Hewlett Packard Enterprise

IT Business Analytics

Software Version: 10.10

Linux operating system

DCS Extractor SDK Guide

Document Release Date: February 2016
Software Release Date: February 2016

Documentation Updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates or to verify that you are using the most recent edition of a document, go to: https://softwaresupport.hp.com/.

This site requires that you register for an HP Passport and to sign in. To register for an HP Passport ID, click **Register** on the HP Support site or click **Create an Account** on the HP Passport logon page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

Visit the HP Software Support site at: https://softwaresupport.hpe.com.

This website provides contact information and details about the products, services, and support that HP Software offers.

HP Software online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support website to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and to sign in. Many also require a support contract. To register for an HP Passport ID, click **Register** on the HP Support site or click **Create an Account** on the HP Passport logon page.

To find more information about access levels, go to: https://softwaresupport.hpe.com/web/softwaresupport/access-levels.

HP Software Solutions Now accesses the HPSW Solution and Integration Portal website. This site enables you to explore HP Product Solutions to meet your business needs, includes a full list of Integrations between HP Products, as well as a listing of ITIL Processes. The URL for this website is http://h20230.www2.hp.com/sc/solutions/index.jsp.

About this PDF Version of Online Help

This document is a PDF version of the online help. This PDF file is provided so you can easily print multiple topics from the help information or read the online help in PDF format. Because this content was originally created to be viewed as online help in a web browser, some topics may not be formatted properly. Some interactive topics may not be present in this PDF version. Those topics can be successfully printed from within the online help.

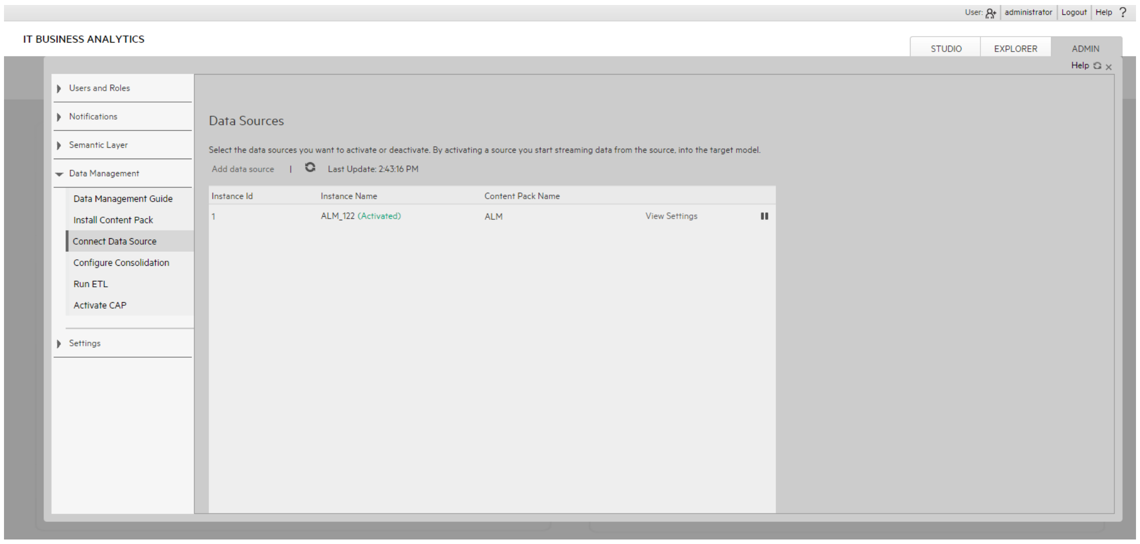
Contents

Getting Started with Data Collection Service (DCS) Extractor

Getting Started with Data Collection Service (DCS) Extractor

Data Collection Service (DCS) enables you to extract data from a specific data source according to the relevant extraction and source model that is generated by the IDE. It consists of a plugable extractor framework for each data source. The extractor gathers data according to the request it receives from the Content Flow Manager, placing it into a set of relevant .TXT files. Each supported data source has a corresponding extractor (or multiple extractors) that is capable of extracting the relevant data out of the data source. All available extractors for Content Packs use the DCS framework.

- **Extraction Mechanism.** The extraction is a self-managed web service. The data source connection information is registered to the framework when adding a new data source in **ITBA > ADMIN > Data Management > Connect Data Source**.



Data Sources

Select the data sources you want to activate or deactivate. By activating a source you start streaming data from the source, into the target model.

Last Update: 2:43:16 PM

Instance Id	Instance Name	Content Pack Name		
1	ALM_122 (Activated)	ALM	View Settings	▮▮

- **Extraction Methodology.** The extraction is done using extractors running on the application container of the Data Warehouse. The extractors use various technologies (for example, JDBC, Web Services, data files, and other kinds of HTTP requests) to extract the data from the data source. Each extraction is an isolated job that cannot be affected by other extraction jobs. Each extraction has a unique batch ID. The batch ID is incremental and cannot duplicated even for different Content Pack instances.

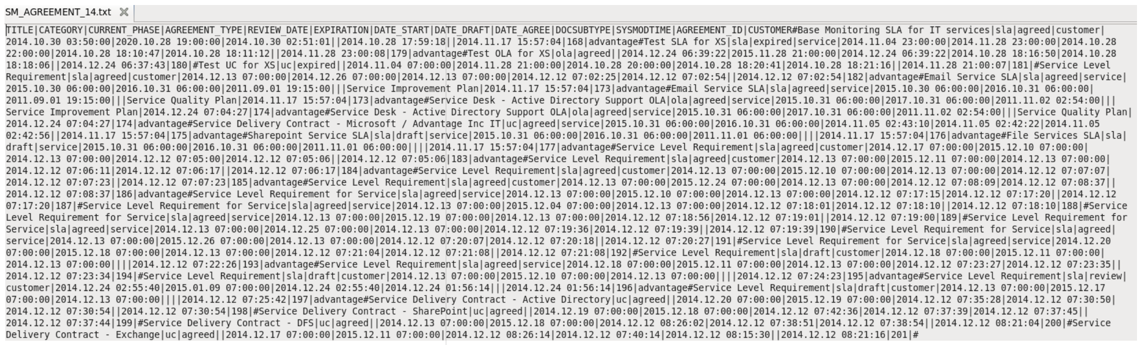
- **ETL Source Extract.** The first stage of the ETL is the Source Extract. In this phase, the Content Flow Manager performs an HTTP request that activates the relevant extractor.

  The DCS extractor extracts the data from the data source into flat files. All Content Packs integrate using DCS, where data is extracted from the data source into >TXT files with a well-defined standard structure.

- **Format of flat files.** The first line of a flat file should be the headers of all columns, separated with "|" symbol. The data follows with the columns values separated with a "|" symbol and the lines separated with a "#" symbol.

