↑Top

目次

改訂情報

変更年月日	変更内容
2014-01-01	初版
2014-05-01	第２版　下記を追加・変更しました ▪ 「*インポートの依存関係* 」を追加
2015-04-01	第３版　下記を追加・変更しました ▪ 「*revive* 」を追加

はじめに

本書の目的

本書ではジョブ情報のインポート・エクスポート機能の詳細について説明します。

説明範囲は以下のとおりです。

- ジョブ情報のインポート・エクスポートの概要
- ジョブ情報のインポート・エクスポートファイルのデータフォーマット
- ジョブ情報のインポート・エクスポートの実行方法
- ジョブ情報のインポート・エクスポートの実行オプション

対象読者

本書では次の利用者を対象としています。

- intra-mart Accel Platform のジョブを管理をする運用担当者
- ジョブ情報のインポート・エクスポート機能を利用したアプリケーションを開発する開発者

本書の構成

- *概要*

  ジョブ情報のインポート・エクスポートで扱う情報について説明します。

- *ファイルフォーマット*

  ジョブ情報（XML形式）のデータフォーマットについて説明します。

- *インポート*

  インポートの処理について説明します。更新モードについても説明します。

- *エクスポート*

  エクスポートの処理について説明します。

- *実行方法*

  インポート・エクスポートの実行方法について説明します。

概要

ジョブのインポート・エクスポート機能では、XML形式でジョブ情報のインポート・エクスポートを行います。
インポート・エクスポートで扱うXMLファイルのフォーマットについては「*XMLファイルフォーマット*」を参照してください。

インポート・エクスポートで扱う情報

インポート・エクスポートで扱うジョブ情報とは、ジョブスケジューラサービスで実行されるジョブに関連する以下の情報です。

ジョブカテゴリ情報

ジョブカテゴリの設定値を保持します。
ジョブカテゴリの階層情報および国際化情報で構成されています。

ジョブ情報

ジョブの設定値を保持します。
ジョブの国際化情報および実行プログラム、実行パラメータで構成されています。

ジョブネットカテゴリ情報

ジョブネットカテゴリの設定値を保持します。
ジョブネットカテゴリの階層情報および国際化情報で構成されています。

ジョブネット情報

ジョブネットの設定値を保持します。
ジョブネットの国際化情報およびジョブネットの構成情報、実行パラメータで構成されています。

トリガー情報

トリガーの設定値を保持します。
「日付指定」、「繰り返し指定」、「営業日指定」のいずれかのトリガー設定情報で構成されます。

注意

ジョブのインポート・エクスポート機能で扱うのはジョブの設定情報のみです。
ジョブの実行プログラム自体は対象外です。

ファイルフォーマット

この章では、ジョブのインポート・エクスポート機能で利用するファイルのフォーマットについて説明します。

XMLファイルフォーマット

<im-job-scheduler-data> タグ内に 1 つのジョブに関連する情報をすべてを記述します。
ジョブに関連する情報については「*インポート・エクスポートで扱う情報*」を参照してください。

複数のジョブに関する情報を一括で扱う場合は、XMLファイルのルートタグ内に`<im-job-scheduler-data>`` タグを複数記述します。

ジョブカテゴリ情報

<im-job-scheduler-data> タグ内に <job-category> タグを記述します。

```
<im-job-scheduler-data>
  <job-category id="sample-job-category">
    <parent-id>parent-job-category</parent-id>
    <localize locale="ja">
      <name>サンプルジョブカテゴリ</name>
    </localize>
  </job-category>
</im-job-scheduler-data>
```

<job-category> に記述できる設定は以下のとおりです。

- <job-category> に記述できる属性

属性	型	必須	デフォルト値	説明

属性	型	必須	デフォルト値	説明
id	文字列	○	（なし）	ジョブカテゴリを一意に識別するIDを指定します。
update-mode	文字列	×	merge	ジョブカテゴリをインポートする際の更新モードを指定します。 更新モードについては「*更新モード*」を参照してください。

- <job-category> に記述できる子要素

要素	型	必須	複数指定	デフォルト値	説明
parent-id	文字列	×	×	（なし）	親カテゴリのIDを指定します。
localize	*カテゴリ国際化情報*	×	○	（なし）	ジョブカテゴリの国際化情報を指定します。

ジョブネットカテゴリ情報

<im-job-scheduler-data> タグ内に <jobnet-category> タグを記述します。

```
<im-job-scheduler-data>
  <jobnet-category id="sample-jobnet-category">
    <parent-id>parent-jobnet-category</parent-id>
    <localize locale="ja">
      <name>サンプルジョブネットカテゴリ</name>
    </localize>
  </jobnet-category>
</im-job-scheduler-data>
```

