成 (EXWorkOrder.view_ cancel)
アクション	back
GUI オプション	3
テキストオプション	3

バンク	1
条件	true
デフォルトのラベル	キャンセル

表示 - フィル

フィールド	値
画面 ID	EXWorkOrder.view
固有 ID	システムによる生成 (EXWorkOrder.view_fill)
アクション	fill
GUI オプション	9
テキストオプション	9
バンク	1
条件	true
デフォルトのラベル	フィル

表示 - 参照

フィールド	値
画面 ID	EXWorkOrder.view
固有 ID	システムによる生成 (EXWorkOrder.view_find)
アクション	find
GUI オプション	8
テキストオプション	8
バンク	1
条件	true
デフォルトのラベル	参照

表示 - 保存

フィールド	値

画面 ID	EXWorkOrder.view
固有 ID	システムによる生成 (EXWorkOrder.view_save)
アクション	save
GUI オプション	4
テキストオプション	4
バンク	1
条件	true
デフォルトのラベル	保存

ステータス定義の作成

作業指示の例のステータス定義は、ユーザが作業指示レコードをオープン、クローズ、または表示するときに、どのプロセスを使用してどのアクションが許可されるのかを指定します。

注 : ステータス定義のフィールドの説明については、「ステータスの定義のフィールドの説明」(25 ページ)を参照してください。

オープンのステータス定義を作成するには:

1. システムナビゲータから、**[カスタマイズ]** > **[文書エンジン]** > **[ステータス]** をクリックします。
   [ステータスの定義] フォームがオープンします。
2. 以下の情報を入力します。

フィールド	値
ステータス	EXWorkOrder.open
表示アプリケーション	EXWorkOrder.open
初期化プロセス	EXWorkOrder.open.initial
フォーマット	$L.format
入力条件	true

3. **[追加]** をクリックします。

クローズのステータス定義を作成するには:

1. システムナビゲータから、**[カスタマイズ]** > **[文書エンジン]** > **[ステータス]** をクリックします。
   [ステータスの定義] フォームがオープンします。
2. 以下の情報を入力します。

フィールド	値
ステータス	EXWorkOrder.close
表示アプリケーション	EXWorkOrder.close
初期化プロセス	EXWorkOrder.close.initial
フォーマット	$L.format
入力条件	true

3. **[追加]**をクリックします。

表示のステータス定義を作成するには:

1. システムナビゲータから、**[カスタマイズ]**>**[文書エンジン]**>**[ステータス]**をクリックします。
[ステータスの定義]フォームがオープンします。
2. 以下の情報を入力します。

フィールド	値
ステータス	EXWorkOrder.view
表示アプリケーション	EXWorkOrder.view
初期化プロセス	EXWorkOrder.view.initial
フォーマット	$L.format
入力条件	true

3. **[追加]**をクリックします。

検索のステータス定義を作成するには:

1. システムナビゲータから、**[カスタマイズ]**>**[文書エンジン]**>**[ステータス]**をクリックします。
[ステータスの定義]フォームがオープンします。
2. 以下の情報を入力します。

フィールド	値
ステータス	EXWorkOrder.search
表示アプリケーション	db.search
初期化プロセス	EXWorkOrder.view.search

フィールド	値
フォーマット	$L.format
入力条件	true

3. **[OK]**をクリックします。

[作業指示のクローズ]ボタンの追加

ユーザがタスクを完了したら、作業指示をクローズする必要があります。この手順では、EXWorkOrder フォームにクローズボタンを追加して、作業指示のステータスをクローズに更新する方法を説明します。この手順では、表示アプリケーションオプションの定義、EXWorkOrder.view のステータス定義、および EXWorkOrder.close のプロセス定義を追加します。

注:この手順では、このアクティビティの検証を含める方法については説明しません。フォーマットコントロールを使用して検証を追加します。

EXWorkOrder.view の表示アプリケーションオプションの定義を作成するには:

1. システムナビゲータから、**[カスタマイズ]**>**[カスタマイズツール]**>**[表示オプション]**をクリックします。[表示アプリケーション定義]フォームがオープンします。
2. [画面 ID]に「EX`WorkOrder.view`」と入力します。
3. 以下の情報を入力します。

フィールド	値
固有 ID	自動生成(EXWorkOrder.view_close)
アクション	close
GUI オプション	5
テキストオプション	5
バンク	1
条件	true
デフォルトのラベル	クローズ

4. **[追加]**をクリックします。
5. **[OK]**をクリックします。

EXWorkOrder.view ステータス定義を作成するには:

1. システムナビゲータから、**[カスタマイズ]**>**[文書エンジン]**>**[ステータス]**をクリックします。[ステータスの定義]フォームがオープンします。
2. [ステータス]フィールドに**「EXWorkOrder.view」**と入力し、**[検索]**をクリックします。EXWorkOrder.view ステータス定義がオープンします。
3. 以下の項目を追加して、[ステータスの定義]フォームを更新します。

