SAP BusinessObjects

カスタム データ プロバイダ プラグイン開発者ガイド

■ SAP BusinessObjects 4.0 Support Package 01

2011-02-17

SAP

著作権

© 2011 SAP AG. All rights reserved.SAP、R/3、SAP NetWeaver、Duet、PartnerEdge、ByDesign、SAP Business ByDesign、および本書に記載されたその他のSAP製品、サービス、ならびにそれぞれのロゴは、ドイツおよびその他の国々におけるSAP AGの商標または登録商標です。Business ObjectsおよびBusiness Objectsロゴ、BusinessObjects、Crystal Reports、Crystal Decisions、Web Intelligence、Xcelsius、および本書で引用されているその他のBusiness Objects製品、サービス、ならびにそれぞれのロゴは、米国およびその他の国々におけるBusiness Objects S.A.の商標または登録商標です。Business ObjectsはSAPのグループ企業です。本書に記載されたその他すべての製品およびサービス名は、それぞれの企業の商標です。本書に記載されたデータは情報提供のみを目的として提供されています。製品仕様は、国ごとに変わる場合があります。これらの文書の内容は、予告なしに変更されることがあります。また、これらの文書はSAP AGおよびその関連会社(「SAPグループ」)が情報提供のためにのみ提供するもので、いかなる種類の表明および保証を伴うものではなく、SAPグループは文書に関する誤記・脱落等の過失に対する責任を負うものではありません。SAPグループの製品およびサービスに対する唯一の保証は、当該製品およびサービスに伴う明示的保証がある場合に、これに規定されたものに限られます。本書のいかなる記述も、追加の保証となるものではありません。

2011-02-17

目次

第 1 章 はじめに 5

1.1 このドキュメントについて 5

1.2 このドキュメントの対象読者 5

1.3 前のリリースからの変更点 5

第 2 章 カスタム データ ソース フレームワークの概要 7

第 3 章 システムのワークフロー 9

3.1 データ ソース作成フェーズ 10

3.2 データ プロバイダ フェーズ 12

第 4 章 プラグイン開発入門ガイド 15

4.1 CustomDSExtension インターフェイス 16

4.2 CustomDSComponent インターフェイス 18

4.3 CDSExtensionDescriptor インターフェイス 19

4.4 CustomDataSource インターフェイス 20

4.5 CustomDataProvider インターフェイス 22

4.6 CDSExtensionBaseDescriptor インターフェイス 24

4.7 拡張タイプ 24

第 5 章 プラグインの設定 27

5.1 概要 27

5.2 プラグインの設定 27

第 6 章 プラグインのデプロイ....29

第 7 章 プラグインのサンプル....31

第 8 章 プラグイン開発時の考慮点....33

第 9 章 ユーティリティ クラス....35

9.1 ユーティリティ クラスの制限....35

第 10 章 CDS Framework の制限....37

第 11 章 お勧めのテクニック....39

付録 A より詳しい情報....41

索引 43

はじめに

1.1 このドキュメントについて

このガイドは、プラグイン開発者にカスタム データ ソース フレームワークについて深く理解してもらうことを目的としています。このガイドの内容は次のとおりです。

- カスタム データ プロバイダ プラグインを開発し、そのプラグインを Web Intelligence Desktop で使用する方法。
- プラグインによってサポートされるデータ ソースに基づいて Web Intelligence ドキュメントを簡単に作成するためのプラグインを設定してデプロイする方法。

1.2 このドキュメントの対象読者

このドキュメントは、SAP BusinessObjects Web Intelligence Desktop によってサポートされていないデータ ソースに基づいて Web Intelligence ドキュメントを作成しようとする開発者を対象としています。

1.3 前のリリースからの変更点

- 前のリリースでは、プラグインの識別とロードに webi_customds_extension.xml 構成ファイルが使用されました。SAP BusinessObjects 4.0 リリースでは、この構成ファイルが削除され、プラグイン バイナリのリソース ディレクトリ META-INF/services にあるプラグイン構成ファイルを解析して、プラグインの検索とロードが行われます。構成ファイルの名前は、プラグインのエントリ ポイント実装の完全修飾バイナリ名です。プラグインのクラスパス依存関係は、プラグイン バイナリに含まれている META-INF/MANIFEST.MF ファイルの MODULE-PATH 属性から取得されます。プラグインのその他の詳細は、CDSExtensionDescriptor インターフェイスと CDSExtensionBaseDescriptor インターフェイスの実装から取得されます。
- 前のリリースでは、プラグイン バイナリを任意の場所にデプロイできました。XI 4.0 では、必ず、プラグイン バイナリを <SAP_BOBJ_INST_DIR>¥SAP BusinessObjects Enterprise XI 4.0¥java¥lib¥PersonalDPPlugins フォルダにデプロイする必要があります。

- 前のリリースでは、User Interface エントリ ポイント実装は必要ありませんでした。プラグインの構成ファイルに DataProviderSource エントリ ポイントに対するエントリがある場合、デフォルトの User Interface 実装が提供されました。SAP BusinessObjects 4.0 リリースでは、User Interface エントリ ポイント実装は必須です。

カスタム データ ソース フレームワークの概要

Web Intelligence カスタム データ ソース (CDS) フレームワークは、SAP BusinessObjects Web Intelligence Desktop コンポーネントで使用できるカスタム データ ソースのデータ プロバイダを簡単に作成するためのプラグイン フレームワークです。フレームワークのコア インターフェイスを実装すること、データ ソース情報とデータ プロバイダ動作の両方をプラグイン化できます。これにより、SAP BusinessObjects Web Intelligence Desktop コンポーネントによって直接実装または提供されていないソースに対して Web Intelligence データ プロバイダを作成できます。

CDS Framework は、カスタム データ ソースから取得したローカル ファイル、URI の場所、インメモリー データ構造などの情報を使用するためのインフラストラクチャを提供し、Web Intelligence ドキュメントを作成できるようにします。

CDS Framework は、データ ソースのインクリメンタル アクセスと非インクリメンタル アクセスを両方ともサポートします。データ ソースのインクリメンタル アクセスでは、カスタム データ ソースにアクセスするためにプラグインがユーザーからの何らかの入力を必要とすることがあります。プラグインは、以前に提供された情報がフレームワークに格納されているとして、毎回新しい/更新された情報のみを CDS Framework に提供します。データ ソースにアクセスするためにプラグインがユーザーから必要な情報を取得すまで、これが反復されます。

データ ソースのインクリメンタル アクセスと非インクリメンタル アクセスの実装については、『プラグインのサンプル 』セクションを参照してください。CDS Framework は、各プラグインを独自のサンドボックス環境で動作するようにできます。

ユーザーは、カスタム データ ソースを使用して Web Intelligence ドキュメントを作成する前に、次のタスクを完了しておく必要があります。

1 データ ソース固有のプラグインの開発。
2 プラグインの設定および SAP BusinessObjects インストールへのデプロイ。

上記のタスクを完了したら、新しく開発されたプラグインによってサポートされるカスタム データ ソースを使用して、.wid ドキュメントを作成できます。

通常、ユーザーは各自の環境にあるデータ ソースについては熟知しています。ただし、Web Intelligence アプリケーションを使用して .wid ドキュメントを作成する際は、問題が発生することがあります。CDS Framework は、OEM などのパートナーの要件に合う実装しやすいソリューションを提供できるように特別に設計されています。CDS Framework を使用することで、プラグインの実装者は、Web Intelligence アプリケーションを使用したデータ分析の複雑さを心配することなく、データ ソースに集中できます。

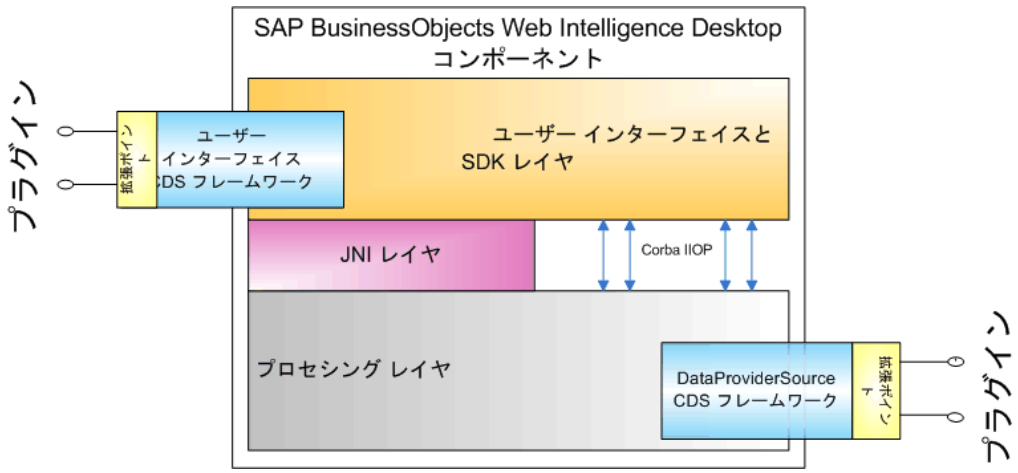
CDS Framework の拡張ポイントは、下の図に示すように、SAP BusinessObjects Web Intelligence Desktop コンポーネント アーキテクチャーの次のレイヤ内に位置します。

- User Interface レイヤ
- DataProviderSource 処理レイヤ

注

Interactive Analysis には DataProviderSource 処理の機能も使用されます。したがって、このドキュメント、Javadoc、およびサンプルでは、DataProviderSource の処理レイヤとサーバー レイヤが同じ意味で使用されます。

CDS Framework は、User Interface (クライアント側) CDS Framework と DataProviderSource (処理/サーバー側) CDS Framework で構成されます。