<jobnet-category> に記述できる設定は以下のとおりです。

- <jobnet-category> に記述できる属性

属性	型	必須	デフォルト値	説明
id	文字列	○	（なし）	ジョブネットカテゴリを一意に識別するIDを指定します。

属性	型	必須	デフォルト値	説明
update-mode	文字列	×	merge	ジョブネットカテゴリをインポートする際の更新モードを指定します。 更新モードについては「*更新モード*」を参照してください。

- `<jobnet-category>` に記述できる子要素

要素	型	必須	複数指定	デフォルト値	説明
parent-id	文字列	×	×	（なし）	親カテゴリのIDを指定します。
localize	*カテゴリ国際化情報*	×	○	（なし）	ジョブネットカテゴリの国際化情報を指定します。

カテゴリ国際化情報

`<localize>` に記述できる設定は以下のとおりです。

- `<localize>` に記述できる属性

属性	型	必須	デフォルト値	説明
locale	文字列	○	（なし）	ロケールIDを指定します。

- `<localize>` に記述できる子要素

要素	型	必須	複数指定	デフォルト値	説明
name	文字列	×	×	（なし）	カテゴリの名前を指定します。

ジョブ情報

`<im-job-scheduler-data>` タグ内に `<job-detail>` タグを記述します。

```
<im-job-scheduler-data>
  <job-detail id="sample-job">
    <category-id>sample-job-category</category-id>
    <job-type>JAVA</job-type>
    <job-path>jp.co.intra_mart.sample.SampleJob</job-path>
    <parameter key="parameter-key">parameter-value</parameter>
    <localize locale="ja">
      <name>サンプルジョブ</name>
      <description></description>
    </localize>
  </job-detail>
</im-job-scheduler-data>
```

<job-detail> に記述できる設定は以下のとおりです。

- <job-detail> に記述できる属性

属性	型	必須	デフォルト値	説明
id	文字列	○	（なし）	ジョブを一意に識別するIDを指定します。
update-mode	文字列	×	merge	ジョブをインポートする際の更新モードを指定します。 更新モードについては「*更新モード*」を参照してください。

- <job-detail> に記述できる子要素

要素	型	必須	複数指定	デフォルト値	説明
category-id	文字列	×	×	（なし）	このジョブのカテゴリを指定します。
job-type	文字列	△ [1]	×	（なし）	ジョブ実行言語を指定します。 ・JAVA：JAVAプログラム ・SCRIPT：スクリプト開発モデル
job-path	文字列	△ [1]	×	（なし）	ジョブプログラムのパスを指定します。 job-type が JAVA の場合はジョブクラスのクラス名を指定します。 job-type が SCRIPT の場合はJSファイルのパスを指定します。

要素	型	必須	複数指定	デフォルト値	説明
parameter	*実行パラメータ情報*	×	○	（なし）	ジョブの実行パラメータを指定します。
localize	*ジョブ国際化情報*	×	○	（なし）	ジョブの国際化情報を指定します。

ジョブ国際化情報

<localize> に記述できる設定は以下のとおりです。

- <localize> に記述できる属性

属性	型	必須	デフォルト値	説明
locale	文字列	○	（なし）	ロケールIDを指定します。

- <localize> に記述できる子要素

要素	型	必須	複数指定	デフォルト値	説明
name	文字列	×	×	（なし）	ジョブの名前を指定します。
description	文字列	×	×	（なし）	ジョブの説明を指定します。

ジョブネット情報

<im-job-scheduler-data> タグ内に <jobnet> タグを記述します。

```
<im-job-scheduler-data>
  <jobnet id="sample-jobnet">
    <category-id>sample-jobnet-category</category-id>
    <parameter key="parameter-key">parameter-value</parameter>
    <localize locale="ja">
      <name>サンプルジョブネット</name>
      <description></description>
    </localize>
    <disallowConcurrent>true</disallowConcurrent>
    <serialize>
      <job-id>sample-job</job-id>
    </serialize>
  </jobnet>
</im-job-scheduler-data>
```

<jobnet> に記述できる設定は以下のとおりです。

- <jobnet> に記述できる属性

属性	型	必須	デフォルト値	説明
id	文字列	○	（なし）	ジョブネットを一意に識別するIDを指定します。
update-mode	文字列	×	merge	ジョブネットをインポートする際の更新モードを指定します。 更新モードについては「*更新モード*」を参照してください。