  If a column value includes special characters like "|", "#" and "\", it should be escaped by adding a "\" symbol before the special character. The DCS framework has a FlatFileWriter will handle the details of writing the headers and values.

  Flat file example:


SM_AGREEMENT_14.txt

```
TITLE|CATEGORY|CURRENT_PHASE|AGREEMENT_TYPE|REVIEW_DATE|EXPIRATION|DATE_START|DATE_DRAFT|DATE_AGREE|DOCSUBTYPE|SYSMODTIME|AGREEMENT_ID|CUSTOMER#Base Monitoring SLA for IT services|sla|agreed|customer|
2014.10.30 03:50:00|2020.10.28 19:00:00|2014.10.30 02:51:01||2014.10.28 17:59:18||2014.11.17 15:57:04|168|advantage#Test SLA for XS|sla|expired|service|2014.11.04 23:00:00|2014.11.28 23:00:00|2014.10.28
22:00:00|2014.10.28 18:10:47|2014.10.28 18:11:12||2014.11.28 23:00:08|179|advantage#Test OLA for XS|ola|agreed||2014.12.24 06:39:22|2015.11.28 21:00:00|2014.12.24 06:39:22|2014.10.28 18:16:50|2014.10.28
18:18:06||2014.12.24 06:37:43|180|#Test UC for XS|uc|expired||2014.11.04 07:00:00|2014.11.28 21:00:00|2014.10.28 20:00:00|2014.10.28 18:20:41|2014.10.28 18:21:16||2014.11.28 21:00:07|181|#Service Level
Requirement|sla|agreed|customer|2014.12.13 07:00:00|2014.12.26 07:00:00|2014.12.13 07:00:00|2014.12.12 07:02:25|2014.12.12 07:02:54||2014.12.12 07:02:54|182|advantage#Email Service SLA|sla|agreed|service|
2015.10.30 06:00:00|2016.10.31 06:00:00|2011.09.01 19:15:00|||Service Improvement Plan|2014.11.17 15:57:04|173|advantage#Email Service SLA|sla|agreed|service|2015.10.30 06:00:00|2016.10.31 06:00:00|
2011.09.01 19:15:00|||Service Quality Plan|2014.11.17 15:57:04|173|advantage#Service Desk - Active Directory Support OLA|ola|agreed|service|2015.10.31 06:00:00|2017.10.31 06:00:00|2011.11.02 02:54:00|||
Service Improvement Plan|2014.12.24 07:04:27|174|advantage#Service Desk - Active Directory Support OLA|ola|agreed|service|2015.10.31 06:00:00|2017.10.31 06:00:00|2011.11.02 02:54:00|||Service Quality Plan|
2014.12.24 07:04:27|174|advantage#Service Delivery Contract - Microsoft / Advantage Inc IT|uc|agreed|service|2015.10.31 06:00:00|2016.10.31 06:00:00|2014.11.05 02:43:10|2014.11.05 02:42:22|2014.11.05
02:42:56||2014.11.17 15:57:04|175|advantage#Sharepoint Service SLA|sla|draft|service|2015.10.31 06:00:00|2016.10.31 06:00:00|2011.11.01 06:00:00|||2014.11.17 15:57:04|176|advantage#File Services SLA|sla|
draft|service|2015.10.31 06:00:00|2016.10.31 06:00:00|2011.11.01 06:00:00|||2014.11.17 15:57:04|177|advantage#Service Level Requirement|sla|agreed|customer|2014.12.17 07:00:00|2015.12.10 07:00:00|
2014.12.13 07:00:00|2014.12.12 07:05:00|2014.12.12 07:05:06||2014.12.12 07:05:06|183|advantage#Service Level Requirement|sla|agreed|customer|2014.12.13 07:00:00|2015.12.11 07:00:00|2014.12.13 07:00:00|
2014.12.12 07:06:11|2014.12.12 07:06:17||2014.12.12 07:06:17|184|advantage#Service Level Requirement|sla|agreed|customer|2014.12.13 07:00:00|2015.12.10 07:00:00|2014.12.13 07:00:00|2014.12.12 07:07:07|
2014.12.12 07:07:23||2014.12.12 07:07:23|185|advantage#Service Level Requirement|sla|agreed|customer|2014.12.13 07:00:00|2015.12.24 07:00:00|2014.12.13 07:00:00|2014.12.12 07:08:09|2014.12.12 07:08:37||
2014.12.12 07:08:37|186|advantage#Service Level Requirement for Service|sla|agreed|service|2014.12.13 07:00:00|2015.12.10 07:00:00|2014.12.13 07:00:00|2014.12.12 07:17:15|2014.12.12 07:17:20||2014.12.12
07:17:20|187|#Service Level Requirement for Service|sla|agreed|service|2014.12.13 07:00:00|2015.12.04 07:00:00|2014.12.13 07:00:00|2014.12.12 07:18:01|2014.12.12 07:18:10||2014.12.12 07:18:10|188|#Service
Level Requirement for Service|sla|agreed|service|2014.12.13 07:00:00|2015.12.19 07:00:00|2014.12.13 07:00:00|2014.12.12 07:18:56|2014.12.12 07:19:01||2014.12.12 07:19:00|189|#Service Level Requirement for
Service|sla|agreed|service|2014.12.13 07:00:00|2014.12.25 07:00:00|2014.12.13 07:00:00|2014.12.12 07:19:36|2014.12.12 07:19:39||2014.12.12 07:19:39|190|#Service Level Requirement for Service|sla|agreed|
service|2014.12.13 07:00:00|2015.12.26 07:00:00|2014.12.13 07:00:00|2014.12.12 07:20:07|2014.12.12 07:20:18||2014.12.12 07:20:27|191|#Service Level Requirement for Service|sla|agreed|service|2014.12.20
07:00:00|2015.12.18 07:00:00|2014.12.13 07:00:00|2014.12.12 07:21:04|2014.12.12 07:21:08||2014.12.12 07:21:08|192|#Service Level Requirement|sla|draft|customer|2014.12.18 07:00:00|2015.12.11 07:00:00|
2014.12.13 07:00:00|||2014.12.12 07:22:26|193|advantage#Service Level Requirement|sla|agreed|service|2014.12.18 07:00:00|2015.12.11 07:00:00|2014.12.13 07:00:00|2014.12.12 07:23:27|2014.12.12 07:23:35||
2014.12.12 07:23:34|194|#Service Level Requirement|sla|draft|customer|2014.12.13 07:00:00|2015.12.10 07:00:00|2014.12.13 07:00:00|||2014.12.12 07:24:23|195|advantage#Service Level Requirement|sla|review|
customer|2014.12.24 02:55:40|2015.01.09 07:00:00|2014.12.24 02:55:40|2014.12.24 01:56:14||2014.12.24 01:56:14|196|advantage#Service Level Requirement|sla|draft|customer|2014.12.13 07:00:00|2015.12.17
07:00:00|2014.12.13 07:00:00||||2014.12.12 07:25:42|197|advantage#Service Delivery Contract - Active Directory|uc|agreed||2014.12.20 07:00:00|2015.12.19 07:00:00|2014.12.12 07:35:28|2014.12.12 07:30:50|
2014.12.12 07:30:54||2014.12.12 07:30:54|198|#Service Delivery Contract - SharePoint|uc|agreed||2014.12.19 07:00:00|2015.12.18 07:00:00|2014.12.12 07:42:36|2014.12.12 07:37:39|2014.12.12 07:37:45||
2014.12.12 07:37:44|199|#Service Delivery Contract - DFS|uc|agreed||2014.12.13 07:00:00|2015.12.18 07:00:00|2014.12.12 08:26:02|2014.12.12 07:38:51|2014.12.12 07:38:54||2014.12.12 08:21:04|200|#Service
Delivery Contract - Exchange|uc|agreed||2014.12.17 07:00:00|2015.12.11 07:00:00|2014.12.12 08:26:14|2014.12.12 07:40:14|2014.12.12 08:15:30||2014.12.12 08:21:16|201|#
```