関連項目

- 31 ページの「プラグインのサンプル」

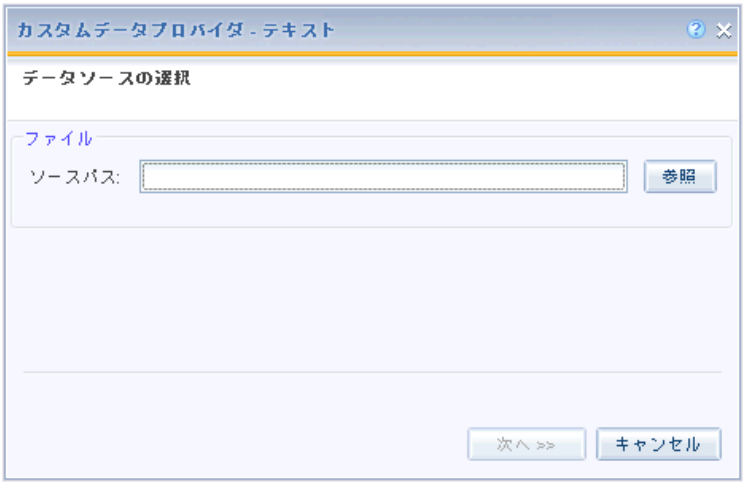
システムのワークフロー

SAP BusinessObjects Web Intelligence Desktop コンポーネントが起動されると、User Interface (UI) および DataProviderSource (DPS) CDS Framework は、<BOBJ_INST_DIR>¥SAP BusinessObjects Enterprise XI 4.0¥java¥lib¥PersonalDPPlugins にある jar ファイルからプラグイン情報を取得します。フレームワークは、プラグインの User Interface エントリ ポイント クラスをロードしてインスタンス化します。インスタンス化に成功すると、フレームワークは、さらに詳細な情報をプラグインに提供して、プラグインが使用準備を完了することを期待します。次に、フレームワークは、プラグインの表示名を取得しようとします。以下の画面が表示されます。ここで、Web Intelligence ドキュメントを作成するためのデータ ソースを選択できます。

注

このセクションで説明するワークフローは、拡張タイプによって異なります。詳細な違いについては、『拡張タイプ 』セクションを参照してください。このセクションで説明するワークフローとシーケンス図は、リソース タイプが FILE の拡張の場合です。これらの拡張は、データ ソースにアクセスし、データ プロバイダを構築するために、ユーザー入力を必要とします。

プラグインを選択すると、ダイアログ ボックスが表示されます。ここで、ユーザーはデータ ソースの場所を指定できます。ユーザーは、[参照]オプションを使用するか、ファイルの場所を手動で入力して、必要な入力を指定できます。



Framework がユーザー入力を取得すると、UI Framework は、データ ソースにアクセスしてデータ プロバイダを構築するための要求を処理レイヤに送信します。この要求には、ユーザーが指定したデータ ソースの場所に関する情報がプラグインに関する情報と共に含まれています。次に、DataProviderSource Framework は、適切なプラグインに関する情報を検索し、DataProviderSource エントリ ポイント クラスをロードしてインスタンス化しようとします。エラーが発生すると、適切な情報がユーザーに表示されます。エラーがない場合、フレームワークは、データ ソースに関するユーザー入力と共に詳細な情報をプラグインに提供します。この時点で、フレームワークは、プラグインが使用準備を完了することを期待し、ユーザーから指定されたデータ ソースをプラグインが処理できるかどうかを検証します。

注

CDS Framework は、正確性またはデータ ソースの有無に関する検証を行いません。検証を行い、ソースの入力が正しくない場合は、プラグインが適切な例外をスローする必要があります。エラーがある場合、フレームワークは、データ ソースのステータスを[無効]にします。

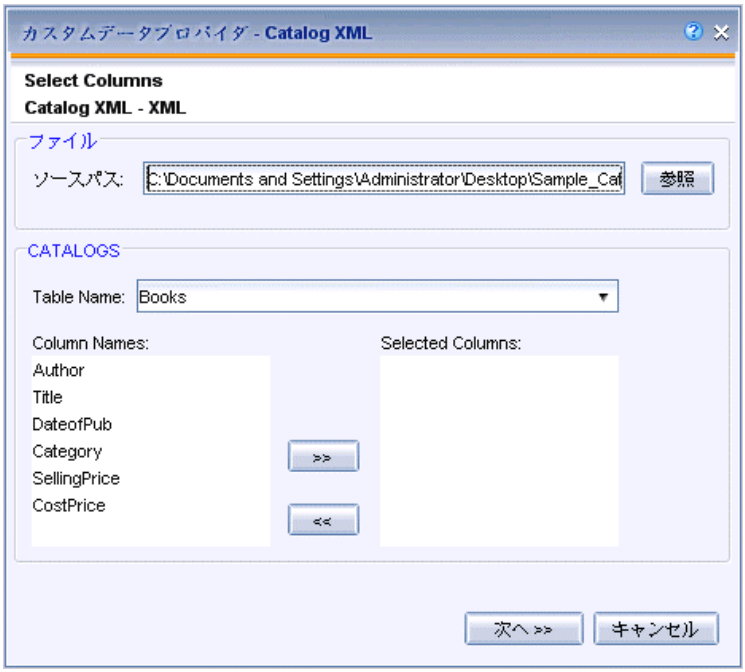
関連項目

- 24 ページの「拡張タイプ」

3.1 データ ソース作成フェーズ

エラーがない場合は、データ ソース アクセス/作成フェーズが開始されます。DataProviderSource CDS Framework は、処理/サーバー側プラグイン実装がデータ ソースにアクセスして Web Intelligence データ プロバイダの構築を支援できるかどうかをチェックします。ソース情報以外のユーザー入力を必要とする拡張の場合 (拡張にプラグイン固有の User Interface コンポーネントが含まれる場合)、最初の呼び出しは false を返します。この場合、プラグインは、必要な情報を提供できません。Framework によるこのチェックが false 値を返す場合、フレームワークは、データ ソースのステータスを incomplete にマークし、独自の User Interface を構築してユーザー入力を取得するためにプラグインから UI 側プラグイン実装に送信される情報の取得を試みます。この情報は、データ ソース パラメータとして処理され、名前と値のペアである必要があります。この名前と値は、null 以外の java.lang.String オブジェクトです。

処理レイヤから DS パラメータを取得した後に、UI Framework は、プラグイン固有の User Interface コンポーネントの取得を試みます。ユーザー インターフェイスの構築時にプラグイン側でエラーが発生した場合は、該当するエラー メッセージが表示されます。エラーがない場合は、ダイアログ ボックスにプラグイン User Interface コンポーネントが表示されます。



注

- CDS Framework は、[次へ]ボタン、[キャンセル]ボタン、およびソース入力パネルを制御します。
- このダイアログ ボックスは、プラグインの UI コンポーネントを取得した後でサイズ変更されます。このダイアログ ボックスのサイズに制限はありません。また、このダイアログ ボックスは、プラグインの UI コンポーネントを取得した後、処理レイヤで別の操作を実行するまでは、自分自身のサイズを動的に変更しません。
- ユーザーが[列の選択]画面の[ヘルプ]アイコン をクリックすると、ブラウザでプラグイン ヘルプの URL が開きます。

新しい画面が表示されたら、ユーザーは入力を行い、[次へ]ボタンをクリックできます。ユーザーが[次へ]ボタンをクリックすると、UI Framework は、プラグインから更新された情報の取得を試みます。

注

ユーザー インターフェイス プラグインでは、どのような種類の入力も検証しないことをお勧めします。入力の検証は、プラグイン実装の処理レイヤ/DataProviderSource レイヤで行う必要があります。

エラーがない場合、UI Framework は、カスタム データ ソースにアクセスして新しい/更新された情報でデータ プロバイダを構築するための要求を処理レイヤに再送信します。

注

フレームワークは、データ ソースのインクリメンタル アクセスと非インクリメンタル アクセスを両方ともサポートします。したがって、プラグインは、データ ソース パラメータの新しい/更新された部分のみを提供することも、全体を提供することもできます。プラグインがいずれかのデータ ソース パラメータを削除する場合は、そのパラメータのキーを空の文字列値に更新する必要があります。

CDS Framework は、カスタム データ ソースにアクセスする要求を受け取ると、処理レイヤ/サーバー側プラグイン実装がカスタム データ ソースを使用して Web Intelligence データ プロバイダを構築できるかどうかをチェックします。拡張タイプに応じて、結果は次のいずれかになります。

1 ユーザーから提供された情報が正しくないか、適切ではありません。
2 更新された情報に基づき、プラグインには、ユーザー入力を必要とする追加の情報セットを持ちます。
3 提供された情報が適切であるため、プラグインは、データ ソースにアクセスして Web Intelligence データ プロバイダの構築を支援できます。

シナリオ 1 および 2 の場合、フレームワークは、プラグインにデータ ソース パラメータを再度要求します。シナリオによっては、DPS プラグインは新しい情報セットを返す必要があります。次に、この情報が UI プラグインに送信されます。PDS プラグインが必要な情報を取得するまで、これが反復されます。

注
ユーザーからの入力が無効な場合は、入力が無効だという警告をプラグインからユーザーに出すことをお勧めします。

DPS プラグイン実装は、必要な情報を取得すると、この情報を入手できたことをフレームワークに通知します。フレームワークは、データ ソースのステータスを created にマークします。

DPS Framework は、Web Intelligence データ プロバイダを構築するために、プラグインから情報の取得を試みます。プラグインは、Web Intelligence データ プロバイダに入れられるオブジェクト/列の詳細/説明を提供する必要があります。エラーがある場合はユーザーに警告されます。エラーがない場合、DPS Framework は、PDS プラグインのデータ プロバイダ インスタンスの取得を試みます。このプラグインのデータ プロバイダ インスタンスは、データの取得に関連する追加の呼び出しを行うためにフレームワークによって使用されます。このオブジェクトが正常に取得されると、データ ソース フェーズの終了と、UI プラグイン実装の UI コンポーネントのライフ サイクルの終了がマークされます。

