目次

改訂情報

変更年月日	変更内容
2013-04-01	初版
2013-10-01	第2版 下記を追加・変更しました ▪ 「*intra-mart Accel Documents へのアクセス* 」のサンプルプログラムを修正しました。 ▪ 「*タグの作成*」、「*タグの付与*」に作成可能なタグ色を追加しました。 ▪ 「*セキュリティ定義の作成* 」を追加しました。 ▪ 「*クラスの定義作成* 」のサンプルプログラムを修正しました。 ▪ 「*属性定義の作成* 」のサンプルプログラムを修正しました。
2014-06-30	第3版 下記を追加しました ▪ 「*付録* 」を追加しました。 ▪ 「*intra-mart Accel Documentsが提供する画面アイテム* 」を追加しました。
2015-03-13	第4版 下記を追加・修正しました ▪ 「*文書作成* 」を修正しました。 ▪ 「*チェックアウト/チェックイン* 」を修正しました。 ▪ 「*属性変更* 」を修正しました。 ▪ 「*保管定義の作成* 」を追加しました。 ▪ 「*クラスの定義作成* 」を修正しました。 ▪ 「*プリファレンスの利用* 」のプリファレンスの使用箇所を修正しました。
2016-08-01	第5版 下記を追加・修正しました ▪ 「*保管設定* 」を追加しました。 ▪ 「*タイムスタンプ設定* 」を追加しました。 ▪ 「*保管定義の作成* 」を修正しました。 ▪ 「*プリファレンスの利用* 」のプリファレンスの使用箇所を修正しました。
2017-04-01	第6版 下記を追加しました ▪ 「*ドロワ作成* 」のサンプルプログラムを追加しました。

変更年月日	変更内容
2018-04-01	第7版 下記を追加・修正しました ・「*文書作成*」のサンプルプログラムを修正しました。 ・「*キャビネット作成*」にノートを追加しました。 ・「*アプリケーション種別「BIS-BIS フロー」で利用できる画面アイテム*」を修正しました。

はじめに

本書の目的

本書では、intra-mart Accel Documentsを利用したプログラムを開発する場合の基本的な方法をサンプルプログラムの実装例を通して説明します。
intra-mart Accel Documentsを利用して開発を行う前にお読みください。

> **コラム**
>
> intra-mart Accel Documentsのデータモデルやインターフェースの詳細は、次を参照してください。
>
> - 「intra-mart Accel Documents/ 仕様書」
> - 「intra-mart Accel Documents/ JavaDoc」

> **注意**
>
> 本書では、「intra-mart Accel Documents」の仕様を記載しています。

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を理解している
- intra-mart Accel Documentsを理解している
- 次の条件を満たす、 intra-mart Accel Documentsを利用する開発者
  - Javaプログラミング言語を理解している開発者

本書の構成

本書では、「intra-mart Accel Documents」で次を操作する場合のサンプルプログラムについて説明します。

- *セッション操作*
  intra-mart Accel Documentsに接続するサンプルプログラムについて説明します。
  intra-mart Accel Documentsを利用したプログラムを開発する場合は必要ですので、必ず読んでください。
- *一覧操作*
  ドロワやフォルダ、文書などの一覧を取得する場合のサンプルプログラムについて説明します。
- *フォルダ操作*
  フォルダを作成する場合のサンプルプログラムについて説明します。
- *文書操作*
  文書の作成や属性変更、アクセス権の変更など、文書に対する操作を行う場合のサンプルプログラムについて説明します。
- *検索操作*
  文書などの検索操作を行う場合のサンプルプログラムについて説明します。
- *タグ操作*
  タグを操作する場合のサンプルプログラムについて説明します。
- *クラス操作*

クラスや属性、セキュリティの定義、保管定義を作成する場合のサンプルプログラムについて説明します。

- *キャビネット操作*
  キャビネットを作成する場合のサンプルプログラムについて説明します。
- *ドロワ操作*
  ドロワを作成する場合のサンプルプログラムについて説明します。
- *プリファレンス操作*
  設定データを保持する場合のサンプルプログラムについて説明します。

> **注意**
> サンプルプログラムで指定しているユーザは、intra-mart Accel Platformのサンプルデータセットアップで作成されるユーザを利用しています。

セッション操作

intra-mart Accel Documents へのアクセス

Accdel Documents にアクセスするためには、まず、KnRepositoryクラスのインスタンスを取得し、
そのインスタンスからKnSessionやKnServiceAdminsessionなどのセッションクラスのインスタンスを生成します。
各セッションクラスのインスタンスで利用可能な機能は以下の通りです。

- KnSessionクラスのインスタンスは、キャビネット単位の一般的な操作およびキャビネット単位の管理操作で利用するAPIにアクセスできます。
- KnServiceAdminSessionクラスのインスタンスは、テナント全体を管理する操作で利用するAPIにアクセスできます。
- KnGuestSessionクラスのインスタンスは、ログインしていないゲストユーザの操作で利用するAPIにアクセスできます。

KnRepositoryクラスのインスタンス取得、および各セッションクラスのインスタンス生成と利用方法について説明します。

サンプルプログラム

KnRepository取得の例

「acceldocuments.properties.xml」に記載された設定内容で、KnRepositoryクラスのインスタンスを取得するサンプルプログラムです。

```
// アプリケーションの設定ファイルからプロパティを作成します。
FileInputStream fis = new FileInputStream("C:/sample/acceldocuments.properties.xml");

// 設定をPropertiesとして取得します。
Properties properties = new Properties();
try {
  properties.loadFromXML(fis);
} finally {
  fis.close();
}

// KnRepositoryを取得します。
KnRepository repository = KnFactory.getRepository(properties);
```

KnServiceAdminSession生成の例

KnServiceAdminSession生成のサンプルコードを記載します。

```
// アカウントコンテキストのユーザを取得します。
String userCode = Contexts.get(AccountContext.class).getUserCd();

// 文書リポジトリ用のユーザ識別子を取得します。
UgId userId = KnUgIdUtil.getUserUgId(userCode);

// サービス管理セッションを生成します。
KnServiceAdminSession serviceAdminSession = repository.createServiceAdminSession(userId);
```

KnSession生成の例

KnSession生成のサンプルコードを記載します。

KnSessionを生成するためには、事前にキャビネットを作成しておく必要があります。

> **コラム**
> キャビネットの作成方法は *キャビネット作成* を参照してください。

```
// クライアントのタイプを指定します。操作履歴に記録されます。
String clientType = "Accel Documents";

// クライアントのアドレスを指定します。通常は、ServletRequest#getRemoteAddr() などで取得したIPアドレスです。
String clientAddress = "127.0.0.1";

// アカウントコンテキストのユーザを取得します。
String userCode = Contexts.get(AccountContext.class).getUserCd();

// 文書リポジトリ用のユーザ識別子を取得します。
UgId userId = KnUgIdUtil.getUserUgId(userCode);

// セッションを生成します。
KnSession session = repository.createSession(cabinetId, userId, clientType, clientAddress);
```

KnGuestSession生成の例

KnGuestSession生成のサンプルコードを記載します。

```
// クライアントのタイプを指定します。操作履歴に記録されます。
String clientType = "Accel Documents";

// クライアントのアドレスを指定します。通常は、ServletRequest#getRemoteAddr() などで取得したIPアドレスです。
String clientAddress = "127.0.0.1";

// ゲストセッションを生成します。
KnGuestSession guestSession = repository.createGuestSession(cabinetId, clientType, clientAddress);
```

KnSessionによるAPI利用の例

KnSessionを使用して、オブジェクト削除のAPIを呼び出すサンプルコードです。

> **コラム**
> サンプルコード中に出現するsessionがKnSessionクラスのインスタンスです。

```
// 削除するオブジェクトのIDを指定します。
ObjectId objectId = ObjectId.fromIdString("kn:folder-01");

// オブジェクトを削除します。
session.removeObject(objectId);
```

> **コラム**
> KnSessionを使用した操作をひと通り終了したら、KnSession#close()でセッションをクローズしてください。

KnSessionを管理者モードに設定する例

KnSessionを管理者モードに設定するサンプルコードを記載します。

管理者モードでは、キャビネット内のオブジェクトにすべての権限が付与された状態でKnSessionの各APIを呼び出すことができます。

> **コラム**
> キャビネット管理者として登録されていないユーザは設定できません。

```
// 現在の状態を取得します。
boolean enabled = session.isAdministratorPrivilegesEnabled();

try {
  // 管理者権限を有効化します。
  session.setAdministratorPrivilegesEnabled(true);

  // この部分でKnSessionから各APIを呼び出します。
  // ...

} finally {
  // 管理者権限をもとの状態に戻します。
  session.setAdministratorPrivilegesEnabled(enabled);
}
```

一覧操作

ドロワー一覧

概要

キャビネット内のドロワー一覧の取得について記載します。

サンプルプログラム

ドロワー一覧を取得するサンプルコードを記載します。

```
// ここでは、フィルタリングしない条件を指定したサンプルを記載しています。
KnFilter filter = KnFilter.FULL;

// フェッチしてくる属性の情報をSetで用意します。
Set<AttributeId> fetchAttributeIds = new HashSet<AttributeId>();
// 名前属性をフェッチしてくる属性に含めます。
AttributeId nameAttrId = KnSystemSchemas.OBJECT_NAME_ATTRIBUTE_ID;
fetchAttributeIds.add(nameAttrId);

// フェッチ条件を生成します。
KnFetch fetch = new KnFetch(fetchAttributeIds, false, false);

// 名前によるソート条件を生成します。
AttributeSortOrder attrSortOrder = new AttributeSortOrder(nameAttrId, Direction.ASCEND);
List<SortOrder> sortOrders = new ArrayList<SortOrder>();
sortOrders.add(attrSortOrder);
SortOrderList sortOrderList = new SortOrderList(sortOrders, null);

// ドロワー一覧を取得します。
BaseIterable<KnDrawer> drawers =
        session.getPublicDrawers(filter, fetch, sortOrderList);
// 取得結果を標準出力に書き出します。
try {
    for (KnDrawer drawer : drawers) {
        System.out.println(drawer.getName());
    }
} finally {
    drawers.close();
}
```

