↑Top

目次

改訂情報

変更年月日	変更内容
2012-12-21	初版
2013-04-01	第２版　下記を追加・変更しました ■「*デフォルトで存在するサブジェクトコンテキストデコレータ*」に「*IPv4AddressDecorator*」を追加 ■「*認可のキャッシュ設定*」を追加 ■「*リソースタイプとアクション*」にキャッシュコントローラについて記述を追加 ■「*部品化*」にコールバックについて記述を追加
2013-07-01	第３版　下記を追加・変更しました ■「*認可のキャッシュ設定*」に「*リソース閉塞状態のキャッシュ*」を追加 ■「*リソース管理*」に「*リソースグループ汎用属性の管理*」を追加 ■「*リソース管理*」の「*ERとテーブル上の表現*」に「*リソースグループ汎用属性テーブル(imaz_resource_group_attr)*」を追加 ■「*認可設定画面*」に「*リソース削除時のバックアップ*」を追加 ■「*初期化仕様*」の一部説明文を修正 ■「*intra-mart Accel Platform に含まれるリソースタイプ*」の一部説明文を修正
2013-10-01	第４版　下記を追加・変更しました ■「*認可のキャッシュ設定*」の説明を修正 ■「*intra-mart Accel Platform に含まれるリソースタイプ*」、「*予約されているリソースグループセット一覧*」にIMBoxを追加 ■「*部品化*」のコールバックについての説明を修正 ■「*imartタグ/カスタムタグによる認可要求*」の説明を修正
2014-01-01	第５版　下記を追加・変更しました ■「*誰が(Subject)*」の提供しているサブジェクト一覧を更新 ■「*リソース削除時のバックアップ*」の説明を修正 ■「*認可判断機能の構造*」の説明を修正 ■「*認可判断モジュール*」の説明を修正 ■「*API*」の使用例コードを修正 ■「*基準日によって名称が変わるサブジェクトタイプ*」の説明を追加 ■「*リソース管理*」に「*閉塞の管理*」を追加 ■「*認可判断機能*」に「*閉塞*」を追加 ■「*サブジェクトリゾルバ*」に「*期間(ImTermSubjectResolver)*」を追加 ■「*デフォルトで存在するサブジェクトコンテキストデコレータ*」に「*TermDecorator*」を追加 ■「*intra-mart Accel Platform に含まれるサブジェクトタイプ*」に「*期間*」を追加 ■「*付録*」に「*運用時のTips*」を追加 ■「*認可情報のインポート・エクスポート*」に「*インポートの依存関係*」を追加 ■「*リソースグループ(imaz_resource_group)*」の説明図を修正

変更年月日	変更内容
2014-04-01	第６版　下記を追加・変更しました ■「*リソース削除時のバックアップ*」の説明を修正 ■パブリックストレージのルートを示すパスの表記を「%PUBLIC_STORAGE_PATH%」に統一しました。 ■アプリケーションサーバ側のルートを示すパスの表記を「%CONTEXT_PATH%」に統一しました。
2014-08-01	第７版　下記を追加・変更しました ■「*認可情報のインポート・エクスポート*」にExcel（xlsx）形式について追記 ■「*認可設定画面*」 -「エクスポート」を「*インポート・エクスポート*」に変更 ■「*認可設定画面*」 -「*インポート・エクスポート*」にExcel（xlsx）形式について追記 ■「*運用時のTips*」に「*任意のタイミングでポリシーの設定状態を更新する*」を追加
2015-04-01	第８版　下記を追加・変更しました ■「*認可のキャッシュ設定*」に以下のキャッシュについての説明を追記 ■■「*ロールサブジェクトの正引きキャッシュ*」 ■■「*ロールサブジェクトの逆引きキャッシュ*」 ■■「*サブジェクト情報のキャッシュ*」 ■■「*サブジェクトグループ情報一覧のキャッシュ*」 ■「*予約されているリソースグループセット一覧*」、「*intra-mart Accel Platform に含まれるリソースタイプ*」に「IM共通マスタ ユーザプロファイル」「IM共通マスタ 個人プロファイル」を追加 ■「*認可 概要*」に「*制限事項*」を追加 ■「*認可サブジェクトコンテキスト*」の「*IM共通マスタ情報*」に、認可で扱う組織はデフォルト組織セットに属するもののみである旨を追記
2015-08-01	第９版　下記を追加・変更しました ■「*intra-mart Accel Platform に含まれるリソースタイプ*」の「IM共通マスタ 個人プロファイル」に定義アクションを追加
2015-12-01	第１０版　下記を追加・変更しました ■「*部品化*」に、「resource-group-authorizer要素」に指定するクラスの型を追記 ■「*サブジェクトリゾルバ*」に、「ロール（IwpRoleSubjectResolver）」はサブロールが含まれれる旨を追記 ■「*認可のキャッシュ設定*」の説明を追記 ■■「*リソースグループ設定のキャッシュ*」に、以下のリソースタイプがデフォルトでキャッシュされる旨を追記 ■■■会社一覧（im_master） ■■■IMBox（imbox-auth） ■■「*ポリシー設定のキャッシュ*」に、以下のリソースタイプがデフォルトでキャッシュされる旨を追記 ■■■会社一覧（im_master） ■「*intra-mart Accel Platform に含まれるリソースタイプ*」に、IM-LogicDesigner のリソースタイプの説明を追記 ■「*予約されているリソースグループセット一覧*」に、IM-LogicDesigner のリソースグループセットの説明を追記

変更年月日	変更内容
2016-04-01	第１１版　下記を追加・変更しました ■ 「*リソースグループ設定のキャッシュ* 」の「キャッシュサイズの計算式」にIMBoxのサイズの計算式を追記 ■ 「*サブジェクトリゾルバ* 」の「認証状態(AuthzMetaSubjectResolver)」に「認証済み外部ユーザ」の説明を追記 ■ 「*認可サブジェクトコンテキスト* 」の「AuthenticationDecorator」に「認証済み外部ユーザ」の説明を追記
2017-08-01	第１２版　下記を追加・変更しました ■ 「*intra-mart Accel Platform に含まれるサブジェクトタイプ*」に プロジェクトチーム機能のサブジェクトタイプの説明を追記

はじめに

本書の目的

本書では認可機構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。

- 認可機構の持つ機能の全体像
- 認可を実現する構造と動作
- 認可機構のもつ情報の操作と認可要求の方法

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 の認可設定を管理する運用担当者の方
- 認可機構を利用するアプリケーションを作成したい開発者の方

本書の構成

- *認可 概要*

  認可機構を提供する上で権限管理をどのようなモデルとしてとらえているかについて説明します。

- *各機能の詳細*

  概要で説明したモデルを実現するために構成を各部毎に説明します。

  - *認可情報管理系*

    認可機構で権限管理の源泉となる情報の内容と管理の手段について説明します。

  - *認可要求インタフェース*

    認可機構に対して認可判断を依頼するための窓口となる部分について説明します。

  - *サブジェクト解決系*

    サブジェクトとは単純にはユーザのことですが、認可機構ではユーザを組織やロールといった形でとらえ、認可判断に使用します。こういったユーザに付随する情報を認可機構に提供する仕組みについて説明します。

  - *認可判断機能*

    認可機構が内部でどのような方式で権限判定を行っているのかを説明します。

- *付録*

  認可の概念とは直接関わらないものの実装上採用している方針や、認可の拡張として標準的に提供されるものの一覧などを収録しています。

サンプルデータの使用について

本書ではAPIの利用方法の説明などにおいてサンプルコードを掲載しています。

このサンプルコード内で一部IM共通マスタのサンプルデータを使用しています。実際にサンプルコードを実行される場合、サンプルデータセットアップを実行して、IM共通マスタのサンプルデータをセットアップした上でお試しください。

認可 概要

intra-mart Accel Platform ではユーザに対する権限判断を行う処理を認可機構として切り出し、統一的な概念のもとにユーザの権限の管理を集約できるようにしています。

アクセス権モデル

ここでは intra-mart Accel Platform におけるアクセス権モデルを説明します。

intra-mart Accel Platform では「誰が」「何を」「どうする」という文脈に対してそれを許可するか禁止するかを権限設定情報として管理します。

これによって例えば以下のようにユーザの権限を表現することができます。

	例1	例2
だれが	管理者ロールのユーザ	営業部の社員
何を	メニュー管理機能	営業日報
どうする	使用（実行）	参照・登録
許可/禁止	許可	許可

誰が (Subject)

「誰が」を表す情報は **サブジェクト** と呼称します。（認可管理画面上では「対象者」と表示しています。）intra-mart Accel Platform では「誰が」を表す情報には複数の種類があります。たとえば、IM共通マスタ をインストールした状態では以下の種類があります。

- ユーザ
- ロール
- 会社組織
- 役職
- パブリックグループ
- パブリックグループ役割
- IPv4アドレス
- ゲストユーザ・認証済みユーザ（認証状態）
- 期間

何を (Resource)

「何を」を表す情報は **リソース** と呼称します。**リソース** はユーザが何らかの操作を行おうとする対象であり、開発者または開発者の作成したアプリケーションによってシステムに追加されます。たとえば intra-mart Accel Platform では ルータ の定義するURLが最も基本的な **リソース** になります。あるページのユーザのアクセス権を管理する場合、このURLに対してのアクセスを許可するか禁止するかを管理者が設定します。

どうする (Action)

「どうする」を表す情報は **アクション** と呼称します。 **アクション** はユーザがリソースに対して行おうとする操作です。リソースに

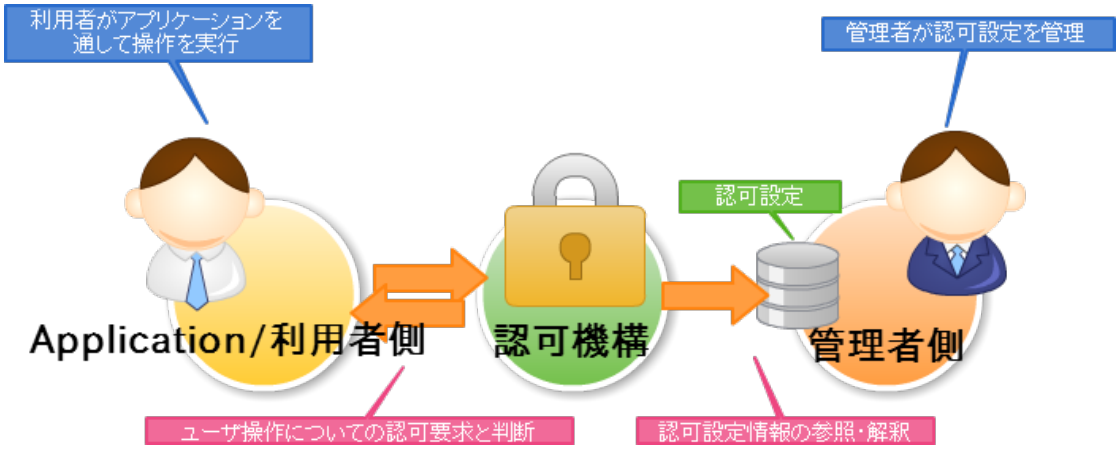
よって取りうる **アクション** は異なります。たとえば ルータ の定義するURL型のリソースでは利用可能なアクションは **実行(execute)** のみです。メニューを管理する際に作成されるメニューグループというリソースがありますが、こちらの場合 **管理(admin)**, **参照(read)** の2種のアクションが定義されています。

認可機構の構造

上記のようなモデルをもとに権限管理を行うに当たって、 intra-mart Accel Platform の認可機構は

1. 認可設定情報（アクセス権設定）を管理する管理者
2. システム上のリソースに対する操作を要求するユーザ
3. その要求を管理者の用意した設定をもとに許可・禁止の認可判定を行う認可機構

という三者の立場で構成されます。



認可設定情報（アクセス権設定）を管理する立場

システムに存在するリソースについて、どういったサブジェクトに対してどういった権限を与えるのかを設定するのが認可管理者の立場です。

認可要求を行う立場

システムに対してアクセスを行うユーザの立場です。ユーザはシステムにアクセスして様々な操作を行おうとしますが、この際にその操作が許可されているかを管理者の設定に従ってチェックされ、操作を制限されます。

実際には、ユーザが何かしらの操作を行おうとする際にアプリケーションがそのユーザが行おうとしている操作を許可してよいかを認可機構に問い合わせ、アプリケーションが操作を続行したり中断したりします。認可機構が直接ユーザの操作を遮断したりするわけではなく、認可機構を利用しようとするアプリケーションがユーザの代わりに認可要求を行い、その結果に応じて振る舞いを変えます。

認可判定を行う立場

ユーザの行おうとしている操作とその操作対象について、管理者の設定する認可設定情報を参照して許可するかどうかを判断します。これを行うのが intra-mart Accel Platform における認可機構になります。

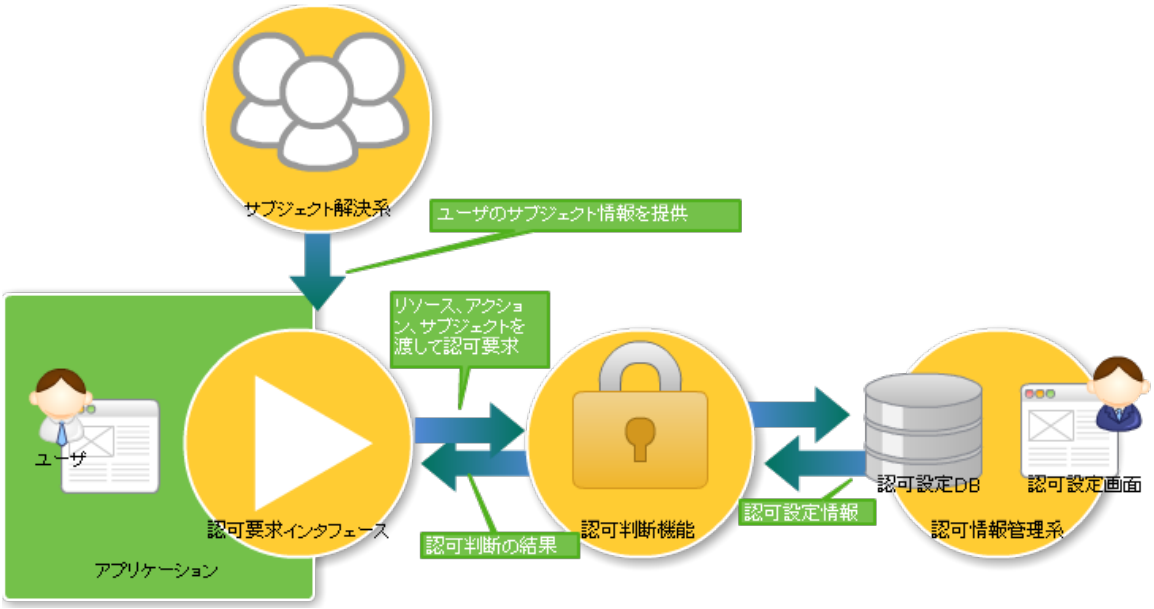
制限事項

認可機構の制限事項については 「 リリースノート 」 - 「 制限事項 」 - 「 認可 」 を参照してください。

各機能の詳細

認可機構を構成する各機能について説明します。

認可 概要 で述べたとおり認可機構では3つの立場があり、 その構造を実現するために大きく4つの部分で構成されています。



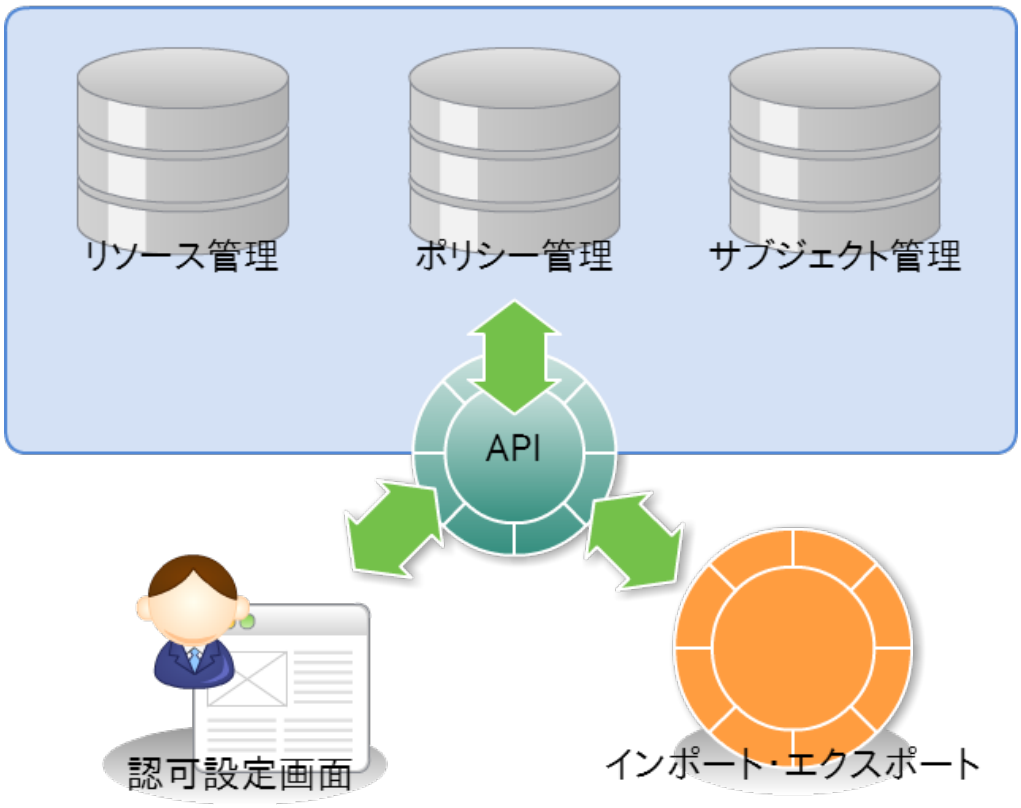
- 認可要求を行う立場に対して **認可要求インタフェース** を提供します。これは認可機構に対して認可要求を行うための入り口となる部分です。
- 認可機構ではIM共通マスタなどの情報を権限設定に使用できます。このためユーザがどの組織に所属しているかといったことが認可判断上とても重要になります。このユーザに付属する情報を捕捉するために**サブジェクト解決系** の機能が認可要求時にユーザ情報を認可機構に対して提供します。
- 認可要求インタフェースがユーザに対する認可がされるかどうかの問い合わせを受けて管理者の設定した情報と照らし合わせて認可するかどうかの判断処理を **認可判断機能** が行います。
- 認可判断機能の判断の根拠となる情報は管理者によって管理される必要があります。管理者が権限情報を管理しやすいよう、**認可情報管理系** の機能が管理者に対して認可設定用のユーザインタフェースやインポート／エクスポート機能、API等を提供しています。