- **Data sources and Content Packs.**

  The following data source types are available for each Content Pack:

Content Pack	Data source type
ALM	ALM
AM	MSSQL, Oracle
AWS	AWS
AWSCW	AWSCW
Azure	GENERIC
CSA	CSA
PPM	Oracle

Content Pack	Data source type
SA	Oracle
SM	MSSQL(Non dbdict), Oracle(Non dbdict), MSSQL(dbdict), Oracle(dbdict), DB2 (dbdict)
CO	CO

- **Troubleshooting Logs.**
  - **$HPBA_Home/glassfish/glassfish/domains/BTOA/logs/dcs.log**: This log describes all of the current activity of the DCS framework as well as the activity of the common utilities and general extractors.
  - **$HPBA_Home/glassfish/glassfish/domains/BTOA/logs/dcs.extractor.log**: This log describes the activity of all the extractors.

Extractor

Integration with data sources is performed using the Data Collection Service (DCS). In order to generate the query executed for the data source, the Content Flow Manager tells the DCS framework which source entity to extract. The DCS then extracts the correlating metadata definitions, and uses them in order to formulate a query.

The extractor developer uses the Source Model .XML files from the Content Pack to enable the extractor functionality. This metadata component is required for the custom extractor development process. The Source Model .XML is a mandatory component of each Content Pack.

Integration Architecture

The following section describes the basic architecture and flow of the DCS extractor process used to create the custom extractor.

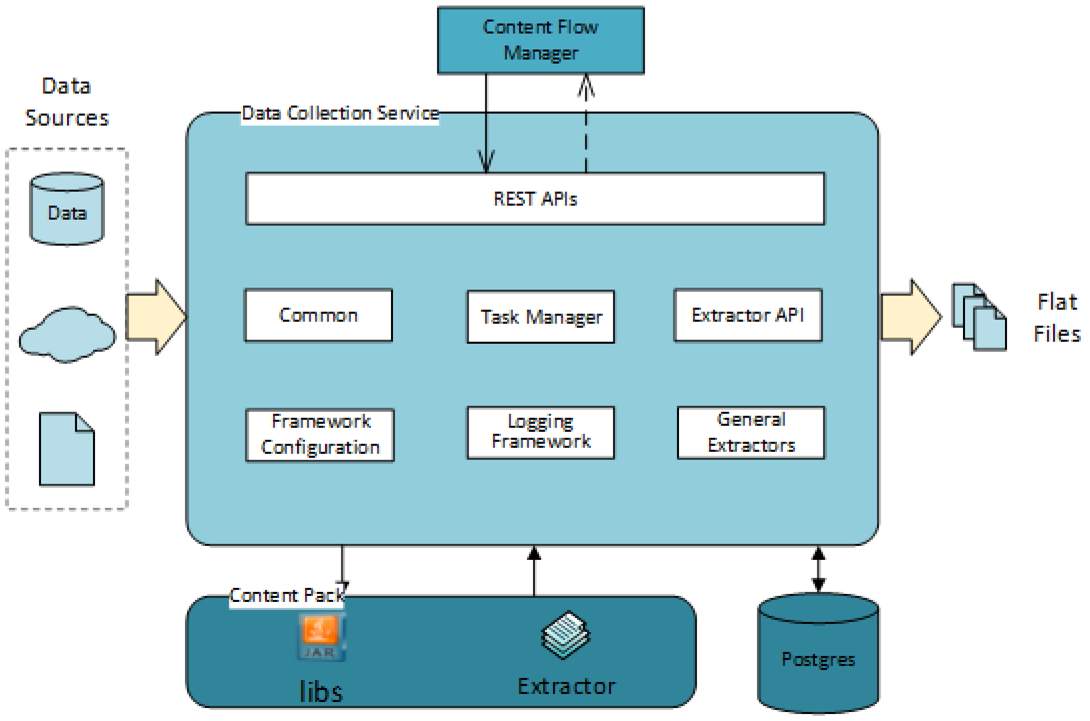
In the extractor process the following information is gathered by the DCS framework:

- **Connection details.** The source credentials that register the DCS framework when adding a new data source.
- **Extraction Model Metadata.**
  - **SourceModel.xml.** Defines the data structure and how this information is extracted from the source (generated by the IDE).

The following depicts the high level architecture of the extraction process:

DCS Process and Architecture

- The Content Flow Manager triggers the extractor according to the schedule settings or runs it immediately
- DCS receives information for the Extraction process.
- DCS extracts the source model and connection details.
- The Extractor begins to extract the data from the data source into a local file.
- The ETL manager imports the flat files into the Data Warehouse (Vertica).
- ETL Workflow takes the flat files and starts the ETL.



a. The Content Flow Manager starts or is scheduled to trigger a new extraction request for some specified Content Package. It passes the necessary information required by the extractor framework through parameters in the **RESTful API**connection properties, such as **CP_ INSTANCE_ID**, to the data source, as well as the last modified time for each entity. and more.

b. The Data Sources that need to be supported should include typical relational databases, web services with SOAP or RESTful APIs, customized data files, and more.

c. The Content Pack contains the customized extractor (.JAR file) and relevant libraries (such as the driver for the database), which can be loaded and replaced dynamically at runtime after the

CP was activated.

d. The REST API is the overall interface for all Data Collection services. It should provide all the necessary RESTful APIs that are needed to interact with users just like the CP Runtime Service, which includes start extraction, abort extraction, get the extraction status of some CP instance, and more.

e. The Common provides all the common classes such as constants, data beans, exception definitions, and utilities. It should be the base of the DCS framework.

f. The Extractor API component defines all the common interfaces for the extractors.

g. The General Extractors implement the interfaces declared in the above section as the typical implementations.

h. The Task Manager is used to manage the extraction tasks in a configurable thread pool. It isolates the execution of a task from other tasks, and also handles the timeout of tasks.

i. The Framework Configuration loads and manages the configuration from a local disk for the DCS framework.