以上のコードでアクセス可能なドロワー一覧を取得できます。 ドロワー一覧は、名前の昇順でソートされて取得されます。

> **コラム**
> BaseIterableなどのイテラブルは、不要になったら、必ずクローズしてください。

フォルダ/文書一覧

概要

ドロワまたはフォルダ直下のフォルダまたは文書一覧の取得について記載します。

サンプルプログラム

サンプルコードでは、フォルダのIDが、”kn:folder-20”のときを例に記載しています。

```
// 一覧を取得するドロワまたはフォルダのIDを指定します。
ObjectId parentObjectId = ObjectId.fromIdString("kn:folder-20");

// ここでは、フィルタリングしない条件を指定したサンプルを記載しています。
KnFilter filter = KnFilter.FULL;

// フェッチしてくる属性の情報をSetで用意します。
Set<AttributeId> fetchAttributeIds = new HashSet<AttributeId>();
// 名前属性をフェッチしてくる属性に含めます。
AttributeId nameAttrId = KnSystemSchemas.OBJECT_NAME_ATTRIBUTE_ID;
fetchAttributeIds.add(nameAttrId);

// フェッチ条件を生成します。
KnFetch fetch = new KnFetch(fetchAttributeIds, false, false);

// 名前によるソート条件を生成します。
AttributeSortOrder attrSortOrder = new AttributeSortOrder(nameAttrId, Direction.ASCEND);
List<SortOrder> sortOrders = new ArrayList<SortOrder>();
sortOrders.add(attrSortOrder);
SortOrderList sortOrderList = new SortOrderList(sortOrders, null);

// フォルダまたは文書一覧を取得します。
BaseIterable<KnObject> objects =
        session.getChildObjects(parentObjectId, filter, fetch, sortOrderList);
// 取得結果を標準出力に書き出します。
try {
    for (KnObject object : objects) {
        System.out.println(object.getName());
        if (object instanceof KnFolder) {
            // フォルダの場合の処理を記載します。
            KnFolder folder = (KnFolder) object;
            System.out.println(folder.getName());
        } else if (object instanceof KnDocument) {
            // 文書の場合の処理を記載します。
            KnDocument document = (KnDocument) object;
            System.out.println(document.getName());
        }
    }
} finally {
    objects.close();
}
```

以上のコードでフォルダ（kn:folder-20）直下のアクセス可能なフォルダまたは文書一覧を取得できます。 フォルダまたば文書の一覧は、名前の昇順でソートされて取得されます。

> **コラム**
> BaseIterableなどのイテラブルは、不要になったら、必ずクローズしてください。

フォルダ操作

フォルダ作成

概要

フォルダ作成について記載します。

サンプルプログラム

フォルダ(kn:folder-20)を指定して直下にフォルダを作成するサンプルコードを記載します。

```
// フォルダを作成するドロワ/フォルダのIDを指定します。
ObjectId parentObjectId = ObjectId.fromIdString("kn:folder-20");

// 作成時に設定する属性値を指定します。
Map<AttributeId, Object> attributeMap =
        new HashMap<AttributeId, Object>();
// 名前属性の値を指定します。
attributeMap.put(KnSystemSchemas.OBJECT_NAME_ATTRIBUTE_ID, "sample folder");

// フォルダを作成します。
session.createFolder(parentObjectId, attributeMap);
```

以上のコードで名前が「sample folder」のフォルダが作成できます。 クラスはデフォルトの「kn:folder」となります。 アクセス権は親ドロワ/フォルダのアクセス権が設定されます。

コラム

クラスやアクセス権を指定してフォルダを作成する場合は、
createFolder(ObjectId, ClassId, Map<AttributeId, ?>, Map<UgId, Set<Permission>>)
を使用します。

文書操作

文書作成

概要

文書を作成する機能について記載します。

サンプルプログラム

フォルダ(kn:folder-01)を指定し、直下に名前が「sampleDocument」の文書を作成するサンプルコードを記載します。

```
// 文書を作成するドロワ/フォルダのIDを指定します。
ObjectId parentObjectId = ObjectId.fromIdString("kn:folder-01");

// 登録する文書のコンテンツをファイルから指定します。
KnFileContent content = new KnFileContent(new File("sample.txt"));

// 作成時に設定する属性値を指定します。
Map<AttributeId, Object> attributeMap =
      new HashMap<AttributeId, Object>();
// 名前属性の値を指定します。
attributeMap.put(KnSystemSchemas.OBJECT_NAME_ATTRIBUTE_ID, "sample.txt");
// 表示用の初期バージョン(0.0.0.1)を指定します。
List<Integer> firstVersion = Arrays.asList(
     Integer.valueOf(0), Integer.valueOf(0), Integer.valueOf(0), Integer.valueOf(1));
attributeMap.put(KnSystemSchemas.DISPLAY_VERSION_NO_ATTRIBUTE_ID, firstVersion);

session.createDocument(parentObjectId, attributeMap, content);
```

以上のコードで名前が「sampleDocument」の文書が作成できます。 クラスはデフォルトの「kn:document」となります。 アクセス権は親ドロワ/フォルダのアクセス権が設定されます。

> **コラム**
>
> クラスやアクセス権を指定して文書を作成する場合は
>
> createDocument(ObjectId, ClassId, Map<AttributeId, ?>, Map<UgId, Set<Permission>>, KnContent)
>
> を使用します。

> **コラム**
>
> IM-Workflowから文書を作成/チェックインする場合は
>
> List<String> workflowMatter = Arrays.asList(%ワークフロー案件名%, %システム案件ID%);
> attributeMap.put(KnSystemSchemas.WORKFLOW_MATTER_ATTRIBUTE_ID, workflowMatter);
>
> のようにワークフロー案件属性を設定します。

チェックアウト/チェックイン

概要

文書をチェックアウトし、チェックインを行う機能について記載します。

チェックアウトは文書のロックを行って他ユーザーの文書編集を抑制し、 チェックインはロックしている文書に対し新規バージョンを追加し、アンロックをすることで実現しています。

サンプルプログラム

最初に文書(kn:document-01)をチェックアウトするサンプルコードを記載します。

```
// チェックアウトをする文書のIDを指定します。
ObjectId documentId = ObjectId.fromIdString("kn:document-01");

session.lockDocument(documentId);
```

次に文書(kn:document-01)をチェックインするサンプルコードを記載します。

```
// チェックインをする文書のIDを指定します。
ObjectId documentId = ObjectId.fromIdString("kn:document-01");

// 登録する文書のコンテンツをファイルから指定します。
KnFileContent content = new KnFileContent(new File("sample.txt"));

// 作成時に設定する属性値を指定します。
Map<AttributeId, Object> attributeMap =
      new HashMap<AttributeId, Object>();
// 名前属性の値を指定します。
attributeMap.put(KnSystemSchemas.OBJECT_NAME_ATTRIBUTE_ID, content.getFileName());
// 表示用のバージョンを指定します。
attributeMap.put(KnSystemSchemas.DISPLAY_VERSION_NO_ATTRIBUTE_ID, newDisplayVersionNo)

session.createDocumentVersion(documentId, attributeMap, content);
session.unlockDocument(documentId);
```

以上のコードで文書(kn:document-01)をチェックアウトし、チェックインできます。

コラム

表示用のバージョン(newDisplayVersionNo)は、チェックインをする文書の最新バージョンを取得して作成します。

```
KnFetch fetch = new KnFetch(
    new HashSet<AttributeId>(
        Arrays.asList(KnSystemSchemas.DISPLAY_VERSION_NO_ATTRIBUTE_ID)),
    false, false);
KnDocument document = session.getDocumentVersion(documentId, VersionId.LATEST,
fetch);
List<Integer> latestDisplayVersionNo = document.getDisplayVersionNo();
newDisplayVersionNo = createNewDisplayVersionNo(latestDisplayVersionNo);
```

> **注意**
>
> KnSystemSchemas.DISPLAY_VERSION_NO_ATTRIBUTE_IDは、表示上の属性であり、自由に設定可能です。
> intra-mart Accel Documentsの内部では整合性のチェックはしていません。

文書ダウンロード

概要

文書のコンテンツを取得する機能について記載します。

サンプルプログラム

文書(kn:document-01)のコンテンツを取得するサンプルコードを記載します。

```
// 取得する文書のIDを指定します。
ObjectId documentId = ObjectId.fromIdString("kn:document-01");

KnDocument document =
      session.getDocumentVersion(documentId, VersionId.LATEST, KnFetch.FULL);
// 文書の名前を取得
String name = document.getName();

// コンテンツを取得
KnContent content = session.getOriginalContent(documentId, VersionId.LATEST);

// コンテンツを読み出すストリームを取得
InputStream stream = content.createInputStream();
try {
   // ストリームから読む
} finally {
   stream.close();
}
```

以上のコードで文書(kn:document-01)のコンテンツが取得できました。

> **コラム**
>
> KnContentから取得したInputStreamは必ずcloseするようにしてください。

文書削除

概要

文書を削除する機能について記載します。

サンプルプログラム

文書(kn:document-01)を削除するサンプルコードを記載します。

```
// 削除する文書のIDを指定します。
ObjectId documentId = ObjectId.fromIdString("kn:document-01");

session.removeObject(documentId);
```

以上のコードで文書(kn:document-01)を削除できました。

アクセス権変更

概要

文書のアクセス権を変更する機能について記載します。

文書のアクセス権は「誰が」にあたる「主体」と「何をできるか」にあたる「パーミッション」の対で表現されています。 「主体」や「パーミッション」の詳細については「intra-mart Accel Documents / 仕様書」を参照してください。

