

SAP BusinessObjects Business Intelligence プラットフォーム
文書バージョン: 4.2 Support Package 02 – 2016-03-09

SAP Business Intelligence プラットフォームトランスレーシ ョンマネジメントツール (TMT) SDK の開発者ガイド

目次

1	ドキュメント履歴.....	4
2	このガイドについて.....	5
2.1	このガイドの対象読者.....	5
2.2	このガイドで扱う内容.....	5
3	技術的要件.....	7
4	TMT 開発者ガイド概要.....	8
4.1	システムダイアグラム.....	8
4.2	翻訳プロセス.....	9
5	TMT SDK 開発ワークフロー.....	10
6	TMT SDK の開発.....	11
6.1	TMT SDK 環境の設定.....	11
6.2	TMT SDK ワークフローの作成.....	16
	CMS から InfoObject をインポートするための実装コード.....	16
	ローカルフォルダから InfoObject をインポートするための実装コード.....	17
	InfoObject から翻訳を抽出するための実装コード.....	18
	トランスレーションマネージャエンジンを取得するための実装コード.....	18
	利用可能なロケール、表示可能なロケール、フォールバックロケールを取得および設定するための実装コード.....	19
	エンティティのプロパティ値とプロパティステータスを取得および設定するための実装コード.....	21
	トランスレーションマネージャドキュメント (.tmgr ファイル) をローカルフォルダに保存するための実装コード.....	22
	トランスレーションマネージャドキュメント (.tmgr ファイル) を CMS にエクスポートするための実装コード.....	23
	トランスレーションマネージャドキュメント (.tmgr ファイル) を XLIFF ファイルにエクスポートするための実装コード.....	23
	トランスレーションマネージャドキュメントを XLIFF ファイルからインポートするための実装コード.....	24
	トランスレーションマネージャドキュメントを XLS ファイルにエクスポートするための実装コード.....	24
	トランスレーションマネージャドキュメントを XLS ファイルからインポートするための実装コード.....	24
	サンプル実装コード.....	25
7	TMT SDK ワークフローのテスト.....	27
8	TMT SDK の使用.....	28

9	制限事項.....	29
10	トラブルシューティングのヒント.....	30
11	よくある質問.....	31

1 ドキュメント履歴

バージョン	日付	説明
SAP BusinessObjects Business Intelligence プラットフォーム 4.2	2015 年 11 月	新しくガイドがリリースされました。
SAP BusinessObjects Business Intelligence プラットフォーム 4.2 サポート パッケージ 2	2016 年 3 月	<ul style="list-style-type: none">コードスニペットが更新されました。CMS から InfoObject をインポートするための実装コード [16 ページ]コードスニペットが更新されました。サンプル実装コード [25 ページ]

2 このガイドについて

このガイドでは、トランスレーションマネジメントツール (TMT) SDK を使用してアプリケーションを開発する方法について説明します。以前は、トランスレーションマネジメントツールに保存された翻訳は、セマンティックレイヤ SDK で表示することも編集することもできませんでした。現在では、独自のアプリケーションの作成に役立つ機能を共有し、トランスレーションマネジメントツールに同じ内容を再入力することなく、変更を再統合できるようになっています。また、ユニバースを分析したり、翻訳の一覧を Excel 形式に抽出したりできる新しい機能も提供されています。つまり、TMT SDK を使用すると、複数の InfoObject を処理して翻訳を実行することができます。

2.1 このガイドの対象読者

このガイドは、翻訳を実行する独自のアプリケーションを作成できる開発者を対象としています。セマンティックレイヤ SDK とトランスレーションマネージャ SDK の両方を使用して、ラベルの翻訳をプログラムにより更新することができます。

例: 顧客やパートナーが翻訳プロセスを自動化する場合は、翻訳 API を使用するか、または XLIFF ファイルのインポートおよびエクスポートを自動化する API を作成して、自動化することができます。

2.2 このガイドで扱う内容

このガイドには以下の情報が掲載されています。

- トランスレーションマネジメントツール (TMT) SDK の概要
- TMT SDK のシステムダイアグラムおよび開発ワークフロー
- TMT SDK の作成、テスト、およびデプロイ
- ローカルフォルダからの InfoObject (.blx、.dfx、.unv) の読み取り方法
- Central Management Server (CMS) リポジトリからの InfoObject の読み取り方法
- InfoObject のプロパティ値およびプロパティステータスの変更
- ローカルフォルダまたは CMS リポジトリからインポートされた InfoObject の翻訳方法
- ローカルフォルダへの変更の保存
- CMS リポジトリへの変更のコミット
- TMT SDK のテスト
- XLIFF ファイルへのトランスレーションマネージャドキュメント (.tmgr) のエクスポートおよび XLIFF ファイルからの .tmgr の読み取り
- XLS ファイルへのトランスレーションマネージャドキュメント (.tmgr) のエクスポートおよび XLS ファイルからの .tmgr の読み取り
- 制限事項