3.2 データ プロバイダ フェーズ

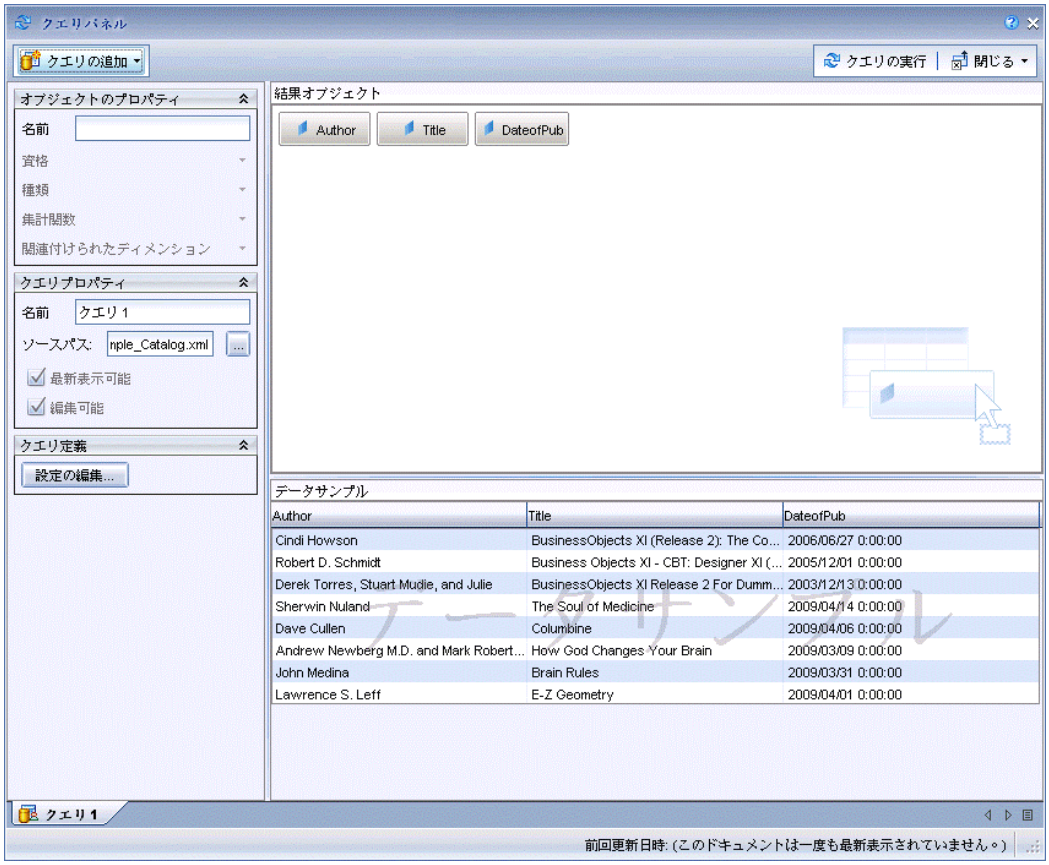
CDS Framework に従い、DPS プラグインは、DataProvider 拡張ポイント実装オブジェクトを取得した後で、必要な手順を実行してデータ反復子を開く必要があります。また、DPS プラグインは、データ プロバイダ オブジェクトに対応するデータ行数について必要な情報を持つ必要があります。プラグインがこの情報を持つ場合、CDS Framework は、次のデータ チャンクがあるかどうかをプラグインに確認する方法ではなく、行数に基づく呼び出しを行うことで、Framework パスを選択します。プラグインは、DPS プラグインから送信された反復子情報の関連プロパティをマークすることで、別のパスの選択を指定できます。詳細については、22 ページの「CustomDataProvider インターフェイス」を参照してください。

DPS プラグインが Web Intelligence データ プロバイダ オブジェクトに対する適切なデータを持つ場合、CDS Framework は、チャンク単位でデータの取得を試みます。CDS Framework から要求されるデータがサンプル抽出の目的に使用される場合、フレームワークは、DPS プラグインにデータ セット全体を要求しません。CDS Framework は、クエリーを実行または更新する場合にのみ、データ セット全体の取得を試みます。

ドキュメントの作成時には、[クエリー パネル]が表示されます。ユーザーは、このパネルを使用して次のアクションを実行できます。

- データ プロバイダ オブジェクトの名前、資格、およびタイプの変更
- 集計関数の変更
- DP ソースの変更

- 現在のデータ ソースの設定の編集

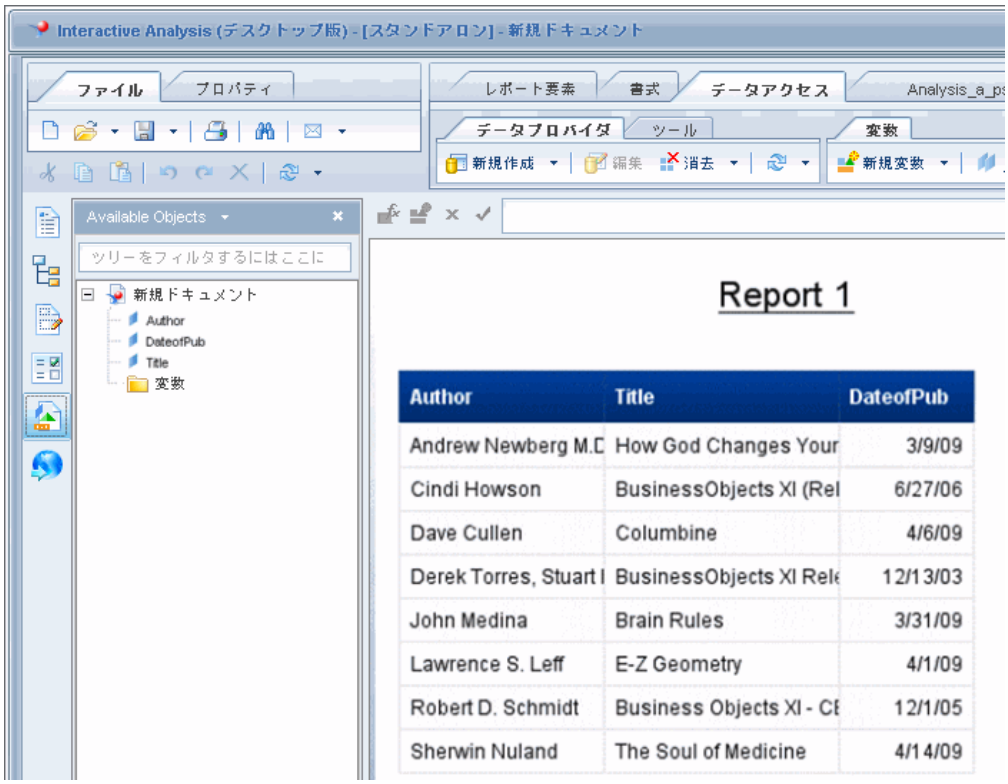


注

- 資格、集計関数、最新表示可能、編集可能などの用語の詳細については、SAP BusinessObjects Web Intelligence Desktop コンポーネントのマニュアルを参照してください。
- フレームワークは、データ プロバイダ オブジェクトのオブジェクト資格の初期割り当てを制御します。
- ユーザーが[クエリー パネル]でソースを変更しようとすると、PDS プラグインは、新しいソースの処理/エラーの生成/新しいソースを処理するための追加情報の要求を行えなくなり、データ ソースのステータスは incomplete とマークされます。この場合、フレームワークは、[クエリー パネル]上のデータ ソース情報の再構築をサポートしません。

ユーザーが[クエリーの実行]をクリックするか、ドキュメントを最新表示すると、DPS Framework は、情報の完全性を検証し、データ プロバイダ オブジェクトの再取得を試みます。エラーがない場合、フレームワークは、データ セット全体に対してデータ プロバイダ フェーズを続行します。完全なデータ セットが正常に取得されると、

ユーザーに Web Intelligence ドキュメントが表示されます。



プラグイン開発入門ガイド

CDS Framework は、CDS Framework プラグインに含まれる Web Intelligence アーキテクチャーの複雑な要素を覆い隠し、さまざまな機能の拡張ポイントとして動作する実装が容易な一連のインターフェイスを提供します。現在のバージョンでは、Web Intelligence (.wid) ドキュメントを作成するために実装する必要があるインターフェイスは 6 つのみです。

注

CDS Framework の現在のバージョンでは、プラグインを作成するには、UI Framework に含まれる CustomDSExtension インターフェイスと CDSExtensionDescriptor インターフェイスを実装する必要があります。UI ベースのプラグインの場合は、CustomDSExtension インターフェイスおよび CDSExtensionDescriptor インターフェイスと共に CustomDSComponent インターフェイスを実装する必要があります。

User Interface CDS Framework では、以下のインターフェイスを実装する必要があります。

- CDSExtensionDescriptor – CDS Framework User Interface プラグインの詳細を提供するインターフェイス
- CustomDSExtension – プラグインの User Interface エントリ ポイント
- CustomDSComponent – プラグインの User Interface コンポーネントを提供するインターフェイス

CustomDSExtension インターフェイスと CustomDSComponent インターフェイスの抽象実装が提供されているため、プラグインの開発者は、これらのインターフェイスを実装するのではなく、それぞれの抽象クラス (AbstractCustomDSExtension および AbstractCustomDSComponent) を拡張する必要があります。

これらのインターフェイス、その抽象実装、および User Interface CDS Framework の他の共通クラスは、<BOBJ_INST_DIR>/SAP BusinessObjects Enterprise XI 4.0/java/lib にある以下のバイナリにバンドルされています。