- <jobnet> に記述できる子要素

要素	型	必須	複数指定	デフォルト値	説明
category-id	文字列	×	×	（なし）	このジョブネットのカテゴリを指定します。
parameter	*実行パラメータ情報*	×	○	（なし）	ジョブネットの実行パラメータを指定します。
localize	*ジョブネット国際化情報*	×	○	（なし）	ジョブネットの国際化情報を指定します。
disallowConcurrent	真偽値	×	×	false	同時実行を禁止する場合、trueを指定します。
serialize	*実行ジョブリスト情報*	△ [1]	×	（なし）	このジョブネットで実行されるジョブを指定します。

ジョブネット国際化情報

<localize> に記述できる設定は以下のとおりです。

- <localize> に記述できる属性

属性	型	必須	デフォルト値	説明
locale	文字列	○	（なし）	ロケールIDを指定します。

- <localize> に記述できる子要素

要素	型	必須	複数指定	デフォルト値	説明
name	文字列	×	×	（なし）	ジョブネットの名前を指定します。
description	文字列	×	×	（なし）	ジョブネットの説明を指定します。

実行ジョブリスト情報

<serialize> に記述できる設定は以下のとおりです。

- <serialize> に記述できる子要素

要素	型	必須	複数指定	デフォルト値	説明
job-id	文字列	○	○	（なし）	実行するジョブのジョブIDを指定します。 ジョブは指定された順に実行されます。

トリガー情報

<im-job-scheduler-data> タグ内に <trigger> タグを記述します。

```
<im-job-scheduler-data>
  <trigger id="sample-trigger">
    <jobnet-id>sample-jobnet</jobnet-id>
    <description></description>
    <enable>false</enable>
    <start-date>2014-03-24T17:21:57.000+09:00</start-date>
    <repeat>
      <count>1</count>
    </repeat>
  </trigger>
</im-job-scheduler-data>
```

<trigger> に記述できる設定は以下のとおりです。

- <trigger> に記述できる属性

属性	型	必須	デフォルト値	説明
id	文字列	○	（なし）	トリガーを一意に識別するIDを指定します。

属性	型	必須	デフォルト値	説明
update-mode	文字列	×	merge	トリガーをインポートする際の更新モードを指定します。 更新モードについては「*更新モード*」を参照してください。

- <trigger> に記述できる子要素

要素	型	必須	複数指定	デフォルト値	説明
jobnet-id	文字列	△ [1]	×	（なし）	このトリガーで実行されるジョブネットのIDを指定します。
parameter	*実行パラメータ情報*	×	○	（なし）	トリガーの実行パラメータを指定します。
description	文字列	×	×	（なし）	トリガーの説明を記述します。
enable	真偽値	×	×	false	このトリガーを有効にする場合、trueを指定します。
start-date	日時（Datetime）	×	×	（なし）	この要素は非推奨です トリガーの開始日を指定します。 トリガーを有効にしている場合、この開始日以降が有効です。
end-date	日時（Datetime）	×	×	（なし）	この要素は非推奨です トリガーの終了日を指定します。 トリガーを有効にしている場合、この終了日まで有効です。
start-point	文字列	×	×	（なし）	この要素は非推奨です ジョブネットを実行する際の開始位置を指定します。
repeat	*繰り返し指定トリガー情報*	△ [1]	×	（なし）	繰り返し指定のトリガーを設定します。
datetime	*日時指定トリガー情報*	△ [1]	×	（なし）	日時指定のトリガーを設定します。

要素	型	必須	複数指定	デフォルト値	説明
business-day	*営業日指定トリガー情報*	△ [1]	×	（なし）	営業日指定のトリガーを設定します。

注意

<repeat> 、<datetime> 、<business-day> 要素は 1 つのトリガー情報に対して 1 つのみ設定できます。
複数のトリガー情報を設定する場合は、 *<im-job-scheduler-data>`* 要素を追加してください。

繰り返し指定トリガー情報

<trigger> タグ内に <repeat> タグを記述します。

```
<repeat>
   <count>5</count>
   <interval>10</interval>
</repeat>
```

<repeat> に記述できる設定は以下のとおりです。

- <repeat> に記述できる子要素

要素	型	必須	複数指定	デフォルト値	説明
count	数値	×	×	（なし）	繰り返し回数を指定します。 指定がない場合は、無限に繰り返しを行います。
interval	数値	×	△	（なし）	繰り返し間隔（秒）を指定します。 繰り返し回数が 1 回以上の場合、必須です。

日時指定トリガー情報

<trigger> タグ内に <datetime> タグを記述します。

```
<datetime>
  <time-zone>Asia/Tokyo</time-zone>
  <years>2015</years>
  <months>11</months>
  <days-of-month>31</days-of-month>
  <hours>0</hours>
  <minutes>0</minutes>
</datetime>
```