フィールド	値
表示アクション	close
プロセス名	EXWorkOrder.close
条件	true

4. **[保存]** をクリックして、**[OK]** をクリックします。

EXWorkOrder.close のプロセス定義レコードを追加するには:

1. システムナビゲータから、**[カスタマイズ]**>**[文書エンジン]**>**[プロセス]**をクリックします。
[プロセス定義]フォームがオープンします。
2. [プロセス名]に「`EXWorkOrder.close`」と入力します。
3. **[追加]**をクリックします。
4. [初期式]タブに、以下の式を入力します。
   - status in $L.file="Closed"
   - $L.mode="closed"
5. **[保存]**をクリックします。
6. [RAD]タブに以下の情報を入力します。

フィールド	値
RAD 呼び出しの前に評価される式	$L.EXaction="update"
RAD アプリケーション	se.base.method
条件	true
パラメータ名	**パラメータ値**
file	$L.file
prompt	$L.EXaction
second file	$L.file.save
record	$L.fc
second.record	$L.object
boolean1	false

7. **[保存]**をクリックして、**[OK]**をクリックします。

作業指示のウィザードの作成

この例では、ウィザードを使用して作業指示を作成します。ウィザードを使用すると、ユーザが作業指示を簡単に開くことができます。この機能は最初からシステムで利用できるようになっています。

ウィザード作成ツールを使用してウィザードを作成し、ユーザがインシデント管理モジュールから作業指示を作成できるようにします。ユーザがインシデントから作業指示を作成すると、ウィザードはユーザに情報の入力を求め、また作業指示の一部のフィールドに自動的に値を入力します。

作業指示フォームのウィザードを作成するには:

1. システムナビゲータから、**[カスタマイズ]**>**[ウィザード]**をクリックします。
   [ウィザード情報]フォームがオープンします。
2. [ウィザード名]に「作業指示の作成 - 1」と入力します。
3. **[追加]**をクリックしてウィザードレコードを作成します。
4. [ウィザード情報]タブに以下の情報を入力します。

フィールド	値
概要	新規作業指示を作成します。
ウィンドウタイトル	作業指示の作成
タイトル	作業指示の作成
開始ノード	一連のウィザードレコードがある場合は、これを選択(true に設定)して、このノードが一連のウィザードのうちの最初のウィザードであることを指定します。

5. **[保存]**をクリックします。
6. [ウィザード情報]フォームの[ファイル選択]タブで、**[$L.file の選択基準]**タブを選択します。
7. [$L.file の選択基準]タブに以下の情報を入力します。

フィールド	値
レコードの作成	true
対象タイプ	EXWorkOrder

8. **[保存]**をクリックします。
9. [使用方法]タブに以下の情報を入力します。

フィールド	値
ウィザードの使用方法	[ウィザードの使用方法]セクションの**[ユーザ入力の要求]**をクリックします。
表示するサブフォーマット	「createWO.assigneeAndCIs」と入力します
表示アプリケーション	「wizard.display」と入力します
[終了]オプションを有効にする	選択(true に設定)して、ウィザードフォーム上の[終了]ボタンを有効にします。