- cdsuiframework.jar
- cdsframework_common.jar

どのタイプのプラグインに対しても DataProviderSource CDS Framework では、以下のインターフェイスを実装する必要があります。

- CDSExtensionBaseDescriptor – CDS Framewok DataProviderSource プラグインの詳細を提供するインターフェイス
- CustomDataSource – プラグインの DataProviderSource エントリ ポイント
- CustomDataProvider – プラグインの DataProvider インターフェイス

CustomDataSource インターフェイスと CustomDataProvider インターフェイスの抽象実装が提供されているため、プラグインの開発者は、これらのインターフェイスを実装するのではなく、それぞれの抽象クラス (AbstractCustomDataSource および AbstractCustomDataProvider) を拡張する必要があります。

これらのインターフェイス、その抽象実装、および Data Provider CDS Framework の他の共通クラスは、<BOBJ_INST_DIR>/SAP BusinessObjects Enterprise XI 4.0/java/lib ディレクトリにある以下のバイナリにバンドルされています。

- cdsframework.jar

- cdsframework_common.jar

最初に、UI 実装用と DPS 実装用の 2 つの新しいプロジェクトを作成します。IDE で、各プロジェクトのクラスパスに cdsuiframework.jar、cdsuiframework_common.jar、および cdsframework.jar を追加し、上記のインターフェイスを実装します。

4.1 CustomDSExtension インターフェイス

CustomDSExtension インターフェイスのメソッドを次に示します。

注
ユーザー インターフェイスが必要ない場合でも、このインターフェイスの実装は必須です。

- getExtensionDescriptor() – フレームワークは、デフォルト コンストラクタを介してプラグインをインスタンス化した後で、このメソッドを呼び出してプラグイン詳細を取得します。
- init(SessionInfo sessinfo) – フレームワークは、getExtensionDescriptor() を呼び出した後で、このメソッドを呼び出します。プラグインは、Framework から使用できるように自分自身を初期化する必要があります。

  SessionInfo メソッドのパラメータには、セッション ID、呼び出し時のインターフェイスのロケール、プラグインに関する情報などのセッションの詳細が含まれます。プラグインのライフサイクルの開始時点によっては、このメソッドの呼び出し時点でセッション ID が空の文字列になる場合があります。ただし、フレームワークは、CDSComponent インスタンスの取得を試みると共に、セッション ID に関連する DataHolder オブジェクトを通して適切な値を送信します。

  init(Sessioninfo sessinfo) のサンプル コード スニペットを以下に示します。

```
public void init(SessionInfo sessInfo) throws CDSExtensionException
{
  interfaceLocale = sessInfo.getInterfaceLocale();
  pluginName = sessInfo.getExtensionName();
  rootHelpPath = sessInfo.getProductHelpPath();
}
Where private Locale interfaceLocale;//used to store the interface locale
private String pluginName; //used to store the extension name of plug-in as described in config file.
private String rootHelpPath; // used to store the root help path of the pug-in.
```

- setLocale(Locale loc) – フレームワークは、このメソッドを呼び出して、プラグインがロケールを設定/リセットできるようにします。

  setLocale のサンプル コードを次に示します。

```
public void setLocale(Locale loc) {
 interfaceLocale = loc;
}
```

- getDisplayName() – フレームワークは、このメソッドを呼び出して、プラグインのローカライズされた表示名を取得します。CDS Framework は、この情報を使用して、プラグインの名前を表示します。

```
public String getDisplayName(){
return"Catalog XML-URL";}
```

- getFileExtnDescription() – フレームワークは、このメソッドを呼び出して、プラグインによってサポートされる、リソース タイプが FILE のファイル拡張の説明を取得します。CDS Framework は、この情報を使用して、[開く] ダイアログ ボックスで、[ファイルの種類] オプションを表示します。

```
public String getFileExtnDescription() {
 return "Xml Files";
}
```

  この情報は、構成ファイルにも提供できます。詳細については、「Configuring the Plug-in」を参照してください。
- getCDSComponent(DataHolder dataHolder) – フレームワークは、このメソッドを呼び出して、CustomDSComponent インスタンスを取得します。CustomDSComponent インスタンスは、プラグインの User Interface コンポーネントを取得するためにフレームワークによって使用されます。DataHolder メソッド パラメータには、プラグインが User Interface を構築するための情報が入っています。User Interface コンポーネントを持たないプラグインの場合、このメソッドは null を返すことができます。

```
public CustomDSComponent getCDSComponent(DataHolder dataHolder) throws CDSExtensionException
{
 return new CatalogUIProvider(dataHolder);
} // where CataloUIProvider is a user-defined class that implements
the CustomDSComponent interface. This interface is responsible for the
look and feel of the plug-in.
```

- getImageIcon() – プラグインの画像アイコンがある場合、フレームワークは、このメソッドを呼び出して取得します。このアイコンは、[最近使用したデータ ソース]メニュー/パネルの表示などの目的で使用されます。

```
public ImageIcon getImageIcon() throws CDSExtensionException {
 return null;//Let the framework get it from FileSystemView
}
```

- isExtensionHelpAvailable() – フレームワークは、このメソッドを呼び出して、プラグイン固有のヘルプ ファイルと製品のヘルプ ファイルのどちらを表示するかを判断します。

  プラグイン開発者は、以下のコードを使用して、プラグインのヘルプ ファイルを提供できます。

```
public boolean isExtensionHelpAvailable() {
 return true;
}
```

- getHelpURL() – フレームワークは、このメソッドを呼び出して、プラグインのヘルプの URL を取得します。この URL には、file:/// などのファイル プロトコルで始まるヘルプ ファイルの絶対パスか、http URL を使用できます。isExtensionHelpAvailable() メソッドが false を返す場合、このメソッドは呼び出されません。

```
public String getHelpURL(){
  StringBuffer helpPath = newStringBuffer();
  helpPath.append("file:///");
  helpPath.append(rootHelpPath);
  helpPath.append(getLanguageFolder());
  helpPath.append("/webintelligence/richclient/html/context.htm");
  return
  helpPath.toString();
}
```

- getHelpURL(String, String) – フレームワークは、このメソッドを呼び出して、プラグインのヘルプの URL を取得します。この URL には、"file:///" で始まるヘルプ ファイルの絶対パスまたは http URL を使用できます。isExtensionHelpAvailable() メソッドが false を返す場合、このメソッドは呼び出されません。

  このメソッドは次の引数を受け取ります。

  - context: 現在のユーザー操作を指します。これは、画面タイトルまたはサブ タイトルになります
  - error message: ユーザー ワークフローで発生したエラー メッセージを指します。

このメソッドが <code>null</code> または空の値を返す場合は、製品のヘルプ ファイルが表示されます。このメソッドは、プラグインの状況依存ヘルプ/エラー メッセージ ヘルプの URL の作成に役立ちます。

注
プラグインは、init(SessionInfo) メソッド呼び出して、指定されたロケールに基づいてローカライズされたヘルプ ファイルを提供する必要があります。

- getDataProviderIcon() – フレームワークは、このメソッドを呼び出して、プラグインのデータ プロバイダ画像アイコンを取得します。

```
public ImageIcon getDataProviderIcon() throws CDSExtensionException{
        return new ImageIcon("C:¥¥icons¥¥officelogo.gif");}
```

- getDataSourceDescription() – フレームワークは、このメソッドを呼び出して、SAP BusinessObjects Web Intelligence Desktop コンポーネントのデータ ソースの一覧に表示されるデータ ソースの説明を取得します。
- getNewActionText() – フレームワークは、このメソッドを呼び出して、Web Intelligence データ ビューから同じデータ ソースの新しいデータ プロバイダを追加するときに表示されるデータ プロバイダの新しい操作テキストを取得します。
- getNewActionToolTip() – フレームワークは、このメソッドを呼び出して、Web Intelligence データ ビューから同じデータ ソースの新しいデータ プロバイダを追加するときに表示されるデータ プロバイダの新しい操作ツールヒントを取得します。
- getNewDataProviderIcon() – フレームワークは、このメソッドを呼び出して、データ プロバイダの新しい画像アイコンを取得します。
- clean() – フレームワークは、このメソッドを呼び出して、プラグインがクリーンアップ アクティビティを実行できるようにします。

AbstractCustomDSExtension は、CustomDSExtension インターフェイスの抽象実装です。

4.2 CustomDSComponent インターフェイス

CustomDSComponent インターフェイスのメソッドを次に示します。

- getUIComponent() – フレームワークは、このメソッドを呼び出して、ユーザー入力ダイアログ ボックスに表示されるプラグインの User Interface コンポーネントを取得します。User Interface コンポーネントは、javax.swing.JComponent タイプである必要があります。
- getActionTitle() – フレームワークは、このメソッドを呼び出して、ユーザーが User Interface コンポーネントで実行する現在のアクションを取得します。

```
public String getActionTitle(){
return "Select Columns";
}
```

- getUpdatedDSParams() – フレームワークは、このメソッドを呼び出して、プラグインから更新されたデータ ソース パラメータを取得します。このメソッドは、ユーザーが入力を提供し、[次へ]ボタンをクリックした後で呼び出されます。
- processMissingDSParams(Map dsParams) – フレームワークは、このメソッドを呼び出して、プラグインが無効、不足、または新しいデータ ソース パラメータを処理できるようにします。プラグインは、無効な入力や不足している入力があることをユーザーに警告したり、別の入力セットを受け入れるために User Interface コンポーネントを更新する必要があります。

- free() – フレームワークは、このメソッドを呼び出して、プラグインが User Interface コンポーネントに対してクリーンアップを実行できるようにします。

AbstractCustomDSComponent は、CustomDSComponent インターフェイスの抽象実装です。

4.3 CDSExtensionDescriptor インターフェイス

CDSExtensionDescriptor インターフェイスは、ユーザー インターフェイス ワークフローに役立つプラグインの詳細を取得するために使用されます。この情報は、Framework の以前のバージョンにある CDS Framework 拡張構成ファイルの置き換えに使用されます。