<datetime> に記述できる設定は以下のとおりです。

- <datetime> に記述できる子要素

要素	型	必須	複数指定	デフォルト値	説明
time-zone	文字列	△ [1]	×	（なし）	指定された日時のタイムゾーンIDを指定します。
years	数値	×	○	（なし）	年を指定します。
months	数値	×	○	（なし）	月を指定します。
days-of-month	数値	×	○	（なし）	日を指定します。
days-of-week	数値	×	○	（なし）	曜日を表す次の数値を指定します。 1：日曜 ～ 7：土曜
hours	数値	×	○	（なし）	時を指定します。
minutes	数値	×	○	（なし）	分を指定します。
seconds	数値	×	○	（なし）	秒を指定します。

営業日指定トリガー情報

<trigger> タグ内に <business-day> タグを記述します。

```
<business-day>
  <calendar-id>JPN_CAL</calendar-id>
  <time-zone>Asia/Tokyo</time-zone>
  <hours>0</hours>
  <minutes>0</minutes>
</business-day>
```

<business-day> に記述できる設定は以下のとおりです。

- <business-day> に記述できる子要素

要素	型	必須	複数指定	デフォルト値	説明
calendar-id	文字列	○	×	（なし）	使用するカレンダーのカレンダーIDを指定します。
time-zone	文字列	△ [1]	×	（なし）	指定された日時のタイムゾーンIDを指定します。
hours	数値	×	○	（なし）	時を指定します。
minutes	数値	×	○	（なし）	分を指定します。
seconds	数値	×	○	（なし）	秒を指定します。

実行パラメータ情報

<parameter> には、key 属性にパラメータのキーを、要素の値にパラメータの値を指定してください。

```
<parameter key="file">sample.xml</parameter>
<parameter key="format-xml">true</parameter>
```

注釈

[1] *(1, 2, 3, 4, 5, 6, 7, 8, 9)* 更新モードが「replace」、または新規登録の場合、必須です。

要素に指定する値について

- ロケールID

  ロケールマスタに定義されているロケールのロケールIDを指定する必要があります。

  > **コラム**
  > ロケールマスタの設定については「設定ファイルリファレンス - ロケールマスタ」を参照してください。

- タイムゾーンID

  タイムゾーンマスタに定義されているタイムゾーンのタイムゾーンIDを指定する必要があります。

  > **コラム**
  > タイムゾーンマスタの設定については「設定ファイルリファレンス - タイムゾーン」を参照してください。

- カレンダーID

カレンダー情報として登録されているカレンダーのカレンダーIDを指定する必要があります。

- 日時指定トリガーの曜日指定

  1（日曜日）、2（月曜日）、3（火曜日）、4（水曜日）、5（木曜日）、6（金曜日）、または、7（土曜日）のみ指定可能です。

XMLファイルサンプル

以下はインポート・エクスポートのXMLファイルの例です。

```
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns="http://www.intra-mart.jp/job-scheduler/data">
  <im-job-scheduler-data>
    <job-category id="sample-job-category">
      <parent-id>parent-job-category</parent-id>
      <localize locale="ja">
        <name>サンプルジョブカテゴリ</name>
      </localize>
    </job-category>
  </im-job-scheduler-data>
  <im-job-scheduler-data>
    <jobnet-category id="sample-jobnet-category">
      <parent-id>parent-jobnet-category</parent-id>
      <localize locale="ja">
        <name>サンプルジョブネットカテゴリ</name>
      </localize>
    </jobnet-category>
  </im-job-scheduler-data>
  <im-job-scheduler-data>
    <job-detail id="sample-job">
      <category-id>sample-job-category</category-id>
      <job-type>JAVA</job-type>
      <job-path>jp.co.intra_mart.sample.SampleJob</job-path>
      <parameter key="parameter-key">parameter-value</parameter>
      <localize locale="ja">
        <name>サンプルジョブ</name>
        <description></description>
      </localize>
    </job-detail>
  </im-job-scheduler-data>
  <im-job-scheduler-data>
    <jobnet id="sample-jobnet">
      <category-id>sample-jobnet-category</category-id>
      <parameter key="parameter-key">parameter-value</parameter>
      <localize locale="ja">
        <name>サンプルジョブネット</name>
        <description></description>
      </localize>
      <disallowConcurrent>true</disallowConcurrent>
      <serialize>
```

```
            <job-id>sample-job</job-id>
          </serialize>
        </jobnet>
    </im-job-scheduler-data>
    <im-job-scheduler-data>
        <trigger id="sample-trigger">
          <jobnet-id>sample-jobnet</jobnet-id>
          <description></description>
          <enable>false</enable>
          <start-date>2014-03-24T17:21:57.000+09:00</start-date>
          <repeat>
            <count>1</count>
          </repeat>
        </trigger>
    </im-job-scheduler-data>
</root>
```