以降の説明ではこれらの機能の詳細を解説していきます。

認可情報管理系

ここでは認可情報の管理系の機能について、個別に解説します。

管理系の機能ではサブジェクトとリソースをそれぞれ一括して管理するサブジェクト管理、リソース管理の機能があり、それらのサブジェクトやリソースを参照する形で作成される認可設定をポリシー管理機能が管理します。



- サブジェクト管理
  認可機構上に存在するサブジェクトやサブジェクトから構成される条件式であるサブジェクトグループを一括して管理します。
  サブジェクトにはサブジェクトタイプという分類があり、この分類ごとに認可機構外の情報と連携を定義しています。このサブジェクトタイプもここで管理しています。
- リソース管理
  認可機構上に存在するリソースや、リソースを階層化する構造情報であるリソースグループを一括して管理します。
  リソースにはリソースタイプという分類があり、リソースURIの記法やリソースの持つアクションを定義しています。このリソースタイプもここで管理しています。
- ポリシー管理
  ポリシー管理は上記のサブジェクト管理で管理されるサブジェクトグループと、リソース管理で管理されるリソースグループとアクションの情報を使用して、認可設定情報をポリシーという形で管理します。
- 認可設定画面
  上記のサブジェクト、リソース、ポリシーの情報を運用管理者が画面から編集するための機能です。編集や認可設定の運用を補助するための機能を提供します。
- インポート・エクスポート
  認可設定画面の用に画面から逐一変更するのではなく、XMLで用意した各種の情報から一括で更新したり出力したりする為の機能を提供します。

以降の章で順に説明します。

サブジェクト管理

ここでは認可情報のサブジェクト管理系の機能について、個別に解説します。

項目

サブジェクトとサブジェクトグループ

サブジェクト

サブジェクトは認可対象者を示す情報です。ただしこれは「特定の誰か」に限らず、ユーザの所属できる組織やロールなどを含みます。ユーザが複数の組織やロールに所属していれば複数のサブジェクトとして解釈されます。

サブジェクトはどのような種類のものであるかにかかわりなく認可機構内でユニークなサブジェクトIDとして管理されます。

サブジェクトグループ

認可機構で権限の付与を行う際、サブジェクトをグループ化した単位で設定を行います。グルーピングには条件指定が可能であり、以下の演算をサポートしています。

OR
　含まれるサブジェクトのいずれかにマッチした場合のみ、そのサブジェクトグループに属すると判断されます。

AND
　含まれるサブジェクトのすべてにマッチした場合のみ、そのサブジェクトグループに属すると判断されます。

NOT
　一致するサブジェクト、またはサブジェクトグループに含まれる場合、そのサブジェクトグループに属さないと判断されます。

たとえば、以下のようなグループを条件として構成することができます。

```
- 開発部 AND 課長
- 総務部 AND NOT（ 協力会社員グループ ）
- （ 開発部 OR 営業部 OR 企画部 ） AND （ NOT 協力会社員グループ ）
```

サブジェクトグループカテゴリ

サブジェクトグループカテゴリはAPIが自動生成する情報です。サブジェクトマネージャを使用して新たなサブジェクトグループを登録すると、サブジェクトグループ内で使用されているサブジェクトタイプを取り出し、そのサブジェクトタイプをもとにサブジェクトグループカテゴリを生成します。

カテゴリの生成にかかわる仕様は以下の通りです。

- グループ内に存在するサブジェクトタイプを収集
- 収集したサブジェクトタイプをもとに以下を決定します。
  - サブジェクトグループカテゴリIDの決定
    - サブジェクトタイプが3種以上の場合 (dummy) のハッシュ値をカテゴリIDとして使用
    - サブジェクトタイプが2種の場合、 : をデリミタとしてサブジェクトタイプIDを連結した文字列のハッシュ値をカテゴリIDとして使用
    - サブジェクトタイプが1種の場合、 サブジェクトタイプIDのハッシュ値をカテゴリIDとして使用
  - サブジェクトグループカテゴリ名の決定（上記IDがすでにテーブルに存在する場合それを使用）
    - サブジェクトタイプが3種以上の場合 「その他複合」
    - サブジェクトタイプが2種の場合、 「サブジェクトタイプA、サブジェクトタイプBの複合」
    - サブジェクトタイプが1種の場合、 サブジェクトタイプ名をそのまま使用

サブジェクトタイプ

サブジェクトタイプは認可機構のサブジェクトとサブジェクトの実体の情報を相互にやり取りするための仕組みです。サブジェクトは必ずいずれかのサブジェクトタイプに属し、サブジェクトタイプを通してサブジェクトの実体情報との相互変換ができます。

たとえば IM共通マスタ では会社組織、役職、パブリックグループ、役割といったサブジェクトタイプを用意しています。そのため認可機構でこれらの情報をサブジェクトとして取り扱うことができます。

サブジェクトタイプは SubjectType インタフェースを実装するJavaクラスで定義されます。このクラスの追う責務は以下のようなものです。

- サブジェクトタイプのID、名称を決定する
- サブジェクトIDと実体の情報を紐づける
- サブジェクトの実体の情報の式表現へ直列化、および、復元を行う

サブジェクトタイプの定義

サブジェクトタイプは必要に応じて、実装を追加することができます。サブジェクトタイプを定義するためには SubjectType インタフェースを実装したクラスと、サブジェクトを扱うためのモデルクラスを作成する必要があります。

サブジェクトタイプは以下のクラスを実装します。

完全修飾クラス名
　jp.co.intra_mart.foundation.authz.model.subjects.SubjectType<T>

型引数Tはサブジェクトのモデルクラスです。このモデルクラスには特に制約はありません。ここで定義したモデルクラスを使用してサブジェクト登録を行ったりすることができるようになるので、役割として適切であればアプリケーションなどで使用している既存のクラスをそのまま使用したほうが取り扱い易くなるでしょう。

> **注意**
> SubjectTypeとサブジェクトモデルクラスは1対1の関係で定義されている必要があります。他のサブジェクトタイプとモデルを共有することはできないので注意してください。

上記で定義したクラスは設定ファイルに記載することで認可機構に認識させます。以下のXMLファイルを作成します。

場所
　%CONTEXT_PATH%/WEB-INF/conf/authz-subject-type-config/ 配下

XMLスキーマ
　%CONTEXT_PATH%/WEB-INF/schema/authz-subject-type-config.xsd

下記は設定例です。下記のように、 **subject-type 要素** の **type-class 属性** にサブジェクトタイプの実装クラスを、 **model-class 属性** にモデルクラスを、それぞれ完全修飾クラス名で記述します。

```
<?xml version="1.0" encoding="UTF-8"?>
<p:authz-subject-type-config
   xmlns:p="http://www.intra-mart.jp/authz/authz-subject-type-config/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.intra-mart.jp/authz/authz-subject-type-config/ ../../schema/authz-subject-type-
config.xsd">

  <subject-type
    type-class="jp.co.intra_mart.foundation.authz.subjecttype.im_master.ImRole"
    model-class="jp.co.intra_mart.foundation.admin.role.model.RoleInfo" />

</p:authz-subject-type-config>
```

サブジェクトタイプの追加にあたっては別途 「認可拡張プログラミングガイド」の「サブジェクト拡張ガイド」の章 で手順や必要な準備について解説しています。サブジェクトタイプを追加する際には、上記の資料もあわせて参照してください。

intra-mart Accel Platform の標準的なモジュール構成においてデフォルトでインストールされる サブジェクトタイプの詳細については「*intra-mart Accel Platform に含まれるサブジェクトタイプ*」を参照してください。

基準日によって名称が変わるサブジェクトタイプ

期間情報を持ち、基準日によって名称が変わる場合は、SubjectType インタフェースを内包する ChangeableNameSubjectType インタフェースを利用できます。
ChangeableNameSubjectType インタフェースを利用したサブジェクトタイプの場合、「認可設定」画面で対象者条件を設定する際に基準日に応じた名称を表示します。

ChangeableNameSubjectType は名称の解決に基準日を使用するメソッドを提供します。 それ以外は SubjectType と同じです。

完全修飾クラス名
　jp.co.intra_mart.foundation.authz.model.subjects.ChangeableNameSubjectType<T>

> **コラム**
> ChangeableNameSubjectType インタフェースは intra-mart Accel Platform 2013 Winter から提供しています。

API

サブジェクト管理に関係する主要なAPI上の操作を説明します。詳細に関してはAPIリストを参照してください。

マネージャインスタンスの取得

サブジェクト管理の主要な操作を行うためには SubjectManager クラスを使用します。
マネージャクラス、および、ファクトリクラスは以下の通りです。

マネージャクラス
　jp.co.intra_mart.foundation.authz.services.admin.SubjectManager

マネージャファクトリ
　jp.co.intra_mart.foundation.authz.services.admin.SubjectManagerFactory

マネージャインスタンスはファクトリクラスを使用して取得します。ファクトリも自身のファクトリメソッドを持っていますので、以下のようにして取得します。

```
// マネージャインスタンスの取得
final SubjectManager subjectManager = SubjectManagerFactory.getInstance().getSubjectManager();
```

APIを利用したサブジェクトの管理

サブジェクトの登録

特定サブジェクトタイプの主キー文字列からサブジェクトを登録する場合は以下のようにします。IM共通マスタ のユーザ aoyagi をサブジェクトとして登録する例です。

```
// サブジェクトタイプIDと、主キー情報を与えてサブジェクトを登録
// キー情報の与え方はサブジェクトタイプの定義によります
final SubjectManager subjectManager = SubjectManagerFactory.getInstance().getSubjectManager();
Subject aoyagiSubject = subjectManager.registerAsSubject("imm_user", "aoyagi");
```

サブジェクトタイプに定義されるモデルクラスを使ってサブジェクトを登録することもできます。この定義は

%CONTEXT_PATH%/WEB-INF/conf/authz-subject-type-config

に格納されているxmlファイルで定義されています。

IM共通マスタ「ユーザ」の場合の実装例は、以下の通りです。

```
// IM共通マスタのUserManagerを利用してユーザのモデルを取得
UserManager userManager = new UserManager();
IUserBizKey bizKey = new UserBizKey();
bizKey.setUserCd("aoyagi");
User user = userManager.getUser(bizKey, new Date());

// ユーザのモデルからサブジェクト登録
Subject aoyagiSubject = subjectManager.registerAsSubject(user);
```

SubjectManager#registerAsSubject() は既に登録されているサブジェクトを再度登録しようとした場合、同じ物を返します。（エラーに成りません）

サブジェクトグループの登録

サブジェクトの条件式を登録する場合はサブジェクトをベースに式からサブジェクトグループを作成します。上記までで青柳のサブジェクトを作成した前提で、「青柳または上田」のサブジェクト条件を作成する例です。

```
// サブジェクトグループに付加する名称（日本語のみ）を作成します
I18nValue<String> name = new I18nValue(Locale.JAPANESE, "上田または青柳");
// サブジェクトグループに付加する説明（日本語のみ）を作成します
I18nValue<String> desc = new I18nValue(Locale.JAPANESE, "上田または青柳です");

// 上田のサブジェクトを登録
Subject uedaSubject = subjectManager.registerAsSubject("imm_user", "ueda");

// OR(S(imm_user:aoyagi),S(imm_user:ueda)) の API での記述
Expression e = SubjectExpression.OR(SubjectExpression.S(aoyagiSubject), SubjectExpression.S(uedaSubject));
// 条件式をサブジェクトグループとして登録
SubjectGroup group = subjectManager.registerSubjectGroup(e, name, desc);
```

サブジェクトグループの取得

式かサブジェクトグループIDでの取得を行えます。存在しない場合、nullが返却されます。

```
// サブジェクトグループIDによる取得
SubjectGroup groupA = subjectManager.getSubjectGroup(group.getSubjectGroupId())

// 式による取得
Expression e = SubjectExpression.OR(SubjectExpression.S(aoyagiSubject), SubjectExpression.S(uedaSubject));
SubjectGroup groupB = subjectManager.getSubjectGroupByExpression(e);
```

サブジェクトグループの名称、説明の更新

サブジェクトグループの名称や説明を更新する場合は個別にマネージャのメソッドをコールします。

```
// サブジェクトグループに付加する名称（日英）
final I18nValue<String> name = new I18nValue(Locale.JAPANESE, "上田または青柳");
name.put(Locale.ENGLISH, "ueda or aoyagi");

// 名前の更新
subjectManager.setSubjectGroupNames(subjectGroupId, name);

// サブジェクトグループに付加する説明（日英）
final I18nValue<String> desc = new I18nValue(Locale.JAPANESE, "上田または青柳です");
desc.put(Locale.ENGLISH, "aoyagi or ueda");

// 説明の更新
subjectManager.setSubjectGroupDescriptions(subjectGroupId, desc);
```

サブジェクトグループの削除

サブジェクトグループを削除すると、関係するサブジェクトが他のサブジェクトグループから参照されていなければ同時に削除します。

```
// サブジェクトグループを削除
subjectManager.removeSubjectGroup(group);
```

式表現

式の文字列表現

サブジェクトグループは文字列化した式表現を持っています。IM共通マスタ「ユーザ」で `aoyagi` を示す場合の例は、以下の通りです。

```
S(imm_user:aoyagi)
```

`S( ... )` はサブジェクトを表す式であり、`imm_user` はこのサブジェクトのサブジェクトタイプを示しています。`:` より右はこのサブジェクトタイプによって定義される識別子で、どういう書式で何を意味するかはサブジェクトタイプが決定します。

この例の場合、`imm_user` は IM共通マスタ が用意しているユーザマスタのサブジェクトタイプIDです。`:` の右側の `aoyagi` はこのサブジェクトタイプの定義するキー情報です。この場合は、 `aoyagi` というユーザコードを意味しています。

また、論理演算子として `OR` `AND` `NOT` が使用できます。

演算子	説明
S	サブジェクトを表す式。括弧内には subject-type-id:subject-id-string の形式でサブジェクトを識別可能な文字列を指定します。このsubject-id-stringの書式はサブジェクトタイプによって定義されます。評価対象がこの文字列によって定義されるサブジェクトに一致する場合のみ真を返します
OR	括弧内に複数の式を記述でき、評価対象がいずれかの式に一致する場合真を返します
AND	括弧内に複数の式を記述でき、評価対象が全ての式に一致する場合真を返します
NOT	括弧内に単一の式のみ記述できます。評価対象が一致しない場合に真を返します

これらを使用した式の書式は、以下の通りです。

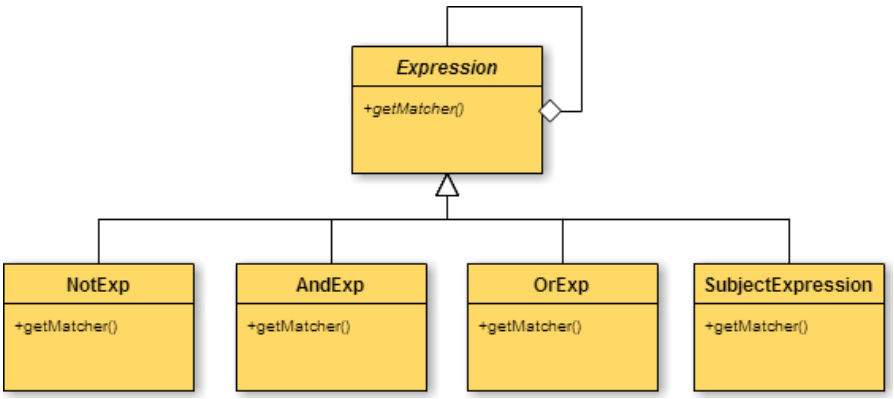
```
式：演算子( 式 | 式 [,式] ... )
```

ユーザコード `aoyagi` または `ueda` を示す場合の例は、以下の通りです。

```
OR(S(imm_user:aoyagi),S(imm_user:ueda))
```

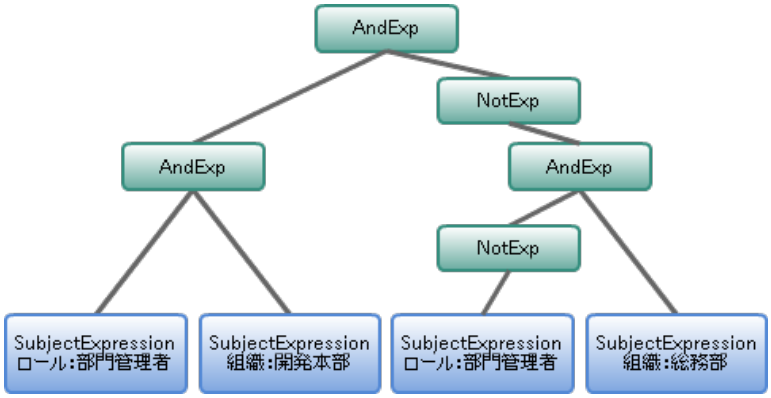
式クラス表現

式は以下のクラスで表現されます。



- 各演算子毎のExpressionクラスが定義されています。
- AndExpクラス , OrExpクラス は複数の式を内包できます。それぞれ文字列表現における OR() 、 AND() に対応します。
- NotExpクラス が内包することができるのは単一の式のみです。文字列表現における NOT() に対応します。
- SubjectExpressionクラス サブジェクトを表現する演算子です。文字列表現における S() に対応します。
  式の文字列表現 で述べている通り、この式の内容はサブジェクトタイプによって定義されます。