CDSExtensionDescriptor インターフェイスのメソッドとサンプル実装を次に示します。

メソッド	サンプル実装
getName()	`public String getName(){ return "Catalog XML"; }`
getVersion()	`public String getVersion(){ return "1.0"; }`
getDataSource Type()	`public String getDataSourceType(){ return "XML"; }`
getDSDefaultLoca tions()	`public String[] getDSDefaultLocations(){ String[] dsDefaultLocations={"C:¥¥PersonalDPFiles¥¥Test.xml","C:¥¥PersonalDPFiles1¥¥Test1.xml"}; return dsDefaultLocations; }`
getDSFileExtension()	`public String[] getDSFileExtension(){ String[] dsFileExtension={"*.xml"}; return dsFileExtension; }`
getDSResource Type()	`public CDSExtensionResourceType getDSResourceType(){ return CDSExtensionResourceType.FILE; }`
requiresExtensionDi alog()	`public boolean requiresExtensionDialog(){ return true; }`

注

getName()、getVersion()、および getDataSourceType() メソッドから返される値は、DataProviderSource プラグイン実装と User Interface プラグイン実装で同じになる必要があります。getDataSourceType() の値は、プラグインごとに一意である必要があります。この値が重複していると、どちらのプラグインも実行時に使用できなくなります。

4.4 CustomDataSource インターフェイス

CustomDataSource インターフェイスのメソッドを次に示します。

- getExtensionBaseDescriptor() – フレームワークは、デフォルトコンストラクタを介してプラグインをインスタンス化した後で、このメソッドを呼び出してプラグイン詳細を取得します。
- init (CustomDataSourceInfo customDataSourceInfo) – フレームワークは、getExtensionBaseDescriptor() メソッドを呼び出した後で、このメソッドを呼び出します。プラグインは、フレームワークが使用できるように自分自身を初期化し、ユーザーから提供されたデータソースを処理できるかどうかを検証する必要があります。

  注
  CDS Framework は、有効期間またはデータ ソースの有無に関する検証を行いません。これはプラグインによって実行されます。ソース入力が正しくない場合、プラグインは、com.businessobjects.customds.exception.DataSourceException タイプの例外をスローする必要があります。CustomDataSourceInfo() メソッドのパラメータには、ソース入力、セッション ID、呼び出し時のインターフェイスのロケール、プラグインに関する情報などの詳細が含まれます。

```
public void init(CustomDataSourceInfo dataSourceInfo) throws DataSourceException{
  CustomDataSourceInfo dsInfo = dataSourceInfo;
  String source = dsInfo.getSource();
  boolean isURLSource = false;
  if(dsInfo.getDataSourceType()!= null)
   {
    isURLSource =
    dsInfo.getDataSourceType().equalsIgnoreCase(URL_DSTYPE);//String to
    identify the URL dstype as configured in the configuration file
   }
  m_catalogParser = new CatalogXMLFileParser(source,
  isURLSource);
  try
   {
      Map catalogDetails = newHashMap();
      catalogDetails.put("Books", "Book");
      catalogDetails.put("Magazines", "Magazine");
      m_catalogParser.parseStructure(catalogDetails);
   }catch (DataSourceException dse) {
     throw dse;
    }
catch (Throwable t) {
 /*
* if the cause is not known better to throw an error with "MINOR_GENERIC"
as Minor code
* also it is a good practice to put the Major code which can closely
relate to the sequence flow
*/
throw new DataSourceException (t.getMessage(),ErrorCodes.MAJOR_GET_INFO_ERROR,
ErrorCodes.MINOR_GENERIC);
  }
}
```

- getDSParameters (final Map currentDSParams) – フレームワークは、このメソッドを呼び出して、データ ソース パラメータを取得します。これらのパラメータは、キーと値のペアの形式である必要があります。キーと値のペアに含まれるキーと値は、null または空でない java.lang.String オブジェクトである必要があります。キーと値のペアに含まれる値が空の場合、フレームワークは、そのペアを保存せずにマップから削除します。

```
public Map getDSParameters(final Map currentDSParams) throws DataSourceException{
  boolean containCatalogDetails = false;
  boolean containColumnDetails = false;
  if (currentDSParams != null)
  {
   /*
   * This check is done to find out if it this call is for edit of the source or a new access call.
   * This kind of check can also be done by plug-ins which has incremental building of DS.
```

```
 * In both cases assumption is that the initial DS params are not removed
 */
Iterator currParamIter = currentDSParams.entrySet().iterator();
  while (currParamIter.hasNext ()) {
    Map.Entry currParamEntry = (Map.Entry)
    currParamIter.next ();
    String currParamKey = currParamEntry.getKey().toString();
          if (currParamKey.startsWith("catalog_xml_details."))
              containCatalogDetails = true;
          if(currParamKey.startsWith("catalog_xml_details_"))
              containColumnDetails = true;
          if(containCatalogDetails &&
              containColumnDetails)
              break;
  }
}
if (containCatalogDetails && containColumnDetails)
    return currentDSParams;
    Map newParams = newHashMap();
    Map catalogDetails = m_catalogParser.getCatalogDetails();
    Iterator paramIter = catalogDetails.entrySet().iterator();
    int index = 1;
    while(paramIter.hasNext ()) {
          Map.Entry paramEntry = (Map.Entry) paramIter.next();
          String paramKey = paramEntry.getKey().toString();
          newParams.put("catalog_xml_details." + index, paramKey);
          Vector paramValue = (Vector)paramEntry.getValue();
          /*
          * If the implementation wants to throw an exception with a message that
          is not available in the pre-defined set of error codes,
          * it is expected to use the minor error code "MINOR_CUSTOM". However, it
          should be noted that in case of such (DataSourceException)
          * exception's with minor code mentioned as "MINOR_CUSTOM", the onus of passing the localized
          string is on the extension point implementation.
          * This localization can be achieved by using the locale passed by the
          framework during init method call.
          * This message is then shown to the user on the client side.
          */
   if (paramValue == null)
      throw new DataSourceException ("Error While Parsing the Parameters",ErrorCodes.MAJOR_GET_INFO_ERROR,
      ErrorCodes.MINOR_CUSTOM);
      Iterator colIter = paramValue.iterator();
      int colIndex = 1;
      while (colIter.hasNext()) {
          Object colObj = colIter.next();
   if( colObj == null) thrownew DataSourceException
      ("Unsupported XML",ErrorCodes.MAJOR_GET_INFO_ERROR, ErrorCodes.MINOR_CUSTOM);
      String colName = colObj.toString();
      newParams.put("catalog_xml_details_"+paramKey+ "."+ colIndex, colName);
      colIndex++;
   }
   if(colIndex == 1)
       throw new DataSourceException ("XML File doesn't contain structure information",ErrorCodes.MAJOR_GET_INFO_ERROR,
       ErrorCodes.MINOR_CUSTOM);
       index++;
   }
 return newParams;
}
```

- isDataSourceParamsComplete (final Map currentDSParams) – フレームワークは、このメソッドを呼び出して、データ ソースにアクセスしてデータ プロバイダの構築を支援できるかどうかに関する情報をプラグインから取得します。この呼び出しが行われた場合、結果は次のいずれかになります。
  - プラグインは、最初のデータ ソース パラメータ セットに対する正しいユーザー入力を受け取りました。ただし、プラグインは、さらに、他のデータ ソース パラメータ セットに対するユーザー入力を必要とします。
  - ユーザーは、適切な情報を提供せずに、次の一連の操作 (選択されたオブジェクトの詳細やサンプルデータの取得など) を実行しようとしました。
  - ユーザーからの入力は、プラグインがデータ ソースにアクセスしてデータ プロバイダの構築を支援するには十分です。

最初の 2 つの場合、プラグインは isDataSourceParamsComplete (final Map currentDSParams) メソッドに対して false を返します。3 つ目の場合、このメソッドは true を返します。

```
public boolean isDataSourceParamsComplete (finalMap currentDSParams) throws DataSourceException {
    boolean hasSelectedCols = false;
    Object obj = currentDSParams.get(CDS_XML_SELECTED_COLS);
    if(obj != null) {
       String strSelCols = obj.toString();
          if(!(strSelCols.equals("")))
              hasSelectedCols = true;
    }
    return (currentDSParams.containsKey(CDS_XML_SELECTED_DETAIL) &&hasSelectedCols);
}
```

- getColumnsInfo (final Map currentDSParams) – フレームワークは、このメソッドを呼び出して、Web Intelligence データ プロバイダの構築に使用されるオブジェクト (列) の詳細を取得します。

  サンプル コードを次に示します。

```
public ColumnsInfo getColumnsInfo (final Map currentDSParams) throws DataSourceException {
  ColumnsInfo columnsInfo = new ColumnsInfo();
  ............
  String colName = obj.toString();
  Integer colType = …;
  ............
  columnsInfo.addColumn(new ColumnSpec(colName,colType.intValue()));
  ............
  return columnsInfo;
}
```

- getChunkSize () – フレームワークは、このメソッドを呼び出して、プラグインの CustomDataProvider 実装から提供される各チャンクのサイズを取得します。

```
public int getChunkSize () throws DataSourceException {
                return DEFAULT_CHUNK_SIZE;
        }
    Where publicstaticfinalint DEFAULT_CHUNK_SIZE = 100;
```

- getCustomDataProvider (ColumnSpec[] columnSpecs) – フレームワークは、このメソッドを呼び出して、プラグインのデータ プロバイダの実装を取得します。

```
 public CustomDataProvider getCustomDataProvider(ColumnSpec[]
columnSpecs) throws DataSourceException {
     return new XMLCatalogDataProvider(m_selCatalogItem, columnSpecs);
 }
```