-
- トラブルシューティングのヒント
 - よくある質問

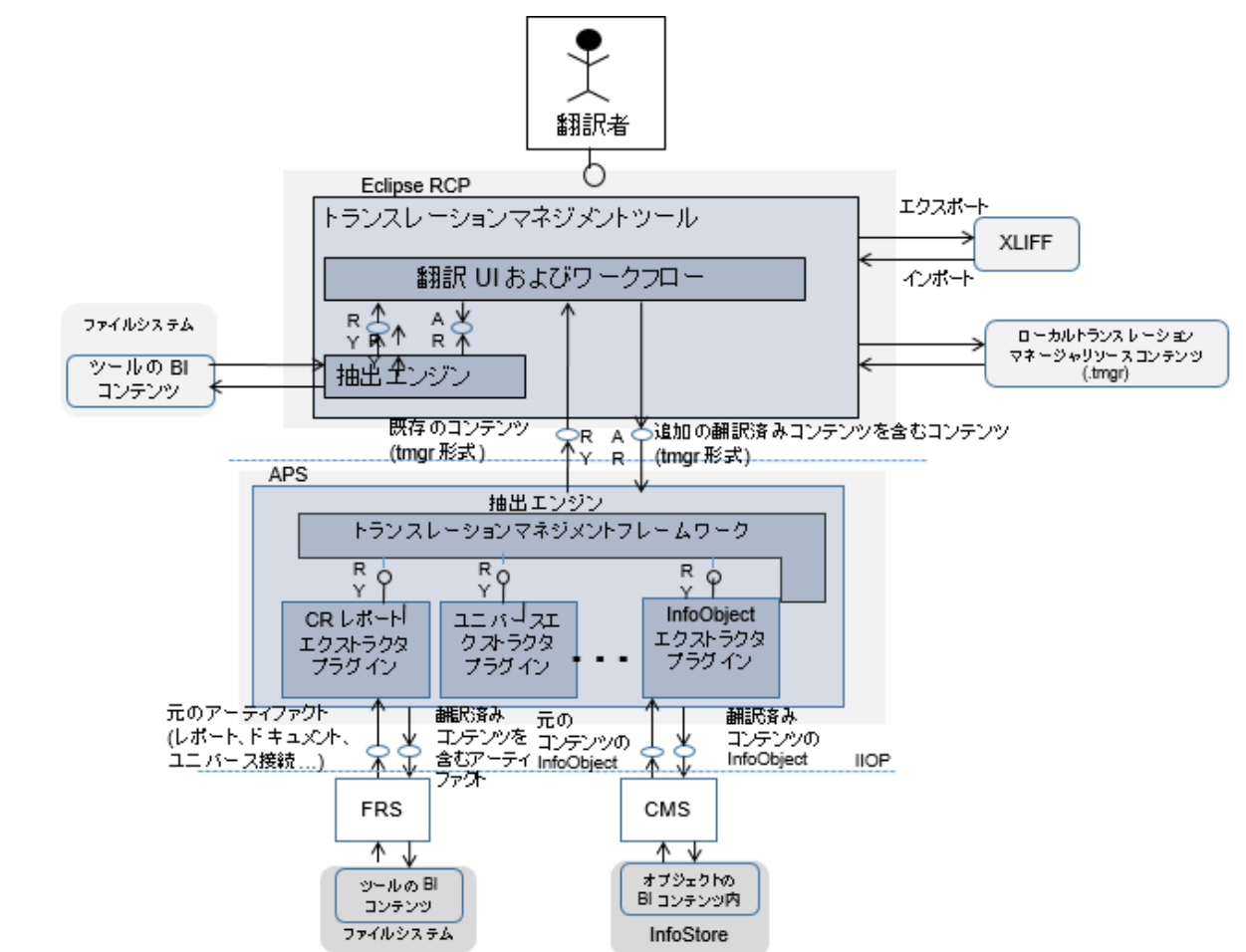
3 技術的要件

トランスレーションマネジメントツールおよび次の基本ワークフローを理解する必要があります。

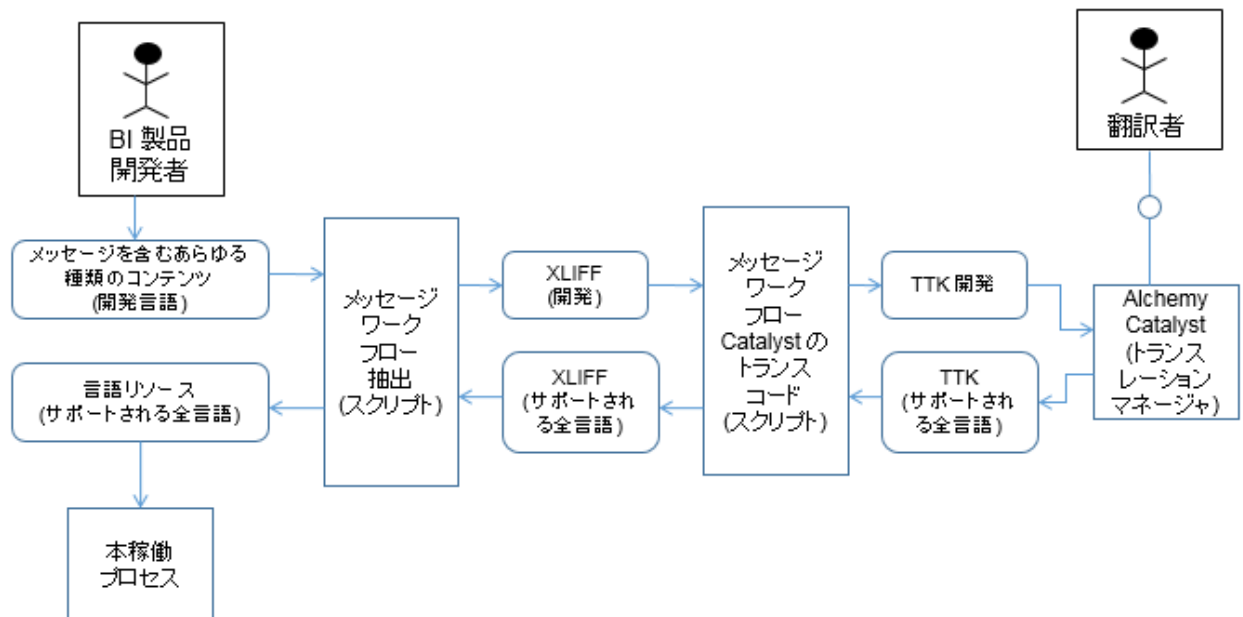
- InfoObject のインポート
- InfoObject の翻訳
- トランスレーションマネージャエンジンの取得
- InfoObject のトランスレーションマネージャドキュメント (.tmgr) としてのローカルフォルダへの保存
- CMS へのトランスレーションマネージャドキュメント (.tmgr) のコミット
- XLIFF ファイルへのトランスレーションマネージャドキュメント (.tmgr) のエクスポート