サンプルプログラム

最初に、文書(kn:document-01)のアクセス権を取得するサンプルコードを記載します。

```
// 取得する文書のIDを指定します。
ObjectId documentId = ObjectId.fromIdString("kn:document-01");

// 文書オブジェクトを取得する
KnObject document = session.getObject(documentId, KnFetch.FULL);

// 文書のアクセス権をMapで取得する
Map<UgId, Set<Permission>> acl = document.getAcl();
```

以上のコードで文書のアクセス権を取得できます。

次に、取得したアクセス権を編集して、ユーザー「hayashi」に対して読み取り権を与えるサンプルコードを記載します。

```
// ユーザーコードからアクセス権を追加するユーザーのUgIdを生成します。
UgId ugId = KnUgIdUtil.getUserUgId("hayashi");

// 読み取り権にあたるパーミッションセットを作成します。
Set<Permission> readPermissions =
        Collections.unmodifiableSet(new HashSet<Permission>(Arrays.asList(
                KnPermissions.READ_ATTRIBUTES, KnPermissions.READ_ACL,
                KnPermissions.READ_VERSIONS,
                KnPermissions.READ_INHERITED_ACL,
                KnPermissions.READ_CHILDREN, KnPermissions.READ_CONTENTS)));

// hayashiというユーザに読み取り権を付与します。
acl.put(ugId, readPermissions);

// 編集したアクセス権をオブジェクトに反映します。
session.setObjectAcl(documentId, acl);
```

以上のコードでアクセス権の編集ができました。

属性変更

概要

文書の属性を変更する方法について記載します。

サンプルプログラム

文書(kn:document-01)の説明属性を変更するサンプルコードを記載します。

```
// 変更する文書のIDを指定します。
ObjectId documentId = ObjectId.fromIdString("kn:document-01");

// 属性マップを作成し、説明属性に値をセットします。
Map<AttributeId, Object> attributeMap = new HashMap<AttributeId, Object>();
attributeMap.put(KnSystemSchemas.DESCRIPTION_ATTRIBUTE_ID, "description");

session.setObjectAttributes(documentId, attributeMap);
```

以上のコードで文書の説明属性を description という値に変更できました。

バージョン操作

概要

文書のバージョンを取得したり、指定したバージョンを削除したりする方法について記載します。

サンプルプログラム

最初に、文書(kn:document-01)のバージョン1のコンテンツを取得するサンプルを記載します。

```
// 取得する文書のIDを指定します。
ObjectId documentId = ObjectId.fromIdString("kn:document-01");

// バージョン1を指定して文書を取得します。
KnDocument document =
      session.getDocumentVersion(documentId, new VersionId(1), KnFetch.FULL);
// 文書の名前を取得。
String name = document.getName();

// コンテンツを取得。
KnContent content = session.getOriginalContent(documentId, VersionId.LATEST);

// コンテンツを読み出すストリームを取得。
InputStream stream = content.createInputStream();
try {
   // ストリームから読みこみます。
} finally {
   stream.close();
}
```

以上のコードで文書(kn:document-01)のバージョン1のコンテンツを取得できました。

> **コラム**
> KnContentから取得したInputStreamは必ずcloseするようにしてください。

次に、バージョン1を指定して削除するサンプルを記載します。

```
session.removeDocumentVersion(documentId, new VersionId(1));
```

以上のコードで文書(kn:document-01)のバージョン1を削除できました。

保管設定

概要

文書に保管責任者や保管期限日を設定する機能について記載します。

サンプルプログラム

文書(kn:document-01)に保管責任者(hayashi)、および、保管期限日を設定するサンプルコードを記載します。

```
// 対象とする文書のIDを指定します。
ObjectId documentId = ObjectId.fromIdString("kn:document-01");

// ユーザーコードから保管責任者として設定するユーザーのUgIdを生成します。
UgId ugId = KnUgIdUtil.getUserUgId("hayashi");

// 保管責任者を指定します。
List<UgId> ownerUgIds = new ArrayList<UgId>();
ownerUgIds.add(ugId);

// 保管期限日を指定します(要件にあわせて、Date型のオブジェクトを用意してください)。
Date expireDate = getExpireDate();

// タイムスタンプ対象とするかどうかを指定します。
boolean isCertChecked = true;

session.setDocumentRetentionParameter(documentId, ownerUgIds,
      expireDate, isCertChecked);

// 保管期限切れ予告通知をセットします。
session.addNoticeExpireDate(documentId);
```

以上のコードで文書に保管設定が付与されます。

> **注意**
> 保管責任者および保管期限日を設定した場合は、必ず保管期限切れ予告通知をセットしてください。

> **コラム**
> タイムスタンプ対象とした文書に、すぐにタイムスタンプを付与する場合は、続けてタイムスタンプ付与のAPIを呼び出す必要があります。
> タイムスタンプ付与については、「*タイムスタンプ設定*」を参照してください。

タイムスタンプ設定

概要

文書にタイムスタンプを設定する機能について記載します。

サンプルプログラム

文書(kn:document-01)にタイムスタンプを付与するサンプルコードを記載します。

```
// 対象とする文書のIDを指定します。
ObjectId documentId = ObjectId.fromIdString("kn:document-01");

// タイムスタンプ付与後のファイルを新規バージョンとして登録する際に設定する属性値を指定します。
Map<AttributeId, Object> attributeMap = new HashMap<AttributeId, Object>();

// 新しく追加するバージョンの表示用属性(例: 1.0.1)
List<Integer> newDisplayVersionNo = Arrays.asList(new Integer(1),
        new Integer(0), new Integer(1));

// 表示用のバージョンを指定します。
attributeMap.put(KnSystemSchemas.DISPLAY_VERSION_NO_ATTRIBUTE_ID, newDisplayVersionNo);

session.attachDocumentTimestamp(documentId, attributeMap);
```

以上のコードで文書にタイムスタンプが付与されます。

> **コラム**
>
> 対象文書はPDFファイルである必要があります。
> また、保管責任者、保管期限日が設定されていて、かつ、タイムスタンプ対象である必要があります。
> 保管設定、タイムスタンプ設定については、「*保管設定*」を参照してください。

> **コラム**
>
> タイムスタンプが付与されたPDFファイルは、対象文書の新規バージョンとして登録されます。

> **コラム**
>
> 引数に指定しているattributeMapは、新規バージョンとして追加する際に、同時に更新する文書の属性を指定するものです。 この引数は、空のMAPで指定しても問題ありませんが、該当文書のクラス定義でバージョン設定を利用している場合は、 同時にバージョン表示用の属性(KnSystemSchemas#DISPLAY_VERSION_NO_ATTRIBUTE_ID)を更新する必要があります。
>
> 上記の例では、バージョン表示用の属性に固定の整数型リストを設定していますが、 通常は、その時点の最新バージョンから次のバージョン用の整数型リストを生成して設定してください。

文書(kn:document-01)にアーカイブタイムスタンプを付与するサンプルコードを記載します。

```
// 対象とする文書のIDを指定します。
ObjectId documentId = ObjectId.fromIdString("kn:document-01");

// アーカイブタイムスタンプ付与後のファイルを新規バージョンとして登録する際に設定する属性値を指定します。
Map<AttributeId, Object> attributeMap = new HashMap<AttributeId, Object>();

// 新しく追加するバージョンの表示用属性(例: 1.0.2)
List<Integer> newDisplayVersionNo = Arrays.asList(new Integer(1),
        new Integer(0), new Integer(2));

// 表示用のバージョンを指定します。
attributeMap.put(KnSystemSchemas.DISPLAY_VERSION_NO_ATTRIBUTE_ID, newDisplayVersionNo);

session.attachArchiveTimestamp(documentId, attributeMap);
```

以上のコードで文書にアーカイブタイムスタンプが付与されます。

> **コラム**
> アーカイブタイムスタンプは、延長用のタイムスタンプです。 通常のタイムスタンプと同様に、タイムスタンプ付与後に新規バージョンとして追加されますので、該当文書のクラスでバージョン設定を利用している場合は、 同時にバージョン表示用の属性を更新する必要があります。

文書(kn:document-01)のタイムスタンプを検証するサンプルコードを記載します。

```
// 対象とする文書のIDを指定します。
ObjectId documentId = ObjectId.fromIdString("kn:document-01");

KnCertValidationStatusType statusType = session.validateTimestamp(documentId);
```

以上のコードで文書のタイムスタンプを検証できます。 検証結果は、返り値のKnCertValidationStatusTypeにて確認可能です。

検索操作

文書の検索

概要

検索機能について記載します。

サンプルプログラム

名前属性と更新日時属性の値を指定して文書を検索するコードのサンプルです。

```
// 文書の名前属性の検索条件に指定する値を定義します。
String documentName = "documentA";

// 文書の更新日時の検索条件に指定する値を定義します。
Calendar cal = Calendar.getInstance();
cal.add (Calendar.MONTH, -1);
Date lastMonth = cal.getTime();

// 文書クラスのクラス識別子を定義します。
ClassId documentClassId = KnSystemSchemas.DOCUMENT_CLASS_ID;

// 名前属性の属性識別子を定義します。
AttributeId nameAttrId = KnSystemSchemas.OBJECT_NAME_ATTRIBUTE_ID;

// 更新日時属性の属性識別子を定義します。
AttributeId modifiedDateAttrId = KnSystemSchemas.MODIFIED_DATE_ATTRIBUTE_ID;

// 検索対象クラスの情報をSetで用意します。
Set<ClassId> documentClassIds = new HashSet();
documentClassIds.add(documentClassId);

// フェッチしてくる属性の情報をSetで用意します。
Set<AttributeId> fetchAttributeIds = new HashSet();
fetchAttributeIds.add(nameAttrId);

// フェッチ条件を生成します。
KnFetch fetch = new KnFetch(fetchAttributeIds, false, false);

// 名前属性の検索条件を生成します。
AttributeCondition<String> nameCondition =
  new AttributeCondition<String>(QueryUtil.value(nameAttrId), QueryUtil.eq(documentName));

// 更新日時の検索条件を生成します。(ge : ～以降)
AttributeCondition<Date> modifiedDateCondition =
  new AttributeCondition<Date>(QueryUtil.value(modifiedDateAttrId), QueryUtil.ge(lastMonth));

// 名前属性と更新日時属性のAND条件を生成します。
List<Term<QueryCondition>> conditions = new ArrayList<Term<QueryCondition>>();
conditions.add(nameCondition);
conditions.add(modifiedDateCondition);
KnQuery query = new KnQuery(QueryUtil.and(conditions), documentClassIds, null, null);

 // 名前によるソート条件を生成します。
 AttributeSortOrder attrSortOrder = new AttributeSortOrder(nameAttrId, Direction.ASCEND);
 List<SortOrder> sortOrders = new ArrayList<SortOrder>();
```

```
 sortOrders.add(attrSortOrder);
 SortOrderList sortOrderList = new SortOrderList(sortOrders, null);

// 検索を実行します。
KnLimitedIterable<KnSearchResult> resultIterable =
          session.searchObjects(query, fetch, SortOrderList.EMPTY, 1000);

// 検索結果を標準出力に書き出します。
try {
  for (KnSearchResult result : resultIterable) {
    KnObject object = result.getObject();
    System.out.println(object.getName());
  }
} finally {
  resultIterable.close();
}
```