- clean () – フレームワークは、このメソッドを呼び出して、プラグインがクリーンアップ アクティビティを実行できるようにします。

AbstractCustomDataSource は、CustomDataSource インターフェイスの抽象実装です。

4.5 CustomDataProvider インターフェイス

CustomDataProvider インターフェイスのメソッドを次に示します。

- openIterator (final IteratorInfo iteratorInfo, ColumnSpec[] columnSpecs) – フレームワークは、このメソッドを呼び出して、Web Intelligence データ プロバイダ オブジェクト/列の詳細とデータ反復子 ID を提供します。プラグインは、IteratorInfo オブジェクトに必要な情報を提供する必要があります。このメソッドを実装する場合は、次の点に注意してください。

フレームワークは、次のいずれかの方法を使用してデータを取得できます。

- 毎回 getNextChunk() メソッドを呼び出す前に、hasNextChunk() メソッドを呼び出して、次のデータ チャンクがあるかどうかをチェックします。
- データ プロバイダで使用できる行の合計数としてプラグインによって指定された値に基づいて、連続して getNextChunk() メソッドを呼び出します。この場合、Framework は hasNextChunk() メソッドは呼び出しません。

フレームワークは、プラグインによって設定された isRowCountBased プロパティの値に基づいて決定を行います。この値が true に設定されている場合、Framework は 2 つ目の方法を使用します。false に設定されている場合は、最初の方法を使用します。デフォルト値は false です。

フレームワークは、一意のデータ反復子 ID を提供します。この ID は、このメソッドの呼び出し中に、IteratorInfo オブジェクトから取得できます。DataProvider 実装へのその他の呼び出しは、この反復子 ID に基づいて行われるため、プラグインはこの情報を保存しておく必要があります。

- setChunkSize (int iIterId, int chunkSize) – フレームワークは、このメソッドを呼び出して、プラグインがチャンク サイズを設定できるようにします。
- hasNextChunk (int iIterId) – フレームワークは、このメソッドを呼び出して、openIterator メソッドの呼び出し中に送信された ID によって識別されるデータ反復子に対して返すチャンクがまだプラグインにあるかどうかを検証します。
- getNextChunk (int iIterId, ColumnSpec[] columnSpecs) – フレームワークは、このメソッドを呼び出して、次のデータ チャンクを取得します。戻り値を null にすることはできません。戻り値が null の場合、Framework は例外をスローします。

サンプル コードを次に示します。

```
public Chunk getNextChunk(ColumnSpec int
iIterId, ColumnSpec[] columnSpecs) throws DataSourceException
{
 Chunk chunk = new Chunk (columnSpecs);
 ……
 Iterator rowsIter = vRows.iterator();
 while(rowsIter.hasNext()) {
 Row row = new Row(columnSpecs.length);
……
 int colIndex = 0;
Iterator selColsIter = …;
while(selColsIter .hasNext()) {
   Integer colType = …;
   switch(colType.intValue()) {
     case ObjectTypes.STRING:
       row.addString(…,colIndex);
       break;
     case ObjectTypes.NUMBER:
       Double doubleVal = …;
       row.addNumber(doubleVal, colIndex);
       break;
     case ObjectTypes.DATE:
       String strColDateVal = …;
       Date dateVal = null;
       if(strColDateVal != null)
{
    SimpleDateFormat sdf = new SimpleDateFormat();
    sdf.applyPattern("MM/dd/yyyy");
    sdf.setTimeZone(TimeZone.getTimeZone("GMT"));
    //set the timezone explicitly otherwise it
    would take //the default locale and millisecond calculation can
    //be not as aspected
    dateVal = sdf.parse(strColDateVal);
 }
 row.addDate(dateVal, colIndex);
break;
}
 colIndex++;
}
chunk.addRow(row);
```

```
}
...
return chunk;
}
```

- cancel (int iIterId) – フレームワークは、このメソッドを呼び出して、openIterator() メソッドの呼び出し中に送信された ID によって識別されるデータ反復子に対して実行されている操作をプラグインがキャンセルできるようにします。
- closeIterator (int iIterId) – フレームワークは、このメソッドを呼び出して、openIterator() メソッドの呼び出し中に送信された ID によって識別されるデータ反復子に関してプラグインに通知します。

AbstractCustomDataProvider は、CustomDataProvider インターフェイスの抽象実装です。

4.6 CDSExtensionBaseDescriptor インターフェイス

CDSExtensionBaseDescriptor インターフェイスは、CDS Framework ユーザー インターフェイスおよびデータ プロバイダ ソースのエントリ ポイントで使用されるプラグインの詳細を取得するために使用されます。この情報は、Framework の以前のバージョンにある CDS Framework 拡張構成ファイルの置き換えに使用されます。

CDSExtensionBaseDescriptor インターフェイスのメソッドとサンプル実装を次に示します。

メソッド	サンプル実装
getName()	`public String getName(){ return "Catalog XML"; }`
getVersion()	`public String getVersion(){ return "1.0"; }`
getDataSourceType()	`public String getDataSourceType(){ return "XML"; }`

注

getName()、getVersion()、および getDataSourceType() メソッドから返される値は、DataProviderSource プラグイン実装と User Interface プラグイン実装で同じになる必要があります。getDataSourceType() の値は、プラグインごとに一意である必要があります。この値が重複していると、どちらのプラグインも実行時に使用できなくなります。

4.7 拡張タイプ

CDS Framework は、さまざまに考えられるプラグイン要件をサポートし、その制限内で、さまざまなユースケース シナリオをサポートします。

拡張は次のカテゴリーに大別できます。

- リソースベース拡張 – リソースベース拡張は、物理的なファイルの場所または URL パス (http URL など) のいずれかの形式で既存のデータ ソースの場所を含む拡張です。
- 非リソースベース拡張 – 非リソースベース拡張は、既存のデータ ソースの場所を含まない拡張です。非リソースベースの例としては、インメモリー データ構造があります。

現在、CDS Framework は次の拡張タイプをサポートします。

- CDSExtensionResourceType.FILE – これらの拡張は、リソースベース拡張のカテゴリーに属します。
- CDSExtensionResourceType.URL – これらの拡張は、リソースベース拡張のカテゴリーに属します。データ ソース URL がファイル プロトコルを使用している場合、拡張のリソース タイプは、URL ではなく、FILE にする必要があります。
- CDSExtensionResourceType.NONE – これらの拡張は、非リソースベース拡張のカテゴリーに属します。
- データ ソースにアクセスして Web Intelligence データ プロバイダの構築を支援するためにユーザー入力を必要とする拡張。これらの拡張は、前述の 3 つの拡張タイプのいずれかと組み合わせることができます。これらの拡張をリソースベース (ファイル/URL) 拡張と組み合わせる場合、ユーザー入力を必要としないという要件は、ユーザーからはソース情報以外の入力を要求しないという意味になります。

プラグインの設定

5.1 概要

プラグインを開発したら、そのプラグインを SAP BusinessObjects Web Intelligence Desktop ユーザーが使用できるように設定してデプロイする必要があります。このセクションでは、プラグインの設定方法について説明します。

5.2 プラグインの設定

User Interface インターフェイスと DataProviderSource インターフェイスを実装したら、User Interface および DataProviderSource CDS Framework にクラスとクラスパスの詳細を指定してプラグインを設定し、User Interface と DataProviderSource のバイナリを作成する必要があります。

プラグインを設定するには、次の手順を実行します。

1. プロジェクト ディレクトリの META-INF フォルダー内に、services フォルダーを作成します。

   注
   services フォルダーは、User Interface プロジェクトと DataProviderSource プロジェクトの両方で作成する必要があります。

2. User Interface プロジェクトの services フォルダーに、com.businessobjects.customds.CustomDSExtension という名前のファイルを作成します。
3. DataProviderSource プロジェクトの services フォルダーに、com.businessobjects.customds.CustomDataSource という名前のファイルを作成します。
4. com.businessobjects.customds.CustomDSExtension ファイルで、完全修飾の CustomDSExtension 実装クラス名を指定します。

   たとえば、com.businessobjects.catalog.CatalogPlugin となります。

5. com.businessobjects.customds.CustomDataSource ファイルで、完全修飾の CustomDataSource 実装クラス名を指定します。

   たとえば、com.businessobjects.catalog.ds.XMLCatalogDataSource となります。

6. MANIFEST.MF ファイルで MODULE-PATH 属性値を指定します。MODULE-PATH 属性は、プラグイン依存 jar のクラスパスを保持します。

   たとえば、MODULE-PATH: $(SHARED_CLASSES_DIR)PersonalDPPlugins/catalog_common.jar となります。

ここで、SHARED_CLASSES_DIR 変数は、<BOBJ_INST_DIR>¥SAP BusinessObjects Enterprise XI 4.0¥java¥lib¥ というパスを参照します。

注
依存 jar が複数ある場合は、jar の場所をセミコロン (;) で区切る必要があります。

7 User Interface と DataProviderSource の jar ファイルを作成します。

プラグインのデプロイ

User Interface と DataProviderSource の jar ファイルを作成したら、その jar ファイル (プラグイン) を <BOBJ_INST_DIR>¥SAP BusinessObjects Enterprise XI 4.0¥java¥lib¥PersonalDPPlugins の場所にデプロイして、SAP BusinessObjects Web Intelligence Desktop ユーザーがそのプラグインを使用できるようにする必要があります。

プラグインをデプロイすると、データ ソースから Web Intelligence (.wid) ドキュメントを作成できます。

プラグインのサンプル

CDS Framework のプラグイン開発が容易になるように、さまざまな拡張タイプのサンプルが用意されています。これらのサンプルは、<BOBJ_INST_DIR>/SAP BusinessObjects Enterprise XI 4.0/Samples/customds ディレクトリに置かれています。各サンプルには ReadMe の .zip ファイルが用意されています。以下のサンプルがあります。