4 TMT 開発者ガイド概要

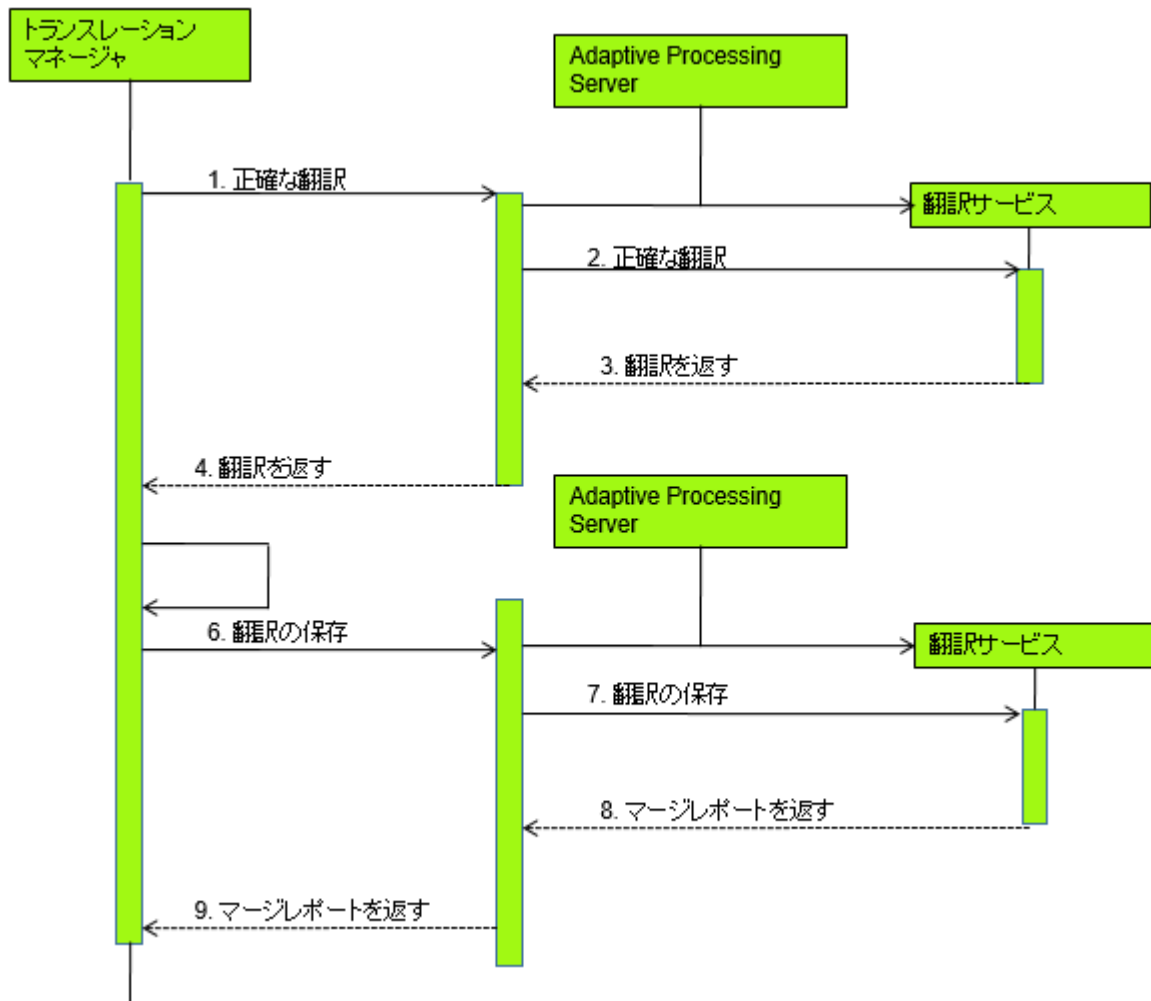
4.1 システムダイアグラム



4.2 翻訳プロセス



5 TMT SDK 開発ワークフロー



6 TMT SDK の開発

6.1 TMT SDK 環境の設定

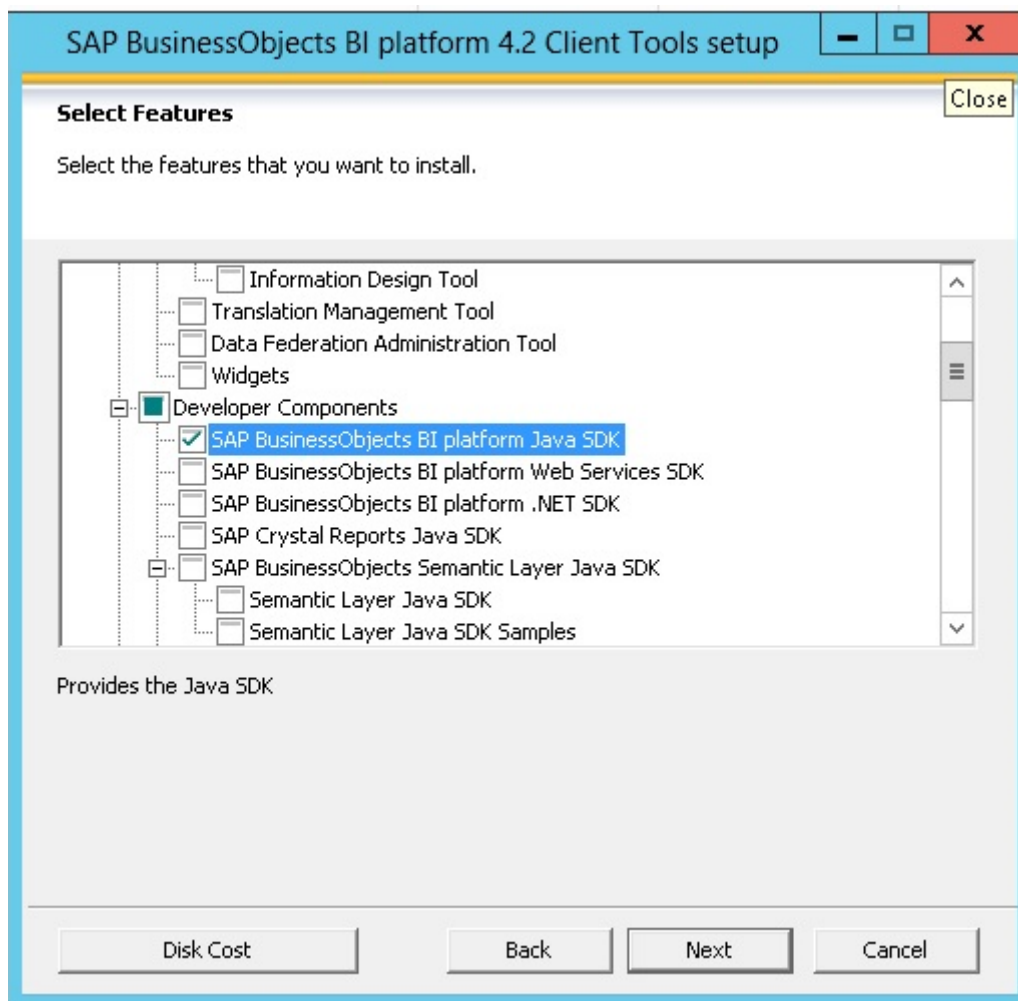
コンテキスト

TMT SDK 環境を設定する方法は 2 つあります。

方法 1:

手順

1. BOE クライアントのインストール中に、[SAP BusinessObjects BI プラットフォーム Java SDK] を選択します。以下のスクリーンショットを参照してください。



2. BOE クライアントのインストール後、次の場所にトランスレーションマネージャ SDK JARS が表示されます。

%BOINSTALLDIR%\java\lib\TranslationManagerSDK

%BOINSTALLDIR%\java\lib\TranslationManagerSDK に表示される JAR ファイル一覧は、次のとおりです。

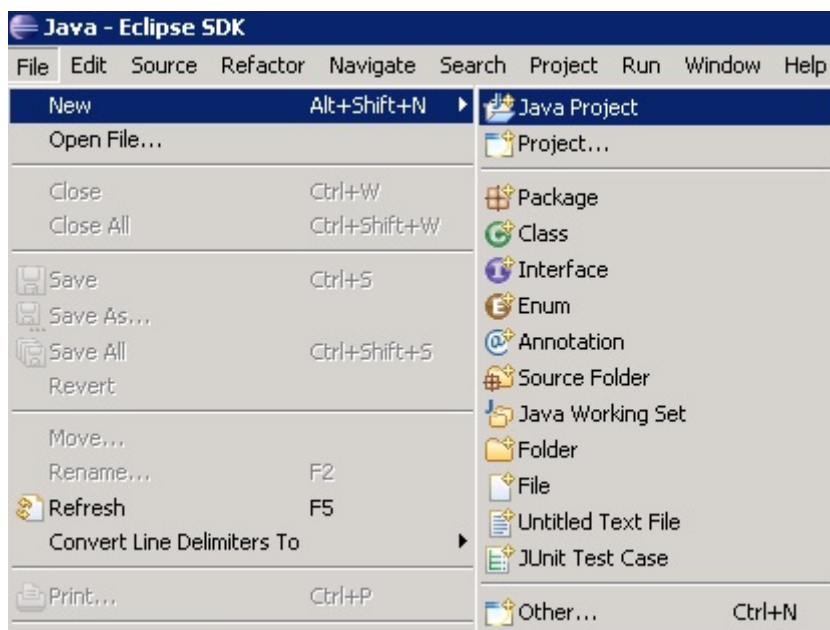
- aspectjrt.jar
- axiom-api-1.2.13.jar
- axiom-impl-1.2.13.jar
- axis2-adb-1.6.2.jar
- axis2-kernel-1.6.2.jar
- bcm.jar
- biplugins.jar
- ceaspect.jar
- cecore.jar
- celib.jar
- cesession.jar
- com.businessobjects.dsl.common.jar
- com.businessobjects.mds.connection.jar
- com.businessobjects.mds.datafoundation.jar
- com.businessobjects.mds.datafoundationsecurityaccess.jar

- com.businessobjects.mds.entity.jar
- com.businessobjects.mds.parameter.jar
- com.businessobjects.mds.repository.jar
- com.businessobjects.mds.resource.jar
- com.businessobjects.mds.toolkit.jar
- com.businessobjects.mds.universe.jar
- com.businessobjects.transmgr.unv.cms.jar
- com.businessobjects.transmgr.unv.jar
- com.businessobjects.transmgr.webi.jar
- com.businessobjects.xi.sdk.jar
- com.sap.translation.cms.client.core.jar
- com.sap.translation.core.jar
- com.sap.translation.sdk.jar
- com.sap.translation.unx.cms.jar
- com.sap.translation.unx.jar
- com.sap.translation.unx.ui.jar
- commons-logging.jar
- corbaidl.jar
- cryptojFIPS.jar
- ebus405.jar
- i18n4j.jar
- logging.jar
- neethi-3.0.2.jar
- org.eclipse.emf.common_2.4.0.v200902171115.jar
- org.eclipse.emf.ecore_2.4.2.v200902171115.jar
- org.xmlpull_1.1.3.4.jar
- poi-3.9-20121203.jar
- SL_plugins.jar
- TraceLog.jar
- translation-service-client.jar
- wicdzcorelib.jar
- wsdl4j-1.6.2.jar
- XmlSchema-1.4.7.jar
- com.sap.translation.sdk.jar