以上のコードで1ヶ月前以降に更新された文書を名前の完全一致で検索した結果が取得できます。

なお、検索結果は名前の昇順で並んでいます。

コラム

KnLimitedIterableなどのイテラブルは、不要になったら、必ずクローズするようにします。

タグ操作

タグの作成

概要

タグの作成機能について記載します。

サンプルプログラム

以下の設定値で個人タグを作成し、共有化するコードのサンプルです。

個人タグを作成するときの設定例

属性	設定値
名前	サンプルタグ
説明	サンプルタグの説明です。
色	赤

```
// タグ作成に必要な属性マップを作成します。
Map<AttributeId, Object> attributeMap = new HashMap<AttributeId, Object>();
// タグの名前を追加します。
attributeMap.put(KnSystemSchemas.OBJECT_NAME_ATTRIBUTE_ID, "サンプルタグ");
// タグの説明を追加します。
attributeMap.put(KnSystemSchemas.DESCRIPTION_ATTRIBUTE_ID, "サンプルタグの説明です。");
// 個人タグを追加します。
attributeMap.put(KnSystemSchemas.IS_PRIVATE_TAG_ATTRIBUTE_ID, Boolean.TRUE);
// タグの色を追加します。
attributeMap.put(KnSystemSchemas.TAG_COLOR_ATTRIBUTE_ID, "red");

// 個人タグを作成します。
ObectId objectId = session.createPrivateTag(attributeMap);

// 作成したタグを共有化します。
session.publishPrivateTag(objectId);
```

コラム

サンプルコード中に出現するsessionはKnSessionを表します。
KnSessionの取得/生成方法は、 *intra-mart Accel Documents へのアクセス* を参照してください。

コラム

タグの色は次の中から選択することができます。
青(blue)、黒(black)、灰色(gray)、緑(green)、ピンク(pink)、紫(purple)、赤(red)、黄(yellow)

タグの付与

概要

タグの付与機能について記載します。

サンプルプログラム

文書(kn:document-01)に対して、個人タグ(kn:tag-01)を付与し、削除するコードのサンプルです。

```
// 付与する個人タグのオブジェクトIDを作成します。
ObjectId tagId = ObjectId.fromIdString("kn:tag-01");

// タグを付与される文書のオブジェクトIDを作成します。
ObjectId documentId = ObjectId.fromIdString("kn:document-01");

// 文書にタグを付与します。
session.attachTag(documentId, tagId);

// 付与したタグを除去します。
session.dettachTag(documentId, tagId);
```

コラム

サンプルコード中に出現するsessionはKnSessionを表します。
KnSessionの取得/生成方法は、 *intra-mart Accel Documents へのアクセス* を参照してください。

コラム

タグの色は次の中から選択することができます。
青(blue)、黒(black)、灰色(gray)、緑(green)、ピンク(pink)、紫(purple)、赤(red)、黄(yellow)

クラス操作

属性定義の作成

概要

属性定義の作成機能について記載します。

サンプルプログラム

以下の条件で属性定義を作成するコードのサンプルです。

属性定義を作成するときの設定例

属性	設定値
ID	sample_attr
名前(標準)	sample attribute
名前(日本語)	サンプル属性
名前(英語)	sample attribute
名前(中国語)	sample attribute
説明(日本語)	サンプル属性の説明
説明(英語)	sample attribute description
説明(中国語)	sample attribute description
入力必須	必須
要素型	文字列

注意

属性の操作は、セッションの管理者モードを有効にする必要があります。
管理者モードについては、 *intra-mart Accel Documents へのアクセス* を参照してください。

```
// 属性IDを作成します。
AttributeId attrId = new AttributeId(KnSystemSchemas.CUSTOM_ATTRIBUTE_ID_PREFIX, "sample_attr");
// 属性の要素型を取得します。
AttributeType attrType = AttributeType.STRING;

// 属性定義のプロトタイプを作成します。
KnAttributeDefinitionPrototype proto = new KnAttributeDefinitionPrototype(attrId, attrType);

// 変更可能として設定します。
proto.setChangeable(true);

// 検索可能として設定します。
proto.setSearchable(true);

// ソート可能として設定します。
proto.setSortable(true);

// バージョン管理不可能として設定します。
proto.setVersionable(false);

// 入力必須として設定します。
proto.setMinMultiplicity(1);

// 属性定義を作成します。
session.createAttributeDefinition(proto);

// 属性定義の表示名を設定します。
Map<Locale, String> attrName = new HashMap<Locale, String>();
attrName.put(Locale.ROOT, "sample attribute");
attrName.put(Locale.JAPANESE, "サンプル属性");
attrName.put(Locale.ENGLISH, "sample attribute");
attrName.put(Locale.SIMPLIFIED_CHINESE, "sample attribute");
String attrNameKey = session.getAttributeDefinitionDisplayNameKey(attrId);
session.putDisplayName(attrNameKey, attrName);

// 属性定義の説明を設定します。
Map<String, String> attrDesc = new HashMap<String, String>();
attrDesc.put(CommonConst.LOCALE_JA, "サンプル属性の説明");
attrDesc.put(CommonConst.LOCALE_EN, "sample attribute description");
attrDesc.put(CommonConst.LOCALE_ZH, "sample attribute description");
String treeId = session.getAttributeDefinitionPreferenceTreeId(attrId);
session.putPreferenceNode(treeId, CommonPreferenceConst.DESCRIPTION_PATH, attrDesc);
```

コラム

サンプルコード中に出現するsessionはKnSessionを表します。
KnSessionの取得/生成方法は、 *intra-mart Accel Documents へのアクセス* を参照してください。

セキュリティ定義の作成

概要

セキュリティ定義の作成機能について記載します。

サンプルプログラム

以下の条件でセキュリティ定義を作成するコードのサンプルです。

セキュリティ定義を作成するときの設定例

属性	設定値
タイプ	セキュリティ付きPDF
名前(標準)	sample security definition
名前(日本語)	サンプルセキュリティ定義
名前(英語)	sample security definition
名前(中国語)	sample security definition
権限パスワード	test
権限	印刷: 許可、変更: 許可、コピー: 許可
文書閲覧のパスワード	必要

注意

セキュリティ定義の操作は、セッションの管理者モードを有効にする必要があります。
管理者モードについては、 *intra-mart Accel Documents へのアクセス* を参照してください。

```
// セキュリティ付きPDF定義のパラメータを作成します。
KnSecurePdfCreateParameter param = new KnSecurePdfCreateParameter();

// 権限パスワードを設定します。
param.setOwnerPassword("test");

// 印刷に関する制限を設定します。
param.setPrintRestriction(KnSecurePdfPrintRestriction.ENABLE);

// 変更に関する制限を設定します。
param.setChangeRestriction(KnSecurePdfChangeRestriction.ENABLE);

// コピーに関する制限を設定します。
param.setCopyRestriction(KnSecurePdfCopyRestriction.ENABLE);

// スクリーンリーダーデバイスによるテキストアクセスに関する制限を設定します。
// コピーと同じ制限にする必要があります。
param.setTextAccessRestriction(KnSecurePdfScreenReaderDeviceTextAccessRestriction.ENABLE);

// 文書を開く際にパスワードが必要な場合、trueを設定します。
param.setRequiredOpenPassword(true);

// セキュリティ定義のパラメタを登録します。
KnSecurityDefinitionRegistParameter<?> registParam
  = new KnSecurePdfSecurityDefinitionRegistParameter(param);

// セキュリティ定義を作成します。
KnSecurityDefinitionId securityDefinitionId = session.createSecurityDefinition(registParam);

// セキュリティ定義の表示名を設定します。
String treeId = session.getSecurityDefinitionPreferenceTreeId(securityDefinitionId);
Map<Locale, String> displayNames = new HashMap<String, String>();
displayNames.put("default", "sample security definition");
displayNames.put("ja", "サンプルセキュリティ定義");
displayNames.put("en", "sample security definition");
displayNames.put("zh_CN", "sample security definition");
session.putPreferenceNode(treeId, "/kiun_repo_web/displayname", displayNames);
```

コラム

サンプルコード中に出現するsessionはKnSessionを表します。
KnSessionの取得/生成方法は、 *intra-mart Accel Documents へのアクセス* を参照してください。