実際にこれらのクラスを使用して式のインスタンスを構築すると以下のような構造を取ります。



- AndExpクラス , OrExpクラス , NotExpクラス は 別の式のインスタンスを内包できるため、階層的な構造をとります。
- SubjectExpressionクラス は必ず末端に記述します。逆にそれ以外の演算子が末端の式に現れることはありません。

式のマッチング仕様

登録されたサブジェクトグループ群に対して、ユーザがサブジェクトグループにマッチするかどうか評価する必要があります。ユーザがどのサブジェクトグループに当たるかの判定はサブジェクト解決系の処理が起点となって行います。サブジェクト解決処理の概要については「*サブジェクト解決系* 」を参照してください。

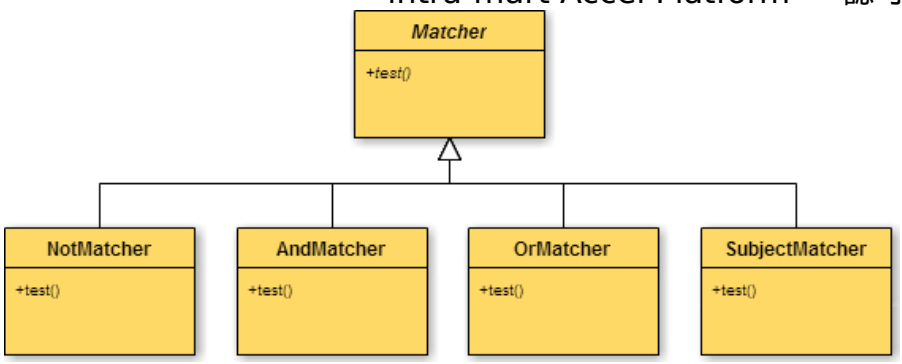
サブジェクト解決系の処理の中では、以下の2段階で処理を行っています。

1. 認可サブジェクトコンテキストまたはサブジェクトリゾルバを使用してユーザのサブジェクトを解決し
2. 解決されたサブジェクトからサブジェクト管理に登録されているサブジェクトグループ（条件）のどれにマッチするかを判断する

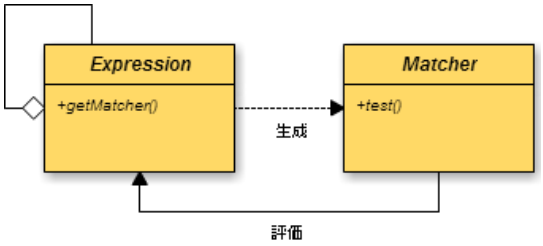
ここでは上記 2 でユーザの該当するサブジェクトが解決された状態で式とのマッチングをどのように行っているかを説明します。

まず、ユーザの該当するサブジェクトから、合致する可能性のあるサブジェクトグループ（条件式）を探します。これは単純にユーザのサブジェクトを使用したサブジェクトグループであるかどうかを条件にデータベース上から取得し、式オブジェクト（ Expression クラスのインスタンス）を構築します。

マッチングにはExpressionクラスと同様の構成をもつMatcherクラスを使用します。以下はマッチングに使用される主要なクラスです。



Matcher クラスはそれぞれ Expression クラスに対応する形で定義されています。Matcher クラスと Expression クラスの関係を下図に示します。



1. Expression クラスの getMatcher() メソッドは Expression クラスの実行時型に対応するMatcher クラスのインスタンスを返します。
2. このインスタンスは生成元の Expression インスタンスを保持しており、 test() メソッドがコールされると生成元 Expression インスタンスから、内包している Expression インスタンスを取り出し
3. 取り出した Expression インスタンスそれぞれに対して getMatcher() を行い
4. 取得したMatcherインスタンスの test() メソッドをコールします。これは再帰的に処理され、最終的に末端である SubjectExpression クラスのインスタンスまで実行されます。
5. SubjectMatcher は引数に与えられた Subject が自分の対応する Expression に一致するかどうかをチェックし、マッチしたかどうかの結果を上位の式の Matcher に返します。
6. この結果は順に上位の式に集約され、最終的に最上位の式の Matcher まで戻ります。
7. 最上位の Matcher が最後の評価を行い、マッチしたかどうかの結果を返します。

省略処理

式は以下のような観点での省略処理をします。

- 無駄なネストの排除
- オペランドのソート

演算子によって以下のように省略処理を行います。

ANDの場合

1. オペランドを順に省略処理する（再起処理）
2. オペランドが AND 演算子を使用（ AND のネスト）している場合、オペランドの AND を外す
   - 例） AND(a, b, AND(c, d)) ⇒ AND(a, b, c, d)
3. オペランドの重複を排除
   - 例） AND(a, b, a, b) ⇒ AND(a, b)
4. オペランドをソート
   - 例） AND(a, b, d, c) ⇒ AND(d, c, b, a)

ORの場合

1. オペランドを順に省略処理する（再起処理）

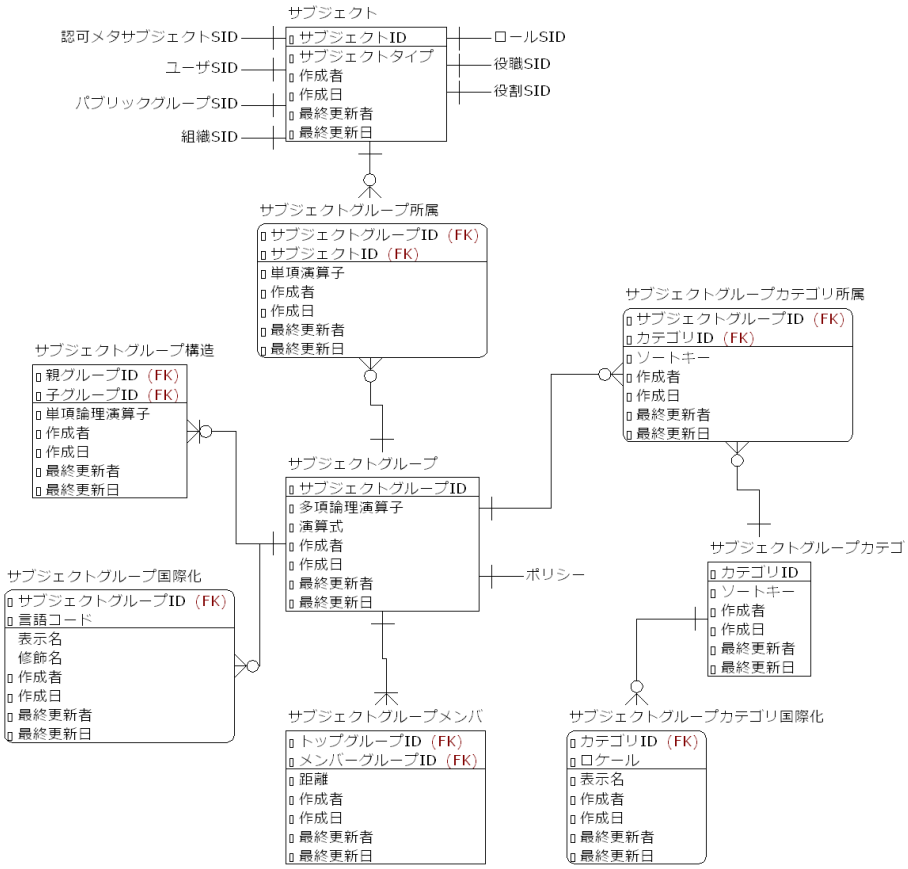
2. オペランドが OR 演算子を使用（ OR のネスト）している場合、オペランドの OR を外す
    - 例） OR(a, b, OR(c, d)) ⇒ OR(a, b, c, d)
3. オペランドの重複を排除
    - 例） OR(a, b, a, b) ⇒ OR(a, b)
4. オペランドをソート
    - 例） OR(a, b, d, c) ⇒ OR(d, c, b, a)

NOTの場合

1. オペランドを省略処理する（再起処理）
2. オペランドがNOT演算子を使用している場合、自身とオペランドの NOT を打ち消す。
    - 例） NOT(NOT(A)) ⇒ A

ERとテーブル上の表現

サブジェクト管理に関係するテーブルの ER は以下の通りです。



ここでは API 上のモデルと ER 上の関連を中心に説明します。テーブルの定義や認可機構全体の ER については別冊のテーブル定義書、 ER 図を参照してください。

サブジェクト(imaz_subject)

主要なフィールド	
論理名	**物理名**
サブジェクトID	sid
サブジェクトタイプ	subject_type

サブジェクトが追加・削除されたタイミングでレコードも追加削除されます。サブジェクトIDはサブジェクトタイプとサブジェクトの実体のキーの値をもとに生成したハッシュ値です。このため、サブジェクトテーブルのレコードに対して更新は原則発生しません。

サブジェクトテーブルは、サブジェクトIDのマスタ情報ですが、サブジェクトIDが紐づけられた実体の情報はサブジェクトタイプクラスを通して引き当てる必要があります。

サブジェクトグループ所属(imaz_subject_group_ath)

主要なフィールド	
論理名	**物理名**
サブジェクトID	sid
サブジェクトグループID	subject_group_id
単項演算子	monadic_operator

単純にサブジェクトグループとサブジェクトを紐づけます。サブジェクトグループの構成の中にサブジェクトも含まれるため、サブジェクトグループを登録する際にこのテーブルのレコードも追加されます。 単項演算子フィールドは NOT 接続かどうかを示します。

たとえば `AND(S(X))` のような式の場合に、AND がサブジェクトグループ、S(X) がサブジェクトとして登録され、その関係をこのサブジェクトグループ所属テーブルが保持します。`ADN(NOT(S(X)))` のような式の場合、間に `NOT` が入っているので、単項演算子の値に `NOT` を示す値 が入ります。

サブジェクトグループ(imaz_subject_group)

主要なフィールド	
論理名	**物理名**
サブジェクトグループID	subject_group_id
論理演算子	logical_operator
式	expression

サブジェクトグループを保持します。サブジェクトグループは内部的には式表現における `AND` または `OR` と同等の構造を表し、式のインスタンス同様、ツリー構造で表します。サブジェクトグループIDは式表現を省略処理した上でハッシュ化した値をもとに生成されるため、式が変更されると別のサブジェクトグループIDに変わります。一度登録されたサブジェクトグループに対して更新が発生することは原則ありません（名称、説明を除き）。

式構造はサブジェクトグループの階層として表現されます。このため式フィールドは登録時メモとして保持していますが、実際に式の復元に使用されません。

サブジェクトグループ国際化(imaz_subject_group_i)

主要なフィールド	
論理名	**物理名**
サブジェクトグループID	subject_group_id
ロケールID	locale_id
表示名	display_name
説明	description

サブジェクトグループについて名称と説明を保持します。双方、各ロケールことに保持することができます。

サブジェクトグループ構造(imaz_subject_group_struct)

主要なフィールド	
論理名	**物理名**
親グループID	parent_group_id
子グループID	child_group_id
単項演算子	monadic_operator

サブジェクトグループ間の直接の親子関係を保持します。サブジェクトグループの階層構造は、条件の式表現を反映したものです。ネストした式を表現するためにある式とその式が内包する式との関係を表しています。

このテーブルもサブジェクトグループ所属テーブル同様、NOT接続のために単項演算子フィールドがあります。

サブジェクトグループメンバ(imaz_subject_group_member)

主要なフィールド	
論理名	**物理名**
トップグループID	top_group_id
メンバーグループID	member_group_id
距離	distance

このテーブルはサブジェクトグループ構造テーブルの参照を高速化するための情報を保持します。APIからサブジェクトグループ登録処理を呼び出した際に、作成されるサブジェクトグループのツリー構造における先頭ノードと各構成ノードの関連を保持します。リソースグループ内包テーブルのように各要素間の情報を保持しているわけではありません。

サブジェクトグループカテゴリ(imaz_subject_group_cat)

主要なフィールド	
論理名	**物理名**
カテゴリID	category_id
ソートキー	sort_key

サブジェクトグループカテゴリはAPIが自動生成する情報です。このテーブルの情報の生成に関する詳細は「*サブジェクトグループカテゴリ*」を参照してください。

カテゴリIDはサブジェクトグループが含むサブジェクトタイプIDのハッシュ値をもとに生成されます。ソートキーを更新することができます。

サブジェクトグループカテゴリ国際化(imaz_subject_group_cat_i)

主要なフィールド	
論理名	**物理名**
カテゴリID	category_id
ロケールID	locale_id
表示名	display_name

サブジェクトグループカテゴリの国際化情報を保持します。この情報はサブジェクトグループカテゴリを生成するタイミングで自動的に作成されます。(*サブジェクトグループカテゴリ* 参照)この国際化情報はサブジェクトタイプの国際化情報と、デリミタなどの国際化情報を使用して、テナントで有効になっているロケール分だけ生成します。

サブジェクトグループカテゴリ所属(imaz_subject_group_cat_ath)

主要なフィールド	
論理名	**物理名**
サブジェクトグループID	subject_group_id
カテゴリID	category_id
ソートキー	sort_key

サブジェクトグループの所属するサブジェクトカテゴリの情報を保持します。サブジェクトグループの情報をもとにサブジェクトグループカテゴリが自動的に生成されるので、同時に作成されます。(*サブジェクトグループカテゴリ* 参照)サブジェクトグループカテゴリ毎にサブジェクトグループのソートキーを保持します。

リソース管理

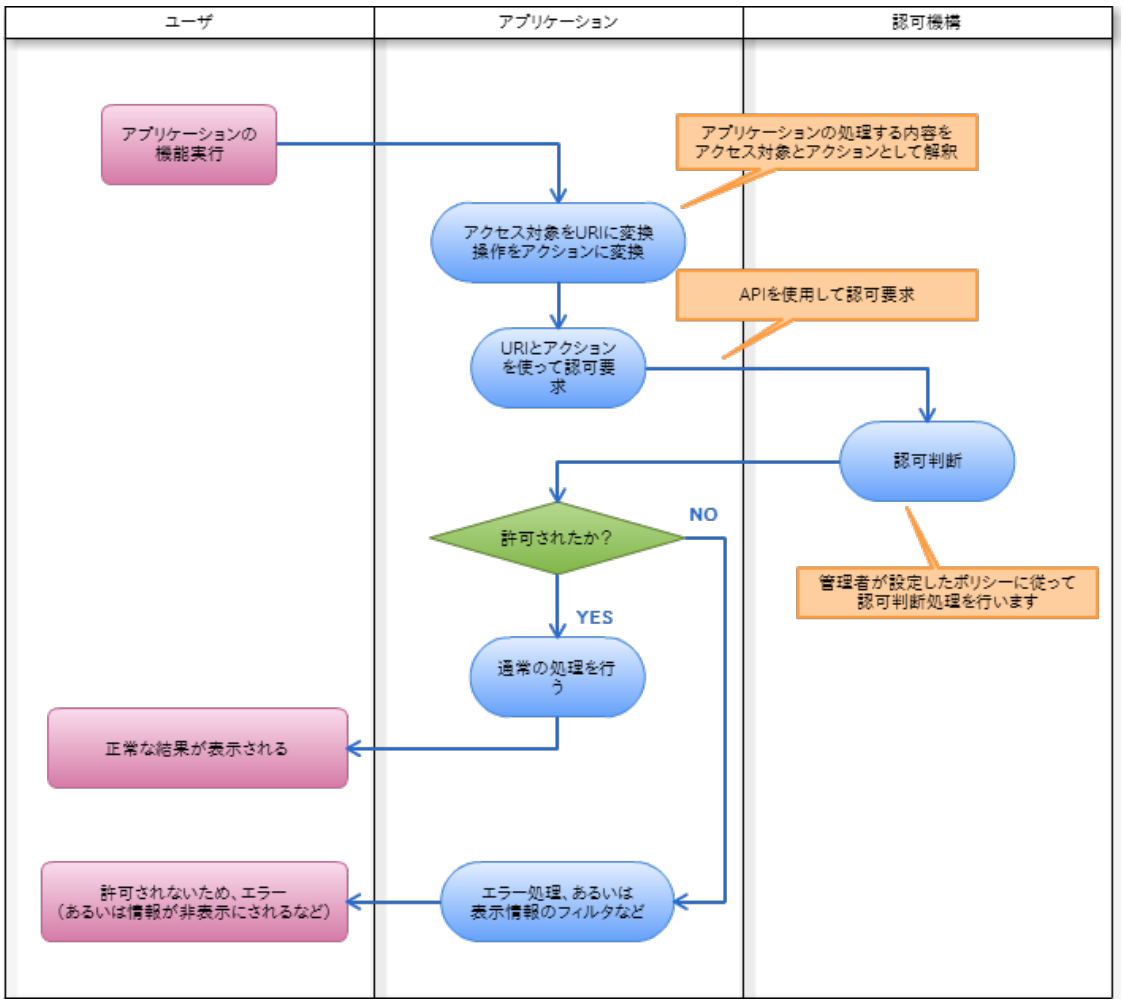
項目

リソースとリソースグループ

認可 概要 で紹介した通り、認可を行う際の「何を」にあたる情報がリソースです。

リソースのキーとなる情報はリソースURIです。アプリケーションで認可機構を利用する場合、認可対象になるもの（「何を」にあたります）をリソースURIとして表現し、予め認可機構に登録します。これによって管理者がそのリソースの権限を管理できるようになります。