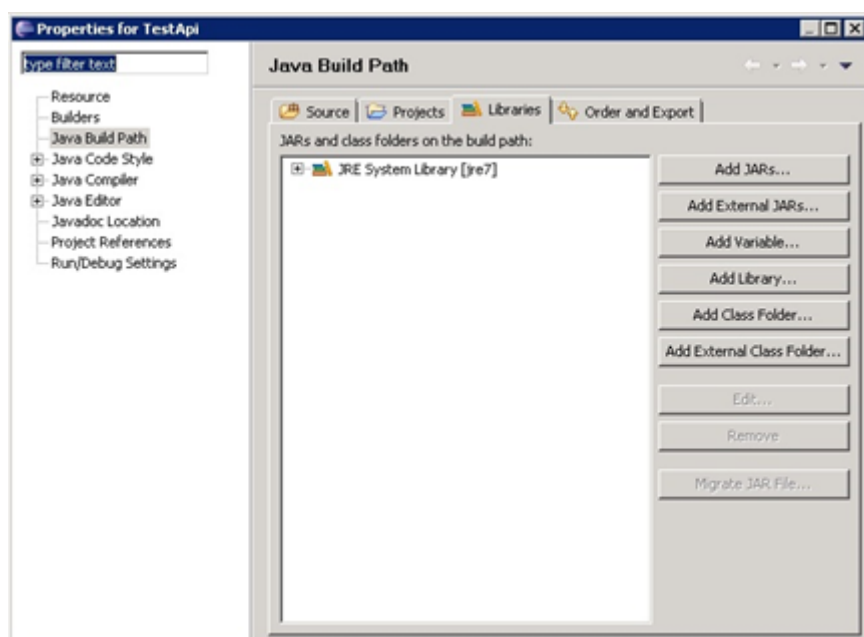
注記

- %BOINSTALLDIR% のデフォルトディレクトリは C:\Program Files (x86)\SAP BusinessObjects です。

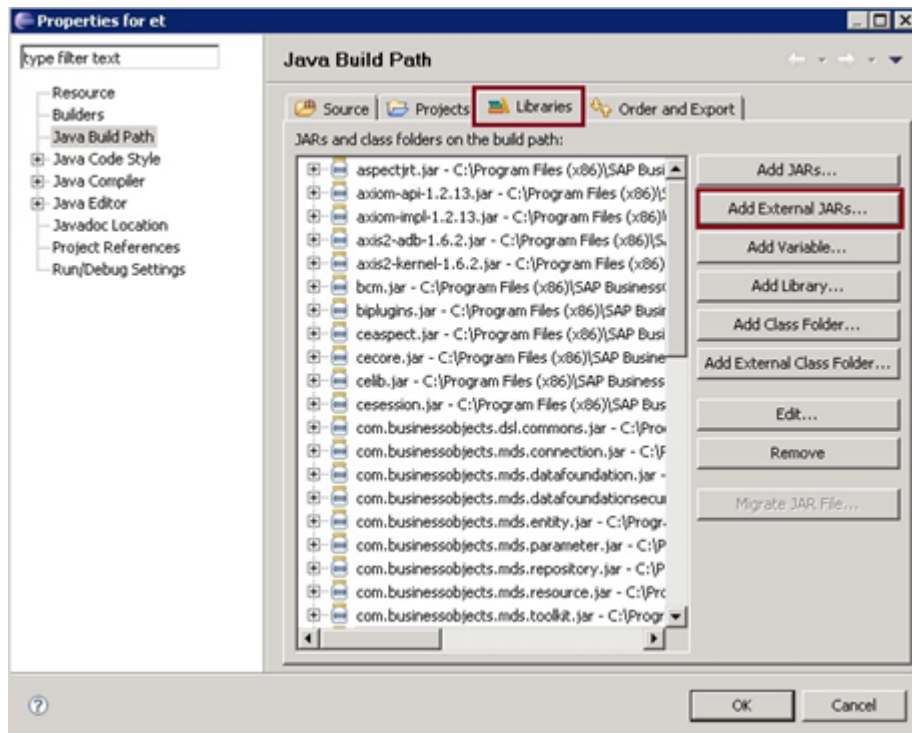
3. Eclipse を起動します (SDK は、Eclipse SDK 3.4.2 バージョンを使用してテストされます)。
4. [ファイル] > [新規作成] > [Java プロジェクト] を選択して、新しい Java プロジェクトを作成します。



5. [Java ビルドパス] で必要な JAR ファイルを追加します (図を参照)。



6. JAR ファイルを追加するには、[ライブラリ] を選択してから、[外部 JAR の追加] を選択し、`%BOINSTALLDIR%\¥java¥lib¥TranslationManagerSDK(C:¥Program Files (x86)¥SAP BusinessObjects¥java¥lib¥TranslationManagerSDK)` パスから、必要な JARS を選択します。次のスクリーンショットを参考にして、ビルドパスにライブラリを追加します。



7. [OK] をクリックします。
8. 環境の設定が正常に完了しました。これで、利用可能な API を使用して、独自のアプリケーションを構築することができます。TMT SDK の詳細については、*SAP Business Intelligent プラットフォームトランスレーションマネージャ SDK Javadoc* を参照してください。

方法 2:

1. 手順 1:
新しいバッチファイルを作成します。
2. 手順 2:
次の詳細を追加し、ファイルをバッチファイル (.bat) として保存します。

i 注記

これは Windows に対してのみ適用されます。

- set LIBHOME=%BOINSTALLDIR%¥java¥lib¥TranslationManagerSDK

i 注記

デフォルトでは、"C:¥Program Files (x86)¥SAP BusinessObjects" は "%BOINSTALLDIR%" です。

- set CLASSPATH=%LIBHOME%¥aspectjrt.jar;%LIBHOME%¥axiom-api-1.2.13.jar;%LIBHOME%¥axiom-impl-1.2.13.jar;%LIBHOME%¥axis2-adb-1.6.2.jar;%LIBHOME%¥axis2-kernel-1.6.2.jar;%LIBHOME%¥bcm.jar;%LIBHOME%¥biplugins.jar;%LIBHOME%¥ceaspect.jar;%LIBHOME%¥cecore.jar;%LIBHOME%¥celib.jar;%LIBHOME%¥cesession.jar;%LIBHOME%¥com.businessobjects.dsl.common.jar;%LIBHOME%¥com.businessobjects.mds.connection.jar;%LIBHOME%¥com.businessobjects.mds.datafoundation.jar;%LIBHOME%¥com.businessobjects.mds.datafoundationsecurityaccess.jar;%LIBHOME%

```

¥com.businessobjects.mds.entity.jar;%LIBHOME%¥com.businessobjects.mds.parameter.jar;
%LIBHOME%¥com.businessobjects.mds.repository.jar
;%LIBHOME%¥com.businessobjects.mds.resource.jar;%LIBHOME%
¥com.businessobjects.mds.toolkit.jar;%LIBHOME%¥com.businessobjects.mds.universe.jar;%LIBHOME
%¥com.businessobjects.transmgr.unv.cms.jar;%LIBHOME%¥com.businessobjects.transmgr.unv.jar;
%LIBHOME%¥com.businessobjects.transmgr.webi.jar
;%LIBHOME%¥com.businessobjects.xi.sdk.jar;%LIBHOME%¥com.sap.translation.cms.client.core.jar;
%LIBHOME%¥com.sap.translation.core.jar;%LIBHOME%¥com.sap.translation.sdk.jar;%LIBHOME%
¥com.sap.translation.unx.cms.jar;%LIBHOME%¥com.sap.translation.unx.jar;%LIBHOME%
¥com.sap.translation.unx.ui.jar
;%LIBHOME%¥commons-logging.jar;%LIBHOME%¥corbaidl.jar;%LIBHOME%¥cryptojFIPS.jar;
%LIBHOME%¥ebus405.jar;%LIBHOME%¥i18n4j.jar;%LIBHOME%¥logging.jar;%LIBHOME%
¥neethi-3.0.2.jar;%LIBHOME%¥org.eclipse.emf.common_2.4.0.v200902171115.jar;%LIBHOME%
¥org.eclipse.emf.ecore_2.4.2.v200902171115.jar;%LIBHOME%¥org.xmlpull_1.1.3.4.jar
;%LIBHOME%¥poi-3.9-20121203.jar;%LIBHOME%¥SL_plugins.jar;%LIBHOME%¥TraceLog.jar;
%LIBHOME%¥translation-service-client.jar;%LIBHOME%¥wicdzcorelib.jar;%LIBHOME%
¥wsdl4j-1.6.2.jar;%LIBHOME%¥XmlSchema-1.4.7.jar;%LIBHOME%¥com.sap.translation.sdk.jar
set PATH=C:¥Windows¥System32;%BOINSTALLDIR%;%PATH%
javac Tapi.java
java -cp C:¥TestApiTests -Djava.library.path="C:¥Windows¥System32;%BOINSTALLDIR%" Tapi

```

i 注記

- "Tapi.java" はサンプルの Java ファイルです。

6.2 TMT SDK ワークフローの作成

6.2.1 CMS から InfoObject をインポートするための実装コード

トランスレーションマネージャ SDK を使用して、複数または単一の InfoObject をインポートすることができます。

CMS から InfoObject をインポートするためのコードスニペットは、次のとおりです。

≡ サンプルコード

```

String strQuery=null;
ISessionMgr sessionManager = CrystalEnterprise.getSessionMgr();
ceEnterpriseSession =
sessionManager.logon("Administrator","Password1","10.160.209.188:6400","secEnterpr
ise");
ceEnterpriseSession.setLocale(Locale.ENGLISH);
IInfostore infostore = (IInfoStore) ceEnterpriseSession.getService("InfoStore");
strQuery = "select * from ci_infoobjects where si_name='" + docname + "'" +
"and SI_PARENT_FOLDER='" + parentfolderid + "'";

```



```

IInfoObjects ceInfoObjects = (IInfoObjects) infostore.query(strQuery);
if (ceInfoObjects.size() != 0)
infoObj = (IInfoObject) ceInfoObjects.get(0);
else
//May be an application object
{
strQuery="select * from ci_appobjects where si_name='" + docname + "'" + "and
SI_PARENT_FOLDER='" + parentfolderid + "'";
ceInfoObjects = (IInfoObjects) infostore.query(strQuery);
if (ceInfoObjects.size() != 0)
infoObj = (IInfoObject) ceInfoObjects.get(0);
}
System.out.println("Importing " + infoObj.getTitle() + " " + infoObj.getKind()
+ " from CMS");
catch(Exception e)
{
return null;
}
}

```

6.2.2 ローカルフォルダから InfoObject をインポートするための実装コード

トランスレーションマネージャ SDK を使用して、データファンデーション (.dfx)、ビジネスレイヤ (.blx)、およびユニバースドキュメント (.unv) をローカルフォルダからインポートし、トランスレーションマネージャドキュメント (.tmgr) をインポートして翻訳を実行することができます。

.unv、.dfx、.blx、.tmgr をインポートするためのコードスニペット:

サンプルコード

```

//Importing the .unv file from local folder
File filepath = new File("C:\\¥¥Universes\\¥¥Univers2.unv");
TranslationSDKManager manager = new TranslationSDKManager();
ITMgrDocument tmgrdoc =
manager.loadArtifact(TranslatableEntity.UNIVERSE,filepath);
ITMgrEngine engine = ITMgrEngine.Factory.createInstance(tmgrdoc);

```

サンプルコード

```

//Importing the .dfx file from local folder
File filepath = new File("C:\\¥¥Universes\\¥¥NConnection.dfx");
TranslationSDKManager manager = new TranslationSDKManager();
ITMgrDocument tmgrdoc =
manager.loadArtifact(TranslatableEntity.DATAFOUNDATION,filepath);
ITMgrEngine engine = ITMgrEngine.Factory.createInstance(tmgrdoc);

```

サンプルコード

```

// Importing .blx from the local folder
File filepath = new File("C:\\¥¥Universes\\¥¥NConnection.blx");
TranslationSDKManager manager = new TranslationSDKManager();
ITMgrDocument tmgrdoc =
manager.loadArtifact(TranslatableEntity.UNIVERSE,filepath);

```

```
ITMgrEngine engine = ITMgrEngine.Factory.createInstance(tmgrdoc);
```

サンプルコード

```
//Importing .tmgr from the local folder
File filepath = new File("C:\\Universe\\NConnection.tmgr");
TranslationSDKManager manager = new TranslationSDKManager();
ITMgrDocument tmgrdoc =
manager.loadArtifact(TranslatableEntity.TRANSMGR, filepath);
ITMgrEngine engine = ITMgrEngine.Factory.createInstance(tmgrdoc);
```

6.2.3 InfoObject から翻訳を抽出するための実装コード

次のコードスニペットを使用して、InfoObject からメタデータ翻訳を抽出します。

サンプルコード

```
TranslationSDKManager manager = new TranslationSDKManager(ceEnterpriseSession);
InputStream translations = manager.extractTranslations(infoObj.getCUID());
return translations;
```

6.2.4 トランスレーションマネージャエンジンを取得するための実装コード

InfoObject からメタデータ翻訳を取得した後、最初にトランスレーションマネージャドキュメントを取得し、ドキュメントからトランスレーションマネージャエンジンを取得する必要があります。次のコードスニペットを使用して、この操作を実行します。

サンプルコード

```
Public static ITMgrDocument GetTmgrDoc (InputStream inputstream, IInfoObject
infoobj) throws TranslationException
{
try
{
ITMgrDocument tmgrdoc = null;
System.out.println("Saving as " + infoobj.getTitle() + ".tmgr file under C:\\sdk\\
folder\\");
FileOutputStream os = new FileOutputStream("C:\\sdk\\" + infoobj.getTitle() +
".tmgr");

byte[] buffer = new byte[1024];
int bytesRead;
//read from is to buffer
while((bytesRead = inputstream.read(buffer)) !=-1){
os.write(buffer, 0, bytesRead);
}
//flush OutputStream to write any buffered data to file
os.flush();
os.close();
//Gets the Translation Manager Document
```

```

return manager.loadArtifact(TranslatableEntity.TRANSMGR,new File("C:¥¥sdk¥
¥"+infoobj.getTitle() + ".tmgr"));
}
catch(FileNotFoundException
{
e.printStackTrace();
return null;
}
catch(IOException e1)
{
e1.printStackTrace();
return null;
}
}
//Gets the Translation Manager Engine
ITMgrEngine
engine=TMgrEngine.Factory.createInstance(GetTmgrDoc(translations,infoObj));