Extractor Prerequisite

The extractor developer uses the Source Model .XML files from the Content Pack to enable the extractor functionality. This metadata component is required for the custom extractor development process. The Source Model .XML is a mandatory component of each Content Pack.

Tasks

Create a Source Model .XML using the IDE

To create a Source Model .XML using the IDE:

1. For a new Content Pack, create your project in the IDE. For details, see Define New Content Pack Project in the *Content Extension Guide*.
2. Add content in the Engineer Stream Designer. For details, see Engineer Stream Designer Tasks in the *Content Extension Guide*.
3. Generate your content for use in the Data Warehouse. Select the **Generate Content Pack** checkbox to create a new Content Pack (not extending content). This creates a new content pack structure along with the generated ETL files. For details, see Generate Content in the *Content Extension Guide*.
4. The source entity .XML is located in the CP output directory under **$HPBA_HOME/ContentPacks/<CP_ Name>/INBUILT/DATAMODEL/DATAINTEGRATION/SOURCE/<version>/<type>/.**
5. The Source Model .XML is then used in the development of the custom extractor. For details, see "Step by Step Develop a Java-based Extractor" on page 10.

Step by Step Develop a Java-based Extractor

The next stage in the DCS Extractor process is to write the actual extractor. You can also modify the Sample Extractor or write your own according to the guidelines

1. Create a Content Pack with IDE

Create the Content Pack with the IDE provided in the ITBA installation package. For details, see *Content Extension Guide*.

2. Unzip, Deploy, and Install the Content Pack in BA

To unzip, deploy, and install the Content Pack in BA:

1. Unzip the zipped Content Pack to folder **$HPBA_HOME/ContentPacks**.
2. Open a shell terminal and run the following commands:

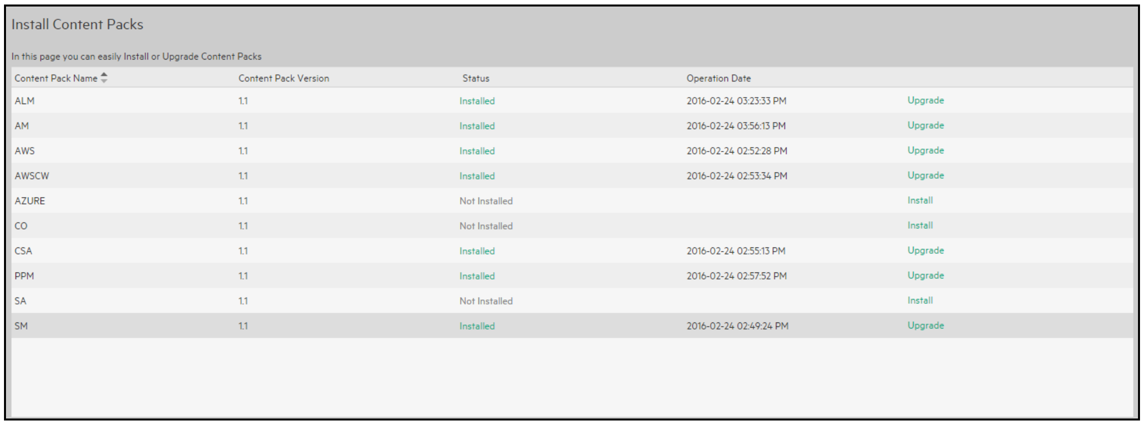
   cd $HPBA_HOME/bin

   ./stopGlassfish.sh

3. If the supervisor is correctly configured, glassfish starts automatically. Otherwise, run the following command

   ./startGlassfish.sh

4. Login as Administrator and click **ADMIN > Data Management > Install Content Pack** and click the relevant **Install** link.



Install Content Packs

In this page you can easily Install or Upgrade Content Packs

Content Pack Name	Content Pack Version	Status	Operation Date	
ALM	1.1	Installed	2016-02-24 03:23:33 PM	Upgrade
AM	1.1	Installed	2016-02-24 03:56:13 PM	Upgrade
AWS	1.1	Installed	2016-02-24 02:52:28 PM	Upgrade
AWSCW	1.1	Installed	2016-02-24 02:53:34 PM	Upgrade
AZURE	1.1	Not Installed		Install
CO	1.1	Not Installed		Install
CSA	1.1	Installed	2016-02-24 02:55:13 PM	Upgrade
PPM	1.1	Installed	2016-02-24 02:57:52 PM	Upgrade
SA	1.1	Not Installed		Install
SM	1.1	Installed	2016-02-24 02:49:24 PM	Upgrade

For details, see Install Content Pack in the *Administrator Guide*.

3. Create a Java Project and Include Dependencies

Create a Java project in Eclipse or Java IDE and include the dependencies described in Dependent Libraries section. For details, see "Dependent Libraries" below.

Dependent Libraries

To develop a DCS-based extractor, add at least the following jars to your build path in your IDE such as Eclipse.

- **$HPBA_HOME/glassfish/glassfish/domains/BTOA/applications/dcs-web/WEB-INF/lib/dcs-common-<version>.jar**
- **$HPBA_HOME/glassfish/glassfish/domains/BTOA/applications/dcs-web/WEB-INF/lib/cp-model-<version>.jar**
- **$HPBA_HOME/glassfish/glassfish/domains/BTOA/lib/crypto-<version>.jar**

4. Implement the Extractor

Implement your own extractor following the guidance and sample extractor. For details, see Extractor Implementation Guidelines.

Extractor Implementation Guidelines

You can create your own extractor. Note the following guidelines when creating a new extractor.