ユーザがそのリソースを利用しようとした際には、アプリケーションはリソースURIで認可機構に問い合わせることで管理者がそのリソースをユーザに対して利用を許可したかどうかを判断することができます。



リソースはリソースグループを使用して権限設定を階層的にまとめることができます。階層的に構成したリソースとリソースグループでは、上位に設定した権限設定を継承させて適用することができます。（この動作の詳細は「*ポリシー解釈器*」を参照してください）

これによってある程度まとまったリソースに対して認可設定を行うことができるようになっています。

認可機構に対してAPIを使ってリソースを作成すると、リソースと対になるリソースグループが1つ作成されます。 APIモデル上ではリソースとリソースグループの持つ情報は以下のようになっています。

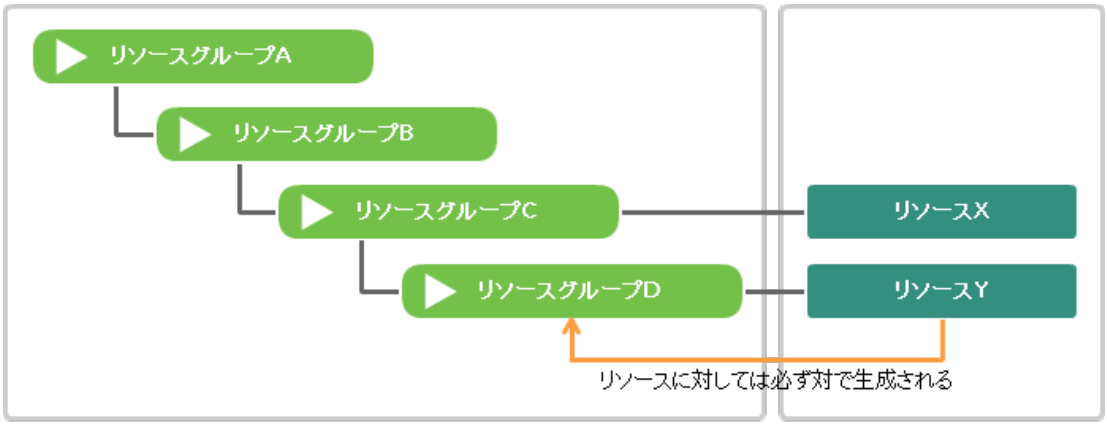
リソース

- リソースURI（、および、ID）
- リソースタイプ

リソースグループ

- ID
- 名称
- 説明
- グループの階層にかかわる情報

リソースは名称や説明を持ちません。リソースと対になっているリソースグループがリソースの名称や説明にあたる情報を保持します。このためAPIモデル上ではリソースとはリソースグループとリソースのセットの事になります。



リソースグループは階層構造を表現するための情報

リソースグループIに対してリソースは0か1

リソースURIの仕様

リソースURIは認可機構がシステム上で認可対象を特定するためのシステム上一意な識別子です。リソースURIは以下のような構成の文字列です。

`RESOURCE-TYPE-ID:IDENTIFIER-COMPONENT`

それぞれの意味合いを説明します。

RESOURCE-TYPE-ID
　このリソースの振る舞いを決める *リソースタイプ* のIDがまず先頭に必要です。

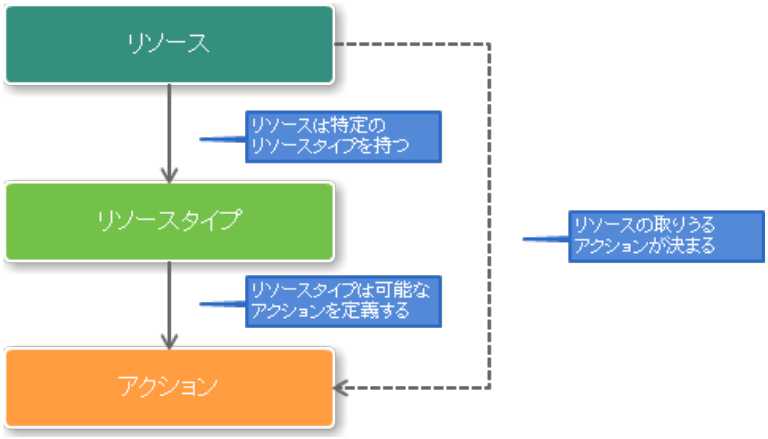
:（コロン）
　`RESOURCE-TYPE-ID` と `IDENTIFIER-COMPONENT` を分ける記号として `:`（コロン）を使います。

IDENTIFIER-COMPONENT
　このリソースをシステムで一意に判別するためのIDを表す文字列です。アプリケーション名やアプリケーション内のコンポーネント名などで階層を区切って、システム上他のアプリケーション等と衝突することの無いよう設計する必要があります。

リソースURIの例) `service://authz/settings/basic`

リソースタイプとアクション

認可 概要 で「どうする」として紹介しているものをアクションと呼びます。アクションはユーザがリソースに対してどういった操作を行うかを表すものです。リソースは必ず単一のリソースタイプを持っており、このリソースタイプがどういったアクションをとりうるのかを決定しています。



リソースタイプとアクションはJavaクラスとして実装されています。リソースタイプは以下のような責務を負っています。

- リソースタイプIDの定義
- このリソースタイプで使用するアクションの定義・取扱い
  - 取りうるアクションの列挙
  - アクションのパース
- リソースタイプにバインドされるリソースモデルの決定

- リソースモデルのURIへの変換

アクションもリソースタイプ同様にJavaクラスとして実装され、 リソースタイプによってその関連が明示されます。

リソースタイプの定義

リソースタイプおよびアクションは必要に応じて実装を追加することができるようになっています。リソースタイプを定義するためには ResourceType インタフェースを実装したクラスと、リソースを扱うためのモデルクラスを作成する必要があります。

リソースタイプは以下のクラスを実装します。

完全修飾クラス名
`jp.co.intra_mart.foundation.authz.model.resources.ResourceType<T>`

型引数Tはリソースのモデルクラスです。このモデルクラスには特に制約はありません。ここで定義したモデルクラスを使用してリソース登録を行ったりすることができるようになるので、役割として適切であればアプリケーションなどで使用している既存のクラスをそのまま使用したほうが取り扱い易くなるでしょう。

> **注意**
> ResourceTypeとリソースモデルクラスは1対1の関係で定義されている必要があります。他のリソースタイプとモデルを共有することはできないので注意してください。

上記で定義したクラスは設定ファイルに記載することで認可機構に認識させます。以下のXMLファイルを作成します。

場所
`%CONTEXT_PATH%/WEB-INF/conf/authz-resource-type-config/` 配下

XMLスキーマ
`%CONTEXT_PATH%/WEB-INF/schema/authz-resource-type-config.xsd`

下記は設定例です。下記のように、 **resource-type 要素** の **type-class 属性** にリソースタイプの実装クラスを、**model-class 属性** にモデルクラスを、それぞれ完全修飾クラス名で記述します。 **cache-class 属性** は任意項目で、リソースのキャッシュコントローラクラスを完全修飾クラス名で記述します。リソースのキャッシュを行わない場合は、記述する必要はありません。

```
<?xml version="1.0" encoding="UTF-8"?>
<p:authz-resource-type-config
   xmlns:p="http://www.intra-mart.jp/authz/authz-resource-type-config/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.intra-mart.jp/authz/authz-resource-type-config/ ../../schema/authz-resource-type-config.xsd">

 <resource-type
    type-class="jp.co.intra_mart.system.router.authz.resourcetype.GeneralServiceResourceType"
    model-class="jp.co.intra_mart.system.router.authz.resourcetype.GeneralServiceModel"
    cache-class="jp.co.intra_mart.system.router.authz.resourcetype.GeneralServiceResourceTypeCacheController" />

</p:authz-resource-type-config>
```

リソースタイプの追加にあたっては別途 「認可拡張プログラミングガイド」の「リソース拡張ガイド」の章 で手順や必要な準備について解説しています。 リソースタイプを追加する際にはこちらも参照してください。

intra-mart Accel Platform の標準的なモジュール構成においてデフォルトでインストールされるリソースタイプの詳細については「*intra-mart Accel Platform に含まれるリソースタイプ*」を参照してください。

リソースのキャッシュコントローラの定義

リソースの情報が頻繁に利用されることが予想される場合、適切なキャッシュを行うことでパフォーマンスを改善することができます。キャッシュを行うためのコントローラクラスを実装することで、リソースタイプごとに異なる戦略でキャッシュを実装することができます。コントローラクラスを定義するためには ResourceTypeCacheController インタフェースを実装したクラスを作成する必要があります。

リソースのコントローラクラスは以下のクラスを実装します。

完全修飾クラス名
`jp.co.intra_mart.foundation.authz.services.admin.impl.cache.ResourceTypeCacheController`

リソースのキャッシュコントローラの追加にあたっては別途 「認可拡張プログラミングガイド」の「リソース拡張ガイド」の章 で手順や必要な準備について解説しています。 リソースのキャッシュコントローラを追加する際にはこちらも参照してください。

intra-mart Accel Platform の標準的なモジュール構成においてデフォルトでインストールされるキャッシュコントローラは、*intra-mart Accel Platform に含まれるリソースタイプ*の「画面・処理(service)」のリソースタイプに設定されている GeneralServiceResourceTypeCacheController クラスです。

リソースグループセット

リソースグループは階層構造をとっていますが、階層の先頭のリソースグループとその配下のツリーをリソースグループセットと呼称しています。

リソース管理上、リソースグループセット毎に特定の目的をもってツリー構造が構成されています。 特定のリソースグループセットとそのIDは intra-mart Accel Platform で予約されています。新たにリソースグループセットを追加する場合は衝突がないよう注意してください。予約されているリソースグループセットについては「*予約されているリソースグループセット一覧* 」を参照してください。

リソースグループ汎用属性

リソースグループには、システム稼動後に認可リソースを登録する機能が自由に設定可能な汎用属性を設定することができます。

認可機能では以下の機能を実現するために利用しています。

- リソースの閉塞

> **コラム**
> リソースの閉塞については「*閉塞*」を参照してください。

設定内容は任意の文字列で構成された属性キーと属性値の組み合わせで、マップのように管理することができます。 属性キーは他の機能と重複しないよう、機能を示す固有ID（モジュールID）を接頭子にします。

> **コラム**
> リソースグループ汎用属性は intra-mart Accel Platform 2013 Summerより利用可能です。

API

リソース管理に関係する主要なAPI上の操作を説明します。詳細に関してはAPIリストを参照してください。

リソースの管理

マネージャインスタンスの取得

リソース管理の主要な操作を行うためには ResourceManager クラスを使用します。
マネージャクラス、および、ファクトリクラスは以下の通りです。

マネージャクラス
`jp.co.intra_mart.foundation.authz.services.admin.ResourceManager`

マネージャファクトリ
`jp.co.intra_mart.foundation.authz.services.admin.ResourceManagerFactory`

マネージャインスタンスはファクトリクラスを使用して取得します。ファクトリも自身のファクトリメソッドを持っていますので、以下のようにして取得します。

```
// マネージャインスタンスの取得
final ResourceManager resourceManager = ResourceManagerFactory.getInstance().getResourceManager();
```

リソースとリソースグループの登録

すでに説明した通り、APIでリソースを登録するとリソースグループがセットで追加されます。このため、概念上はリソースは必ずしもツリーの末端になるわけではありません。

リソースグループのツリーの中に、リソースURIを持つノードが存在することになります。ただし、認可要求はリソースURIを使用

してしか行えない点に注意してください。

以下にリソースとリソースグループの構成を登録する例を示します。

```
// トップとなるリソースグループを登録
final I18nValue<String> topGroupName = new I18nValue<String>(Locale.JAPANESE, "トップグループ");
final I18nValue<String> topGroupDesc = new I18nValue<String>(Locale.JAPANESE, "1段目のグループです。");
final ResourceGroup topGroup = resourceManager.registerResourceGroup("top-group-id", topGroupName,
topGroupDesc);

// 中間リソースグループを登録
final I18nValue<String> subGroupName = new I18nValue<String>(Locale.JAPANESE, "中間グループ");
final I18nValue<String> subGroupDesc = new I18nValue<String>(Locale.JAPANESE, "2段目のグループです。");
final ResourceGroup subGroup = resourceManager.registerSubGroup("sub-group-id", topGroup, subGroupName,
subGroupDesc);

// 中間リソースの配下にリソースを登録
// 登録の結果の戻り値はリソースグループです。
final I18nValue<String> resourceName = new I18nValue<String>(Locale.JAPANESE, "中間グループ");
final I18nValue<String> resourceDesc = new I18nValue<String>(Locale.JAPANESE, "2段目のグループです。");
final ResourceGroup resource = resourceManager.registerAsResource("service://sample/sample_path", resourceName,
resourceDesc, "sample-id", subGroup);
```

リソースグループの取得

リソースグループはIDによる取得のほか、リソースに当たるリソースグループの場合URIで取得することができます。

```
// IDによるリソースグループの取得
final ResourceGroup groupA = resourceManager.getResourceGroup(topGroup.getResourceGroupId());

// URIによるリソースにあたるグループの取得
final ResourceGroup groupB = resourceManager.getResourceGroupByUri("service://sample/sample_path");
```

リソースグループの名称、説明の更新

リソースグループに対して名称と説明を更新できます。

```
// 名称の更新
final I18nValue<String> name = new I18nValue<String>(Locale.JAPANESE, "トップグループ");
name.put(Locale.ENGLISH, "Top Group");
resourceManager.setResourceGroupNames("top-group-id", name);

// 説明の更新
final I18nValue<String> desc = new I18nValue<String>(Locale.JAPANESE, "1段目のグループです。");
desc.put(Locale.ENGLISH, "Top level of groups");
resourceManager.setResourceGroupDescriptions("top-group-id", desc);
```

リソースグループの削除

リソースに当たるリソースグループを削除すると、リソースも削除されます。

```
// リソースグループがリソースにあたるグループだった場合、リソースも削除されます。
resourceManager.removeResourceGroup(resource);

// 削除するリソースグループに配下がある場合、それらも削除されます。
resourceManager.removeResourceGroup("top-group-id");
```

リソースグループ汎用属性の管理

マネージャインスタンスの取得

リソースグループ汎用属性の操作を行うためには ResourceAttributeManager クラスを使用します。
マネージャクラス、および、ファクトリクラスは以下の通りです。

マネージャクラス
　jp.co.intra_mart.foundation.authz.services.admin.ResourceAttributeManager

マネージャファクトリ
　jp.co.intra_mart.foundation.authz.services.admin.ResourceAttributeManagerFactory

マネージャインスタンスはファクトリクラスを使用して取得します。ファクトリも自身のファクトリメソッドを持っていますので、以下のようにして取得します。

```
// マネージャインスタンスの取得
final ResourceAttributeManager resourceAttributeManager =
ResourceAttributeManagerFactory.getInstance().getResourceAttributeManager();
```

使い方の詳細についてはAPIリストを参照してください。

閉塞の管理

閉塞状態の管理は「*リソースグループ汎用属性*」を利用しますが、閉塞状態の管理するAPIとして「ResourceBlocker」を提供しています。 閉塞状態の管理は ResourceAttributeManager ではなく、ResourceBlocker を利用します。

> **コラム**
> リソースの閉塞については「*閉塞*」を参照してください。

> **コラム**
> ResourceBlockerは intra-mart Accel Platform 2013 Winterより利用可能です。

マネージャインスタンスの取得

マネージャクラス、および、ファクトリクラスは以下の通りです。

マネージャクラス
　jp.co.intra_mart.foundation.authz.services.admin.block.ResourceBlocker

マネージャファクトリ
　jp.co.intra_mart.foundation.authz.services.admin.block.ResourceBlockerFactory

マネージャインスタンスはファクトリクラスを使用して取得します。ファクトリも自身のファクトリメソッドを持っていますので、以下のようにして取得します。

```
// マネージャインスタンスの取得
final ResourceBlocker resourceBlocker = ResourceBlockerFactory.getInstance().getResourceBlocker();
```

閉塞設定

指定したリソースグループとその配下のすべてのリソースグループに対して閉塞設定を行います。

アクションを指定しての閉塞

閉塞を行うリソースグループのID、リソースタイプとアクションを指定して閉塞を行います。
指定したリソースグループ、および、配下のリソースグループの特定のアクションが閉塞されていた場合、指定したアクションがそのリソースグループが閉塞を行うアクションの一つとして追加されます。

```
// メニューグループの「参照」アクションを閉塞
resourceBlocker.block("im-menu-group-cat-im_global_nav_pc", "im-menu-group", "read");
```

リソースグループ自体の閉塞

閉塞を行うリソースグループのIDを指定して閉塞を行います。
指定したリソースグループ、および、配下のリソースグループの特定のアクションが閉塞されていた場合、そのリソースグループ自体を閉塞する状態として更新されます。

```
// リソースグループ自体の閉塞
resourceBlocker.block("im-menu-group-cat-im_global_nav_pc");
```

閉塞設定の解除

指定したリソースグループとその配下のすべてのリソースグループに対して閉塞状態の解除を行います。

アクションを指定しての閉塞解除

閉塞を行うリソースグループのID、リソースタイプとアクションを指定して閉塞解除を行います。
指定したリソースグループ、および、配下のリソースグループの特定のアクションが閉塞されていた場合、指定したアクションがそのリソースグループが閉塞を行うアクションから削除されます。
リソースグループ自体が閉塞されている場合、閉塞状態が維持されます。

```
// メニューグループの「参照」アクションの閉塞を解除
resourceBlocker.unblock("im-menu-group-cat-im_global_nav_pc", "im-menu-group", "read");
```

リソースグループ自体の閉塞解除

閉塞を行うリソースグループのIDを指定して閉塞を行います。
指定したリソースグループ、および、配下のリソースグループの特定のアクションが閉塞されていた場合も、その閉塞状態が解除されます。

```
// リソースグループの閉塞の解除
resourceBlocker.unblock("im-menu-group-cat-im_global_nav_pc");
```