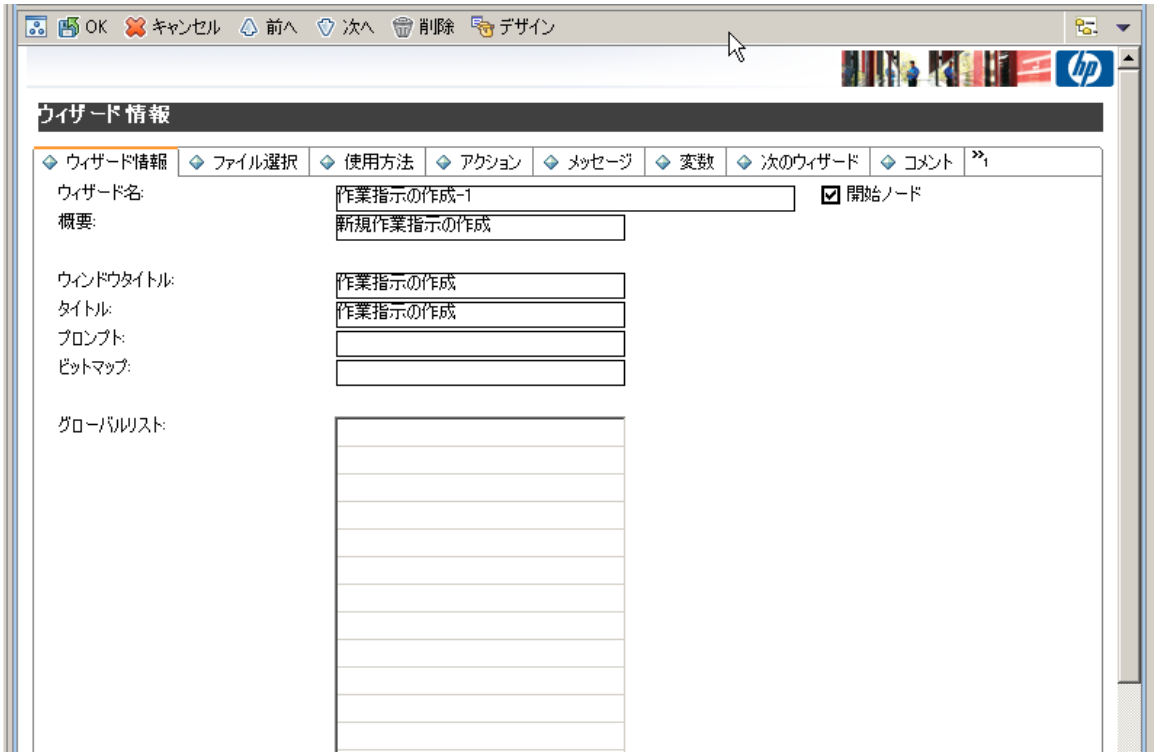
10. **[保存]**をクリックします。
11. [アクション]タブに以下の情報を入力します。

フィールド	値
アクションの実行対象	現在のファイル($L.file)
[実行するアクション]>[式]>	initiator in $L.file=operator() status in $L.file="New" RelatedID in $L.file=number in $relatedRec category in $L.file=category in $relatedRec impact in $L.file=initial.impact in $relatedRec urgency in $Lfile=severity in $relatedRec priority in $L.file=priority.code in $relatedRec
完了時にレコードを表示	**[完了時にレコードを表示]**をクリックします。
モード	[追加]を選択します。

12. **[保存]**をクリックします。
13. [キャンセル式]タブに以下の情報を入力します。

フィールド	値
キャンセル時に実行される式	cleanup($relatedRec)

14. **[保存]**をクリックして、**[OK]**をクリックします。



プロセス定義レコードの追加

プロセス定義レコードは、ユーザのアクションに対するシステムの応答を定義します。プロセス定義は RAD 式、JavaScript、既存の RAD アプリケーションの呼び出しを使用して、現在のレコード(この場合は作業指示レコード)に対してアクションを実行します。

プロセス定義レコードを追加するには:

1. システムナビゲータから、**[カスタマイズ]**>**[文書エンジン]**>**[プロセス]**をクリックします。
   [プロセス定義]フォームがオープンします。
2. [プロセス名]に「**create.WorkOrder**」と入力します。
3. **[追加]**をクリックします。
4. [初期式]タブに、以下の式を入力します。
   - **$L.void=fduplicate($relatedRec, $L.file)**
   - **$relatedCIs={}**
5. **[保存]**をクリックします。
6. [初期 JavaScript]タブに、以下の式を入力します。

```
system.vars.$relatedCIs=system.library.BSGFunctions.getMembers(system.vars.$L-
_file.affected_item, false, 3)
```

注:この式は 1 行で入力する必要があります。

7. **[保存]**をクリックします。
8. [RAD]タブに以下の情報を入力します。

フィールド	値
RAD 呼び出しの前に評価される式	以下の 2 つの式を入力します。 $L.wiz.name="作業指示の作成 - 1" if (not null(logical.name in $L.file)) then ($relatedCIs=insert ($RelatedCIs, 1, 1, logical.name in $L.file)) **注**:この式は 1 行で入力してください。また、式を入力するときに insert という単語の後にスペースを入力しないようにしてください。例えば、上記の式の単語 'insert' の部分は 次のようになります。 insert($RelatedCIs, 1, 1, logical.name in $L.file))
RAD アプリケーション	wizard.run
パラメータ名	name
パラメータ値	$L.wiz.name
パラメータ名	text
パラメータ値	$L.exit
条件	true

9. **[保存]**をクリックします。
10. [最終式]タブに、「`cleanup($relatedRec)`」と入力します。
11. **[保存]**をクリックして、**[OK]**をクリックします。