保管定義の作成

概要

保管定義の作成機能について記載します。

サンプルプログラム

以下の条件で 保管定義を作成するコードのサンプルです。

保管定義を作成するときの設定例

属性	設定値
名前(標準)	sample retention definition
名前(日本語)	サンプル保管定義
名前(英語)	sample retention definition
名前(中国語)	sample retention definition
文書の保護	有効
期限日	期限変更を許可する
期限切れ文書	期限切れ文書の操作を制限する
デフォルト保管責任者	hayashi
デフォルト保管期間	起算日: 今年度の開始日、期間: 1年間
デフォルトタイムスタンプ対象	デフォルトでタイムスタンプ対象に設定する
文書コピー時の動作	保管責任者と期限日にコピー元の値を設定する
期限切れ予告通知日	10日前
通知方法	メール、ポートレット、IMBox
設定レベル	文書登録時に保管設定を許可する

注意

保管定義の操作は、セッションの管理者モードを有効にする必要があります。
管理者モードについては、 *intra-mart Accel Documents へのアクセス* を参照してください。

```
Map<AttributeId, Object> attributeMap = new HashMap<AttributeId, Object>();

// 文書の保護を設定します。
attributeMap.put(KnSystemSchemas.RESTRICT_DELETION_ATTRIBUTE_ID,
    Boolean.TRUE);

// 期限日の変更許可を設定します。
attributeMap.put(KnSystemSchemas.ACCEPT_EXTENTION_PERIOD_ATTRIBUTE_ID,
    Boolean.TRUE);

// 期限切れ文書の操作制限を設定します。
attributeMap.put(KnSystemSchemas.RESTRICT_EXPIRED_DOCUMENT_OPERATIONS_ATTRIBUTE_ID,
    Boolean.TRUE);

// デフォルト保管責任者を設定します。
attributeMap.put(KnSystemSchemas.DEFAULT_OWNERS_ATTRIBUTE_ID,
    Arrays.asList("kn_user:hayashi"));

// 保管期間の起算日タイプを設定します。
attributeMap.put(KnSystemSchemas.RETENTION_PERIOD_INITIAL_DATE_TYPE_ATTRIBUTE_ID,
    KnRetentionPeriodInitialDateType.FIRST_DAY_OF_BUSINESS_YEAR.name());

// 保管期間の年度の開始月を設定します。
attributeMap.put(KnSystemSchemas.FIRST_MONTH_OF_BUSINESS_YEAR_ATTRIBUTE_ID,
    Integer.valueOf(4));
```

```
// デフォルト保管期間を設定します。
attributeMap.put(KnSystemSchemas.DEFAULT_RETENTION_PERIOD_ATTRIBUTE_ID,
    Integer.valueOf(1));

// デフォルト保管期間単位を設定します。
attributeMap.put(KnSystemSchemas.DEFAULT_RETENTION_PERIOD_UNIT_ATTRIBUTE_ID,
    KnDefaultRetentionPeriodUnit.YEAR.name());

// デフォルトタイムスタンプ対象を設定します。
attributeMap.put(KnSystemSchemas.DEFALUT_CERT_TARGET_ATTRIBUTE_ID,
    Boolean.TRUE);

// 文書コピー時の保管責任者を設定します。
attributeMap.put(KnSystemSchemas.OWNER_OF_COPY_DESTINATION_ATTRIBUTE_ID,
    KnOwnerOfCopyDestination.VALUE_OF_SOURCE.name());

// 文書コピー時の保管期限日を設定します。
attributeMap.put(KnSystemSchemas.RETENTION_PERIOD_OF_COPY_DESTINATION_ATTRIBUTE_ID,
    KnRetentionPeriodOfCopyDestination.VALUE_OF_SOURCE.name());

// 期限切れ予告通知日を設定します。
attributeMap.put(KnSystemSchemas.NOTICE_DAYS_BEFORE_ATTRIBUTE_ID,
    Integer.valueOf(10));

// 通知方法を設定します。
attributeMap.put(KnSystemSchemas.NOTICE_METHODS_ATTRIBUTE_ID,
    Arrays.asList(KnSubscriptionType.MAIL.name(),
        KnSubscriptionType.PORTAL.name(),
        KnSubscriptionType.BOX.name()));

// 設定レベルを設定します。
attributeMap.put(KnSystemSchemas.CONFIGURATION_LEVEL_ATTRIBUTE_ID,
    KnConfigurationLevel.CONFIGURABLE.name());

// 保管定義を作成します。
KnRetentionDefinitionId retentionDefinitionId =
    session.createRetentionDefinition(attributeMap);

// 保管定義の表示名を設定します。
String treeId = session.getRetentionDefinitionPreferenceTreeId(retentionDefinitionId);
Map<String, String> displayNames = new HashMap<String, String>();
displayNames.put("default", "sample retention definition");
displayNames.put("ja", "サンプル保管定義");
displayNames.put("en", "sample retention definition");
displayNames.put("zh_CN", "sample retention definition");
session.putPreferenceNode(treeId, "/kiun_repo_web/displayname", displayNames);
```

コラム

サンプルコード中に出現するsessionはKnSessionを表します。
KnSessionの取得/生成方法は、 *intra-mart Accel Documents へのアクセス* を参照してください。

クラスの定義作成

概要

クラス定義の作成機能について記載します。

サンプルプログラム

以下の条件でクラス定義を作成するコードのサンプルです。

クラス定義を作成するときの設定例

属性	設定値
ID	sample_doc_class
名前(標準)	sample document class
名前(日本語)	サンプル文書クラス
名前(英語)	sample document class
名前(中国語)	sample document class
基本クラス	文書
説明(日本語)	サンプル文書クラスの説明
説明(英語)	sample document class description
説明(中国語)	sample document class description
使用可能属性	knc_attr:sample_attr (サンプル属性)

注意

属性の操作は、セッションの管理者モードを有効にする必要があります。
管理者モードについては、 *intra-mart Accel Documents へのアクセス* を参照してください。

```
// 定義するクラスの基本クラスのクラスIDを作成します。
ClassId superClassId = ClassId.fromIdString(KnSystemSchemas.ABSTRACT_DOCUMENT_CLASS_ID);

// クラス定義のプレフィックスの取得します。
String prefix = KnSystemSchemas.CUSTOM_DOCUMENT_CLASS_ID_PREFIX;

// 定義するクラスのクラスIDを作成します。
ClassId classId = new ClassId(prefix, "sample_doc_class");

// クラス定義のプロトタイプを作成します。
KnClassDefinitionPrototype proto = new KnClassDefinitionPrototype(classId);

// 基本クラスをプロトタイプに設定します。
proto.setSuperClassId(superClassId);

// クラス定義を登録します。
session.createClassDefinition(proto);

// クラスに設定する属性定義を取得します。
AttributeDefinition attrDef = session.getAttributeDefinition(attrId);

// クラス定義に属性を設定します。
// 属性のデフォルト値を取得する関数(getDefaultValue)を作成してください。
Object defaultValue = getDefaultValue(attrDef.getType());
session.addAttributeToClassDefinition(classId, attrDef.getId(), defaultValue);

// クラス定義の表示名を設定します。
Map<Locale, String> className = new HashMap<String, String>();
className.put(Locale.ROOT, "sample document class");
className.put(Locale.JAPANESE, "サンプル文書クラス");
className.put(Locale.ENGLISH, "sample document class");
className.put(Locale.SIMPLIFIED_CHINESE, "sample document class");
String classNameKey = session.getClassDefinitionDisplayNameKey(classId);
session.putDisplayName(classNameKey, className);

// クラス定義の説明を設定します。
Map<String, String> classDesc = new HashMap<String, String>();
classDesc.put(CommonConst.LOCALE_JA, "サンプル文書クラスの説明");
classDesc.put(CommonConst.LOCALE_EN, "sample document class description");
classDesc.put(CommonConst.LOCALE_ZH, "sample document class description");
String treeId = session.getClassDefinitionPreferenceTreeId(classId);
session.putPreferenceNode(treeId, CommonPreferenceConst.DESCRIPTION_PATH, classDesc);

// クラスにセキュリティ定義を設定します。
// セキュリティ定義はセキュア文書クラスにのみ設定が可能です。
KnSecurityDefinitionId refSecurityDefinitionId = securityDefinitionId ;
session.relateClassDefinition(classId, refSecurityDefinitionId);

// クラスに保管定義を設定します。
// 保管定義は文書クラス、セキュア文書クラスにのみ設定が可能です。
KnRetentionDefinitionId refRetentionDefinitionId = retentionDefinitionId ;
session.relateClassDefinitionToRetentionDefinition(classId, refRetentionDefinitionId);
```

コラム

サンプルコード中に出現するsessionはKnSessionを表します。
KnSessionの取得/生成方法は、 *intra-mart Accel Documents へのアクセス* を参照してください。

コラム

サンプルコード中に出現するattrIdは事前に作成したAttributeIdを表します。
AttributeIdの作成方法は、 *属性定義の作成* を参照してください。

コラム

サンプルコード中に出現するsecurityDefinitionIdは事前に作成したKnSecurityDefinitionIdを表します。
KnSecurityDefinitionIdの作成方法は、 *セキュリティ定義の作成* を参照してください。

コラム

サンプルコード中に出現するretentionDefinitionIdは事前に作成したKnRetentionDefinitionIdを表します。
KnRetentionDefinitionIdの作成方法は、 *保管定義の作成* を参照してください。

キャビネット操作

キャビネット作成

概要

キャビネットの作成方法について記載します。

サンプルプログラム

管理者「aoyagi」を指定してキャビネットを作成し、そのキャビネットに利用者「ueda」を設定するコードのサンプルです。

サービス管理セッションを生成するユーザには、Accel Documents管理者用のロールが割り当てられている必要があります。

```
// ユーザコードからサービス管理セッションを生成するユーザのUgIdを生成します。このユーザがキャビネットの作成者となります。
UgId serviceAdminUgId = KnUgIdUtil.getUserUgId("hayashi");

// サービス管理セッションを生成します。
KnServiceAdminSession serviceAdminSession = repository.createServiceAdminSession(serviceAdminUgId);

try {
  // キャビネットの属性をMAPで用意します。
  Map<AttributeId, Object> attributeMap = new HashMap<AttributeId, Object>();
  attributeMap.put(KnSystemSchemas.CABINET_NAME_ATTRIBUTE_ID, name);

  // キャビネットの管理者群をListで用意します。
  List<UgId> adminUgIdList = new ArrayList<UgId>();
  adminUgIdList.add(KnUgIdUtil.getUserUgId("aoyagi"));

  // キャビネットを作成します。
  KnCabinetId cabinetId = serviceAdminSession.createCabinet(attributeMap, adminUgIdList);

  // キャビネットの利用者群をListで用意します。
  List<UgId> accessibleUgIdList = new ArrayList<UgId>();
  accessibleUgIdList.add(KnUgIdUtil.getUserUgId("ueda"));

  // キャビネットの利用者を登録します。
  serviceAdminSession.updateAccessiblePrincipal(cabinetId, accessibleUgIdList);
} finally {
  // サービス管理セッションをクローズします。
  serviceAdminSession.close();
}
```