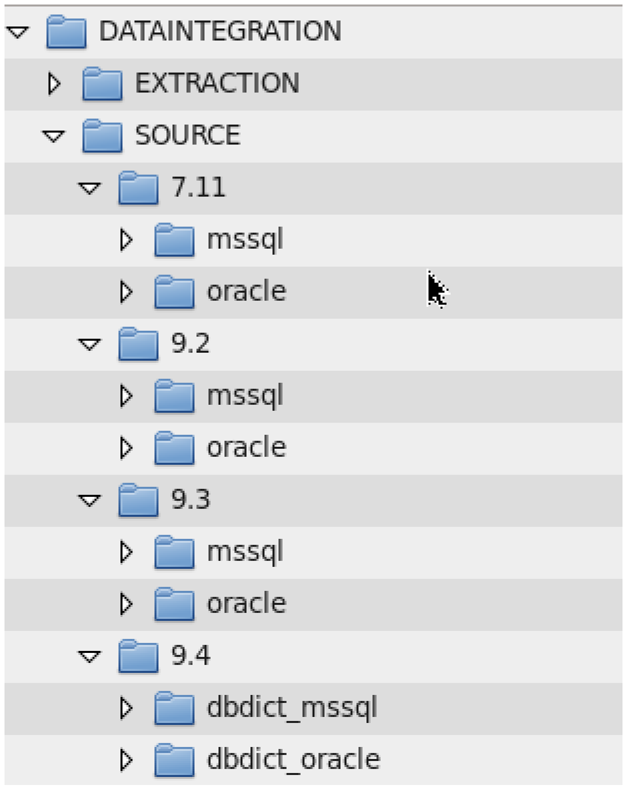
The DCS Extractor implementation requires you to create the **XXXExtractor** extractor that extends the **BaseExtractor** extractor.

All the extractors are packaged in a .JAR that you copy and deploy. The .JAR is reloaded dynamically when a new extraction is triggered without rebooting BA. For deployment, the .JAR must align with the extractor path (relative path to the **extractor_manifest.xml** file) that is defined in the **extractor_ manifest.xml** file.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<extractors>
    <extractor>
      <productName>SM</productName>
      <dataSourceType>MSSQL,ORACLE,DBDICT_MSSQL</dataSourceType>
      <productVersion>dbdict,7.11,9.2,9.3,9.4</productVersion>
      <className>com.hp.itba.dwh.extractor.sm.SMExtractor</className>
      <extractorPath>extractor-sm</extractorPath>
      <extractorVersion>1.0</extractorVersion>
</extractor>
<extractor>
       …extractor2…
</extractor>
</extractors>
```

You should define the extractor information in the manifest file of your own Content Pack (CP). The extractor .JAR file must be located in the path of each Content Pack **$HPBA_ HOME/ContentPacks/<CP_Name>/EXTRACTOR/extractor_manifest.xml**. The DCS framework loads and parses the manifest file to get these items once the extraction is triggered. The extractor manifest may contains several extractors information and each extractor need to specify the product name, data source type, product version, class name extractor path, and extractor version. The product name and product version may have multiple values separated by a comma. They are case sensitive, which means the Cartesian product of these values can be supported by the specified extractor class.

The product version and type should follow the source model folder structure show below.



Note: To make sure the content pack can be activated and the extractor correctly found by the DCS framework, the data source type and product version should fully aligned between the **extractor_manifest.xml**, **dataSourceDef.xml** and the source folder structure.

The following describes the basic API and workflow for implementation.

XXXExtractor

This extractor extends the BaseExtractor and requires at least the following implementation:

```
public boolean checkConnection() throws ExtractorException
protected abstract void doExtract(String batchId, Map<String, String>
lastModifiedDateMap, List<DcsEntity> entities) throws ExtractorException,
            InterruptedException;
```

The **batchId** is a unique number indicates a distinguished data extraction job.

The **lastModifiedDateMap** contains all the last modified date for all the entities, which is critical for delta data extraction.

The **entities** contains all the valid entities for the current extraction. Entities that appears in the extraction model but not in the source model are ignored and are not included in the entities parameter.

For the initial load, the CFM (Content Flow Manager) transforms the initial load period to a specified date value as the last modified date.

For example, if the current time is ‘2015-06-01 00:00:00’ and the initial load period is 6 months, the calculated last modified date for all entities is ‘2015-01-01 00:00:00’. The date time is formatted with the time zone in the ITBA server. Each entity will have a separate entry in the **lastModifiedDateMap**.

For a delta load, CFM fetches the last modified date from the TSNP tables according to the column that is marked as **lastmodifieddate=’Y’** in the extraction model.

Activation & Registration

- Test the connection.
- Verify that the relevant source is available and that the Data Warehouse can connect to it.
- The connection details of the specific source such as product version, data source type, connection properties must be send to the extractor, initialized, and must invoke the **checkConnection** method. Once the check connection is successful, its details are registered into the DCS framework and saved for later data extraction. Otherwise, an exception is thrown to the upper layer.

  The check operation uses these details in order to validate that the source is accessible from the Data Warehouse.

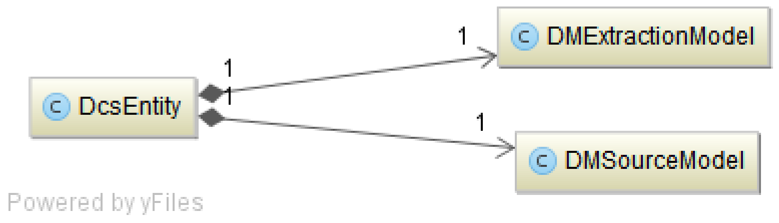
Data Source Extraction

- Extract the relevant data from the source.
- Use the Source Model:
  - Describes the required data and data structure of the source.
  - Defines the structure of the result output text file.

    The **doExtract(…)** should connect to the source and read the relevant data defined in the SourceModel. For each extractor, there may be more than one entity for extraction. All the entities can be extracted in sequence order in the extractor that you defined. Another option is that you create an entity level extractor and let the entity extractor run in parallel. The CP level extractor will manage all the entity level extractor tasks.

    Every entity extraction should use the **createFlatFileWriter(…)** that is implemented in the **BaseExtractor** to create the **FileFileWriter** and write the result. Generally, **FileFIleWriter** has two methods: **writeHeader(String[] headers)** and **writeLine(String[] values)**, which are used to write the header values and the values for each row data. The headers and values are escaped so that the flat file can be imported into the Vertica DB.

SourceModel stores information about the data structure of the source. It defines the name of the entity and the fields related to this entity. All the entities contains both the extraction model and the source model, and they are loaded by the **BaseExtractor** before the data extraction. Each entity is related to a **DcsEntity** object, which contains corresponding source model and extraction model. You can get the field details through the extraction model and source model reference.



In order to get the fields and their order, you can use **getColumns()** of the **DMExtractionModel** (defined and initialized in the **BaseExtractor**).

- Initial load and delta load
  - The initial load means that all the data is loaded according the range defined by the initial load period. This process takes a lot of time depending on the length of the period you want to extract. A one time initial load is executed after setting up a new data source connection.
  - After the initial load, the following data extraction is taken as a delta load. Generally, it extracts the data that changed since the last extraction time except for delete cases.
- Delete case
  - If an entity is marked as delete **(isdelete="Y"**) in the extraction model, it means that the columns that are defined as business key (**businesskey="Y"**) need to be fully extracted from the data source, whatever it is initial load or delta load. You should ignore the last modified date value for that entity or use the value in the initial load.
- Extraction status: The Extractor manages and stores the status to DB (PostgreSQL).

  To store the operation status info to DB:

  ```
  status.setStatus(StatusEnum.COMPLETED.getValue());
  flushStatus();
  ```

  It is very important that the extractor developer save the status of all the entities because the **BaseExtractor** includes a logic that checks all the statuses of the entities. An extractor may have a lot of entities that can be extracted in parallel. **If and only if** all the entities are successfully extracted, the whole extractor can be successful. If any of the entity fails to extract data, the whole extractor fails.

Sample Extractor

```
package com.hp.itba.dwh.extractor.test;

import java.util.List;
import java.util.Map;

import com.hp.btoe.security.crypto.ActiveCrypto;
import com.hp.itba.dwh.dcs.api.IDataSource;
import com.hp.itba.dwh.dcs.api.impl.BaseExtractor;
import com.hp.itba.dwh.dcs.bean.DcsEntity;
import com.hp.itba.dwh.dcs.bean.DcsEntityStatus;
import com.hp.itba.dwh.dcs.common.FlatFileWriter;
import com.hp.itba.dwh.dcs.common.StatusEnum;
import com.hp.itba.dwh.dcs.exception.ExtractorException;
import com.hp.itba.dwh.dcs.util.DateUtil;

public class SampleExtractor extends BaseExtractor {


    @Override
    public String getPlatformVersion() {
         return "10.10.0"; }

    @Override
       public boolean checkConnection() throws ExtractorException {
               logger.debug("start to check the connection for CP {}.", cpKey);
               IDataSource dataSource = metadata.getDataSource();
        List<String[]> properties = dataSource.getConnProperties();
        for (String[] prop : properties) {
            String value = prop[1];
            if ("true".equalsIgnoreCase(prop[2]) && value != null) {
                try {
                    // decrypt the the property value if necessary
                    value = ActiveCrypto.getDefault().decrypt(value);
                } catch (Exception e) {
                    throw new ExtractorException("fail to decrypt password!", e);
                }
            }
        }
        // setup the connection to the data source
        // ...
        logger.debug("end of checking the connection for CP {}.", cpKey);
        return true;
        }

@Override
       protected void doExtract(String batchId, Map<String, String>
               lastModifiedDateMap, List<DcsEntity> entities) throws
ExtractorException,InterruptedException{
               logger.debug("start to extract data for CP {}. batchId: {}", cpKey,batchId);
       // get data source connection information
               IDataSource dataSource = metadata.getDataSource();
               List<String[]> properties = dataSource.getConnProperties();
               for (String[] prop : properties) {
                       String value = prop[1];
                       if ("true".equalsIgnoreCase(prop[2]) && value != null) {
                               try {
                                       // decrypt the the property value if necessary
                                       value = ActiveCrypto.getDefault().decrypt(value);
                               } catch (Exception e) {
                                       throw new ExtractorException("fail to decrypt password!", e);
                               }
                       }
               }
               // setup the connection to the data source
               // ...
               // check whether user click the abort button. If aborted, the wholeextraction
should be interrupted
               stopSign.checkState();
               try {
                       for (DcsEntity entity : entities) {
                               stopSign.checkState();
                               DcsEntityStatus entityStatus = entity.getEntityStatus();
                               // mark entity status as started
                               entityStatus.setStartTime(DateUtil.getCurrentTime());
                               entityStatus.setEntityStatus(StatusEnum.ENTITY_STARTED.getValue());
                               statusHelper.flushStatus();
                               // create the FlatFileWriter with header written
                               FlatFileWriter writer = createFlatFileWriter(entity);
                               List<com.hp.itba.dwh.common.cp.model.DMExtractionModelColumn>
                               outputColumns = entity.getOutputColumns();
                               int totalRowCount = 0;
                               // for each line in the fetched data
                               String[] values = new String[outputColumns.size()];
                               // populate the values with the data fetched from the data source
                               // ...
                               writer.writeLine(values);
                               // record the total count of rows extracted and mark the entity as co
                               entityStatus.setResultCount(totalRowCount);
                               entityStatus.setEntityStatus(StatusEnum.ENTITY_COMPLETED.getValue());
                               entityStatus.setEndTime(DateUtil.getCurrentTime());
```

```
                    statusHelper.increaseSuccessEntityCount();
                    statusHelper.flushStatus();
            }
    } catch (ExtractorException e) {
            throw e;
    } catch (Throwable ex) {
            logger.error("error occurs when extracting data!", ex);

    }

 }

}
```

5. Create a Unit Test to Check the Extractor

Create a unit test so that your extractor can be work fine. For details, see "Test the Extractor (Unit Test)" below.

Test the Extractor (Unit Test)

You can write a unit test to test your extractor to check that your extractor works. In the test, you must inject a dummy **IExtractorMetadata** instance. The **IExtractorMetadata** is used for initialization by the extractor. The dummy instance provides the extractor with information such as CP name, data source type, product version, connection property details and so on.

If required, download and import the Junit and EasyMock library to your project from the following sites:

http://junit.org/

http://easymock.org/

```
package com.hp.itba.dwh.extractors.sample.test;

import java.io.File;
import java.util.ArrayList;
import java.util.List;

import junit.framework.Assert;

import org.easymock.EasyMock; import org.easymock.IMocksControl; import
org.junit.BeforeClass; import org.junit.Ignore;
import org.junit.Test;

import com.hp.itba.dwh.dcs.api.IDataSource;
import com.hp.itba.dwh.dcs.api.IExtractor;
import com.hp.itba.dwh.dcs.api.IExtractorMetadata;
import com.hp.itba.dwh.dcs.exception.ExtractorException;
import com.hp.itba.dwh.extractor.test.SampleExtractor;

public class SampleExtractorTest {
static IExtractor extractor;
```

```
public static void setup() {
extractor = new SampleExtractor();
IMocksControl control = EasyMock.createControl();
IExtractorMetadata metadata = control.createMock(IExtractorMetadata.class);

EasyMock.expect(metadata.getCpName()).andReturn("Sample").anyTimes();

EasyMock.expect(metadata.getDataSourceType()).andReturn("generic").anyTimes();

EasyMock.expect(metadata.getProductVersion()).andReturn("1.0").anyTimes();