```

6.2.5 利用可能なロケール、表示可能なロケール、フォールバックロケールを取得および設定するための実装コード

6.2.5.1 利用可能なロケールの設定および取得

TMT SDK を使用して、利用可能なロケールから単一または複数のロケールを追加することができます。

例:

サンプルコード

```

//Adds the Available Locale
engine.addAvailableLocale(Locale.US);
engine.getAvailableLocales(); //Getting the available locales

```

6.2.5.2 翻訳されたロケール値でのトランスレーションエンジンの設定

翻訳されたロケールの値でトランスレーションエンジンを設定することができます。

例:

サンプルコード

```

//Sets the Translation Engine with translated locale value
engine.setModifiedLocale(Locale.US);

```

6.2.5.3 利用可能なロケールの削除

TMT SDK を使用して、単一または複数のロケールを削除することができます。

例:

サンプルコード

```
//Removes the Available Locale  
engine.removeAvailableLocale(Locale.US);
```

6.2.5.4 表示可能なロケールの設定および取得

TMT SDK を使用すると、特定のドキュメントに対して、利用可能なロケールの一覧に含まれる任意のロケールを表示可能に設定することができます。

例:

サンプルコード

```
//Sets the locale as visible  
engine.setLocaleVisible(Locale.US,true);  
  
//Gets the visible locales  
engine.getVisibleLocales();
```

6.2.5.5 フォールバックロケールの設定および取得

TMT SDK を使用すると、特定のドキュメントに対して、利用可能なロケールの一覧に含まれる任意のロケールをフォールバックとして設定することができます。

注記

単一のロケールを追加した場合、フォールバックロケールはデフォルトで元のコンテンツ言語になります。

例:

サンプルコード

```
//Sets the locale as fallback
engine.setFallbackLocale(Locale.US,true); //Setting the fallback locale
//Gets the fallback locale
engine.getFallbackLocale();
```

6.2.6 エンティティのプロパティ値とプロパティステータスを取得および設定するための実装コード

トランスレーションマネージャ SDK を使用して、トランスレーションマネージャドキュメント (.tmgr) 内のエンティティのプロパティ値とプロパティステータスを取得および設定することができます。

次のコードスニペットを使用して、トランスレーションマネージャドキュメント (.tmgr) ファイルのルートエンティティのプロパティ値を取得および設定することができます。

サンプルコード

```
System.out.println("--MODIFYING THE VALUES--WITH--"+name+"---AND--"+description);
Locale locales[] = FormLocales();
Object rootEntity = engine.getRoot();
IPropertyInfo[] propertyInfo = engine.getLocalizedProperties(rootEntity);
if(propertyInfo.length == 0)
{
    Object[] obj = engine.getChildren(rootEntity);
    if(obj.length > 0) //this means it is only rootentity
    {
        for(int index=0;index < obj.length;index++)
        {
            IPropertyInfo[] propertyInfo = engine.getLocalizedProperties(obj[index]);
            for(int index1=0; index1
                < locales.length; index1++)
            {
                System.out.println("Id is -->" + propertyInfo[0].getId());
                System.out.println("Description is -->" + propertyInfo[1].getId());
                System.out.println("Locales index is -->" +
                    locales[index1].getDisplayLanguage());
                propertyInfo[0].setValue(obj[index], locales[index1], name
                    +locales[index1].getCountry());
                propertyInfo[1].setValue(obj[index], locales[index1], description
                    +locales[index1].getCountry());
            }
        }
    }
}
else
{
    IPropertyInfo[] propertyInfo = engine.getLocalizedProperties(rootEntity);
    for(int index1=0; index1 < locales.length; index1++)
    {
        propertyInfo[0].setValue(rootEntity, locales[index1], name
            +locales[index1].getCountry());

        propertyInfo[1].setValue(rootEntity, locales[index1], description
            +locales[index1].getCountry());
    }
}
```

```
}
```

次のコードスニペットを使用して、トランスレーションマネージャドキュメント (.tmgr) のルートエンティティのプロパティステータスを取得および設定することができます。

サンプルコード

```
System.out.println("--MODIFYING THE STATUS--WITH--"+status.name());
Locale locales[] = {Locale.US, Locale.FRANCE};
Object rootEntity = engine.getRoot();
IPropertyInfo[] propertyInfoRoot = engine.getLocalizedProperties(rootEntity);
if(propertyInfoRoot.length==0)
{
    Object[] obj = engine.getChildren(rootEntity);
    if(obj.length > 0) //this means it is only rootentity
    {
        for(int index=0;index < obj.length;index++)
        {
            IPropertyInfo[] propertyInfo = engine.getLocalizedProperties(obj[index]);
            for(int index1=0; index1 < locales.length; index1++)
            {
                propertyInfo[0].setStatus(obj[index], locales[index1], status);
                propertyInfo[1].setStatus(obj[index], locales[index1], status);
            }
        }
    }
}
else
{
    IPropertyInfo[] propertyInfo = engine.getLocalizedProperties(rootEntity);
    for(int index1=0; index1 < locales.length; index1++)
    {
        propertyInfo[0].setStatus(rootEntity, locales[index1], status);
        propertyInfo[1].setStatus(rootEntity, locales[index1], status);
    }
}
```

6.2.7 トランスレーションマネージャドキュメント (.tmgr ファイル) をローカルフォルダに保存するための実装コード

次のコードスニペットを使用して、トランスレーションマネージャドキュメント (.tmgr) ファイルをローカルに保存し、ファイルを開き、翻訳を実行することができます。

サンプルコード

```
//Saves the translation manager document(.tmgr).
engine.save();
```

6.2.8 トランスレーションマネージャドキュメント (.tmgr ファイル) を CMS にエクスポートするための実装コード

次のコードスニペットを使用して、翻訳実行後に、InfoObject をシステムリポジトリにエクスポートすることができます。

サンプルコード

```
try
{
engine.save();
ByteArrayOutputStream tmgrStream = new ByteArrayOutputStream();
FileDocumentFactory.exportToTMgr(engine,tmgrStream);
InputStream saveResult =
manager.saveTranslations(infoObj.getCUID(),tmgrStream,true);
System.out.println("InfoObject cuid is " + infoObj.getCUID());
TMgrMergeReport mergeResult = new TMgrMergeReport(saveResult);
if(mergeResult.hasConflict())
{
System.out.println("Conflict has occurred");
}
else
{
System.out.println("Translations are saved");
}
catch(Exception e)
{
e.printStackTrace();
}
```

6.2.9 トランスレーションマネージャドキュメント (.tmgr ファイル) を XLIFF ファイルにエクスポートするための実装コード

次のコードスニペットを使用して、トランスレーションマネージャドキュメントを XLIFF ファイルにエクスポートすることができます。

サンプルコード

```
try
{
IEntityInfo info = engine.getEntityInfo(engine.getRoot());
IXliffExporter exportxliff = manager.createXliffExporter();
exportxliff.setSourceLocale(Locale.US,true);
exportxliff.setTargetLocale(Locale.ITALY,true);
File path = new File("C:¥¥sdk¥¥sundar.xliff");
exportxliff.write(info,path);
}
catch(TranslationException ex)
{
ex.getMessage();
}
```