インポート

項目

ジョブのインポートはXML形式で行うことが可能です。
インポートファイルのフォーマットについては「*XML ファイルフォーマット*」を参照してください。

この章では、インポートがどのように行われるかを説明します。
また *更新モード* による更新方法の違いについて説明します。

更新モード

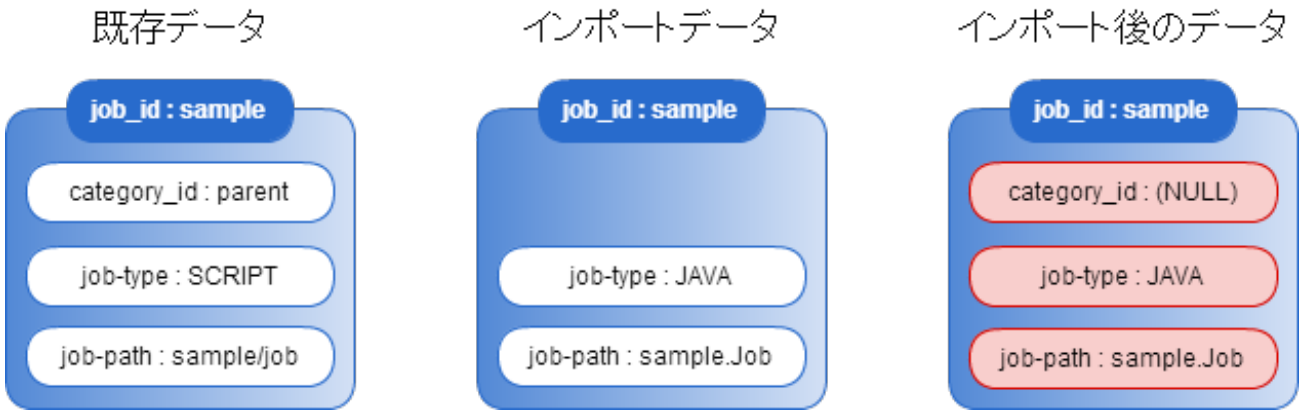
更新モードを利用することで、インポートファイルのデータがデータベース上に存在する場合（更新を行う場合）の
データの更新方法を変更することができます。
更新モードには *merge* と *replace* が提供されています。
各インポート要素の update-mode 属性を指定することでモードを設定します。
特に指定していない場合は、*merge* モードで動作します。

merge

インポートファイルのデータとデータベース上のデータをマージして更新します。
インポートファイルに存在しない項目は既存のデータをそのまま設定されます。

replace

データベース上のデータをインポートファイルのデータに置き換えて更新します。
インポートファイルに存在しない項目はデフォルト値が設定されます。



revive

インポートファイルのデータと同じキーのデータがデータベースに存在しない場合、インポートファイルのデータが新規追加されます。
また、インポートファイルのデータと同じキーのデータがデータベースに存在するが、多言語データが欠損している場合は多言語データのみインポートを行います。

インポート実行オプション

ジョブのインポートでは、インポートで扱うファイルなどの設定を変更するためのオプションが用意されています。
オプションはジョブまたはジョブネットの実行パラメータに指定することで設定できます。

実行オプションには、以下のオプションが指定できます。

名前	キー名	型	デフォルト値	導入バージョン
エンコーディング	encoding	文字列	UTF-8	2012 Autumn
ファイルパス	file	文字列	（なし）	2012 Autumn
コミット件数	commit-count	数値	0	2012 Autumn
XML検証フラグ	validate-xml	真偽値	true	2012 Autumn
無効にするトリガ	disable_trigger	文字列	RepeatTrigger	2013 Winter

エンコーディング

キー名　encoding

インポートするXMLファイルの文字エンコーディングを指定します。

ファイルパス

キー名　file

インポートするXMLファイルのパス（パブリックストレージのルートからの相対パス）を指定します。

コミット件数

キー名　commit-count

インポート処理で、コミットを行うまでのデータ件数を指定します。
コミット件数に「0」（デフォルト値）が指定された場合は、インポート処理が完了するまでコミットが行われません。