閉塞状態の取得

アクションを指定しての閉塞確認

指定したリソースグループがアクションに対して閉塞されているかどうかを取得できます。

```
// メニューグループの「参照」アクションが閉塞されているかどうか
resourceBlocker.isBlock("im-menu-group-cat-im_global_nav_pc", "im-menu-group", "read");
```

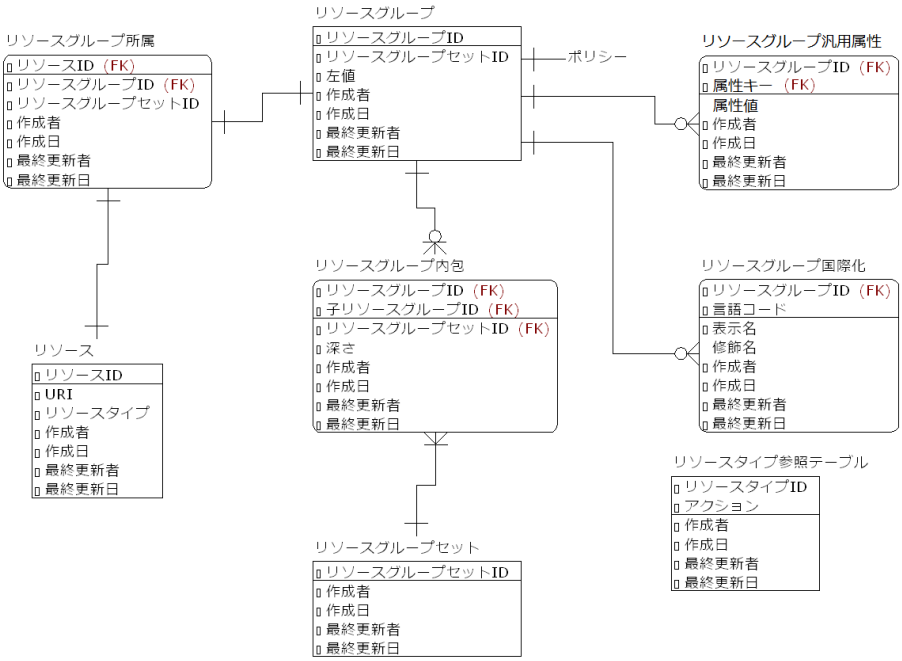
リソースグループ自体の閉塞確認

指定したリソースグループ自体が閉塞されているかどうかを取得できます。

```
// リソースグループ自体が閉塞されているかどうか
resourceBlocker.isBlock("im-menu-group-cat-im_global_nav_pc");
```

ERとテーブル上の表現

リソース管理に関係するテーブルのERを以下に示します。



ここではAPI上のモデルとER上の関連を中心に説明します。テーブルの定義や認可機構全体のERについては別冊のテーブル定義書、ER図を参照してください。

リソースグループ(imaz_resource_group)

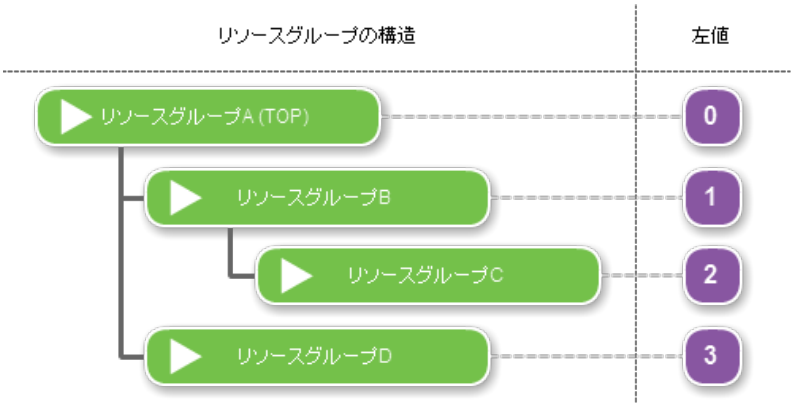
主要なフィールド	
論理名	**物理名**
リソースグループID	resource_group_id
リソースグループセットID	resource_group_set_id
左値	left_value

リソースグループを表します。

リソースグループIDはシステムで一意なIDを持ちます。

リソースグループは階層構造を持つことができ、特定のトップ配下のツリーをリソースグループセットという単位で管理しています。このセットを示すのがリソースセットIDです。リソースグループの階層構造のトップにあたるリソースグループはリソースグループIDとリソースグループセットIDが同じ値になります。

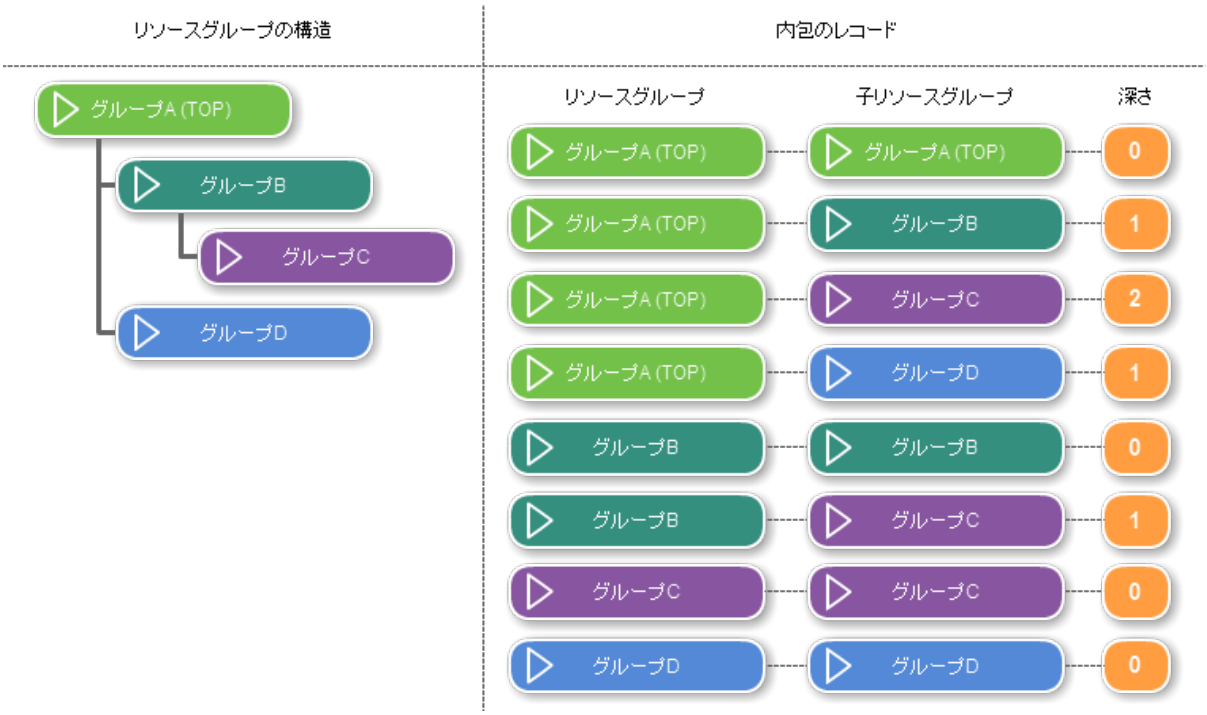
左値はリソースグループセット内で一意の順序を示す0から始まる整数値です。リソースグループは画面上での表示順を保持しており、その順序を保持する値が左値です。この値はAPIによって自動的にコントロールされます。あくまで表示上の順序なので、リソースグループツリーの根から葉全体に対して連番が振られる形になっています。簡単な例を以下に示します。



リソースグループ内包(imaz_resource_group_inc)

主要なフィールド	
論理名	**物理名**
リソースグループID	resource_group_id
子リソースグループID	child_resource_group_set_id
リソースグループセットID	resource_group_set_id
深さ	depth

セット内における親子関係のすべてを保持します。簡単な例を以下に示します。



リソースグループ国際化(imaz_resource_group_i)

主要なフィールド	
論理名	**物理名**
リソースグループID	resource_group_id
ロケールID	locale_id
表示名	diaplay_name
説明	description

リソースグループに対して表示名と説明を保持します。それぞれ各ローケルに対して保持することができます。

リソースグループセット(imaz_resource_group_set)

主要なフィールド	
論理名	**物理名**
リソースグループセットID	resource_group_set_id

リソースグループは階層構造を持つことができ、特定のトップ配下のツリー一式をリソースグループセットと定義しています。このセットを示すのためのIDがリソースグループセットIDです。この値は原則トップのリソースグループのリソースグループIDと同じ値です。

リソース(imaz_resource)

主要なフィールド	
論理名	**物理名**
リソースID	resource_id
URI	uri
リソースタイプ	resource_type

リソースを表すテーブルです。リソースの登録・削除に応じて追加・削除されます。リソースIDはURIのハッシュ値から生成される値のため、任意の値に設定することはできません。リソースタイプはこのリソースを取り扱うリソースタイプのIDです。

リソースグループ所属(imaz_resource_group_ath)

主要なフィールド	
論理名	**物理名**
リソースID	resource_id
リソースグループID	resource_group_id
リソースグループセットID	resource_group_set_id

リソースとリソースグループの関連を示します。リソースを作成すると対になるリソースグループが作成されるため、リソースが登録されるタイミングで登録されます。リソースに対応するリソースグループを削除するとリソースも連動して削除されるので、リソースグループ所属も合わせて削除されます。

リソースタイプ参照テーブル(imaz_resource_type_ref)

主要なフィールド	
論理名	**物理名**
リソースタイプ	resource_type
アクション	action

登録されているリソースタイプと、リソースタイプが定義しているアクションをデータベース上で取り扱うためのテーブルです。このテーブルの情報は起動時か、最初にResourceManagerを使用するタイミングでリソースタイプの設定情報をもとに自動的に作成されます。

このテーブルの情報は認可設定画面でアクションを適切に提示するために使用しています。

リソースグループ汎用属性テーブル(imaz_resource_group_attr)

主要なフィールド	
論理名	**物理名**
リソースグループID	resource_group_id
属性キー	attr_key
属性値	attr_value

リソースグループに紐付ける汎用的な属性を取り扱うためのテーブルです。認可にリソースグループを登録した後、任意のキーで属性値を付加することができます。

ポリシー管理

ポリシー

ポリシーとは *認可 概要* で紹介した通り、

- 「誰が」（サブジェクト）
- 「何を」（リソース）
- 「どうする」（アクション）

に対して

- 「許可／禁止」（エフェクト）

を定めたものです。

サブジェクト管理機能で管理されている「誰が」とリソース管理機能で管理されている「何を」「どうする」について「許可／禁止」を管理するのがポリシー管理機能です。リソースはまとめて権限を管理するためにリソースグループで設定をまとめることができ、サブジェクトは実際には条件式を使用してサブジェクトグループという単位で利用されます。このため、ポリシー管理は実際には

- 「誰が」：サブジェクトグループ
- 「何を」：リソースグループ
- 「どうする」：アクション

に対して「禁止」「許可」の設定を管理する形になっています。

ポリシー管理では上記の情報のセットを大量に保持しているだけなので、ポリシー情報の取り扱い自体はとても単純です。

API

ポリシー管理に関係する主要なAPI上の操作を説明します。詳細に関してはAPIリストを参照してください。

マネージャインスタンスの取得

ポリシー管理の主要な操作を行うためには PolicyManager クラスを使用します。
マネージャクラス、および、ファクトリクラスは以下の通りです。

マネージャクラス
`jp.co.intra_mart.foundation.authz.services.admin.PolicyManager`

マネージャファクトリ
`jp.co.intra_mart.foundation.authz.services.admin.PolicyManagerFactory`

マネージャインスタンスはファクトリクラスを使用して取得します。ファクトリも自身のファクトリメソッドを持っていますので、以下のようにして取得します。

```
final PolicyManager policyManager = PolicyManagerFactory.getInstance().getPolicyManager();
```

ポリシーの設定

特定のリソースグループ、サブジェクトグループ、アクションに対してエフェクトを設定します。`Permit` か `Deny` が設定できます。

```
// ポリシーの設定
final Policy policy1 = policyManager.setPolicy("resource-group-id", "subject-group-id", "service", "execute", "permit");
final Policy policy2 = policyManager.setPolicy(resourceGroup, action, subjectGroup, Effect.PERMIT);
```

ポリシーの取得

特定のリソースグループ、サブジェクトグループ、アクションに対して設定されているエフェクトを取得します。この時、単純に設定したポリシーを取得するか、リソースグループの継承関係を考慮するかによってメソッドが分かれています。

```
// 継承関係を考慮せずポリシーを取得する（テーブルの値からそのまま取得）
Policy declaredPolicy = policyManager.getDeclaredPolicy("resource-group-id", "subject-group-id", "service", "execute");

// リソースグループの継承関係を考慮してポリシーを取得する
Policy actualPolicy = policyManager.getActualPolicy("resource-group-id", "subject-group-id", "service", "execute");
```

ポリシーの削除

特定のリソースグループ、サブジェクトグループ、アクションに対して設定されているポリシーを削除すると「未設定」状態になります。通常上位のリソースグループに設定されている権限を参照するようになります。

```
// 単一の設定を削除
// 削除すると通常継承扱いになります。
policyManager.removePolicy(policy1);
policyManager.removePolicy(policy2.getPolicyId());

// リソースグループ、サブジェクトグループについて一括で削除
policyManager.removePoliciesForResourceGroup("resource-group-id");
policyManager.removePoliciesForSubjectGroup("subject-group-id");
```

テーブル上の表現

ここではAPI上のモデルとテーブル上の関連を中心に説明します。

ポリシー(imaz_policy)

ポリシーテーブルには概要で説明した項目をほぼそのまま格納します。

主要なフィールド	
論理名	**物理名**
ポリシーID	policy_id
リソースグループID	resource_group_id
サブジェクトグループID	subject_group_id
リソースタイプ	resource_type
アクション	action
エフェクト	effect

ポリシーIDは内部的に採番されるIDです。リソースグループID、リソースタイプ、アクション、サブジェクトグループIDで一意になるよう制約がかかっており、通常APIなどからポリシーを変更する際にはこれらの値を使用します。

認可情報のインポート・エクスポート

項目

認可情報のインポータ・エクスポータ

認可機構では以下のインポータ、エクスポータを用意しています。

- サブジェクト インポータ/エクスポータ（XML形式）
- リソースグループ インポータ/エクスポータ（XML形式）
- リソース インポータ/エクスポータ（XML形式）
- ポリシー インポータ/エクスポータ（XML形式）
- ポリシー置換インポータ（XML形式）
- 認可設定画面で呼び出せる部分エクスポータ（XML形式）
- 認可設定画面で呼び出せる部分インポータ/エクスポータ（Excel（xlsx）形式）

コラム

- XML形式

  XML形式のインポータは、ポリシーを除き、データの削除処理を行いません。デフォルトではテーブルに存在するデータとXMLに記載された情報をマージする動作をします。
  XML形式のインポータは、共通して update-mode 属性（更新モード）があり、これに replace を指定することでデータを置き換える動作ができるものがあります。

  ポリシーのインポータは、明示的に「未設定」をエフェクトに指定した場合に限り、ポリシーの削除処理を行います。
  そのため、「未設定」のエフェクトを指定していない定義ファイルでシステム全体のポリシーを同期したい場合のために、ポリシー置換インポータというインポータを特別に用意しています。
  このインポータはポリシーを一度削除したうえで、指定のXMLファイルを読み込みます。

- Excel（xlsx）形式

  Excel（xlsx）形式のインポータは、ポリシーのみを扱います。テーブルに存在するデータを、ファイル内の各セルに記載された情報に置き換える動作をします。
  サブジェクトやリソースグループ、リソースの追加・更新・削除は行いません。

  Excel（xlsx）形式のインポータ・エクスポータは、intra-mart Accel Platform 2014 Summer(Honoka)から提供しています。

インポータ/エクスポータの使用方法

一括で入出力するインポータ/エクスポータはジョブ、および、ジョブネットとして登録されています。
ジョブネットを実行してインポート/エクスポートを行うことができます。

使用可能なジョブ、ジョブネットの詳細情報、指定可能なオプションに関しては、「IM-Authz（認可）インポート・エクスポート仕様書 」を参照してください。
ジョブネットの実行操作に関しては、「テナント管理者操作ガイド - ジョブを設定する 」を参照してください。

認可機構用のジョブとジョブネット

実行パラメータの詳細は「ジョブ・ジョブネット リファレンス - テナントマスタ 」を参照してください。

ジョブ

- エクスポート

名称	説明
認可（ポリシー）エクスポート	XMLファイルへポリシー設定をエクスポートします。
認可（リソース）エクスポート	XMLファイルへリソースをエクスポートします。
認可（リソースグループ）エクスポート	XMLファイルへリソースグループをエクスポートします。
認可（サブジェクト、および、グループ）エクスポート	XMLファイルへサブジェクトグループとサブジェクトの情報を条件としてエクスポートします。

- エクスポート（Excel）

名称	説明
認可エクスポート（Excel）	Excel（xlsx）ファイルへ、ポリシー設定と関連するリソースグループ、リソース、サブジェクト情報をエクスポートします。

- インポート

名称	説明
認可（ポリシー）インポート	XMLファイルからポリシー設定をインポートします。
認可（ポリシー）一括削除	現在テーブルに入っているポリシーをすべて削除します。 ポリシー全体を置き換えたい場合などにジョブネットの一部として使用します。
認可（リソース）インポート	XMLファイルからリソースをインポートします。
認可（リソースグループ）インポート	XMLファイルからリソースグループをインポートします。
認可（サブジェクト、および、グループ）インポート	XMLファイルから条件の情報をサブジェクトグループとサブジェクトとしてインポートします。