6.2.10 トランスレーションマネージャドキュメントを XLIFF ファイルからインポートするための実装コード

次のコードスニペットを使用して、トランスレーションマネージャドキュメントを XLIFF ファイルからインポートすることができます。

サンプルコード

```
IXliffImporter importxliff = manager.createXliffImporter();
File path3 = new File("C:¥¥sdk¥¥sundar.xliff");
File path2 = new File("C:¥¥sdk¥¥POC4.tmgr");

ITMgrDocument tmgrdocim=manager.loadArtifact(TranslatableEntity.TRANSMGR,path2);
TMgrEngine engine2 = new TMgrEngine(tmgrdocim);
engine2.save();
importxliff.load(engine2,path3);
```

6.2.11 トランスレーションマネージャドキュメントを XLS ファイルにエクスポートするための実装コード

次のコードスニペットを使用して、トランスレーションマネージャドキュメントを XLS ファイルにエクスポートすることができます。

注記

Excel (.xlsx) 形式はサポートされていません。

サンプルコード

```
IExcelExporter exportexcel = manager.createExcelExporter();
path = new File("C:¥¥sdk¥¥sundar.xls");
Locale loc[] = { Locale.JAPAN,Locale.ITALY};
exportexcel.setSourceLocale(Locale.US);
exportexcel.setTargetLocale(loc);
exportexcel.write(engine, path);
```

6.2.12 トランスレーションマネージャドキュメントを XLS ファイルからインポートするための実装コード

次のコードスニペットを使用して、トランスレーションマネージャドキュメントを Excel (.xls) ファイルからインポートすることができます。

サンプルコード

```
IExcelImporter importexcel = manager.createExcelImporter();
importexcel.load(engine,path,new File("C:¥¥sdk¥¥excel.properties"));
engine.saveAs(new File("C:¥¥sdk¥¥POCXLS.tmgr"));
```

6.2.13 サンプル実装コード

CMS からの InfoObject の抽出、メタデータ翻訳の取得、利用可能なロケール/フォールバックロケール/表示可能なロケールの設定、プロパティステータスの設定、エンジンの保存および CMS へのエクスポートを実行するサンプル実装コード

サンプルコード

```
String strQuery=null;
ISessionMgr sessionManager = CrystalEnterprise.getSessionMgr();
ceEnterpriseSession =
sessionManager.logon("Administrator","Password123","localhost:
6400","secEnterprise");
ceEnterpriseSession.setLocale(Locale.ENGLISH);
IInfostore infostore = (IInfoStore) ceEnterpriseSession.getService("InfoStore");
strQuery = "select * from ci_infoobjects where si_name='" + docname + "'" + "and
SI_PARENT_FOLDER='" + parentfolderid + "'";

IInfoObjects ceInfoObjects = (IInfoObjects) infostore.query(strQuery);
if (ceInfoObjects.size() != 0)
infoObj = (IInfoObject) ceInfoObjects.get(0);
else //May be an application object
{
strQuery="select * from ci_appobjects where si_name='" + docname + "'" + "and
SI_PARENT_FOLDER='" + parentfolderid + "'";
ceInfoObjects = (IInfoObjects) infostore.query(strQuery);
if (ceInfoObjects.size() != 0)
infoObj = (IInfoObject) ceInfoObjects.get(0);
}
System.out.println("Importing " + infoObj.getTitle() + " " + infoObj.getKind()
+ " from CMS");

manager = new TranslationSDKManager(ceEnterpriseSession);
InputStream translations =
manager.extractTranslations(infoObj.getCUID());
ITMgrEngine engine =
ITMgrEngine.Factory.createInstance(GetTmgrDoc(translations,infoObj));
engine.removeAvailableLocale(Locale.US);
engine.addAvailableLocale(Locale.US);
engine.setFallbackLocale(Locale.US);
engine.setVisibleLocale(Locale.US,true);
IPropertyInfo[] propertyInfo = engine.getLocalizedProperties(rootEntity);
propertyInfo[0].setStatus(rootEntity,Locale.US,status);
engine.setModifiedLocale(Locale.US);
engine.save();
ByteArrayOutputStream tmgrStream = new ByteArrayOutputStream();

FileDocumentFactory.exportToTmgr(engine,tmgrStream);

InputStream saveResult =
manager.saveTranslations(infoObj.getCUID(),tmgrStream,true);

System.out.println("InfoObject cuid is " + infoObj.getCUID());
```

```
TMgrMergeReport mergeResult = new TMgrMergeReport(saveResult);  
if (mergeResult.hasConflict())  
{  
    System.out.println("Conflict has occurred");  
}  
else  
{  
    System.out.println("Translations are saved");  
}
```

7 TMT SDK ワークフローのテスト

テストを行う前に、次の要件を確認する必要があります。

1. SAP Business Intelligence Server がインストールされ (ローカルまたはリモート)、CMS が実行されている必要があります。
2. Adaptive Processing Server で翻訳サービスが実行されている必要があります。
3. 翻訳サービス JAR は、`%BOINSTALLDIR%\SAP BusinessObjects Enterprise XI 4.0\java\pjs\services\TranslationService\lib path` にあります。

i 注記

- クライアントおよびサーバのインストールが利用可能である必要があります (サーバはローカルまたはリモートにインストールすることができます)。
- Adaptive Processing Server (APS) がサーバマシン内で翻訳サービスをホストし、翻訳に必要な対応するプラグインもサーバに存在することを確認します。

8 TMT SDK の使用

アプリケーションの開発およびテストが完了したので、このアプリケーションを使用して次のワークフローを実行することができます。

- CMS からの InfoObject のインポート: [CMS から InfoObject をインポートするための実装コード \[16 ページ\]](#)
- ローカルフォルダからの InfoObject のインポート: [ローカルフォルダから InfoObject をインポートするための実装コード \[17 ページ\]](#)
- メタデータ翻訳の抽出: [InfoObject から翻訳を抽出するための実装コード \[18 ページ\]](#)
- トランスレーションマネージャエンジンの取得: [トランスレーションマネージャエンジンを取得するための実装コード \[18 ページ\]](#)
- 利用可能なロケールの設定および取得: [利用可能なロケールの設定および取得 \[19 ページ\]](#)
- 翻訳されたロケール値でのトランスレーションエンジンの設定: [翻訳されたロケール値でのトランスレーションエンジンの設定 \[19 ページ\]](#)
- 表示可能なロケールの設定および取得: [表示可能なロケールの設定および取得 \[20 ページ\]](#)
- フォールバックロケールの設定および取得: [フォールバックロケールの設定および取得 \[20 ページ\]](#)
- プロパティ値とプロパティステータスの設定および取得: [エンティティのプロパティ値とプロパティステータスを取得および設定するための実装コード \[21 ページ\]](#)
- トランスレーションマネージャドキュメント (.tmgr) ファイルの保存: [トランスレーションマネージャドキュメント \(.tmgr ファイル\) をローカルフォルダに保存するための実装コード \[22 ページ\]](#)
- XLIFF ファイルへのエクスポート: [トランスレーションマネージャドキュメント \(.tmgr ファイル\) を XLIFF ファイルにエクスポートするための実装コード \[23 ページ\]](#)
- XLIFF ファイルからのインポート: [トランスレーションマネージャドキュメントを XLIFF ファイルからインポートするための実装コード \[24 ページ\]](#)
- XLS ファイルへのエクスポート: [トランスレーションマネージャドキュメントを XLS ファイルにエクスポートするための実装コード \[24 ページ\]](#)
- XLS ファイルからのインポート: [トランスレーションマネージャドキュメントを XLS ファイルからインポートするための実装コード \[24 ページ\]](#)
- CMS への翻訳のエクスポート: [トランスレーションマネージャドキュメント \(.tmgr ファイル\) を CMS にエクスポートするための実装コード \[23 ページ\]](#)