> **注意**
> commit-count を指定した場合、インポート実行元で管理しているトランザクションがコミットされる可能性があります。

XML検証フラグ

キー名　validate-xml

インポートするXMLファイルの構文を検証するかどうかを指定します。

指定する値	説明
true	XML構文の検証を行います。（デフォルト値）
false	XML構文の検証を行いません。

無効にするトリガ

キー名　disable_trigger

インポート実行時に無効扱いで登録するトリガを指定します。
複数のトリガをまとめて無効にしたい場合は、カンマ区切りで値を指定してください。

指定する値	説明

指定する値	説明
RepeatTrigger	繰り返し指定を無効にします。（デフォルト値）
DatetimeTrigger	日時指定を無効にします。
BusinessDayTrigger	営業日指定を無効にします。

注意
RepeatTrigger が有効な場合、インポート直後にジョブの実行が開始されます。

インポートの依存関係

インポートするデータによっては、他のデータが登録されていることが前提となっていることがあります。
その場合、前提となるデータが未登録のままインポートを実施すると、インポートに失敗する可能性があります。

インポートデータの依存関係は、以下の通りです。

- ジョブカテゴリ情報
  - 親となるジョブカテゴリを指定する場合、対象のジョブカテゴリがすでに登録されている必要があります。
- ジョブネットカテゴリ情報
  - 親となるジョブネットカテゴリを指定する場合、対象のジョブネットカテゴリがすでに登録されている必要があります。
- ジョブ情報
  - ジョブカテゴリを指定する場合、対象のジョブカテゴリがすでに登録されている必要があります。
- ジョブネット情報
  - ジョブネットカテゴリを指定する場合、対象のジョブネットカテゴリがすでに登録されている必要があります。
- トリガー情報
  - 対象のジョブネットが登録されている必要があります。

エクスポート

項目
- エクスポート実行オプション

ジョブのエクスポートはXML形式で行うことが可能です。
エクスポートファイルのフォーマットについては「*XML ファイルフォーマット*」を参照してください。

この章では、エクスポートがどのように行われるかを説明します。

エクスポート実行オプション

ジョブのエクスポートでは、エクスポートで出力するファイルなどの設定を変更するためのオプションが用意されています。
オプションはジョブまたはジョブネットの実行パラメータに指定することで設定できます。

実行オプションには、以下のオプションが指定できます。

名前	キー名	型	デフォルト値	導入バージョン
エンコーディング	encoding	文字列	UTF-8	2012 Autumn
ファイルパス	file	文字列	（なし）	2012 Autumn
XML整形フラグ	format-xml	真偽値	false	2012 Autumn
ルートタグ名	root-tag-name	文字列	root	2012 Autumn
書き込み件数	flush-count	数値	5000	2012 Autumn

エンコーディング

キー名　encoding

エクスポートするXMLファイルの文字エンコーディングを指定します。

ファイルパス

キー名　file

エクスポートするXMLファイルのパス（パブリックストレージのルートからの相対パス）を指定します。

XML整形フラグ

キー名 format-xml

エクスポートするXMLファイルを整形するかどうかを指定します。

指定する値	説明
true	XMLの整形を行います。
false	XMLの整形を行いません。（デフォルト値）

ルートタグ名

キー名 root-tag-name

エクスポートするXMLファイルのルートタグ名を指定します。

書き込み件数

キー名 flush-count

エクスポートするXMLファイルに一度に書き込むデータ件数を指定します。

実行方法

項目

この章では、インポート・エクスポートを実行する手段を紹介します。

注意

インポートしたデータをエクスポートする場合、またはエクスポートしたデータをインポートする場合は、「エンコーディング」などの対応するオプションは同じ値を指定する必要があります。

ジョブスケジューラを利用する

ジョブスケジューラの機能を利用してインポート・エクスポートを実行する方法を紹介します。
ジョブスケジューラの詳細については「ジョブスケジューラ仕様書」を参照してください。
intra-mart Accel Platform ではジョブのインポート・エクスポートを行うためのジョブ・ジョブネットを提供しています。
この項では、intra-mart Accel Platform が標準で提供しているジョブのインポート・エクスポートを行うジョブ・ジョブネットの情報を紹介します。

ジョブ

- ジョブインポート

ジョブカテゴリ	テナントマスタ > インポート
ジョブID	job-import
ジョブ名	ジョブインポート

- ジョブエクスポート

ジョブカテゴリ	テナントマスタ > エクスポート
ジョブID	job-scheduler-export
ジョブ名	ジョブエクスポート

ジョブネット

- ジョブインポート

ジョブネットカテゴリ	テナントマスタ > インポート
ジョブネットID	job-import-jobnet
ジョブネット名	ジョブインポート

- ジョブエクスポート

ジョブネットカテゴリ	テナントマスタ > エクスポート
ジョブネットID	job-export-jobnet
ジョブネット名	ジョブエクスポート

コラム

ジョブスケジューラ利用時のオプションについて

ジョブスケジューラを利用してインポート・エクスポートを実行する場合は、ジョブ・ジョブネットの実行パラメータにオプションを指定します。

オプションの詳細は、「*インポート実行オプション*」および「*エクスポート実行オプション*」を参照してください。

コラム

ジョブスケジューラ利用時のトランザクション管理について

ジョブスケジューラを利用したインポートでは、オプション commit-count を指定しない場合、インポート処理が完了後に一括してコミットを行います。
必要に応じてオプション commit-count の値を変更してご利用ください。