コラム

intra-mart Accel Documents アプリケーションでは、キャビネット作成後にメニュー表示設定のインポート処理などを行っています。 APIのキャビネット作成ではインポート処理は行われないため、以下の公開ソースコードを参照して必要な情報のインポート処理を追加で行ってください。

jp.co.fujixerox.kiun.repository.web.common.util.KnCabinetUtil#settingAfterCreateCabinet(KnSession)

ドロワ操作

ドロワ作成

概要

ドロワの作成方法について記載します。

サンプルプログラム

サンプル会社(comp_sample_01)に書き込み権があるドロワを作成するサンプルコードを記載します。

> **注意**
>
> ドロワを作成する場合は、セッションの管理者モードを有効にする必要があります。
> 管理者モードについては、 *intra-mart Accel Documents へのアクセス* を参照してください。

```
// 作成時に設定する属性値を指定します。
Map<AttributeId, Object> attributeMap = new HashMap<AttributeId, Object>();
// 名前属性の値を指定します。
attributeMap.put(KnSystemSchemas.OBJECT_NAME_ATTRIBUTE_ID, " sample drawer");

// 作成時に設定するアクセス権を指定します。
Map<UgId, Set<Permission>> acl = new HashMap<UgId, Set<Permission>>();
// アクセス権を追加する組織(サンプル会社)のUgIdを生成します。
UgId ugId = KnUgIdUtil.getDepartmentUgId(
      "comp_sample_01", "comp_sample_01", "comp_sample_01");
// 書き込み権にあたるパーミッションセットを作成します。
Set<Permission> writePermissions =
      Collections.unmodifiableSet(new HashSet<Permission>(Arrays.asList(
            KnPermissions.READ_ATTRIBUTES,
            KnPermissions.WRITE_ATTRIBUTES, KnPermissions.READ_ACL,
            KnPermissions.READ_VERSIONS, KnPermissions.CREATE_VERSIONS,
            KnPermissions.READ_INHERITED_ACL,
            KnPermissions.READ_CHILDREN, KnPermissions.ADD_CHILDREN,
            KnPermissions.REMOVE_CHILDREN, KnPermissions.READ_CONTENTS,
            KnPermissions.LOCK)));
// サンプル会社に書き込み権を付与します。
acl.put(ugId, writePermissions);

// 作成時に設定する継承ACLとして空のMapを指定します。
Map<UgId, Set<Permission>> inheritedAcl = new HashMap<UgId, Set<Permission>>();

// 属性、アクセス権、継承ACLを指定してドロワを作成します。
ObjectId drawerId = session.createPublicDrawer(attributeMap, acl, inheritedAcl);

// フォルダ/文書作成者に管理権を付与する場合は、次の設定をします。
session.setObjectCreatorPermissionsForDrawer(drawerId, KnPermissions.ALL_PERMISSIONS);
```

> **コラム**
>
> サンプルコード中に出現するsessionはKnSessionを表します。
> KnSessionの取得/生成方法は、 *intra-mart Accel Documents へのアクセス* を参照してください。

コラム

ドロワ作成時に指定できる継承ACLとは、ドロワ配下のフォルダ/文書に対して適用される共通のアクセス権です。
継承ACLが設定されている場合、継承ACLとフォルダ/文書のアクセス権の論理和でアクセス権が評価されます。

注意

intra-mart Accel Documents アプリケーションでは継承ACLを使用していません。
そのため、intra-mart Accel Documents アプリケーションでは、上記のサンプルと同様に継承ACLに空のMapを
指定してドロワを作成しています。
APIを使って作成したドロワをintra-mart Accel Documents アプリケーションで利用する場合は、
継承ACLに空のMapを指定することを推奨します。

プリファレンス操作

プリファレンスの利用

目次

概要

プリファレンス機能についての説明と使い方について記載します。

プリファレンス機能は、アプリケーション独自の情報などをキャビネットやオブジェクトなどに紐付けて保持する機構です。

- データ構造
  - 階層ツリー内のノードには、次の要素を設定します。
    - ノード名
      ノードを識別する名前を指定します。
      1 文字以上128 文字以内の文字列で、英数字と「_」、「.」、「:」、または「-」が利用できます。
      正規表現で記述すると次のとおりです。
      [a-zA-Z0-9_:.-]{1,128}
      例）「メンテナンス」→「アプリケーション設定」の「メールアドレス」設定の場合
      mailAddress
    - 絶対パス名
      ノードの階層ツリー内での位置をあらわすパスを指定します。
      階層ツリーの一番上のノード（ルートノード）の絶対パス名は、「/」（スラッシュ）です。
      ルートノード以外のノードの絶対パス名は、ノード名を「/」で連結した文字列です。
      例）「メンテナンス」→「アプリケーション設定」の「メールアドレス」設定の場合
      /kiun_repo_web/tenant/
      絶対パス名は最大で4000文字です。
  - 各ノードには、文字列を「キー」とするマップ型でデータを持ちます。
  - ノードの値は、属性型で定義されている型の値を使用できます。
  - 階層ツリーの同一階層内で、同じノード名は作成できません。

> **コラム**
>
> プリファレンスにアクセスするためには、ノードの絶対パスを使用します。

> **注意**
>
> プリファレンスの設定データは、検索できません。

- ツリー管理

プリファレンスのツリーは、intra-mart Accel Documentsを利用するアプリケーション側で作成できます。
ツリーには、次の要素があります。

- 識別子
  ツリーを識別する文字列です。 作成されたツリーは、識別子によって識別します。
  1 文字以上128 文字以内の文字列で、英数字、「_」、「.」、「:」、または「-」が利用できます。
  正規表現で記述すると次のとおりです。
  [a-zA-Z0-9_:.-]{1,128}
  ドロワやフォルダなどオブジェクトごとにプレフィックスを付けたツリー識別子を指定します。
  例） 検索フォルダの場合
  object.kn:queryFolder-XXX

> **注意**
>
> intra-mart Accel Documentsで使用しているTreeIDは操作しないでください。
> intra-mart Accel Documentsが正しく動作しなくなる場合があります。

> **コラム**
>
> オブジェクトと生存期間を一緒にしたい場合は、オブジェクトやスキーマ定義が削除した場合に、
> 対応するツリーを一緒に削除します。

- 使用されるプレフィックスの一覧
  ツリーの識別子に含めるプレフィックスを次に説明します。

object	オブジェクト用のツリーのプレフィックスです。 このプレフィックスに、固有のオブジェクト識別子を連結したものをツリー識別子として指定します。
classDefinition	クラス用のプレフィックスです。 このプレフィックスに、固有のクラス識別子を連結したものをツリー識別子として指定します。
attributeDefinition	属性用のツリーのプレフィックスです。 このプレフィックスに、固有の属性識別子を連結したものをツリー識別子として指定します。
application	アプリケーション用の設定を保持するツリーのプレフィックスです。 このプレフィックスに、固有のアプリケーション名を連結したものをツリー識別子として指定します。

プリファレンスの使用箇所

intra-mart Accel DocumentsのWebアプリケーションで使用しているプリファレンスについて説明します。

- 「メンテナンス」→「アプリケーション設定」→「メールアドレス設定」で設定するメールアドレス情報
  - ツリー識別子
    application.service
  - パス
    /kiun_repo_web/tenant/mailAddress
    - キー （メールアドレス）
      typical_mail_address

        - キー （メールアドレス名：日本語）
          ja
        - キー （メールアドレス名：英語）
          en
        - キー （メールアドレス名：中国語）
          zh_CN
- 「メンテナンス」→「タイムスタンプオプション設定」で設定するタイムスタンプ設定情報
    - ツリー識別子
      application.cert
    - パス
      /kiun_repo_system/pades
        - キー （TSAの接続先URL）
          tsa_url
        - キー （TSAへの接続ID）
          tsa_id
        - キー （TSAへの接続パスワード）
          tsa_password
        - キー （TSAのポリシー）
          tsa_policy
        - キー （ハッシュアルゴリズム）
          algorithm

> **コラム**
> TSAへの接続パスワードは暗号化された状態で保存されています。

- キャビネット属性の最大コンテントサイズ
    - ツリー識別子
      application.cabinet
    - パス
      /kiun_repo_web/document
        - キー （制限サイズ）
          content_size_limit
- キャビネット属性の最大検索結果数
    - ツリー識別子
      application.cabinet
    - パス
      /kiun_repo_web/search
        - キー （制限サイズ）
          result_limit
- キャビネット属性の保管期限切れ文書の表示設定
    - ツリー識別子
      application.cabinet
    - パス
      /kiun_repo_web/retention
        - キー （表示設定フラグ）
          display_expired_document
        - キー （設定日時）
          modified_date
- 属性定義の説明

- ツリー識別子
  attributeDefinition.（属性ID）
- パス
  /kiun_repo_web/description
  - キー（日本語）
    ja
  - キー（英語）
    en
  - キー（中国語）
    zh_CN

- 属性定義の候補値
  - ツリー識別子
    attributeDefinition.（属性ID）
  - パス
    /kiun_repo_web/candidate
    - キー（候補値）
      （ソート順のインデックス）