- インポート（Excel）

名称	説明
認可インポート（Excel）	Excel（xlsx）ファイルからポリシー情報をインポートします。

ジョブネット

- エクスポート

名称	説明
認可（ポリシー）エクスポート	同名のジョブを実行するためのジョブネット定義です。
認可（リソース）エクスポート	同名のジョブを実行するためのジョブネット定義です。
認可（リソースグループ）エクスポート	同名のジョブを実行するためのジョブネット定義です。
認可（サブジェクト、および、グループ）エクスポート	同名のジョブを実行するためのジョブネット定義です。

- エクスポート（Excel）

名称	説明
認可エクスポート（Excel）	同名のジョブを実行するためのジョブネット定義です。

- インポート

名称	説明
認可（ポリシー）インポート	同名のジョブを実行するためのジョブネット定義です。 このジョブネットではポリシー設定はマージされます。
認可（ポリシー）置換インポート	「認可（ポリシー）一括削除」ジョブを実行の後、「認可（ポリシー）インポート」ジョブを実行します。 これによりインポートファイルの内容でポリシー設定が置き換えられます。
認可（リソース）インポート	同名のジョブを実行するためのジョブネット定義です。
認可（リソースグループ）インポート	同名のジョブを実行するためのジョブネット定義です。
認可（サブジェクト、および、グループ）インポート	同名のジョブを実行するためのジョブネット定義です。

- インポート（Excel）

名称	説明
認可インポート（Excel）	同名のジョブを実行するためのジョブネット定義です。

インポートの依存関係

インポートの依存関係は、インポートする情報とファイル形式によって異なります。
依存関係については、「IM-Authz（認可）インポート・エクスポート仕様書 - インポート」の、各形式の「インポートの依存関係」項を参照してください。

インポート・エクスポートで使用するファイルの書式

インポート・エクスポートで使用するファイルは、XML形式とExcel（xlsx）形式の２種類あります。
それぞれの書式については、「IM-Authz（認可）インポート・エクスポート仕様書 - ファイルフォーマット」を参照してください。

認可設定画面で呼び出せる部分エクスポータ

認可設定画面から部分的なポリシーと、関連するリソースグループ、リソース、および、サブジェクトグループのエクスポートができます。
エクスポートしたファイルは、認可設定画面上からダウンロードできます。
詳細は「*認可設定画面*」を参照してください。

認可設定画面

ここでは認可設定画面の仕様について説明します。

ポリシー編集機能

認可設定画面では

- リソース、リソースグループの追加・編集・削除
- サブジェクトグループ（およびサブジェクト）の追加・編集・削除
- ポリシーの変更

が行えます。

画面の縦軸がリソース、およびリソースグループで、横軸がサブジェクトグループ（画面表示上は条件）になっており、格子状に並んでいるのが各リソース／サブジェクトに対する許可・禁止の設定です。

編集ロック状態

認可設定画面では取り扱う情報の対象が多いため変更は原則として即座にコミットされます。誤操作でポリシーを変更してしまうことを防止するために、画面を表示した時点では変更操作がロックされています。

右上の「権限設定を開始する」ボタンを押下することでロックを解除できます。

権限の設定と表現

ポリシー設定には「許可」「禁止」「未設定」の3つの状態を設定できます。編集ロックを解除した状態でセルをクリックする事で「許可」「禁止」「未設定」を順番に切り替えます。画面のアイコン上は以下のように表現されます。

-  **濃い緑のチェックマーク：**明示的に「許可」を設定している状態

-  **濃い赤の×マーク：**明示的に「禁止」を設定している状態
-  **薄い緑のチェックマーク（矢印のオーバレイ付き）：**未設定。上位リソースグループの「許可」設定を継承している状態
-  **薄い赤の×マーク（矢印のオーバレイ付き）：**未設定。上位リソースグループの「禁止」設定を継承している状態

薄いアイコンはその箇所が未設定であることを示します。未設定の場合はポリシー解釈器によってどのように判断するかが決定されます。

デフォルトではポリシー解釈器は以下のように判断します。

- 未設定状態の場合、リソースグループの構造を参照してより上位のリソースグループの設定を参照します。
- 最上位のリソースグループまで何も設定されていない場合は「禁止」になります。

ポリシー解釈器の詳細は「*ポリシー解釈器* 」を参照してください。

権限設定が重複する場合の取り扱い

ユーザは複数のサブジェクトグループに当てはまることがあるので、認可設定上単一のリソースに対して判定しようとしても複数の設定が該当する可能性があります。あるロールには許可、ある組織には禁止されているリソースがあり、ユーザがそれら両方に所属しているような場合です。

この場合の振る舞いも権限の継承同様ポリシー解釈器によってどのように判断するかが決定されます。ポリシー解釈器の詳細は「*ポリシー解釈器* 」を参照してください。

> **注意**
> デフォルトの判断ではユーザがマッチするサブジェクトグループのうち一つでも「許可」が設定されていれば「許可」と判断するようになっています。

特定リソースについての権限の保護

認可設定画面を操作するために必要な権限は、設定画面の仕様として保護しています。

具体的には現在ログイン中のユーザのサブジェクトグループについて、認可設定画面用の権限をすべて外したり禁止したりしようとすると、エラーとして操作が中断されます。 保護されるリソースは設定ファイルに記載されています。詳細は「設定ファイルリファレンス」の「認可設定画面」「保護リソース設定」の項を参照してください。

サブジェクトグループの追加・編集・削除

サブジェクトグループ（対象者の条件）を追加・編集・削除することができます。

- サブジェクト（画面上では対象者）を、複数設定することができます。
  - 組織や役職の場合「上位」や「下位」といった付加条件を付けることができます。
- 各サブジェクトに対して「除外」条件を付与できます。
- 「すべてに一致する」か「いずれかに一致する」かを選択できます。

ただし、以下の制約があります。

- 左上のセレクトボックスが「すべてに一致する」の場合、全サブジェクトに対して「除外」を設定して条件を作成することはできません。
- 「いずれかに一致する」の場合、「除外」を使うことはできません。

認可設定画面でサブジェクトグループを選択して「条件の削除」ボタンを押下すると、条件を削除することができます。ただし以下の制約に注意する必要があります。

- 「認証」カテゴリのサブジェクトグループは削除することができません。
- サブジェクトグループはシステム全体で共有されています。サブジェクトグループを削除すると他のリソースの種類（*リソースの種類の切り替え* 参照）などにも影響を及ぼすため、注意してください。

リソース参照編集機能

リソースやリソースグループの詳細な情報を参照できます。リソース／リソースグループの欄にマウスカーソルをあてると、右側に

小さなアイコンが表示されます。子のアイコンをクリックするとリソース／リソースグループの詳細表示画面が表示されます。

編集ロックが解除された状態の場合、この画面にメニューが追加され、リソースやリソースグループの追加・編集・削除ができるようになります。

リソースグループを削除すると、配下のグループやリソースも再帰的に削除して、さらに削除されたグループに紐づくポリシー設定も削除されます。

> **注意**
>
> リソースは原則としてプログラムの必要に応じて追加されるため、削除するとプログラムの想定外の動作を引き起こす可能性があります。
> テナント環境セットアップで登録されたり、システムが自動的に追加したリソースは原則削除しないようにしてください。
> 削除する際は本当に必要のないリソースであることを確認してから実施してください。
>
> リソースを削除した場合は、削除したリソースと関係する認可設定情報のバックアップが自動的に保存されます。
> 詳細については「*リソース削除時のバックアップ*」を参照してください。

編集を補助する機能

リソースの説明
縦軸（リソース、リソースグループ）の一覧の部分にマウスカーソルを当てるとリソースやリソースグループに設定されている説明がポップアップで表示されます。

絞り込み検索
縦軸（リソース、リソースグループ）や横軸（サブジェクトグループ）は名称の部分一致検索で絞り込んで表示することができます。画面左上の虫眼鏡アイコンをクリックすると、縦軸、横軸それぞれの検索ボックスが表示されます。キーワードを入力して検索ボタンを押下することで、それぞれの軸が絞り込み表示されます。この条件は検索ボックスを閉じてもクリアされません。絞り込みを解除する場合は検索ボックスにあるクリアボタンを押下してください。

絞り込み検索後の解除
縦軸は絞り込み検索後、いずれかのリソース／リソースグループをダブルクリックするとその位置を中心に絞り込みを解除することができます。

アクションの絞り込み
画面に表示しているアクションが複数ある場合、いずれかのアクションのみに表示を絞り込むことができます。画面上側の「アクションの種類」のセレクトボックスから、表示したいアクションを選択すると、縦軸のアクションが選択したもののみに絞られます。

現在の階層表示
縦軸をスクロールしていくとリソースグループ階層のどの位置を見ているかがわかりにくくなるため、現在表示している位置の階層が縦軸の上に表示されるようになっています。

リサイズ
リソースやリソースグループの名称が長すぎで表示しきれない場合、枠線をドラッグすることで表示領域を左右に広げることができます。サブジェクトグループの欄についても同様に上下に広げることができます。

サブジェクトグループ・カテゴリの並び替え
サブジェクトグループや、サブジェクトグループのカテゴリを並べ替えることができます。縦に並んでいる順番がソート順になります。ドラッグアンドドロップで並び順を変更することができます。

赤色のサブジェクトグループ
画面上赤く表示されているサブジェクトグループは、現在操作中のユーザが該当するサブジェクトグループです。

インポート・エクスポート

XML形式

> **コラム**
>
> XML形式のエクスポート機能は、intra-mart Accel Platform 2012 Winter(Bourbon)から提供しています。

認可設定画面では、認可設定の情報を特定の範囲に限って、XML形式でエクスポートすることができます。
認可設定画面上に表示されている任意のポリシーと、関連するリソースグループ・リソース・サブジェクトグループを同一の「リソースの種類」内で出力することができます。

認可設定画面での操作手順に関しては、「テナント管理者操作ガイド - 認可を設定する」を参照してください。

なお、XML形式のインポートは認可設定画面上から実行することはできません。ジョブスケジューラを利用してください。

> **コラム**
>
> XML形式のインポート・エクスポートは、編集ロック状態でも実行できます。
> 編集ロックを解除した状態でエクスポート対象を選択しようとすると意図せずポリシーを変更してしまう可能性があるため、編集ロックを行った状態でエクスポートに関する操作を行うことをお勧めします。

Excel（xlsx）形式

> **コラム**
>
> 「IM-Authz（認可）Excelインポート・エクスポート」モジュールは、intra-mart Accel Platform 2014 Summer(Honoka)から提供しています。

認可設定画面上で選択可能な「リソースの種類」のうち、すべて、または、現在画面上で選択しているもののいずれかを選択して、Excel（xlsx）形式で対象の認可設定状態を出力することができます。
すべての「リソースの種類」を出力する場合、「リソースの種類」ごとにシートが作成されます。

また、認可設定画面では、Excel（xlsx）形式に限ってインポートすることができます。
認可設定画面、または、ジョブスケジューラによってエクスポートしたExcel（xlsx）形式の編集済みファイルを認可設定画面上からアップロードしてインポートすることができます。

認可設定画面での操作手順に関しては、「テナント管理者操作ガイド - 認可を設定する 」を参照してください。

> **注意**
>
> - Excel（xlsx）形式のエクスポートを使用するためには、別途「追加機能」 - 「IM-Authz（認可）Excelインポート・エクスポート」モジュールを導入する必要があります。
> - 認可設定画面を「*部品化* 」によって開いた場合、認可設定画面上からExcel（xlsx）形式のインポート・エクスポートを実行できません。
> - 認可設定画面上からExcel（xlsx）形式のインポート・エクスポートを実行した場合、処理が終了するまで認可設定の編集ロックを解除することはできません（画面上から権限設定を変更できません）。
> - 認可設定画面上からExcel（xlsx）形式のインポート・エクスポートを実行した場合、処理は非同期で実行されます。
>   これは、認可設定のデータ量によってはExcel（xlsx）形式の処理に時間がかかることがあるためです。
>   この非同期処理はテナントごとにアプリケーションロックがかかります。そのため、同じテナント内で、認可設定画面から同時に複数のインポート・エクスポート処理を実行することはできません。
>   インポート・エクスポートの処理が終わらず、やむを得ず処理を中断させる場合は、以下の順序に従って処理を強制終了させてください。
>
>   1. システム管理者に依頼して、システム管理画面からログインします。
>   2. 「システム管理」 - 「非同期 - タスクキュー一覧」をクリックします。
>   3. 「並列タスクキュー」に表示されているキューの「詳細」ボタンをクリックします。
>   4. 「処理中タスク」の「開始日時」に、インポート・エクスポートの実行を開始した時刻が表示されているタスクの「終了」ボタンをクリックします。
>   5. 「キューの先頭に再登録」を「しない」、「キューの状態」を「停止する」を選択して「決定」ボタンをクリックします。

> **コラム**
>
> Excel（xlsx）形式のインポート・エクスポートは、編集ロック状態でも実行できます。
> 編集ロックを解除した状態でエクスポート対象を選択しようとすると意図せずポリシーを変更してしまう可能性があるため、編集ロックを行った状態でエクスポートに関する操作を行うことをお勧めします。

リソース削除時のバックアップ

> **コラム**
>
> リソースのバックアップ機能は、intra-mart Accel Platform 2013 Summer(Damask)から提供しています。

認可設定画面上でリソースグループ、または、リソースを削除した場合、「*認可設定画面の設定* 」の設定内容に応じて、削除したリソースグループ、リソースに関係する認可設定情報が自動的にバックアップされます。

リソースを削除したことで想定外の動作を引き起こしてしまった場合、バックアップされたファイルをジョブネット経由で復元することで、リソースを削除前の状態に戻すことができます。

バックアップされたファイルは認可設定画面の設定ファイルに設定された場所に保存されています。ジョブ、または、ジョブネットの実行パラメータに、「file」をキーにして、値にファイルパスを設定してから実行してください。

用意されているジョブとジョブネットの一覧については「*認可情報のインポート・エクスポート* 」を参照してください。

> **注意**
>
> 削除したリソースによっては、ジョブネット管理画面を開くことができなくなる場合があります。
> この場合、以下の順序に従って復旧を行ってください。
>
> 1. ジョブネット管理画面を開くことができる権限を持つユーザでログインします。
> 2. 以下の URL にアクセスして、グローバルナビを無効にした状態で、ジョブネット管理画面を直接開きます。
>   `http://<HOST>:<PORT>/<CONTEXT_PATH>/tenant/job_scheduler/jobnet_maintenance?imui-theme-builder-module=headwithcontainer`
>
> 上記手順でジョブネット管理画面を開くことができない場合は、システム管理画面から認可設定画面とジョブネット管理画面を開くための権限付与を行ってください。
> 権限付与の方法は、「システム管理者操作ガイド 」の「テナント管理」章を参照してください。
> 権限付与後、再度上記手順に従って操作しなおしてください。

> **コラム**
>
> 認可設定情報をバックアップから復旧する場合は、以下の順序でジョブネットを実行してインポートを行ってください。
>
> - 「テナントマスタ」ー「インポート」ー「認可（リソースグループ）インポート」
> - 「テナントマスタ」ー「インポート」ー「認可（リソース）インポート」
> - 「テナントマスタ」ー「インポート」ー「認可（サブジェクト、および、グループ）インポート」
> - 「テナントマスタ」ー「インポート」ー「認可（ポリシー）インポート」 ※
>
> ※「認可（ポリシー）置換インポート」は使用しないでください。

認可設定画面の設定

認可設定画面でリソースグループ、または、リソース削除時に、自動的に保存するバックアップ処理の変更を行えます。

設定ファイル配置場所
`%CONTEXT_PATH%/WEB-INF/conf/authz-editor-config.xml`

XMLスキーマファイル
`%CONTEXT_PATH%/WEB-INF/schema/authz-editor-config.xsd`

以下のサンプルをもとに設定内容について説明します。

```
<?xml version="1.0" encoding="UTF-8"?>
<authz-editor-config
   xmlns="http://www.intra-mart.jp/authz/authz-editor-config"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.intra-mart.jp/authz/authz-editor-config ../schema/authz-editor-config.xsd">

 <resource-group-backup>
   <path>im_authz/backup</path>
   <max-size>30</max-size>
 </resource-group-backup>

</authz-editor-config>
```

- **resource-group-backup要素：**リソースグループ、または、リソース削除時のバックアップを設定する要素です。

- **path要素：**バックアップを保存するパブリックストレージのパスを設定する要素です。
- **max-size要素：**バックアップの最大保存件数を設定する要素です。

バックアップ先の保存形式

認可設定情報のバックアップは、「*認可設定画面の設定*」で設定された場所に、バックアップ実行時の現在日時（システムタイムゾーン）と、リソース削除時の実行者ユーザコードが付加されたディレクトリが作成されます。

ディレクトリ名の形式は `《現在日時（yyyyMMdd_HHmmss）》_《ユーザコード》` です。ただし、すでに同じディレクトリが存在する場合は枝番が付けられます。

そのディレクトリ内に以下のファイルが保存されます。

ファイル名	説明
authz-resource-group.xml	リソースグループ情報
authz-resource.xml	リソース情報
authz-subject-group.xml	サブジェクト、サブジェクトグループ情報
authz-policy.xml	ポリシー情報

また、バックアップ保存時にディレクトリの数が認可設定画面の設定ファイルで設定された最大保存数を超えた場合、最も古い日時形式の名前で保存されたディレクトリから順次削除されます。

> **注意**
>
> 認可設定画面の設定ファイルで設定されたバックアップの保存先には、手動でディレクトリを追加しないでください。予期せず削除される場合があります。

キャッシュのクリア

認可機構の持つキャッシュをクリアします。

認可機構ではユーザがリソースにアクセスする毎に権限があるかどうかを認可判断機能で判断しなければなりません。毎回ポリシー設定を取得して判断処理を行うとユーザのアクセス頻度などによってパフォーマンスの低下が著しくなります。これを緩和するために認可判断の結果をキャッシュできるようになっています。（*キャッシュ設定* により変更可能です）