Javaから実行する

JavaのAPIを利用してインポート・エクスポートを実行する方法を紹介します。

インポート

DataImportExecutor#importData(String, InputStream, Map) を利用してインポートを行います。

- 完全修飾クラス名

  jp.co.intra_mart.foundation.data.importer.DataImportExecutor

第１引数にはインポータIDを指定します。インポータIDは以下を利用します。

- インポータID

  `jp.co.intra_mart.import.StandardJobSchedulerXmlImporter`

第２引数にはインポート元を `InputStream` で指定します。

第３引数にはインポートオプションを `Map<String, Object>` で指定します。
詳細は、「*インポート実行オプション*」を参照してください。

`DataImportExecutor` の詳細については「DataImportExecutorクラスのAPIリスト」を参照してください。

以下はXML形式でインポートを行うサンプルプログラムです。

```
package sample;

import java.io.IOException;
import java.io.InputStream;
import java.util.HashMap;
import java.util.Map;

import jp.co.intra_mart.foundation.data.OptionKeyName;
import jp.co.intra_mart.foundation.data.exception.DataImporterException;
import jp.co.intra_mart.foundation.data.importer.DataImportExecutor;
import jp.co.intra_mart.foundation.service.client.file.PublicStorage;

/**
 * ジョブのインポートを行うクラスです。
 */
public class JobImporter {

    private static final String IMPORTER_ID =
"jp.co.intra_mart.import.StandardJobSchedulerXmlImporter";

    /**
     * ジョブのインポートを行います。
     * @throws DataImporterException インポートで何らかの例外が発生した場合。
     */
    public void doImport() throws DataImporterException {
        final DataImportExecutor executor = new DataImportExecutor();
        final Map<String, Object> options = new HashMap<String, Object>();
        options.put(OptionKeyName.ENCODING.value(), "UTF-8");
        options.put(OptionKeyName.VALIDATE_XML.value(), true);
        options.put(OptionKeyName.COMMIT_COUNT.value(), 100);
        // パブリックストレージ直下のaccount.xmlを選択
        final PublicStorage storage = new PublicStorage("job-scheduler.xml");
        try {
            // PublicStorageからInputStreamを取得
            final InputStream stream = storage.open();
            try {
                executor.importData(IMPORTER_ID, stream, options);
            } finally {
                stream.close();
            }
        } catch (final IOException e) {
            throw new DataImporterException(e);
        }
    }

}
```

注意

第２引数の InputStream を指定した場合は、オプション file は利用できません。
オプション file を利用したい場合は、第２引数に null を指定してください。