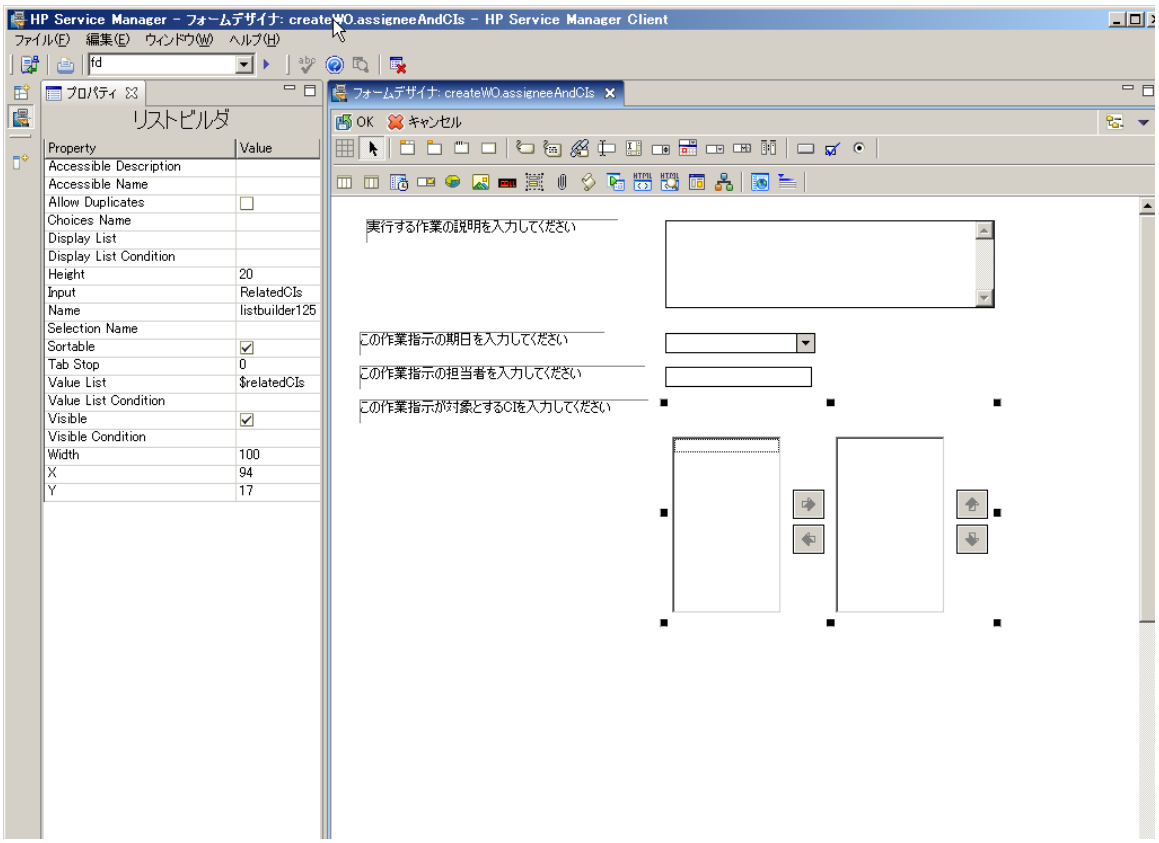
ウィザードの入力フォームの作成

このフォームは、作業指示ウィザードで最初に表示されるフォームです。ユーザは、ウィザードが作業指示レコードを作成するために必要な情報を入力します。フォームデザイナを使用してこのフォームを作成します。この例では、フォーム名を createWO.assigneeAndCIs とします。

ウィザードの入力フォームを作成するには:

1. システムナビゲータから、**[カスタマイズ]**>**[フォームデザイナ]**をクリックします。
   フォームデザイナの検索/新規作成フォームが開きます。
2. [フォーム]にフォーム名として「**createWO.assigneeAndCIs**」と入力します。
3. **[新規]**をクリックします。このフォームにはフォームデザイナウィザードを使用する必要はありません。
4. フォームに以下の入力フィールドを作成します。
   - 説明(実行する作業の説明を入力してください)
   - 期日(この作業指示の期日を入力してください)
   - 担当者(この作業指示の担当者を入力してください)
   - CI(この作業指示が対象とするCIを入力してください)
5. このフォームのプロパティには、以下の項目が含まれます。
   - Input:**RelatedCIs**
   - Value List:$**relatedCIs**
   - Sortable:オン
6. **[保存]**をクリックします。

次の図は、EXWorkorder ウィザード入力フォームの createWO.assigneeAndCIs フォームの例を示します。



インシデントのクローズ、およびインシデントの更新用フォームを変更

この作業指示の例では、インシデントのクローズ、およびインシデントの更新フォームを変更して、インシデントに割り当てられた作業指示を表示し、インシデントから作業指示をダブルクリックして作業指示を読み取ったり編集できるようにします。フォームデザイナを使用して、*IM.update.incident* と *IM.close.incident* フォームを更新します。これには、IM.update.incident および IM.close.incident フォームの両方にタブを追加し、さらにそのタブ上にフォームを追加する必要があります。