認可設定画面でポリシーを変更した場合は該当部分のキャッシュは同期的にクリアするようになっており、管理者が設定した情報とキャッシュの間でずれが発生することを防止しています。

アプリケーションサーバが分散構成をとっている場合、キャッシュは各アプリケーションサーバ毎に保持しています。この場合でもポリシー変更時には分散構成のすべてのアプリケーションサーバのキャッシュが破棄されます。

このため管理者が認可設定画面からポリシーを変更しても通常キャッシュのクリアを行う必要はありません。

ただしネットワークのエラーなどの原因で、いくつかのアプリケーションサーバがキャッシュのクリアに失敗した場合、それらのアプリケーションサーバでは古い認可設定のキャッシュが残り続けることになります。このようにキャッシュをクリアできなかった場合、認可設定画面上ではエラーとして通知されます。古い認可設定のキャッシュを削除するためにネットワークの状態を修復した上で、この機能を使用して手動でキャッシュクリアを行ってください。

認可設定画面の設定

認可設定画面でリソースグループ、リソース、および、ポリシー更新時にキャッシュ不整合によるエラーが発生した場合の挙動の変更を行えます。

設定ファイル配置場所
`%CONTEXT_PATH%/WEB-INF/conf/authz-editor-config.xml`

XMLスキーマファイル
`%CONTEXT_PATH%/WEB-INF/schema/authz-editor-config.xsd`

以下のサンプルをもとに設定内容について説明します。

```
<?xml version="1.0" encoding="UTF-8"?>
<authz-editor-config
   xmlns="http://www.intra-mart.jp/authz/authz-editor-config"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.intra-mart.jp/authz/authz-editor-config ../schema/authz-editor-config.xsd">

  <policy-update>
    <cache-update-error rollback="true" />
  </policy-update>
  <resource-group-update>
    <cache-update-error rollback="true" />
  </resource-group-update>

</authz-editor-config>
```

- **policy-editor-config要素：**設定のルートになる要素です。
  - **policy-update要素：**ポリシー更新時の挙動を設定する要素です。
    - **cache-update-error要素：**キャッシュ更新に問題が発生した場合の挙動を設する要素です。
      - **rollback属性：**キャッシュ更新で問題が発生した場合、ポリシーの更新をロールバックするかどうかを設定します。
  - **resource-group-update要素：**リソースグループ、または、リソース更新時の挙動を設定する要素です。
    - **cache-update-error要素：**キャッシュ更新に問題が発生した場合の挙動を設する要素です。
      - **rollback属性：**キャッシュ更新で問題が発生した場合、リソースグループとリソースの更新をロールバックするかどうかを設定します。

リソースの種類の切り替え

リソースは種類によって設定できるサブジェクトが異なるため、リソースによって切り替えて管理する必要があります。左上の「リソースの種類」のセレクトボックスを選択して表示するリソースの種類を切り替えます。

リソースの種類については後述する *部品化* で定義されたものが一覧で表示されます。

部品化

認可設定画面は設定画面を部分的に切り出して部品化して使用することができます。

この機能は、特定のアプリケーション用に準備したリソース群についてのみ権限設定できる認可設定画面を、そのアプリケーションの管理機能の一環として使用するという形を想定しています。

そのために以下が必要になります。

1. 部品化された画面が管理するリソースグループセットの作成
   部品化機能を利用しようとするアプリケーションが、認可設定のために使用するリソース／リソースグループの構造を定義しなければなりません。intra-mart Accel Platform でもメニューやポータルが利用するために特定のリソースグループセットを定義しています。intra-mart Accel Platform で予約しているリソースセットについては *予約されているリソースグループセット一覧* を参照してください。

2. 部品化された画面についての設定（XMLファイル）の設置
   部品化のためのID、上記で定義したリソースグループセット、使用するサブジェクトタイプについて定義した設定ファイルを記述します。

   設定ファイルの配置場所
   `%CONTEXT_PATH%/WEB-INF/conf/authz-partial-policy-edit-config/`

   XMLスキーマファイル
   `%CONTEXT_PATH%/WEB-INF/schema/authz-partial-policy-edit-config.xsd`

   以下サンプルをもとに設定内容について説明します。

```
<?xml version="1.0" encoding="UTF-8"?>
<authz-partial-policy-edit-config
   xmlns="http://www.intra-mart.jp/authz/authz-partial-policy-edit-config"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.intra-mart.jp/authz/authz-partial-policy-edit-config ../../schema/authz-partial-policy-edit-config.xsd">

 <part-config>
  <part-id>im_authz_impl_router</part-id>
  <caption-cd>CAP.Z.IWP.ROUTER.AUTHZ.PARTCONFIG.TITLE</caption-cd>
  <resource-groups>
   <resource-group-id>http-services</resource-group-id>
  </resource-groups>
  <subject-types>
   <subject-type-id>im_authz_meta_subject</subject-type-id>
   <subject-type-id>imm_user</subject-type-id>
   <subject-type-id>imm_company_post</subject-type-id>
   <subject-type-id>imm_department</subject-type-id>
   <subject-type-id>imm_public_grp</subject-type-id>
   <subject-type-id>imm_public_grp_role</subject-type-id>
   <subject-type-id>b_m_role</subject-type-id>
  </subject-types>
  <callbacks>
    <resource-group-authorizer>jp.co.intra_mart.system.authz.ResourceGroupAuthorizer</resource-group-authorizer>
  </callbacks>
 </part-config>

</authz-partial-policy-edit-config>
```

- **part-config要素：**設定のルートになる要素です。
  - **part-id要素：**この設定のIDを付与します。他の設定と重複しないよう注意してください。
  - **caption-cd要素：**この設定の名称を引くためのメッセージコードを指定します。認可設定画面のリソースの種類の欄に表示されるため、国際化を考慮してメッセージリソースを用意してください。
  - **resource-groups要素：**部品化した認可設定画面の編集対象にするリソースグループのIDを設定します。
    - **resource-group-id要素：**対象にするリソースグループのIDを設定します。resource-groups要素内に複数指定可能できますが、単一のリソースグループセット内である必要があります。
  - **subject-types要素：**部品化した認可設定画面で使用できるサブジェクトタイプIDを設定します。ここに挙げられている以外のサブジェクトタイプは選択できなくなります。
    - **subject-type-id要素：**使用するサブジェクトタイプIDを設定します。subject-types要素内に複数指定できます。
  - **callbacks要素：**認可設定画面を部品として呼び出す場合に認可から呼び出されるリソース群の表示可否判断を行うコールバックを指定できます。任意項目です。
    - **resource-group-authorizer要素：**認可設定画面を部品として呼び出す場合に指定されたリソースグループに対する権限設定可否判断クラスを指定できます。権限設定可否判断クラスとは、jp.co.intra_mart.foundation.authz.partial.AuthzPartialResourceGroupAuthorizer インタフェースを実装したクラスです。任意項目です。

3. <imart> タグまたはカスタムタグによる呼び出し
   以下のように指定すると、画面にボタンが表示され、ポップアップ画面として認可設定画面を呼び出せるようになります。

```
<imart type="imAuthzPolicyEditor" displayType="button" partId="sample_part_id">ボタンで表示</imart>
```

partId には 2. で作成した設定ファイルで指定した part-id を指定します。resource-group-authorizer のクラスの詳細については「認可 拡張プログラミングガイド」を参照してください。そのほかのパラメータや、使い方の詳細についてはAPIリストを参照してください。

編集できない領域

認可設定画面では *部品化* で説明した設定を収集し、「リソースの種類」の一覧に表示しています。逆に、 *部品化* で説明した設定で指定されている範囲のリソース以外のものは、認可設定画面には表示されません。この領域はユーザからは権限設定できない領域になるため注意してください。

対象者検索の拡張領域

サブジェクトタイプを独自に追加した場合などに認可設定画面でも追加したサブジェクトタイプを対象者として検索できるようにする必要があるため、検索画面を開くためのボタンを追加できるようになっています。

> **コラム**
> サブジェクトタイプの追加の詳細については 「認可拡張プログラミングガイド」の「サブジェクト拡張ガイド」の章を参照してください。

ここでは検索画面を開くためのボタンを追加するための仕様について説明します。

追加のボタンはプラグインの機構を使用します。プラグインに関してはAPIリストの PluginManager を参照してください。以下のような設定ファイルを作成します。

配置先

`%CONTEXT_PATH%/WEB-INF/plugin/<Plugin ID>/plugin.xml`

拡張ポイント

`jp.co.intra_mart.authz.subject.search`

以下IM共通マスタのユーザの場合をサンプルに設定内容を説明します。

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
 <extension
    point="jp.co.intra_mart.authz.subject.search" >
  <subjectSearchConfig
    name="IM-Master Authz Subject Search Config"
    id="jp.co.intra_mart.im_master.authz.subject.search"
    version="8.0.0"
    rank="10"
    enable="true">

   <!-- サブジェクト共通のヘッダ埋込ページ -->
   <headerPage>im_master_subjecttypes/authz/subject/search</headerPage>

   <buttons>
    <!-- ボタン情報。複数のボタンを記述可能 -->
    <!-- [属性] subjectType サブジェクトタイプ -->
    <!-- [属性] displayName 表示名（多言語化） -->
    <!-- [属性] action ボタン押下時の関数名 function(pluginInfo：この設定情報)。"a.b.c" のようにオブジェクトプロパティの指定が可能。 -->
    <button
      subjectType="imm_department"
      displayName="%button.department"
      action="immSubjectSearch"
      >
     <!-- 付加条件セットからこのサブジェクトで使う設定を選ぶ。 -->
     <!-- [属性] id この検索ボタンで利用する付加条件セットのID -->
     <!-- [属性] value 付加条件セットのうち初期表示する付加条件の種別 -->
     <condition id="im_master.condition" value="eq"/>

     <!-- ボタン表示イメージ置き換え用ページ。 -->
     <!-- これを指定するとボタンではなく指定のjsspをインクルードします -->
     <!-- <viewPage>sample/authz/subject_plugins/sample/view</viewPage> -->

     <!-- オプション key/value で設定。内容は任意です -->
     <options>
      <target>jp.co.intra_mart.master.search.department</target>
      <callback>immDeptSearchCallback</callback>
     </options>
    </button>
   </buttons>

   <!-- 付加条件セットマスタ -->
   <conditions>
    <!-- 付加条件セット 複数記述可能 -->
    <!-- [属性] id 付加条件セットのID -->
    <condition id="im_master.condition">
     <!-- Comparator 定義。複数記述可能。Selectボックスに表示される。 -->
     <!-- typeが 選択肢のvalue値、displayが表示名 -->
     <!-- [属性] type 付加条件の値 -->
     <!-- [属性] displayName 表示名 -->
     <comparator type="gt" displayName="より大きい" />
     <comparator type="ge" displayName="以上" />
     <comparator type="eq" displayName="等価" />
     <comparator type="le" displayName="以下" />
     <comparator type="lt" displayName="より小さい" />
    </condition>
   </conditions>
  </subjectSearchConfig>
 </extension>
</plugin>
```

<subjectSearchConfig> 要素までは PluginManager で既定している書式になりますので、 ここでは <subjectSearchConfig> より下の要素について説明します。

- **headerPage 要素:** ヘッダとして読み込むページを指定します。追加したボタンでのイベントハンドラ等を記述する場所です。
- **buttons 要素:** 下記のボタン要素のコンテナです。

- **button 要素:** 追加するボタンを表す要素です。複数記述可能です。
  - **subjectType 属性:** このボタンで扱うサブジェクトタイプIDを指定します。
  - **displayName 属性:** このボタンの表示名を指定します。先頭が%のコードを指定するとメッセージリソースからそのコードに設定された文字列を取得します。
  - **action 属性:** このボタンをクリックされた場合に実行するイベントハンドラを指定します。このイベントハンドラはheaderPageで指定したページに記述しておきます。
  - **condition 要素:** 付加条件として使用する条件セットを指定します。下記のconditions要素で指定したものから選択します。
    - **id 属性:** 下記のconditions要素で設定しているものから使用する条件セットを指定します。
    - **value 属性:** デフォルト選択値を指定します。
    - **viewPage 要素:** ボタンの代わりにjsspを読み込んで表示させたい場合、この要素に読み込むjsspのパスを指定します。
    - **options 要素:** 任意の値を設定できます。タグ名をキー、タグ内のテキストを値としたハッシュ配列がボタンに対して設定されます。
- **conditions 要素:** 付加条件として使用する条件セットのマスター定義です。
  - **condition 要素:** 付加条件のひとまとまり（条件セット）を表します。
    - **id 属性:** この条件セットのIDです。
    - **comparator 要素:** 付加条件の選択肢を表します。
      - **type 属性:** 選択肢の値になります。
      - **displayName 属性:** 選択肢の表示名になります

認可要求インタフェース

項目

ここでは認可を要求するための機能について、個別に解説します。

認可機構に対して、認可判断の結果を問い合わせる（認可要求を行う、といいます）場合に使用するためのインタフェースが認可要求インタフェースです。これにはAPIによる方法とimartタグ/カスタムタグによる方法があります。

アプリケーションなどが認可機構との連携を行いたい場合このインタフェースを使用して認可判断を要求します。認可要求インタフェースは認可判断機能を呼び出して、その結果を呼び出し側に取り次ぐ役割を持っています。

APIによる認可要求

APIにより認可判断要求を行う場合、AuthorizationClient を使用します。 AuthorizationClient のインスタンスを取得するには AuthorizationClientFactory を使用します。 完全修飾クラス名は以下の通りです。

AuthorizationClient
`jp.co.intra_mart.foundation.authz.client.AuthorizationClient`

AuthorizationClientFactory
`jp.co.intra_mart.foundation.authz.client.AuthorizationClientFactory`

このインタフェースの詳細はAPIリストを参照してください。以下に認可要求を行うサンプルを示します。

```
// 1. ファクトリからインスタンスを取得
final AuthorizationClient client = AuthorizationClientFactory.getInstance().getAuthorizationClient();

// 2-a. URIを使用した認可要求
//      ここで指定しているURIはルーティングテーブルで認可設定画面のURLに設定しているURI
final AuthorizeResult result1 = client.authorize("service://authz/settings/basic", "execute");

if (result1.equals(AuthorizeResult.Permit)) {
   /*-- 許可された場合の操作 --*/
}

// 2-b. リソースタイプのサポートするリソースモデルを使用した認可要求
//      セットしているパスは2-aで指定しているリソースを指しているURL
final GeneralServiceModel service = new GeneralServiceModel();
service.setPath("/tenant/authz/settings");

final AuthorizeResult result2 = client.authorize(service, "execute");
if (result2.equals(AuthorizeResult.Permit)) {
   /*-- 許可された場合の操作 --*/
}
```

imartタグ/カスタムタグによる認可要求

imartタグ、JSPのカスタムタグとして認可機構との連携情報を埋め込んで、認可判断の結果からコンテンツの表示・非表示をコントロールすることができます。

imAuthzタグ

リソースのURIとアクションを直接指定してコンテンツの表示／非表示をコントロールするタグです。以下に認可と連携したページ表示をコントロールする例を示します。指定可能な属性の詳細はAPIリストを参照してください。

Java EE 開発モデル（ JSP ）の例

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="http://www.intra-mart.co.jp/taglib/im-tenant" prefix="imt" %>


<h1>認可要求のサンプル</h1>

<imt:imAuthz uri="service://authz/settings/basic" action="execute" >
  <div>認可されていればここが表示されます。</div>
</imt:imAuthz>

<imt:imAuthz uri="service://authz/settings/basic" action="execute" negative="<%= Boolean.TRUE %>">
  <div>認可されていなければここが表示されます。</div>
</imt:imAuthz>
```

> **注意**
> 属性 negative は intra-mart Accel Platform 2013 Autumn 以降のバージョンで利用可能です。

スクリプト開発モデル（プレゼンテーションページ）の例

```
<h1>認可要求のサンプル</h1>

<imart type="imAuthz" uri="service://authz/settings/basic" action="execute" >
  <div>認可されていればここが表示されます。</div>
</imart>

<imart type="imAuthz" uri="service://authz/settings/basic" action="execute" negative>
  <div>認可されていなければここが表示されます。</div>
</imart>
```

> **注意**
> 属性 negative は intra-mart Accel Platform 2013 Autumn 以降のバージョンで利用可能です。

imUrlAuthzタグ

ルーティングテーブルに設定されているパスを指定してコンテンツの表示／非表示をコントロールするタグです。このタグは内部でルータにパスに相当するリソースURIとアクションを問い合わせ、その結果から認可要求を行い、表示／非表示を判断しています。指定可能な属性の詳細はAPIリストを参照してください。

Java EE 開発モデル（ JSP ）の例

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="http://www.intra-mart.co.jp/taglib/im-tenant" prefix="imt" %>


<h1>認可要求のサンプル</h1>

<imt:imUrlAuthz url="/tenant/authz/settings" >
  <div>認可されていればここが表示されます。</div>
</imt:imUrlAuthz>