- 属性定義のデフォルト値
  - ツリー識別子
    attributeDefinition.（属性ID）
  - パス
    /kiun_repo_web/defaultValue
    - キー（デフォルト値）※文字列の場合
      default_value_string
    - キー（デフォルト値）※真偽値の場合
      default_value_boolean
    - キー（デフォルト値）※日時の場合
      default_value_date
    - キー（デフォルト値）※整数の場合
      default_value_integer
    - キー（デフォルト値）※長整数の場合
      default_value_long
    - キー（デフォルト値）※固定小数点の場合
      default_value_bigdecimal
    - キー（デフォルト値）※ＵＧ識別子の場合
      default_value_ugid
    - キー（作成日をデフォルト値とするかどうか）※日時の場合
      is_created_date
    - キー（操作ユーザをデフォルト値とするかどうか）※ＵＧ識別子の場合
      is_registered_ugId

- クラス定義の説明
  - ツリー識別子
    classDefinition.（クラスID）
  - パス
    /kiun_repo_web/description
    - キー（日本語）
      ja
    - キー（英語）
      en

        - キー（中国語）
          zh_CN
- クラス定義のバージョン設定
    - ツリー識別子
      classDefinition.（クラスID）
    - パス
      /kiun_repo_web/version
        - キー（プレフィックス）
          prefix
        - キー（初期バージョン）
          firstVersion
        - キー（バージョンの桁数）
          numberOfVersion
        - キー（サフィックス）
          suffix
- クラス定義の属性表示設定
    - ツリー識別子
      classDefinition.（クラスID）
    - パス
      /kiun_repo_web/attrViewSetting
        - キー（表示属性）
          items
- 検索フォルダの検索条件（簡易検索）
    - ツリー識別子
      object.（クラスID）
    - パス
      /kiun_repo_web/query_folder/condition_simple
        - キー
          検索条件
- 検索フォルダの検索条件（詳細検索）
    - ツリー識別子
      object.（クラスID）
    - パス
      /kiun_repo_web/query_folder/condition_advance
        - キー
          検索条件
- 一覧表示設定の名前
    - ツリー識別子
      application.listViewSetting
    - パス
      /kiun_repo_web/（一覧表示設定ID）/name
        - キー（標準表示名）
          default
        - キー（日本語）
          ja
        - キー（英語）
          en
        - キー（中国語）
          zh_CN

- 一覧表示設定の説明
  - ツリー識別子
    application.listViewSetting
  - パス
    /kiun_repo_web/（一覧表示設定ID）/description
    - キー （日本語）
      ja
    - キー （英語）
      en
    - キー （中国語）
      zh_CN
- 一覧表示設定の表示属性
  - ツリー識別子
    application.listViewSetting
  - パス
    /kiun_repo_web/（一覧表示設定ID）/items
    - キー （表示属性）
      attributes
- 一覧表示設定の表示属性の表示幅
  - ツリー識別子
    application.listViewSetting
  - パス
    /kiun_repo_web/（一覧表示設定ID）/width
    - キー （表示属性の表示幅）
      （属性ID）
- 一覧表示設定の初期ソート条件
  - ツリー識別子
    application.listViewSetting
  - パス
    /kiun_repo_web/（一覧表示設定ID）/sort
    - キー （ソート条件）
      （ソート条件のインデックス）
- ドロワ/フォルダの一覧表示設定
  - ツリー識別子
    object.（オブジェクトID）
  - パス
    /kiun_repo_web/listViewSetting
    - キー （デフォルト設定）
      default
    - キー （一覧表示設定）
      settings
- メニュー表示設定の名前
  - ツリー識別子
    application.menuViewSetting
  - パス
    /kiun_repo_web/settings/（メニュー表示設定ID）/name
    - キー （標準表示名）
      default
    - キー （日本語）

ja
    - キー（英語）
      en
    - キー（中国語）
      zh_CN
- メニュー表示設定の説明
  - ツリー識別子
    application.menuViewSetting
  - パス
    /kiun_repo_web/settings/（メニュー表示設定ID）/description
    - キー（日本語）
      ja
    - キー（英語）
      en
    - キー（中国語）
      zh_CN
- メニュー表示設定の表示属性
  - ツリー識別子
    application.menuViewSetting
  - パス
    /kiun_repo_web/settings/（メニュー表示設定ID）/items
    - キー（一括操作）
      batch_operation
    - キー（新規登録(フォルダ)）
      create_folder
    - キー（新規登録(文書)）
      create_document
    - キー（新規登録(セキュア文書)）
      create_secure_document
    - キー（ドロワ操作）
      drawer_operation
    - キー（フォルダ操作）
      folder_operation
    - キー（文書操作）
      document_operation
    - キー（セキュア文書操作）
      secure_document_operation
    - キー（検索条件属性）
      attribute_condition
    - キー（検索対象クラス）
      class_condition
- デフォルトのメニュー表示設定
  - ツリー識別子
    application.menuViewSetting
  - パス
    /kiun_repo_web/default
    - キー（デフォルトのメニュー表示設定ID）
      id
- ドロワ/フォルダのメニュー表示設定

- ツリー識別子
  object.（オブジェクトID）
- パス
  /kiun_repo_web/menuViewSetting
  - キー （デフォルト設定）
    id

- セキュリティ定義の名前
  - ツリー識別子
    application.securityDefinition.（セキュリティ定義のID）
  - パス
    /kiun_repo_web/displayname
    - キー （標準表示名）
      default
    - キー （日本語）
      ja
    - キー （英語）
      en
    - キー （中国語）
      zh_CN
- 保管定義の名前
  - ツリー識別子
    application.reteintionDefinition.（保管定義のID）
  - パス
    /kiun_repo_web/displayname
    - キー （標準表示名）
      default
    - キー （日本語）
      ja
    - キー （英語）
      en
    - キー （中国語）
      zh_CN
- メッセージテンプレート
  - ツリー識別子
    application.cabinet
  - パス
    /kiun_repo_web/template/（通知タイプ）/（通知種類）/（通知言語）
    - 通知タイプ
      - imbox（IMBox）
      - mail（メール）
    - 通知種類
      - notice（通知先指定通知）
      - monitor（モニタ通知）
      - share_st/gst（ゲストユーザ向け文書公開開始通知）
      - share_ed/gst（ゲストユーザ向け文書公開終了通知）
      - share_cl/gst（ゲストユーザ向け文書公開終了通知（公開中止））
      - share_st/user（認証済みユーザ向け文書公開開始通知）
      - share_ed/user（認証済みユーザ向け文書公開終了通知）

- share_cl/user（認証済みユーザ向け文書公開終了通知（公開中止））
- owner（保管責任者変更通知）
- expiration（保管期限切れ予告通知）
- timestamp_attach_error（タイムスタンプ付与失敗通知）
- validationExpireDate（検証有効期限終了予告通知）

- キー （本文）
  body
- キー （件名）
  subject

サンプルプログラム

intra-mart Accel Documents の検索フォルダ機能は、プリファレンス機能を利用して、検索条件をフォルダに紐付けています。

この機能における利用方法をサンプルとして記載します。

> **コラム**
>
> サンプルコード中のsessionはKnSessionを表します。
>
> 生成方法は *intra-mart Accel Documents へのアクセス*、 を参照してください。

プリファレンス登録の例

```
// 検索フォルダオブジェクトのIDを指定します。
ObjectId folderId = ObjectId.fromIdString("kn:queryFolder-01");

// 各アプリケーションごとに設定するパスを定義します。
String applicationPath = "/kiun_repo_web";

// 「検索フォルダ機能」で使用するパスを定義します。
String queryFolderPath = applicationPath + "/query_folder";

// 検索フォルダの検索条件用のパスを定義します。
String conditionPath = queryFolderPath + "/condition";

// 検索条件値をマップにセットするためのKEYを定義します。
String queryKey = "query";

// フォルダオブジェクトの識別子からツリーIDを取得します。
String treeId = session.getObjectPreferenceTreeId(folderId);

// 検索条件をMAPで用意します。
Map<String, Object> valueMap = new HashMap<String, Object>();
valueMap.put(queryKey, "QueryValue");

// 検索条件のMAPをプリファレンスに登録します。
session.putPreferenceNode(treeId, conditionPath, valueMap);
```

保存された検索条件は以下のようにツリー構造で保存されます。

コラム

サンプルのツリー構造は、intra-mart Accel Documents アプリケーションの検索フォルダ機能における実際のツリー構造とはパス名およびKEYが異なります。

```
/

/kiun_repo_web

/kiun_repo_web/query_folder

/kiun_repo_web/query_folder/condition
  ・ query : "QueryValue" (String)
```

プリファレンスからの値取得の例

次のコードはフォルダに紐付けた条件を取得するコードのサンプルです。

```
// 検索フォルダオブジェクトのIDを指定します。
ObjectId folderId = ObjectId.fromIdString("kn:queryFolder-01");

// 検索フォルダの検索条件用のパスを定義します。
String conditionPath = "/kiun_repo_web/query_folder/condition";

// 検索条件値をマップから取得するためのKEYを定義します。
String queryKey = "query";

// フォルダオブジェクトの識別子からツリーIDを取得します。
String treeId = session.getObjectPreferenceTreeId(folderId);

// プリファレンスのノードを取得します。
KnPreferenceNode node = session.getPreferenceNode(treeId, conditionPath);

if (node != null) {
  // ノードから検索条件のマップを取得します。
  Map<String, Object> valueMap = node.getValueMap();

  // マップから検索条件を取得します。
  String queryString = (String) valueMap.get(queryKey);
}
```