EasyMock.expect(metadata.getFieldDelimiter()).andReturn("|").anyTimes();
EasyMock.expect(metadata.getRowDelimiter()).andReturn("#").anyTimes();
EasyMock.expect(metadata.getOutputPath()).andReturn(new File
("output").getAbsolutePath()).anyTimes();
IDataSource ds = control.createMock(IDataSource.class); List<String[]> value = new
ArrayList<String[]>();
value.add(new String[] {"productName", "Sample", "false"});
//…
EasyMock.expect(ds.getConnProperties()).andReturn(value).anyTimes();
EasyMock.expect(metadata.getDataSource()).andReturn(ds).anyTimes();

EasyMock.expectLastCall().anyTimes();
control.replay();
try {
extractor.init("sample_1", metadata);
// extractor.setDbHelper(dbHelper);
} catch (ExtractorException e) { Assert.fail(e.getMessage());
}

}

public void testCheckConnection() {
try {
extractor.checkConnection();
} catch (ExtractorException e) { Assert.fail(e.getMessage());
}
}

public void testExtract() {
try {
extractor.extract("1", null);
} catch (ExtractorException e) { Assert.fail(e.getMessage());
}
}
}
```

6. Package the Extractor as a JAR File

The next step after testing is to perform the compilation and packaging process of the DCS extractor project. You need to compile and package your extractor project into a .JAR file and put it under the location defined in the **extractor_manifest.xml** file. The dependencies of your extractor should be located in the **lib** folder under your extractor location.

7. Copy and Deploy the Extractor JAR and Libraries

After packing the extractor into a .JAR file, you can take the .JAR from the project and deploy it on the server. Perform the following steps to deploy the compiled custom extractor.

1. Build the extractor Project: **extractor-xxx-1.0.0.jar**.
2. On the ITBA server, backup the jar file in the following location if the file exists: **$HPBA_HOME/ContentPacks/<CP_Name>/EXTRACTOR/<extractor-folder>**
3. On the ITBA server, copy the **extractor-xxx-1.0.0.jar** file to the following location: **$HPBA_HOME/ContentPacks/<CP_Name>/EXTRACTOR/<extractor-folder>**.
4. Copy the dependent libraries to the **$HPBA_HOME/ContentPacks/<CP_Name>/EXTRACTOR/<extractor-folder>/lib** folder.

8. Modify the extractor_manifest.xml File

Modify the **extractor_manifest.xml** file. For details, see "Extractor Implementation Guidelines" on page 14.

9. Add a New Data Source for the Content Pack in ITBA

Add a new data source for the Content Pack in ITBA. If it fails check "3. Create a Java Project and Include Dependencies" on page 13, "4. Implement the Extractor" on page 14, "5. Create a Unit Test to Check the Extractor" on page 21, "6. Package the Extractor as a JAR File" on page 24, "7. Copy and Deploy the Extractor JAR and Libraries" on page 25, or "8. Modify the extractor_manifest.xml File" on the previous page, and retry.

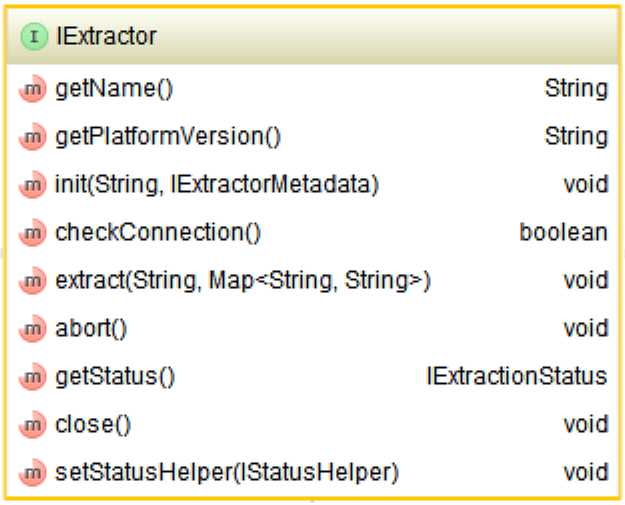
10. Run ETL

Run the ETL that corresponds to the Content Pack. For details, see Run ETL - Content Flow Management in the *Administrator Guide*.

If it fails check "3. Create a Java Project and Include Dependencies" on page 13, "4. Implement the Extractor" on page 14, "5. Create a Unit Test to Check the Extractor" on page 21, "6. Package the Extractor as a JAR File" on page 24, "7. Copy and Deploy the Extractor JAR and Libraries" on page 25, or "8. Modify the extractor_manifest.xml File" on page 26, and retry.

Extractor API

The Extractor API defines the interfaces that are used to extract all the entities of the CP. All extractors should implement the interface defined below:



IExtractor	
getName()	String
getPlatformVersion()	String
init(String, IExtractorMetadata)	void
checkConnection()	boolean
extract(String, Map<String, String>)	void
abort()	void
getStatus()	IExtractionStatus
close()	void
setStatusHelper(IStatusHelper)	void

- The **init** method is invoked to initialize the connections to a specified data source. It also loads all the entity configurations from the metadata files to the CP folder. The endpoint argument defines the necessary information to connect to the data source.
- The **checkConnection** method is called to check whether the connection to the data source can be established or not.
- The **extract** method is responsible for doing the actual extraction work for all the entities. The entities can be extracted in sequence or in parallel, depending on the implementation of the extractors. The status is returned once the task is finished or failed.
- The **abort** method supports the abort extraction functionality during the extraction. If the execution was aborted, it returns **true**; otherwise, it returns **false**.

The **BaseExtractor** implements the **IExtractor** interface and handles all the common tasks for all extractors, such as initializing the extractor, loading the extraction model and source models, loading the settings for each extractor, handling the status persistence and so on. So that it is recommended to implement customized extractors based on the **BaseExtractor**.

Generally, all the extractors that extends BaseExtractor should overwrite the following methods:

```
1. getPlatformVersion()
2. checkConnection()throws ExtractorException
```

```
3. doExtract(String batchId, Map<String,String> lastModifiedMap, List<DcsEntity>
entities) throws ExtractorException, InterrruptedException
```

The **getPlatformVersion** method is used to indicate which platform the current extractor is targeted to be based on. The version number should have 3 parts: <major version>, <minor version> and <patch version>. For ITBA 10.10, the platform version is **10.10.0**.

The **checkConnection** and the **doExtract** methods are the most important methods that you must implement:

- The **checkConnection** is used to test the connection when adding new data sources.
- The **doExtract**extracts the data when you click the start ETL or ETL is triggered by the scheduler.

Every check connection or data extraction is a separate process and cannot have an impact on other processes. This means that you cannot share the fields for different batches of extractor execution. For example, if you want to count the times of execution in your extractor, you must defined\ a non-static field named **count** and increase it in the **doExtract** method. You cannot get the correct result because for each extraction, a new extractor class instance is created and the count value is always 0 at the beginning.

The extractor class is dynamically loaded by the DCS framework. So that you can easily replace the extractor .JAR file to make your changes work immediately in the next batch execution.

General Extractors

A base extractor is provided to extend all general extractors. General extractors for typical databases like MySQL, DB2, MSSQL, Oracle, and PostgreSQL are provided within the DCS framework. They provide the most common functionality for the extraction. You can customize extractors by extending the general extractors or extending the base extractor itself.

You can simply define the following **extractor_manifest.xml** file to use the general SQL extractor:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<extractors>
    <extractor>
      <productName>TEST</productName>
      <dataSourceType>oracle</dataSourceType>
      <productVersion>1.0</productVersion>
<className>com.hp.itba.dwh.extractor.common.sql.SqlExtractor</className>
      <extractorVersion>1.0</extractorVersion>
      <extractorPath>.</extractorPath>
    </extractor>
</extractors>
```

where **dataSourceType** can be one of the following: **mysql**, **db2**, **mssql**, **oracle**, or **postgresql**.