DataExportExecutor#exportData(String, OutputStream, Map) を利用してエクスポートを行います。

- 完全修飾クラス名

  jp.co.intra_mart.foundation.data.exporter.DataExportExecutor

第１引数にはエクスポータIDを指定します。エクスポータIDは以下を利用します。

- エクスポータID

  jp.co.intra_mart.export.StandardJobSchedulerXmlExporter

第２引数にはエクスポート先を OutputStream で指定します。

第３引数にはエクスポートオプションを Map<String, Object> で指定します。
詳細は、「*エクスポート実行オプション*」を参照してください。

DataExportExecutor の詳細については「DataExportExecutorクラスのAPIリスト」を参照してください。

以下はXML形式でエクスポートを行うサンプルプログラムです。

```
package sample;

import java.io.IOException;
import java.io.OutputStream;
import java.util.HashMap;
import java.util.Map;

import jp.co.intra_mart.foundation.data.OptionKeyName;
import jp.co.intra_mart.foundation.data.exception.DataExporterException;
import jp.co.intra_mart.foundation.data.exporter.DataExportExecutor;
import jp.co.intra_mart.foundation.service.client.file.PublicStorage;

/**
 * ジョブのエクスポートを行うクラスです。
 */
public class JobExporter {

    private static final String EXPORTER_ID =
"jp.co.intra_mart.export.StandardJobSchedulerXmlExporter";

    /**
     * ジョブのエクスポートを行います。
     * @throws DataExporterException エクスポートで何らかの例外が発生した場合。
     */
    public void doExport() throws DataExporterException {
        final DataExportExecutor executor = new DataExportExecutor();
        final Map<String, Object> options = new HashMap<String, Object>();
        options.put(OptionKeyName.ENCODING.value(), "UTF-8");
        options.put(OptionKeyName.FORMAT_XML.value(), false);
        options.put(OptionKeyName.FETCH_COUNT.value(), 10);
        // パブリックストレージ直下のaccount.xmlを選択
        final PublicStorage storage = new PublicStorage("job-scheduler.xml");
        try {
            // PublicStorageからOutputStreamを取得
            final OutputStream stream = storage.create();
            try {
                executor.exportData(EXPORTER_ID, stream, options);
            } finally {
                stream.close();
            }
        } catch (final IOException e) {
            throw new DataExporterException(e);
        }
    }

}
```

注意

第２引数の OutputStream を指定した場合は、オプション file は利用できません。
オプション file を利用したい場合は、第２引数に null を指定してください。

スクリプト開発モデルプログラムから実行する

スクリプト開発モデルのAPIを利用してインポート・エクスポートを実行する方法を紹介します。

インポート

`DataImportExecutor#importData(String, ByteReader, Object)` を利用してインポートを行います。

第１引数にはインポータIDを指定します。インポータIDは以下を利用します。

- インポータID

  `jp.co.intra_mart.import.StandardJobSchedulerXmlImporter`

第２引数にはインポート元を `ByteReader` で指定します。

第３引数にはインポートオプションをObject形式で指定します。
詳細は、「*インポート実行オプション*」を参照してください。

`DataImportExecutor` の詳細については「DataImportExecutorオブジェクトのAPIリスト」を参照してください。

以下はXML形式でインポートを行うサンプルプログラムです。

```
var IMPORTER_ID = 'jp.co.intra_mart.import.StandardJobSchedulerXmlImporter';

/**
 * ジョブのインポートを行います。
 */
function doImport() {
    var executor = new DataImportExecutor();
    var options = {
        'encoding': 'UTF-8',
        'validate-xml': true,
        'commit-count': 100
    };
    // パブリックストレージ直下の「job-scheduler.xml」を選択
    var storage = new PublicStorage('job-scheduler.xml');
    // ファイルからデータをインポート
    storage.openAsBinary(function(reader, error) {
        if (error) {
            // ファイルの読み込みに失敗 -> 例外処理
            Logger.getLogger().error(error.message);
            return;
        }
        result = executor.importData(IMPORTER_ID, reader, options);
        if (result.error) {
            // インポート失敗 -> 例外処理
            Logger.getLogger().error(result.errorMessage);
        }
    });
}
```

注意

第２引数の ByteReader を指定した場合は、オプション file は利用できません。
オプション file を利用したい場合は、第２引数に null を指定してください。

エクスポート

DataExportExecutor#exportData(String, ByteWriter, Object) を利用してエクスポートを行います。

第１引数にはエクスポータIDを指定します。エクスポータIDは以下を利用します。

- エクスポータID

  jp.co.intra_mart.export.StandardJobSchedulerXmlExporter

第２引数にはエクスポート元を ByteWriter で指定します。

第３引数にはエクスポートオプションをObject形式で指定します。
詳細は、「*エクスポート実行オプション*」を参照してください。

DataExportExecutor の詳細については「DataExportExecutorオブジェクトのAPIリスト」を参照してください。

以下はXML形式でインポートを行うサンプルプログラムです。

```
var EXPORTER_ID = 'jp.co.intra_mart.export.StandardJobSchedulerXmlExporter';

/**
 * ジョブのエクスポートを行います。
 */
function doExport() {
    var executor = new DataExportExecutor();
    var options = {
        'encoding': 'UTF-8',
        'format-xml': false,
        'fetch-count': 10
    };
    // パブリックストレージ直下の「job-scheduler.xml」を選択
    var storage = new PublicStorage('job-scheduler.xml');
    // ファイルにデータをエクスポート
    storage.createAsBinary(function(writer, error) {
        if(error) {
            // ファイルの作成に失敗 -> 例外処理
            Logger.getLogger().error(error.message);
            return;
        }
        var result = executor.exportData(EXPORTER_ID, writer, options);
        if (result.error) {
            // エクスポートに失敗 -> 例外処理
            Logger.getLogger().error(result.errorMessage);
        }
    });
}
```

注意

第２引数の ByteWriter を指定した場合は、オプション file は利用できません。
オプション file を利用したい場合は、第２引数に null を指定してください。