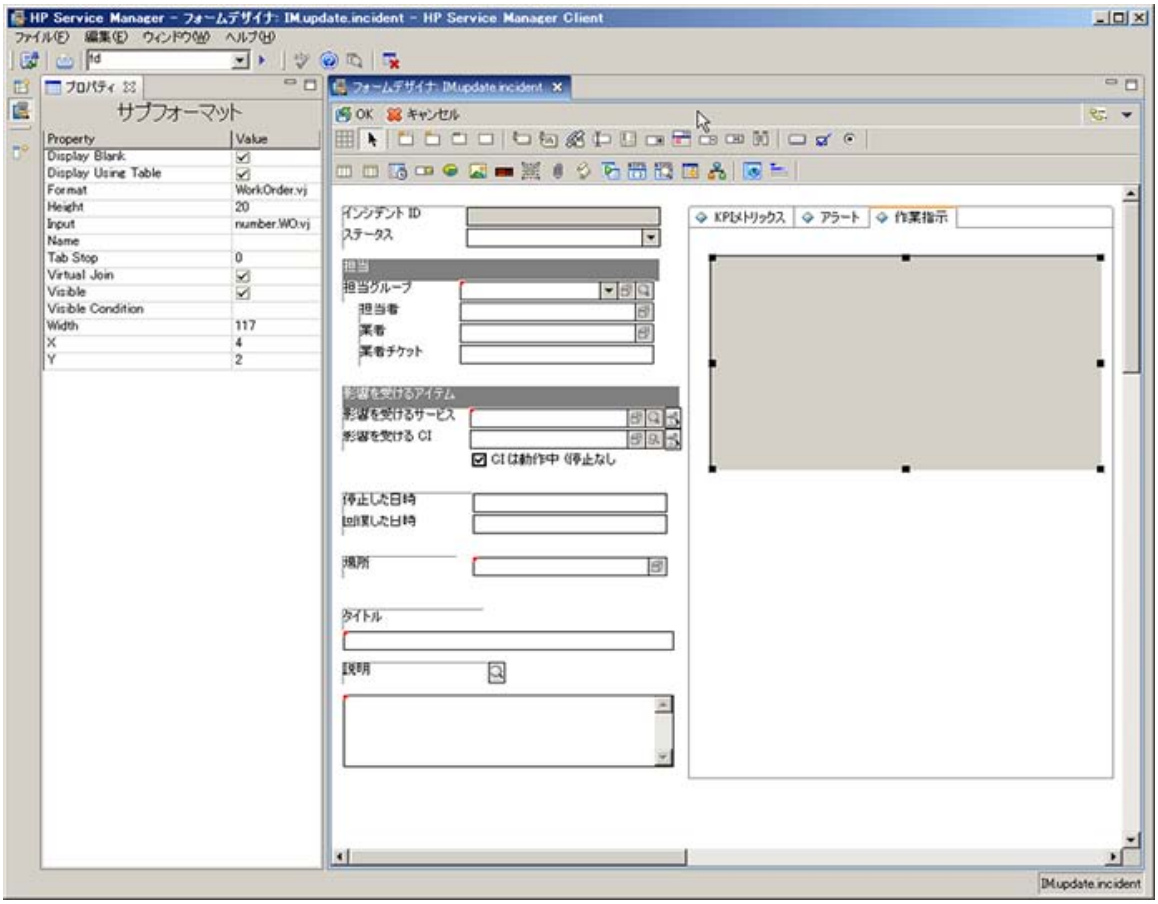
インシデントの更新フォームを変更するには:

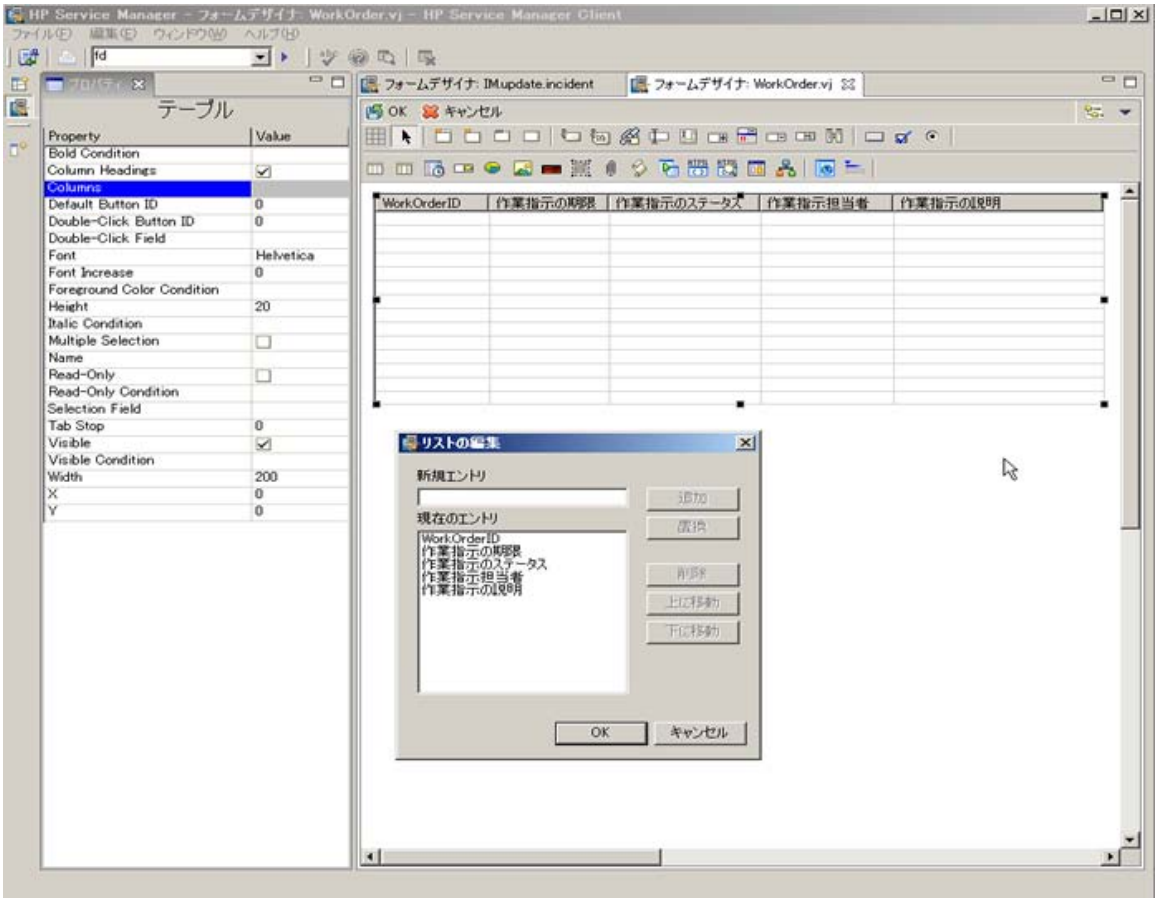
注: 全般的に同じ手順を用いて、インシデントのクローズフォームを更新します。

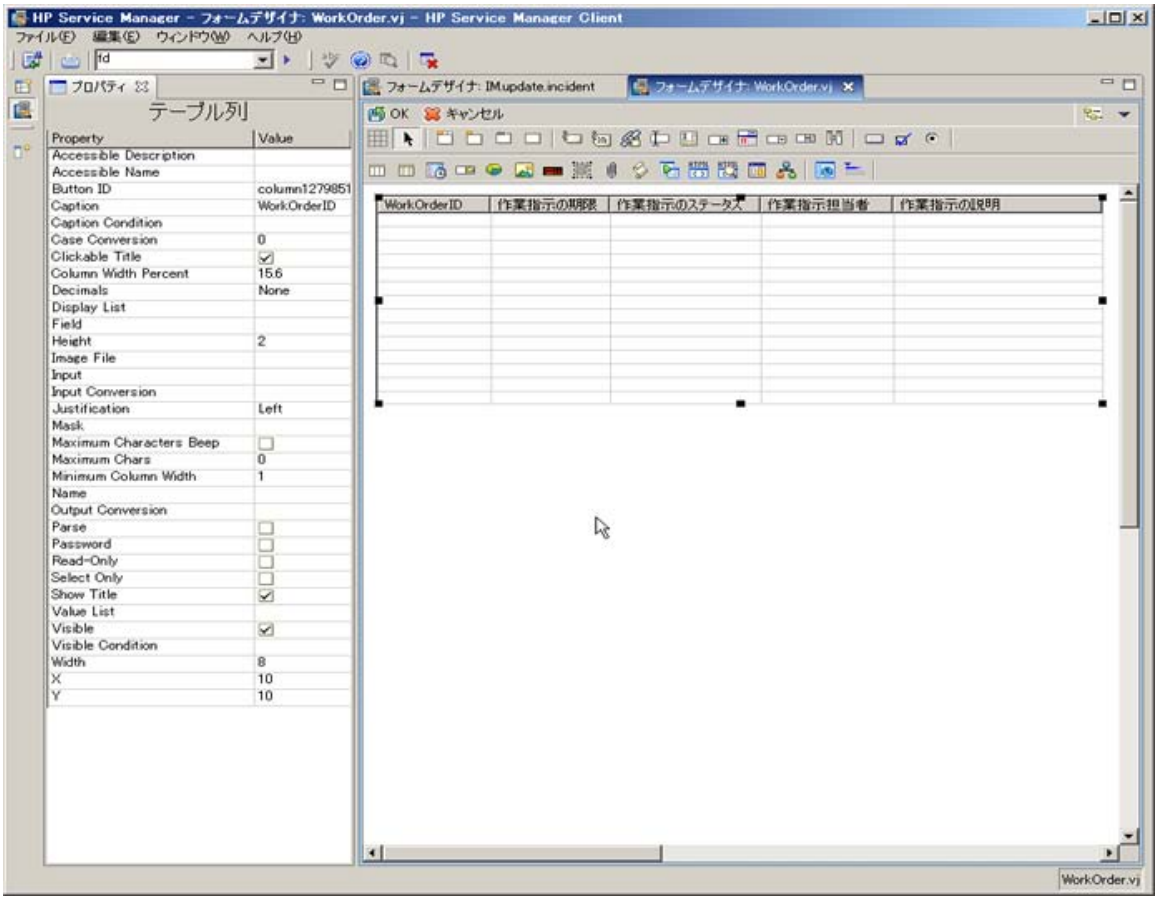
1. システムナビゲータから、**[カスタマイズ]**>**[フォームデザイナ]**をクリックします。
   フォームデザイナの検索フォームが開きます。
2. [フォーム]にフォーム名として「**IM.update.incident**」と入力します。
3. **[検索]**をクリックします。
4. **[デザイン]**をクリックします。
5. 既存のノートブックに新規タブを追加します。
6. タブのキャプションを「作業指示」に設定します。
7. 以下のプロパティを持つサブフォーマットを追加します。
   - Visible:オン
   - Format:WorkOrder.vj
   - Virtual Join:オン
   - Display Blank:オン
   - Display Using Table:オン
   - Input:number.WO.vj
8. **[保存]**をクリックします。
9. フォームデザイナツールを使用して、WorkOrder.vj フォームを作成します。
   - WorkOrderID:ID
   - 作業指示の期限:deadline
   - 作業指示のステータス:status
   - 作業指示担当者:assign.name
   - 作業指示の説明:description
10. **IM.close.incident** に手順 1 から 8 を繰り返します。