プリファレンスからの部分ツリー削除の例

次のコードは検索フォルダの条件を削除するコードのサンプルです。

```
// 検索フォルダオブジェクトのIDを指定します。
ObjectId folderId = ObjectId.fromIdString("kn:queryFolder-01");

// 文書管理アプリケーションの検索フォルダ機能用のパスを定義します。
String queryFolderPath = "/kiun_repo_web/query_folder";

// フォルダオブジェクトの識別子からツリーIDを取得します。
String treeId = session.getObjectPreferenceTreeId(folderId);

// プリファレンスから検索フォルダ機能の条件を削除します。(子ツリーも一緒に削除されます。)
session.removePreferenceNode(treeId, queryFolderPath);
```

付録

intra-mart Accel Documentsが提供する画面アイテム

アプリケーション種別「標準」で利用できる画面アイテム

AccelDocuments 添付文書

Accel Documentsで管理されている文書をフォームに添付するための画面アイテムです。
画面アイテムには文書、または、フォルダのリンクが埋め込まれます。リンクにアクセスすることで文書のダウンロードや文書やフォルダの表示が可能となります。

アイテム

アイテム名

同一フォーム内で画面アイテムを識別するための名前を指定します。

画面の種類（行項目）

1. 登録
   Webアプリケーション(標準)での登録画面の時の表示タイプを設定します。
2. 編集
   Webアプリケーション(標準)での更新画面の時の表示タイプを設定します。
3. 参照
   Webアプリケーション(標準)での参照(詳細)画面の時の表示タイプを設定します。

表示・入力タイプ（列項目）

1. 表示
   html上に画面アイテムを存在させます。
2. 非表示
   html上に画面アイテムを存在させません。

表示タイプ：表示

表示スタイル

ラベル

ラベルに設定した名称は、リンクの左に表示します。

URL

文書、フォルダにアクセスするURLを指定します。
URLには、「acceldocuments」以降のURLを指定します。
指定するURLとして、以下のURLを利用できます。

- 文書をダウンロードするURL
  文書の属性表示画面に表示されるURL

指定例：acceldocuments/rep/storage/download/<キャビネットID>/<文書ID>

- フォルダ、ドロワを表示するURL
  フォルダ、ドロワの一覧画面を表示しているURL
  指定例：acceldocuments/rep/storage/list/<キャビネットID>#st-<フォルダID>

アイテムサイズ・配置
フォーム内での表示の位置・高さ・幅を指定します。
幅
画面アイテムとして指定した領域(「フォーム・デザイナ」画面上で赤い点線で囲まれる範囲)の横の長さ(幅)をピクセル単位で指定します。
高
画面アイテムとして指定した領域(「フォーム・デザイナ」画面上で赤い点線で囲まれる範囲)の縦の長さ(高さ)をピクセル単位で指定します。
X
画面アイテムとして指定した領域(「フォーム・デザイナ」画面上で赤い点線で囲まれる範囲)の左上頂点のフォーム左上からの横位置をピクセル単位で指定します。
Y
画面アイテムとして指定した領域(「フォーム・デザイナ」画面上で赤い点線で囲まれる範囲)の左上頂点のフォーム左上からの縦位置をピクセル単位で指定します。

アプリケーション種別「IM-Workflow」で利用できる画面アイテム

AccelDocuments 添付文書

Accel Documentsで管理されている文書をフォームに添付するための画面アイテムです。
画面アイテムには文書、または、フォルダのリンクが埋め込まれます。リンクにアクセスすることで文書のダウンロードや文書やフォルダの表示が可能となります。

アイテム

アイテム名

同一フォーム内で画面アイテムを識別するための名前を指定します。

画面の種類（行項目）

1. 申請
   ワークフローの申請画面の時の表示タイプを設定します。
2. 再申請
   ワークフローの再申請画面の時の表示タイプを設定します。
3. 承認
   ワークフローの承認画面の時の表示タイプを設定します。
4. 参照
   ワークフローの確認・参照画面等の表示タイプを設定します。

表示・入力タイプ（列項目）

1. 表示
   html上に画面アイテムを存在させます。
2. 非表示
   html上に画面アイテムを存在させません。

表示タイプ：表示

表示スタイル

ラベル

ラベルに設定した名称は、リンクの左に表示します。

URL

文書、フォルダにアクセスするURLを指定します。
URLには、「acceldocuments」以降のURLを指定します。
指定するURLとして、以下のURLを利用できます。

- 文書をダウンロードするURL
  文書の属性表示画面に表示されるURL
  指定例：acceldocuments/rep/storage/download/<キャビネットID>/<文書ID>
- フォルダ、ドロワを表示するURL
  フォルダ、ドロワの一覧画面を表示しているURL
  指定例：acceldocuments/rep/storage/list/<キャビネットID>#st-<フォルダID>

アイテムサイズ・配置
フォーム内での表示の位置・高さ・幅を指定します。
幅
画面アイテムとして指定した領域(「フォーム・デザイナ」画面上で赤い点線で囲まれる範囲)の横の長さ(幅)をピクセル単位で指定します。
高
画面アイテムとして指定した領域(「フォーム・デザイナ」画面上で赤い点線で囲まれる範囲)の縦の長さ(高さ)をピクセル単位で指定します。
X
画面アイテムとして指定した領域(「フォーム・デザイナ」画面上で赤い点線で囲まれる範囲)の左上頂点のフォーム左上からの横位置をピクセル単位で指定します。
Y
画面アイテムとして指定した領域(「フォーム・デザイナ」画面上で赤い点線で囲まれる範囲)の左上頂点のフォーム左上からの縦位置をピクセル単位で指定します。

アプリケーション種別「BIS-BISフロー」で利用できる画面アイテム

AccelDocuments 添付文書

Accel Documentsで管理されている文書をフォームに添付するための画面アイテムです。
画面アイテムには文書、または、フォルダのリンクが埋め込まれます。リンクにアクセスすることで文書のダウンロードや文書やフォルダの表示が可能となります。

アイテム

アイテム名

同一フォーム内で画面アイテムを識別するための名前を指定します。

画面の種類（行項目）

1. 処理

BISフローの処理画面の時の表示タイプを設定します。

2. 参照
   BISフローの参照画面の時の表示タイプを設定します。

表示・入力タイプ（列項目）

1. 表示
   html上に画面アイテムを存在させます。
2. 非表示
   html上に画面アイテムを存在させません。

表示タイプ：表示

表示スタイル

ラベル

ラベルに設定した名称は、リンクの左に表示します。

URL

文書、フォルダにアクセスするURLを指定します。
URLには、「acceldocuments」以降のURLを指定します。
指定するURLとして、以下のURLを利用できます。

- 文書をダウンロードするURL
  文書の属性表示画面に表示されるURL
  指定例：acceldocuments/rep/storage/download/<キャビネットID>/<文書ID>
- フォルダ、ドロワを表示するURL
  フォルダ、ドロワの一覧画面を表示しているURL
  指定例：acceldocuments/rep/storage/list/<キャビネットID>#st-<フォルダID>

アイテムサイズ・配置
フォーム内での表示の位置・高さ・幅を指定します。
幅
画面アイテムとして指定した領域(「フォーム・デザイナ」画面上で赤い点線で囲まれる範囲)の横の長さ(幅)をピクセル単位で指定します。
高
画面アイテムとして指定した領域(「フォーム・デザイナ」画面上で赤い点線で囲まれる範囲)の縦の長さ(高さ)をピクセル単位で指定します。
X
画面アイテムとして指定した領域(「フォーム・デザイナ」画面上で赤い点線で囲まれる範囲)の左上頂点のフォーム左上からの横位置をピクセル単位で指定します。
Y
画面アイテムとして指定した領域(「フォーム・デザイナ」画面上で赤い点線で囲まれる範囲)の左上頂点のフォーム左上からの縦位置をピクセル単位で指定します。

アプリケーション種別「BIS-ワークフロー」で利用できる画面アイテム

AccelDocuments 添付文書

Accel Documentsで管理されている文書をフォームに添付するための画面アイテムです。

画面アイテムには文書、または、フォルダのリンクが埋め込まれます。リンクにアクセスすることで文書のダウンロードや文書やフォルダの表示が可能となります。

アイテム

アイテム名

同一フォーム内で画面アイテムを識別するための名前を指定します。

画面の種類（行項目）

1. 申請
   ワークフローの申請画面の時の表示タイプを設定します。
2. 再申請
   ワークフローの再申請画面の時の表示タイプを設定します。
3. 承認
   ワークフローの承認画面の時の表示タイプを設定します。
4. 参照
   ワークフローの確認・参照画面等の表示タイプを設定します。

表示・入力タイプ（列項目）

1. 表示
   html上に画面アイテムを存在させます。
2. 非表示
   html上に画面アイテムを存在させません。

表示タイプ：表示

表示スタイル

ラベル

ラベルに設定した名称は、リンクの左に表示します。

URL

文書、フォルダにアクセスするURLを指定します。
URLには、「acceldocuments」以降のURLを指定します。
指定するURLとして、以下のURLを利用できます。

- 文書をダウンロードするURL
  文書の属性表示画面に表示されるURL
  指定例：acceldocuments/rep/storage/download/<キャビネットID>/<文書ID>
- フォルダ、ドロワを表示するURL
  フォルダ、ドロワの一覧画面を表示しているURL
  指定例：acceldocuments/rep/storage/list/<キャビネットID>#st-<フォルダID>

アイテムサイズ・配置
フォーム内での表示の位置・高さ・幅を指定します。
幅
画面アイテムとして指定した領域(「フォーム・デザイナ」画面上で赤い点線で囲まれる範囲)の横の長さ(幅)をピクセ

ル単位で指定します。

高

画面アイテムとして指定した領域(「フォーム・デザイナ」画面上で赤い点線で囲まれる範囲)の縦の長さ(高さ)をピクセル単位で指定します。

X

画面アイテムとして指定した領域(「フォーム・デザイナ」画面上で赤い点線で囲まれる範囲)の左上頂点のフォーム左上からの横位置をピクセル単位で指定します。

Y

画面アイテムとして指定した領域(「フォーム・デザイナ」画面上で赤い点線で囲まれる範囲)の左上頂点のフォーム左上からの縦位置をピクセル単位で指定します。