- カタログ XML サンプル

  このサンプルは、リソース タイプとして FILE および URL を使用する拡張の実装を示します。

  このサンプルには、次の zip ファイルが含まれます。

  - catalogui.zip：このファイルには、User Interface 実装のバイナリ、ソースコード、およびこの拡張のデータ ソースとして使用されるサンプル XML が含まれます。
  - ds_Catalog.zip：このファイルには、DataProviderSource 実装のバイナリ、ソースコード、およびこの拡張のデータ ソースとして使用されるサンプル XML が含まれます。
  - catalog_common.zip：このファイルには、User Interface 実装および DataProviderSource 実装の共通クラスが含まれます。

  カタログ XML サンプルをデプロイするには、上記の zip ファイルを展開し、jar ファイルを <BOBJ_INST_DIR>¥SAP BusinessObjects Enterprise XI 4.0¥java¥lib¥PersonalDPPlugins ディレクトリにコピーします。

- カタログ XML – NoDSParams サンプル

  このサンプルは、リソース タイプとして FILE を使用する拡張の実装を示します。このサンプルは、データ ソースにアクセスするためにソース情報以外のユーザー入力を必要としません。

  このサンプルには、次の zip ファイルが含まれます。

  - catalogui_NoDSParams.zip：このファイルには、User Interface 実装のバイナリ、ソースコード、およびこの拡張のデータ ソースとして使用されるサンプル XML が含まれます。
  - ds_Catalog_NoDSParams.zip：このファイルには、DataProviderSource 実装のバイナリ、ソースコード、およびこの拡張のデータ ソースとして使用されるサンプル XML が含まれます。

  カタログ XML サンプルをデプロイするには、上記の zip ファイルを展開し、jar ファイルを <BOBJ_INST_DIR>¥SAP BusinessObjects Enterprise XI 4.0¥java¥lib¥PersonalDPPlugins ディレクトリにコピーします。

- カタログ XML – Incremental サンプル

  このサンプルは、リソース タイプとして FILE を使用してデータ ソースにインクリメンタル アクセスする拡張の実装を示します。

  このサンプルには、次の zip ファイルが含まれます。

  - catalogui_incremental.zip：このファイルには、User Interface 実装のバイナリ、ソースコード、およびこの拡張のデータ ソースとして使用されるサンプル XML が含まれます。

- ds_Catalog_Incremental.zip：このファイルには、DataProviderSource 実装のバイナリ、ソースコード、およびこの拡張のデータ ソースとして使用されるサンプル XML が含まれます。

カタログ XML サンプルをデプロイするには、上記の zip ファイルを展開し、jar ファイルを <BOBJ_INST_DIR>¥SAP BusinessObjects Enterprise XI 4.0¥java¥lib¥PersonalDPPlugins ディレクトリにコピーします。

- 従業員サンプル

このサンプルは、リソース タイプとして NONE を使用する拡張の実装を示します。

このサンプルには、次の zip ファイルが含まれます。

- resource_none_ui.zip：このファイルには、User Interface 実装のバイナリ、ソースコード、およびこの拡張のデータ ソースとして使用されるサンプル XML が含まれます。
- ds_Resource_None.zip：このファイルには、DataProviderSource 実装のバイナリ、ソースコード、およびこの拡張のデータ ソースとして使用されるサンプル XML が含まれます。

カタログ XML サンプルをデプロイするには、上記の zip ファイルを展開し、jar ファイルを <BOBJ_INST_DIR>¥SAP BusinessObjects Enterprise XI 4.0¥java¥lib¥PersonalDPPlugins ディレクトリにコピーします。

- 従業員 – No UI サンプル

このサンプルは、リソース タイプとして NONE を使用する拡張の実装を示します。また、データ ソースにアクセスして Web Intelligence データ プロバイダの構築を支援するためにユーザー入力を必要としません。

このサンプルには、次の zip ファイルが含まれます。

- resource_none_ui_NoDSParams.zip：このファイルには、User Interface 実装のバイナリ、ソースコード、およびこの拡張のデータ ソースとして使用されるサンプル XML が含まれます。
- ds_Resource_None_NoDSParams.zip：このファイルには、DataProviderSource 実装のバイナリ、ソースコード、およびこの拡張のデータ ソースとして使用されるサンプル XML が含まれます。

カタログ XML サンプルをデプロイするには、上記の zip ファイルを展開し、jar ファイルを <BOBJ_INST_DIR>¥SAP BusinessObjects Enterprise XI 4.0¥java¥lib¥PersonalDPPlugins ディレクトリにコピーします。

プラグイン開発時の考慮点

プラグインの開発時には、以下の点を考慮してください。

- 各プラグインは、一意のデータ ソース タイプを持つ必要があります。重複している場合は、どちらのプラグインも使用可能リストから削除されます。ユーザーにメッセージは表示されません。ただし、この情報はログファイルに記録されます。
- プラグインどうしで表示名が重複している場合、フレームワークは、ユーザーに使用可能なデータ ソースのリストを提供する際に、<プラグイン表示名>(<プラグイン名>_<プラグイン DS タイプ>_<プラグイン バージョン>) を表示して、この重複を解決できるようにします。
- 各プラグインは、サンドボックス環境で動作します。したがって、システム (SAP BusinessObjects Web Intelligence Desktop) バイナリを除いて、プラグインが依存するバイナリの場所と名前を指定する必要があります。
- データ ソースにアクセスして Web Intelligence データ プロバイダの構築を支援するために User Interface レイヤと処理レイヤの間で交換される情報は、名前と値のペアの形式にする必要があります。この名前と値は、null または空でない java.lang.String オブジェクトです。キーと値のペアに含まれる値が空の場合、フレームワークは、そのペアを保存せずにマップから削除します。
- パラメータを削除する必要がある場合、プラグインは、その値を空の文字列に設定してからフレームワークに渡す必要があります。
- CDS Framework のログ情報は、Tracelog ログ ファイルに記録されます。このファイルは、BO_trace.ini を使用して Web Intelligence Desktop のログを有効にすると生成されます。詳細については、SAP BusinessObjects Web Intelligence Desktop のドキュメントを参照してください。
- 同じ名前の列が複数存在する場合は、列名を表示する際にサフィックス (_1、_2 など) が追加されます。
- フレームワークは、データ ソース パラメータの暗号化と解読を行いません。DS パラメータの暗号化は、必要に応じてプラグインが実行する必要があります。
- 空の文字列も有効な値と見なされるデータ ソース パラメータがプラグインに含まれる場合は、プラグインの実装者が UI プラグイン実装と DPS プラグイン実装の間の動作を調整する必要があります。プラグイン実装は識別子を含むことがあります。これが見つかった場合は、プラグインによって空の文字列値と解釈されます。
- ユーザーがソース、URL、またはファイルを選択し、処理レイヤにエラーがない場合は、[次へ]ボタンがデフォルトで有効になります。フレームワークは、[次へ]ボタンを無効または有効にするためのコールバック メカニズムをプラグイン開発者に提供していません。
- フレームワークは、インクリメンタルおよび非インクリメンタルの両方のシナリオで、processMissingDSParameters() メソッドを呼び出します。つまり、フレームワークは、データ ソースにアクセスしてデータ プロバイダを構築するための要求を処理レイヤに行って得られた出力に、追加の情報セットが含まれても、無効または不足している入力に関する情報が含まれても、User Interface プラグインの同じメソッドを呼び出します。
- 一連の操作は、サンプルのフェッチ操作でも実際のデータのフェッチ操作でほとんど同じです。
- SAP BusinessObjects Web Intelligence Desktop の[クエリー パネル]でユーザーが行う[ソースの変更]操作は、互換性のあるデータ ソースに対して使用することを目的としています。既存のデータ ソース パラメータが新しいデータ ソースに対して不十分である場合、プラグインはデータ ソース パラメータ セットを再構築できません。そのため、この場合は例外が発生します。
- インターフェイスのロケールに関する情報には、国固有の情報が含まれない場合があります。

- 現在のバージョンでは、CDS Framework 拡張を使用して作成されたクエリーに対して、Web Intelligence プロンプト/クエリー フィルタは使用できません。
- 現在のバージョンでは、CDS Framework 拡張を使用して作成されたクエリーに対して、BI ラウンチパッドから Web Intelligence Rich Internet Application/Web Intelligence を使用してクエリーを編集することはサポートされていません。
- テキスト、Excel ファイルなどの個人用データ ソース、および Web サービスなどのカスタム データ ソースから作成された Web Intelligence ドキュメントの最新表示は、BI ラウンチパッドからサポートされます。ファイルベースのプラグインについては、Web Intelligence Desktop のオンライン ヘルプを参照してください。
- 現在のバージョンでは、CDS Framework 拡張を使用して作成されたクエリーに対して、オブジェクト階層の指定はサポートされません。
- 現在のバージョンでは、リソースタイプとして NONE を使用する拡張で編集操作を実行することはできません。
- CDS Framework は、CDS プラグインの User Interface のルック アンド フィールを制御しません。
- CDS Framework は、プラグインから提供されるエラー メッセージをローカライズできません。
- 現在のバージョンでは、CDS Framework は、プラグインがログを記録するために使用できるログ メカニズムを提供しません。必要に応じて、プラグインは独自のログ メカニズムを持つ必要があります。
- プラグインにカスタム データ ソース ファイル用の画像アイコンがある場合は、画像アイコンのパスをプラグインに渡す必要があります。そのパスに画像が見つからない場合、CDS Framework は、システム内で関連するアイコンを検索します。CDS Framework がシステム内で関連するアイコンを見つけられなかった場合、そのデータ ソースには Web Intelligence のデフォルト アイコンが表示されます。

プラグインをデバッグするには、次のようなレジストリ エントリを追加します。

[HKEY_LOCAL_MACHINE¥SOFTWARE¥SAP BusinessObjects¥Suite XI 4.0¥default¥WebIntelligence¥Rich Client¥JVMOptions]

サンプル レジストリ エントリを次に示します。

"3"="-Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=9100"