9 制限事項

TMT SDK の制限事項は次のとおりです。

- トランスレーションマネージャ SDK を使用して、ローカルフォルダから Web Intelligence ファイル (.wid) を開き、翻訳を実行することはできません。
- トランスレーションマネージャドキュメント (.tmgr) を XLS ファイルにエクスポートする場合、プロパティ値は XLS ファイルに保存されますが、プロパティステータスは XLS ファイルに保存されません。
- SDK を使用してロケールが追加された後に、同じロケールを追加しようとしても、現時点では例外はスローされません。ただし、このような特定の状況でエラーがスローされるように SDK コードをカスタマイズすることができます。
- SDK を使用する場合、まだ追加されていないロケールを削除しようとしても例外はスローされません。ただし、以下の SDK コードを使用して、このような状況を回避することができます。

サンプルコード

```
Locale loc[] = engine.getAvailableLocales();
Locale toremove = Locale.US;
for(int index = 0; index < loc.length; index++)
{
    If (toremove.getDisplayName().compareTo(loc[index].getDisplayName() ) == 0)
        Engine.removeLocale(toremove);
    else
        Throw TranslationException(e);
}
```

10 トラブルシューティングのヒント

- 問題:
トランスレーションマネージャ SDK コードの記述時に、`ClassNotFoundException` がスローされました。
アクション:
`<CLASSPATH>` に `%BOINSTALLDIR%\java\lib\TranslationManagerSDK` が設定され、その下に必要なすべての JAR ファイルが追加されていることを確認します。
- 問題:
Central Management Server (CMS) からの `InfoObject` のインポート中に、例外がスローされます。
アクション:
CMS が起動および実行され、翻訳サービスが Adaptive Processing Server (APS) に対して利用可能であることを確認します。
- 問題:
`engine.save ()` の使用時に、トランスレーションマネージャドキュメント (`.tmgr` ファイル) の保存場所を見つけられません。
アクション:
デフォルトでは、トランスレーションマネージャドキュメントは `C:\Users\Administrator` パスの下に保存されます。
- 問題:
トランスレーションマネージャドキュメント (`.tmgr` ファイル) が破損しています。SDK を介してトランスレーションマネージャドキュメント (`.tmgr` ファイル) を開くことができません。
アクション:
利用可能なすべてのロケールを削除し、再度追加してから、翻訳を実行します。
- 問題:
ワークフローの実行中にエラーが発生しました。
アクション:
トランスレーションマネジメントツール (TMT) を使用して同じワークフローを実行し、TMT で正常に動作するかどうかを確認します。
- 問題:
`NullPointerException` 例外が発生しました。
アクション:
CMS の `InfoObject` が正常にインポートされていることを確認した後、続いて `InfoObject` からメタデータを取得します。

11 よくある質問

1. すべての JAR ファイルがトランスレーションマネージャ SDK で利用可能であることを確認するには、どうすればよいですか。
Business Objects クライアントをインストールしたら、[\[PlatformJavaSDK\]](#) をクリックし、関連するすべての JAR ファイルが `%BOINSTALLDIR%\java\lib\TranslationManagerSDK` に存在することを確認します。JAR ファイルの一覧は、このガイドの [TMT SDK 環境の設定の節](#)を参照してください。
2. *Adaptive Processing Server* の翻訳サービスが稼働中であることを確認するには、どうすればよいですか。
Business Objects サーバをローカルまたはリモートでインストールしたら、以下の操作を実行して、翻訳サービスが稼働中かどうかをチェックします。
 1. セントラル管理コンソール (CMC) ポータルを起動します。
 2. [\[サーバ\]](#) を選択します。
 3. サーバー一覧で、[\[コアサービス\]](#) を選択し、[\[Adaptive Processing Server\]](#) を選択します。
 4. [\[サービス\]](#) ラベルの下にある [\[Adaptive Processing Server\]](#) を右クリックし、[\[メトリクス\]](#) を選択してホストされているサービスの一覧を表示します。
3. トランスレーションマネージャ SDK を使用して基本ワークフローを実行するには、どうすればよいですか。
最初に、トランスレーションマネージャ API を使用して `ITMgrEngine` を取得する必要があります。取得したら、`ITMgrEngine` を使用して、「[TMT SDK ワークフローの作成](#)」の節に記載されているすべての操作を実行することができます。

重要免責事項および法的情報

コードサンプル

この文書に含まれるソフトウェアコード及び / 又はコードライン / 文字列 (「コード」) はすべてサンプルとしてのみ提供されるものであり、本稼働システム環境で使用することが目的ではありません。「コード」は、特定のコードの構文及び表現規則を分かりやすく説明及び視覚化することのみを目的としています。SAP は、この文書に記載される「コード」の正確性及び完全性の保証を行いません。更に、SAP は、「コード」の使用により発生したエラー又は損害が SAP の故意又は重大な過失が原因で発生させたものでない限り、そのエラー又は損害に対して一切責任を負いません。

アクセシビリティ

この SAP 文書に含まれる情報は、公開日現在のアクセシビリティ基準に関する SAP の最新の見解を表明するものであり、ソフトウェア製品のアクセシビリティ機能の確実な提供方法に関する拘束力のあるガイドラインとして意図されるものではありません。SAP は、この文書に関する一切の責任を明確に放棄するものです。ただし、この免責事項は、SAP の意図的な違法行為または重大な過失による場合は、適用されません。さらに、この文書により SAP の直接的または間接的な契約上の義務が発生することは一切ありません。

ジェンダーニュートラルな表現

SAP 文書では、可能な限りジェンダーニュートラルな表現を使用しています。文脈により、文書の読者は「あなた」と直接的な呼ばれ方をされたり、ジェンダーニュートラルな名詞 (例: 「販売員」又は「勤務日数」) で表現されます。ただし、男女両方を指すとき、三人称単数形の使用が避けられない又はジェンダーニュートラルな名詞が存在しない場合、SAP はその名詞又は代名詞の男性形を使用する権利を有します。これは、文書を分かりやすくするためです。

インターネットハイパーリンク

SAP 文書にはインターネットへのハイパーリンクが含まれる場合があります。これらのハイパーリンクは、関連情報を見いだすヒントを提供することが目的です。SAP は、この関連情報の可用性や正確性又はこの情報が特定の目的に役立つことの保証を行いません。SAP は、関連情報の使用により発生した損害が、SAP の重大な過失又は意図的な違法行為が原因で発生したものでない限り、その損害に対して一切責任を負いません。すべてのリンクは、透明性を目的に分類されています (<http://help.sap.com/disclaimer> を参照)。



go.sap.com/registration/
contact.html

© 2016 SAP SE or an SAP affiliate company. All rights reserved.
本書のいかなる部分も、SAP SE 又は SAP の関連会社の明示的な許可なくして、いかなる形式でも、いかなる目的にも複製又は伝送することはできません。本書に記載された情報は、予告なしに変更されることがあります。SAP SE 及びその頒布業者によって販売される一部のソフトウェア製品には、他のソフトウェアベンダーの専有ソフトウェアコンポーネントが含まれています。製品仕様は、国ごとに変わる場合があります。
これらの文書は、いかなる種類の表明又は保証もなしで、情報提供のみを目的として、SAP SE 又はその関連会社によって提供され、SAP 又はその関連会社は、これら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 又はその関連会社の製品及びサービスに対する唯一の保証は、当該製品及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。
本書に記載される SAP 及びその他の SAP の製品やサービス、並びにそれらの個々のロゴは、ドイツ及びその他の国における SAP SE (又は SAP の関連会社) の商標若しくは登録商標です。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。
商標に関する情報および表示の詳細については、<http://www.sap.com/corporate-en/legal/copyright/index.epx> をご覧ください。