    注:手順 9 で作成した WorkOrder.vj フォームは、仮想結合に使用することができます。

以下の図は追加するタブのサンプルを示したものです。


HP Service Manager - フォームデザイナ: IM.update.incident - HP Service Manager Client
ファイル(F) 編集(E) ウィンドウ(W) ヘルプ(H)
プロパティ
サブフォーマット
Property
Value
Display Blank
Display Using Table
Format
WorkOrder.vj
Height
20
Input
number.WO.vj
Name
Tab Stop
0
Virtual Join
Visible
Visible Condition
Width
117
X
4
Y
2
フォームデザイナ: IM.update.incident
OK
キャンセル
インシデント ID
ステータス
担当
担当グループ
担当者
業者
業者チケット
影響を受けるアイテム
影響を受けるサービス
影響を受ける CI
CI は動作中 (停止なし
停止した日時
回復した日時
場所
タイトル
説明
KPIメトリックス
アラート
作業指示
IM.update.incident


HP Service Manager - フォームデザイナ: WorkOrder.vj - HP Service Manager Client
ファイル(F) 編集(E) ウィンドウ(W) ヘルプ(H)
fd
テーブル
Property
Value
Bold Condition
Column Headings
Columns
Default Button ID
Double-Click Button ID
Double-Click Field
Font
Helvetica
Font Increase
Foreground Color Condition
Height
20
Italic Condition
Multiple Selection
Name
Read-Only
Read-Only Condition
Selection Field
Tab Stop
Visible
Visible Condition
Width
200
X
Y
フォームデザイナ: IMupdate.incident
フォームデザイナ: WorkOrder.vj
OK
キャンセル
WorkOrderID
作業指示の期限
作業指示のステータス
作業指示担当者
作業指示の説明
リストの編集
新規エントリ
現在のエントリ
WorkOrderID
作業指示の期限
作業指示のステータス
作業指示担当者
作業指示の説明
追加
置換
削除
上に移動
下に移動
OK
キャンセル



probsummary テーブルにリンク用エイリアスを作成する

作業指示の例では、インシデントに関連付けられている作業指示があれば、ユーザはインシデントレコードから作業指示にアクセスできる必要があります。このためには、probsummary テーブル内にエイリアスを作成して、probsummary テーブルと EXWorkOrder テーブル間のリンクを作成する必要があります。

probsummary テーブルにエイリアスを追加するには:

1. システムナビゲータから、**[カスタマイズ]**>**[データベースディクショナリ]**をクリックします。
   [データベースディクショナリ]フォームがオープンします。
2. [ファイル名]に「`probsummary`」と入力します。
3. **[検索]**をクリックします。
4. [フィールド]タブで、number フィールドを選択して**[フィールド/キーの編集]**をクリックします。
5. **[エイリアスの作成]**をクリックして、名前として「`number.WO.vj`」を入力し、タイプに文字を選択します。
6. **[OK]**をクリックします。

probsummary テーブルに EXWorkOrder テーブルをリンクするには:

1. システムナビゲータから、**[カスタマイズ]**>**[カスタマイズツール]**>**[リンク]**をクリックします。
   [リンクファイル]フォームがオープンします。
2. [名前]に「`probsummary`」と入力します。
3. **[検索]**をクリックします。
4. 最後のエントリの下の行をクリックして、新規エントリのための空きの行を作成します。
5. [ソースフィールド名]に、「`number.WO.vj`」と入力します。
6. 新しい行全体を選択して、**[行の選択]**をクリックします。
7. 以下の情報を入力します。

フィールド	値
フィールド(リンク元/ソース)	number.WO.vj
ファイル(リンク先/ターゲット)	EXWorkOrder
フィールド(リンク先/ターゲット)	RelatedID
クエリ	$query
式	$query="RelatedID=\""+number in $File+"\""

8. **[保存]**をクリックして、**[戻る]**をクリックします。
9. 同じ手順を繰り返して ID フィールドのリンク行を作成します。
10. 以下の情報を入力します。

フィールド	値
フィールド(リンク元/ソース)	ID
ファイル(リンク先/ターゲット)	EXWorkOrder
フィールド(リンク先/ターゲット)	ID
クエリ	$query
式	$query="ID=\""+nullsub(cursor.field.contents(), "xxx")+"\""

11. **[保存]**をクリックして、**[戻る]**をクリックします。

im.set.close プロセス定義の変更

インシデントに関してオープン状態の作業指示がある場合、ユーザがインシデントをクローズできないように、このプロセスを変更する必要があります。

プロセス定義レコードを変更するには:

1. システムナビゲータから、**[カスタマイズ]**>**[文書エンジン]**>**[プロセス]**をクリックします。
[プロセス定義]フォームがオープンします。
2. [プロセス名]に「**im.set.close**」と入力します。
3. **[検索]**をクリックします。
4. [初期 JavaScript]タブに、以下の JavaScript を入力します。

```
var WO=new SCFile ("EXWorkOrder")

var FoundOpenWO=WO.doSelect ("RelatedID=\""+system.vars.$L_
file.number + "\""+" and status ~=\"" + "Closed" + "\""

if (FoundOpenWO == RC_SUCCESS)

{

system.vars.$openWO=true;

}

else

{

system .vars.$openWO=false;

}
```

5. **[保存]**をクリックします。
6. [RAD]タブに以下の情報を入力します。

フィールド	値
注:[RAD]タブ上の一部のフィールドには、変更が必要のない値が自動的に入力されます。	
RAD アプリケーション - RAD アプリケーション us.consume.wrapper をコールしているセクションの下の空のセクションを選択して、以下の情報を入力します。	
RAD 呼び出しの前に評価される式	$L.text="オープン状態の作業指示があります。このインシデントはクローズできません。"
RAD アプリケーション	apm.mb.ok
条件	$openWO=true

パラメータ名	text
パラメータ値	$L.text

7. **[保存]**をクリックします。
8. [最終式]タブに以下の情報を入力します。
   - if ($openWO=true) then ($L.exit="badval")
   - $L.exit="closestate"
9. **[保存]**をクリックして、**[OK]**をクリックします。

作業指示の例をテスト

作業指示システムを作成するタスクを完了したら、このタスクが正常に動作することを確認する必要があります。作業指示の例の基本機能を確認するには、以下の手順に従います。

- インシデントマネージャで、オープン状態のインシデントを見つけるか、インシデントを作成します。
- オプションメニューまたはツールバーのボタンを使用して、作業指示を作成します。オプションメニューに[作業指示の作成]と表示されるはずです。
- 作業指示ウィザードで、インシデントに関する作業指示を作成するためのデータを入力します。
- インシデントに作業指示を追加して、インシデントへの変更を保存します。
- インシデントを開いて編集し、[作業指示]タブで作業指示を編集します。
- 変更を保存します。
- インシデントを再度開きます。今回は作業指示を編集して閉じます。
- インシデントをクローズできるはずです。
- これらのステップを繰り返します。ただし、今回は 1 つのインシデントに 2 つの作業指示を作成します。
- 1 つの作業指示のみをクローズして、インシデントのクローズを試みます。インシデントにオープンされている作業指示があるため、インシデントをクローズできないというエラーメッセージが生成されるはずです。
- インシデントのすべての作業指示をクローズします。インシデントをクローズできるはずです。