注

IDE でリモート デバッグ情報を作成する際は、このレジストリに示されているポート番号を使用する必要があります。

ユーティリティ クラス

データ ソースには複雑な構造が含まれる場合があり、それらをデータ ソース パラメータの入力として使用できる名前と値のペアに変換することは、非常に困難な作業になります。このような作業を支援するために、CDS Framework は、複雑な構造を入力として受け取り、それらを DS パラメータとして要求される名前と値のペアに変換するためのユーティリティを備えています。その機能のために使用されるクラスは、com.businessobjects.customds.utils.ParamsSerializer です。このユーティリティ クラスは、次の目的で使用できます。

- 以下の構造を持つ java.util.Map をシリアライズして、文字列 (非 null) キーと文字列値のペアを 1 つ含む java.util.Map に変換する。
- 文字列 (非 null) キーと文字列値のペアを 1 つ含むシリアライズされた java.util.Map (通常はシリアライズ呼び出しによって生成) をシリアライズ解除して、元の java.util.Map と同じ構造を持つマップに変換する。

このユーティリティの構造は次のようになります。

元の java.util.Map には、1 つの文字列 (非 null) から成る 1 つのキーが入ります。java.util.Map.Entry 値には、java.util.Map (任意のレベルまで。ただし、要素は最初のレベルの構造に準拠する必要がある)、文字列要素のみを含む java.util.List、文字列要素、または任意のレベルでこの 3 つの組み合わせが入ります。

9.1 ユーティリティ クラスの制限

ユーティリティ クラスの制限を次に示します。

- シリアライズ解除された java.util.Map には、いくつかの java.util.Map を表す (存在する場合) java.util.HashMap のインスタンスと、いくつかの List を表す (存在する場合) java.util.ArrayList のインスタンスが含まれます。
- シリアライズされた java.util.Map には、シリアライズ解除に使用される追加的な名前と値のペアがフレームワークによって挿入されている場合があります。
- シリアライズ解除された java.util.Map 内の要素の順序は、元の java.util.Map 内の要素の順序と一致しない場合があります。
- de-serialize メソッドの入力は、serialize メソッド呼び出しの出力から変更されていない java.util.Map である必要があります。したがって、シリアライズされたマップをシリアライズ解除する必要がある場合、シリアライズされたマップを変更することはできません。
- 特定のキーワードが ParamsSerializer クラスによって内部的に識別子として使用されます。現在のところ、シリアライズされるマップで使用されるキーに、これらのいずれかのキーワードが含まれている場合は、シリアライズ解除が正しく実行されない可能性があります。マップ エントリの値が空または null の場合にも同様の問題が発生する可能性があります。

前述のクラスの使用例を次に示します。

```
Map servicesMap = new HashMap();
Map portsMap1 = new HashMap();
Map portsMap2 = new HashMap();
Map methodsMap1 = new HashMap();
Map methodsMap2 = new HashMap();
servicesMap.put("QueryService",portsMap1);
servicesMap.put("ReportEngineService",portsMap2);
portsMap1.put("8080",methodsMap1);
portsMap2.put("9090",methodsMap2);
methodsMap1.put("Method1","InputMethod");
methodsMap1.put("Method1.2","OutputMethod");
methodsMap2.put("Method2", new HashMap().put(1, "$null$"));
System.out.println("Map is : - "+servicesMap);
ParamsSerializer paramSerialiser = new ParamsSerializer();
Map serialisedMap = paramSerializer.serialize(servicesMap);
System.out.println("Serialised Map is : "+serialisedMap);
Map deserialisedMap = paramSerialiser.deserialize(serialisedMap);
System.out.println("De Serialized Map is : "+deserialisedMap);
```

出力は次のようになります。

```
 Map is: - {ReportEngineService={9090={Method2=null}},
QueryService={8080={Method1.2=OutputMethod,
Method1=InputMethod}}}
Serialized Map is :
{QueryService#%pm*8080#%pm*Method1.2#%pv*=OutputMethod,
QueryService#%pm*8080#%pm*Method1#%pv*=InputMethod,
ReportEngineService#%pm*9090#%pm*Method2#%pv*=#%pnl*}
 De Serialized Map is :
{ReportEngineService={9090={Method2=null}},
QueryService={8080={Method1.2=OutputMethod,
Method1=InputMethod}}}
```

CDS Framework の制限

CDS Framework は、次の制限内でデータ ソースを使用します。

- 表形式 (列と行) でデータを提供できるデータ プロバイダ ソース拡張ポイント実装のみがサポートされます。
- CDS Framework は、データ ソース アクセスのための認証機能を追加するダイレクト/ビルトイン機能を備えていません。ただし、データ ソース パラメータを使用して間接的に認証機能を実現することができます。
- CDS Framework は、データ ソースへのウィザード ベースのアクセスをサポートしません。ただし、データ ソースへのインクリメンタル アクセスを実装することで、(完全なウィザードの機能はないが) ウィザードのようなワークフローをシミュレートできます。
- CDS Framework は、データ ソースのプラグインの自動生成をサポートしません。ただし、さまざまな拡張の実装を示すサンプルが用意されています。

お勧めのテクニック

プラグインの開発時に考慮するとよいベスト プラクティスを以下に示します。

- 技術的な制限はありませんが、将来の機能拡張に備えて、User Interface プラグイン クラス/インターフェイスと DPS プラグイン バイナリ (jar) に含まれるクラス/インターフェイスが相互に参照しないようにすることをお勧めします。
- DPS プラグイン実装と User Interface プラグイン実装は、個別の .jar ファイルにパッケージ化することをお勧めします。

より詳しい情報

情報リソース	場所
SAP BusinessObjects 製品情報	http://www.sap.com
SAP ヘルプ ポータル	http://help.sap.com/businessobjects/ へアクセスし、[SAP BusinessObjects Overview] サイドパネルから [All Products] をクリックします。 SAP ヘルプ ポータルでは、すべての SAP BusinessObjects 製品とそのデプロイメントについて扱った最新のドキュメンテーションにアクセスできます。PDF 版またはインストール可能な HTML ライブラリのダウンロードが可能です。 一部のガイドは SAP サービス マーケットプレイスに格納されており、SAP ヘルプ ポータルからは入手できません。ヘルプ ポータルのガイド一覧で、そのようなガイドには SAP サービス マーケットプレイスへのリンクが付いています。保守契約を締結されたお客様には、このサイトにアクセスするための正規ユーザー ID が付与されます。ID の入手方法については、お客様担当のカスタマー サポート担当者までお問い合わせください。
SAP サービス マーケットプレイス	http://service.sap.com/bosap-support > ドキュメンテーション ・ インストール ガイド:https://service.sap.com/bosap-instguides ・ リリース ノート:http://service.sap.com/releasenotes SAP サービス マーケットプレイスには、一部のインストール ガイド、アップグレードおよび移行ガイド、デプロイメント ガイド、リリース ノート、サポート対象プラットフォームに関するドキュメントが格納されています。保守契約を締結されたお客様には、このサイトにアクセスするための正規ユーザー ID が付与されます。ID の入手方法については、お客様担当のカスタマー サポート担当者までお問い合わせください。SAP ヘルプ ポータルから SAP サービス マーケットプレイスにリダイレクトされた場合は、左側のナビゲーション ペインのメニューを使用して、アクセスするドキュメンテーションが含まれているカテゴリを探します。
Docupedia	https://cw.sdn.sap.com/cw/community/docupedia Docupedia は追加のドキュメンテーションリソース、協調的なオーサリング環境、および対話型のフィードバックチャネルを提供します。

情報リソース	場所
開発者向けリソース	https://boc.sdn.sap.com/ https://www.sdn.sap.com/irj/sdn/businessobjects-sdklibrary
SAP Community Network 上の SAP BusinessObjects に関する記事	https://www.sdn.sap.com/irj/boc/businessobjects-articles これらの記事は、以前はテクニカル ペーパーという名称でした。
ノート	https://service.sap.com/notes これらのノートは、以前はナレッジ ベース記事という名称でした。
SAP Community Network 上のフォーラム	https://www.sdn.sap.com/irj/scn/forums
トレーニング	http://www.sap.com/services/education 弊社では、従来のクラス型の学習から目標を定めた eラーニング セミナーまで、学習ニーズや好みの学習スタイルに合わせたトレーニング パッケージを提供しています。
オンライン カスタマー サポート	http://service.sap.com/bosap-support SAP サポート ポータルには、カスタマー サポート プログラムとサービスに関する情報が含まれています。また、さまざまなテクニカル情報およびダウンロードへのリンクも用意されています。保守契約を締結されたお客様には、このサイトにアクセスするための正規ユーザー ID が付与されます。ID の入手方法については、お客様担当のカスタマー サポート担当者までお問い合わせください。
コンサルティング	http://www.sap.com/services/bysubject/businessobjectsconsulting コンサルタントは、初期の分析段階からデプロイメントプロジェクトの実現まで一貫したサポートを提供します。リレーショナル データベースと多次元データベース、接続、データベース設計ツール、カスタマイズされた埋め込みテクノロジなどのトピックに関する専門的なサポートを行います。

索引

C

CDS Framework
- DataProviderSource 9
- User Interface 9
- データ ソース 10
- データ プロバイダ 12

か

開発
- プラグイン 15

拡張タイプ
- NONE 24
- URL 24
- ファイル 24

カスタム データ ソース フレームワーク 7
- プラグイン 5

せ

制限 37
- ユーティリティ クラス 35

て

データ ソース
- URI の場所 7
- インメモリー 7
- ローカル ファイル 7

デプロイ
- サンプル 31

ふ

プラグイン
- MODULE-PATH 27

プラグイン (続き)
- 設定する 27
- デプロイ 29

プラグイン インターフェイス
- CDSExtensionBaseDescriptor 24
- CDSExtensionDescriptor 19
- CustomDataProvider 22
- CustomDataSource 20
- CustomDSComponent 18
- CustomDSExtension 16

へ

ベスト プラクティス 39

ゆ

ユーティリティ クラス 35