Configure the “Connect Data Source” UI for ITBA Integration

After the integration test, you must define the end point settings for the data source required for the activation of the integration. The Connect Data Source User Interface enables you to activate, deactivate and configure data source parameters based on the Content Pack data source .XML file. For new content you can customize the **datasourceDef.xml** file and configure according to the requirements of your Content Pack. Once you have created a new Content Pack directory, it should contains the .XML configuration file in the following directory:
$HPBA_HOME/ContentPacks/<CP_Name>/conf/dataSourceDef.xml

Any configuration changes made in the .XML file can then be viewed in the Connect Data Source UI. For details, see Connect the Data Sources in the *Administrator Guide*.

Deploy the Configuration

The Connect Data Source UI supports new data source types that can be extended at runtime. You deploy the configuration using the .XML file listed below. It provides a dynamic list of attributes that are required to connect to a new source type.

The Data Source Activation process uses all of these attributes to connect to the data source. When you want to integrate a new data source data into ITBA, you create a directory for the data source.

Each Content Pack includes a custom Connect Data Source User Interface definition file in .XML format (**dataSourceDef.xml**). It is located by default at:
$HPBA_HOME/ContentPacks/<CP_name>/conf

1. Configure the **dataSourceDef.xml** file that is used to create the Connect Data Source UI page for the corresponding data source. The page is used by the administrator to enter the details of the data source.

   The following is a sample **dataSourceDef.xml** file:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
 <dataSourceType typeId="AM">
   <fields>
      <field fieldType="LIST" required="false" fieldId="version">
          <localizedLabel>
               <key>AMVersion</key>
               <defaultString>AM Version</defaultString>
          </localizedLabel>
          <options>
               <option key="5.1" label="5.1"/>
               <option key="5.2" label="5.2/9.3"/>
          </options>
               <preselectedValue>5.2</preselectedValue>
   </field>
      <field fieldType="LIST" required="false" fieldId="timezone">
  <generatorClass>com.hp.btoe.dw.dsm.server.TimeZonesGenerator</generatorClass>
               <localizedLabel>
                       <key>wizardTimeZone</key>
                       <defaultString>Time Zone</defaultString>
               </localizedLabel>
       </field>
       <field fieldType="SPACER" >
       </field>
       <field fieldType="LIST" required="false" fieldId="productType">
               <localizedLabel>
                       <key>wizardDBProductName</key>
                       <defaultString>Data Source Type</defaultString>
               </localizedLabel>
                   <options>
                       <option key="mssql" label="MSSQL"/>
                       <option key="oracle" label="Oracle"/>
                    </options>
               <preselectedValue>mssql</preselectedValue>
       </field>
       <field fieldType="SPACER" >
       </field>
       <field fieldType="TEXT" required="true" fieldId="user">
               <localizedHint>
                       <key>wizardUsernameHint</key>
                       <defaultString>&lt;&lt;Enter username&gt;&gt;</defaultString>
               </localizedHint>
               <localizedLabel>
                       <key>wizardUsername</key>
                       <defaultString>Username</defaultString>
               </localizedLabel>
 </field>
```

- The .XML file contains, per Content Pack, the definition of the different properties. Each Property field can have the following types:
  - **LIST.** The user must select from a list of parameters.
  - **TEXT.** The user must enter the relevant text in the field.
  - **SPACER.** Configures a space between fields in the UI.

- **PASSWORD.** The user enters an encrypted password in the field.
- **BOOLEAN.** The user selects a checkbox.

◦ The following options can be configured for field values:

- **preselectedValue.** The displayed default value.

  Example: For the AM Version field the default value is 5.2.

```
<key>AMVersion</key>
 <defaultString>AM Version</defaultString>
 </localizedLabel>
 <options>
 <option key="5.1" label="5.1"/>
 <option key="5.2" label="5.2/9.3"/>
 </options>
 <preselectedValue>5.2</preselectedValue>
 </field>
```

- **required.** A mandatory field. Mandatory fields are indicated by a red asterisk in the UI.

  Example: The **Username** field is mandatory:

```
</field>
 <field fieldType="TEXT" required="true" fieldId="user">
 <localizedHint>
 <key>wizardUsernameHint</key>
 <defaultString>&lt;&lt;Enter username&gt;&gt;</defaultString>
 </localizedHint>
 <localizedLabel>
 <key>wizardUsername</key>
 <defaultString>Username</defaultString>
 </localizedLabel>
 </field>
```

- **dependentField/Value.** A field that is displayed only when a specific value is selected.

  Example: The string **SID** is only enabled when the selected value in the dependent **productType** field is **Oracle**.

```
<localizedLabel>
 <key>wizardSid</key>
 <defaultString>SID</defaultString>
 </localizedLabel>
 <dependentField>productType</dependentField>
 <dependentValues>oracle</dependentValues>
 </field>
```

- **localizedHint.** Text that is displayed to assist the user in entering or selecting a value.

Example: The Username field displays a hint for the user: "Enter username".

```
<field fieldType="TEXT" required="true" fieldId="user">
 <localizedHint>
 <key>wizardUsernameHint</key>
 <defaultString>&lt;&lt;Enter username&gt;&gt;</defaultString>
 </localizedHint>
 <localizedLabel>
 <key>wizardUsername</key>
```

2. **Deploy the Configuration File:** When you have completed the .XMLfile, save it as **$HPBA_ HOME/ContentPacks/<CP_Name>/conf/dataSourceDef.xml**, in which **<CP_Name>** is the name of the Content Pack you are creating for the data source you want to integrate with ITBA.

Send Documentation Feedback

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on DCS Extractor SDK Guide (IT Business Analytics 10.10)

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to SW-Doc@hpe.com.

We appreciate your feedback!

Hewlett Packard
Enterprise

Java™
COMPATIBLE