<imt:imUrlAuthz url="/tenant/authz/settings" negative="<%= Boolean.TRUE %>">
  <div>認可されていなければここが表示されます。</div>
</imt:imUrlAuthz>
```

> **注意**
> 属性 negative は intra-mart Accel Platform 2013 Autumn 以降のバージョンで利用可能です。

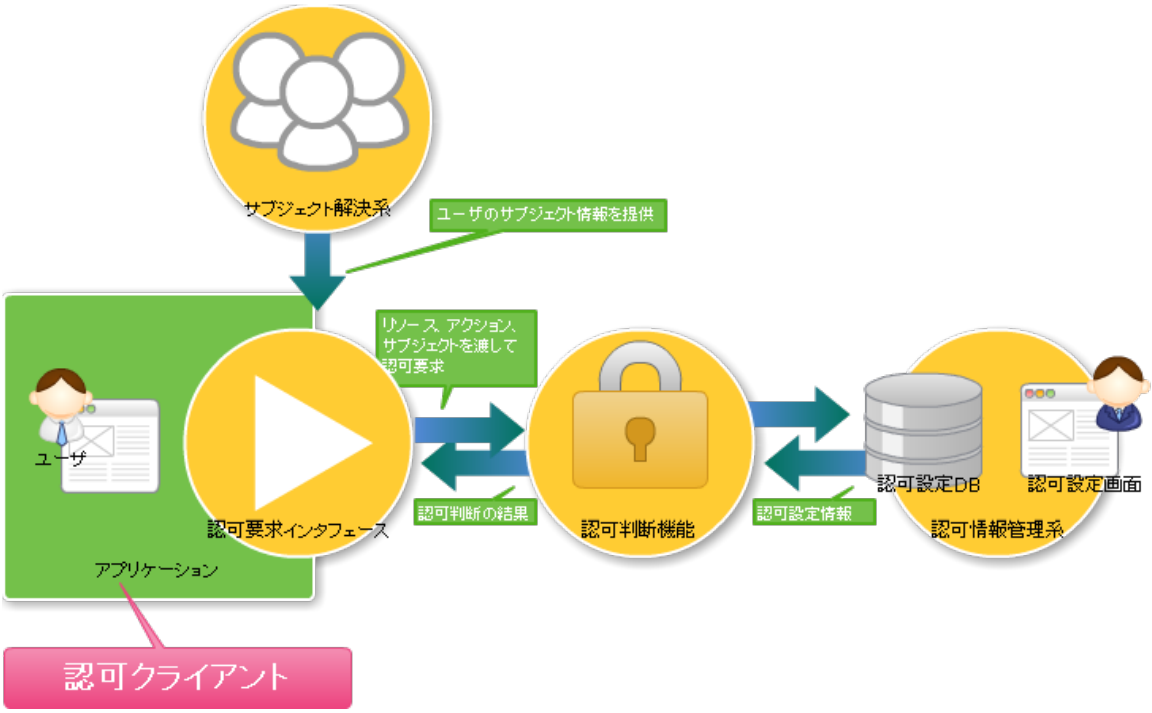
スクリプト開発モデル（プレゼンテーションページ）の例

```
<h1>認可要求のサンプル</h1>

<imart type="imUrlAuthz" url="/tenant/authz/settings" >
  <div>認可されていればここが表示されます。</div>
</imart>

<imart type="imUrlAuthz" url="/tenant/authz/settings" negative>
  <div>認可されていなければここが表示されます。</div>
</imart>
```

> **注意**
> 属性 negative は intra-mart Accel Platform 2013 Autumn 以降のバージョンで利用可能です。

認可クライアント

認可と連携して情報のアクセス制御や表示制御を行っているアプリケーションを認可クライアントと呼称しています。



デフォルトで存在する認可クライアント

intra-mart Accel Platform の中にはすでに認可クライアントとなっているコンポーネントが存在します。以下にそのコンポーネントと、認可とのかかわりの概要について説明します。実際にどのような処理が行われているかの詳細については各コンポーネントの仕様書を参照してください。

- ルータ（テナント管理機能モジュール）
  ルータはルーティングテーブルに定義されたURLのパスと実行するプログラムの実体を結びつける処理を行っています。ルータはURLにアクセスされるタイミングでそのURLへのアクセスが許可されるかを認可要求インタフェースを使用して問い合わせ、その結果許可されていなければそのページの処理を行わず、エラーステータスを返却します。

  ルーティングテーブルにはリソースURI・アクションを定義できるようになっており、URLを一つのリソースとして認可機構に対して認可要求を行っています。

  ルータは認可問い合わせに必要なリソースを自動的には管理しないため、ルーティングテーブルに対してURLを追加した場合、それに対応するリソースの定義を認可機構にも追加しなければなりません。
- メニュー（テナント管理機能モジュール）
  メニューはデータ構造的にはメニューグループとメニューアイテムに分かれています。メニューグループはひとまとまりのメニューのツリーを表し、メニューアイテムは各機能へのリンクとしてURLを保持している情報です。

  メニューでは2段階の認可を行っており、まずメニューグループの認可要求を行い、それが許可されればその配下の各アイテムに対する認可要求を個別に行います。メニュー表示時には、認可で許可されたアイテムのみがメニューとして表示されます。

  メニューAPIはメニューグループと1対1で対応するリソースを自動的に管理しており、メニューグループを追加・削除した場合同期的に認可機構側のリソースを追加・削除するようになっています。
- ポータルモジュール
  ポータルは3種類のリソースを定義しています。

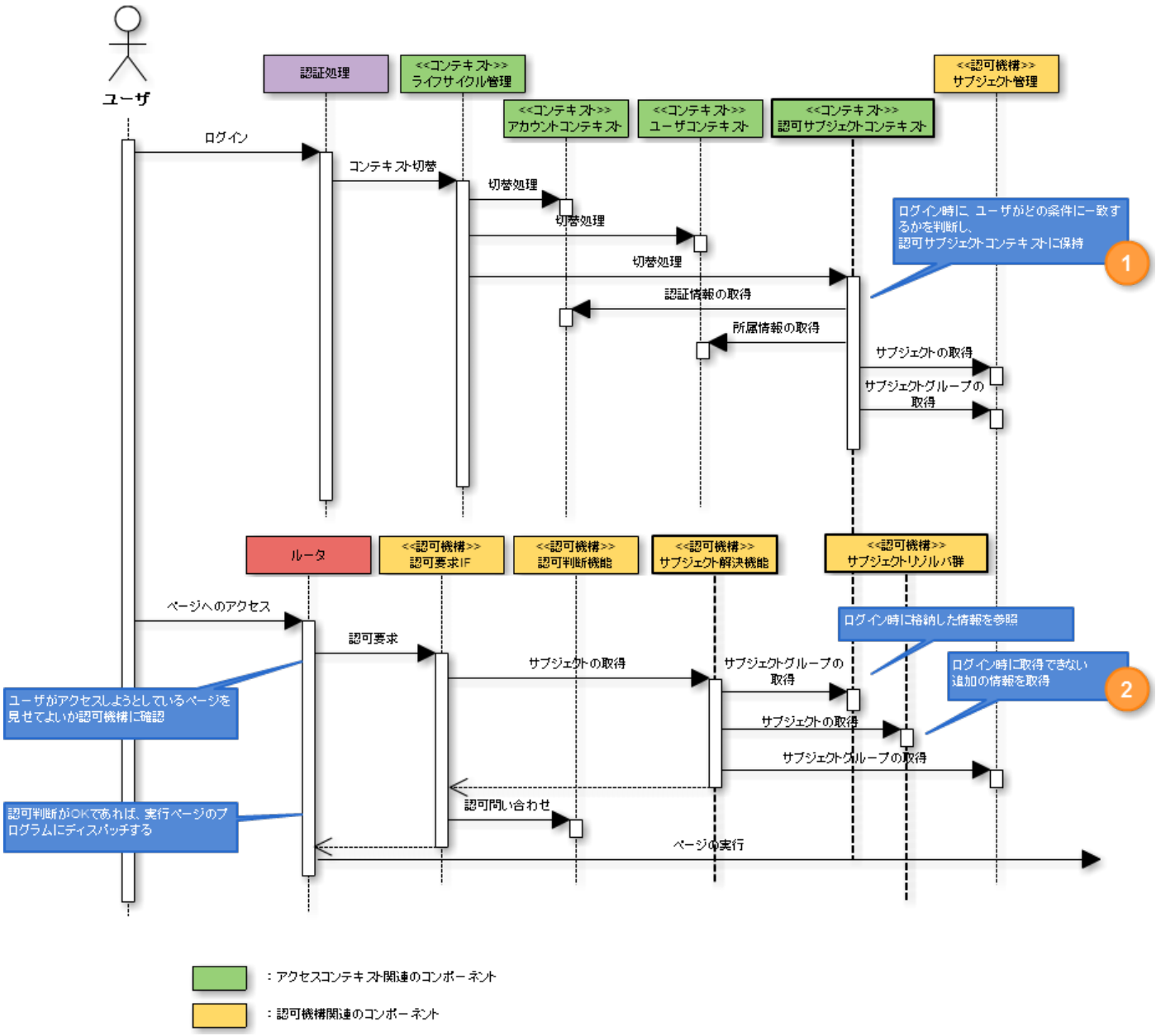
  1. ポータルの権限 ポータルを表すリソースです。ポータルの使用権限を管理します。ユーザはここで許可されたポータルを見ることができます。
  2. ポートレットの使用権限 ポートレットを表すリソースです。ポートレットの使用権限を管理します。 ユーザは許可されたポートレットのみ、自分のポートレットに追加することができます。
  3. ポートレット設定の変更権限 ポートレットの設定を変更する権限を管理します。 ユーザは許可されたポートレットのみ、ポートレット設定を変更することができます。

  それぞれポータルやポートレットの追加や削除に伴って同期的に認可機構側のリソースも変更されます。
- IM共通マスタ 認可連携モジュール
  IM共通マスタは会社に関して認可処理を行っています。特定の会社の情報を参照できるかどうかを認可機構を使用して管理します。この認可設定によって検索画面などにおいて許可されていない会社の情報は表示されなくなります。IM共通マスタで会社を追加・削除すると同期的に認可機構側のリソースも変更されます。

- WebService認証・認可モジュール
  WebService認証・認可モジュールを使用すると、Apache Axis 2と連携してWebServiceを作成することができるようになります。このモジュールは認可機構と連携しており、Apache Axis の使用する service.xmlの中に<authz>タグを追加できるようになっています。ここにルータ同様にサービスに対応するリソースURI・アクションを設定します。ルータ同様認可設定で許可されていなければサービスの実行を行わず、エラーとします。
- SAStruts Framework on Accel Platformモジュール
  SAStruts Framework on Accel Platformモジュールを使用すると、@Authz アノテーションを使ってアクションクラスやメソッドに対してルータ同様にリソースURI・アクションを定義することができます。ルータ同様認可設定で許可されていなければ処理の実行を行わず、エラーとします。

サブジェクト解決系

ここではユーザがどのサブジェクトに該当するかを判断するための機能について、説明します。



認可機構において、ユーザのサブジェクト解決処理は大きく2段階に分かれています。

1. まずログイン時に、ログインユーザについてその時点で明確になるサブジェクトとサブジェクトグループを解決し、認可サブジェクトコンテキストに保持します。（上図の１）
2. ページへのアクセスが発生したタイミングで認可要求が発生し、サブジェクト解決機能がこの時点でのサブジェクト解決を再度試みます。まず認可サブジェクトコンテキストに格納されている情報を取得し、これ以外に、このタイミングで追加で解決されるサブジェクトがあるかどうかをサブジェクトリゾルバを使って確認します。（上図の２）

認可サブジェクトコンテキストは、ユーザについて状況に左右されないサブジェクトをログインのタイミングでサブジェクト解決してコ

ンテキストの情報として保持します。

具体的には、ユーザの所属組織や所属パブリックグループなどがこれにあたります。組織やパブリックグループはユーザがログインした時点でどの組織に所属しているか明確であるため、ログイン時に一括して取得し、該当するサブジェクト情報としてコンテキストに保持しておきます。

この時解決できたサブジェクトからサブジェクトグループについても解決を行い、保持します。サブジェクトグループの解決はサブジェクトグループの表す式からマッチするかどうかを判断します。サブジェクトグループのマッチング処理については *式のマッチング仕様* を参照してください。

これに対して後者のサブジェクトリゾルバはリソースにアクセスを要求する時点になって初めて判明するサブジェクトや、ログインユーザ以外についてのサブジェクトを得たい場合に使用されます。このためサブジェクトリゾルバはコンテキストのように一定期間保持される情報ではなく、直接データベースなどの一次情報からユーザのサブジェクトを解決することを求められます。

これら認可サブジェクトコンテキストやサブジェクトリゾルバとして情報を収集する実装はサブジェクトタイプ同様に拡張可能になっており、サブジェクトタイプとほぼセットで追加されなければなりません。

認可サブジェクトコンテキスト、および、サブジェクトリゾルバの詳細は以下を参照してください。

サブジェクトリゾルバ

役割

サブジェクトリゾルバはユーザに対して該当するサブジェクトIDを解決します。

サブジェクトを解決するにはサブジェクトの実体を表す情報の取り扱いが不可欠になるため、リゾルバの実装自体は認可機構では提供しておらず、後述するインタフェースのみ定義しています。このインタフェースを実装したクラスを適宜プラグインすることでサブジェクト解決処理を実現しています。

サブジェクトリゾルバは2種のインタフェースが定義されています。

- DeclaredSubjectResolver
- OnDemandSubjectResolver

以降で順に説明します。

DeclaredSubjectResolver

インタフェース

完全修飾クラス名
　jp.co.intra_mart.foundation.authz.client.DeclaredSubjectResolver

このインタフェースを実装するリゾルバは、ユーザに対してログイン時から明確になっている、状況に依存しないサブジェクトの解決処理を担当します。

例としては所属会社や所属ロールなどです。

このインタフェースを持つ実装は、認可サブジェクトコンテキストを利用できる状況下では呼び出されません。ログインユーザ以外のユーザについてのサブジェクトを解決する場合などに呼び出されます。

設定

上記のインタフェースを実装するクラスを作成したら、それを認可機構に読み込ませる為の設定が必要です。

場所
　%CONTEXT_PATH%/WEB-INF/conf/declared-subject-resolvers/ 配下

XMLスキーマ
　%CONTEXT_PATH%/WEB-INF/schema/declared-subject-resolvers.xsd

下記は設定例です。下記のように、 **class-name 要素** に実装クラスの完全修飾クラス名を記述します。

```
<?xml version="1.0" encoding="UTF-8"?>
<p:declared-subject-resolvers
   xmlns:p="http://www.intra-mart.jp/authz/declared-subject-resolvers/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.intra-mart.jp/authz/declared-subject-resolvers/ ../../schema/declared-subject-resolvers.xsd">

 <p:class-name>jp.co.intra_mart.foundation.master.authz.subjectresolver.ImCompanyPostSubjectResolver</p:class-name>
 <p:class-name>jp.co.intra_mart.foundation.master.authz.subjectresolver.ImDepartmentSubjectResolver</p:class-name>
 <p:class-name>jp.co.intra_mart.foundation.master.authz.subjectresolver.ImPublicGroupRoleSubjectResolver</p:class-name>
 <p:class-name>jp.co.intra_mart.foundation.master.authz.subjectresolver.ImPublicGroupSubjectResolver</p:class-name>
 <p:class-name>jp.co.intra_mart.foundation.master.authz.subjectresolver.ImUserSubjectResolver</p:class-name>

</p:declared-subject-resolvers>
```

OnDemandSubjectResolver

インタフェース

完全修飾クラス名
　jp.co.intra_mart.foundation.authz.client.OnDemandSubjectResolver

ユーザに対して状況依存のサブジェクトの解決を担当します。認可要求時のリソースに応じてサブジェクトが変わる場合このインタフェースを実装します。

このインタフェースを持つ実装は認可判断時に毎回呼び出されます。

設定

上記のインタフェースを実装するクラスを作成したら、それを認可機構に読み込ませる為の設定が必要です。

場所
　%CONTEXT_PATH%/WEB-INF/conf/ondemand-subject-resolvers/ 配下

XMLスキーマ
　%CONTEXT_PATH%/WEB-INF/schema/ondemand-subject-resolvers.xsd

下記は設定例です。下記のように、**class-name 要素** に実装クラスの完全修飾クラス名を記述します。

```
<?xml version="1.0" encoding="UTF-8"?>
<p:ondemand-subject-resolvers
   xmlns:p="http://www.intra-mart.jp/authz/ondemand-subject-resolvers/"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.intra-mart.jp/authz/ondemand-subject-resolvers/ ../../schema/ondemand-subject-resolvers.xsd">

 <p:class-name>sample.TimeDependResolver</p:class-name>

</p:ondemand-subject-resolvers>
```

デフォルトで存在するサブジェクトリゾルバ

標準的なモジュール構成に含まれるサブジェクトリゾルバについて説明します。モジュール構成時の選択によって含まれるリゾルバは異なる場合があります。

基盤・テナント情報

認証状態(AuthzMetaSubjectResolver)

完全修飾クラス名
　jp.co.intra_mart.foundation.authz.subjectresolver.core.AuthzMetaSubjectResolver

実装インタフェース
　DeclaredSubjectResolver

処理の内容
　認証状態を解決するリゾルバです。「認証済み」か、「未認証（ゲストユーザ）」かのサブジェクトを解決します。
　また、招待機能と外部ユーザ モジュールを含めた環境では、解決するサブジェクトに「認証済み外部ユーザ」が追加されます。

ロール(IwpRoleSubjectResolver)

完全修飾クラス名
　jp.co.intra_mart.foundation.authz.subjectresolver.im_master.IwpRoleSubjectResolver

実装インタフェース
　DeclaredSubjectResolver

処理の内容
　ロールのサブジェクトを解決するリゾルバです。ユーザに割り当てられているロール（サブロールを含む）が、認可設定画面でサブジェクトとして登録されている場合、そのロールをサブジェクトとして解決します。

期間(ImTermSubjectResolver)

完全修飾クラス名
　jp.co.intra_mart.foundation.authz.subjectresolver.base.ImTermSubjectResolver

導入バージョン
　intra-mart Accel Platform 2013 Winter

実装インタフェース
　DeclaredSubjectResolver

処理の内容
　認可要求を行ったタイミングの時間が設定された期間内であるかを判断するリゾルバです。 期間が認可設定画面で定義されておりなおかつ、認可要求を行った時間が定義した期間内である場合、その期間をサブジェクトとして解決します。 期間の判断は、ユーザが設定しているタイムゾーンでの日付にて期間内であるかを判断します。

IM共通マスタ情報

会社組織(ImDepartmentSubjectResolver)

完全修飾クラス名
　jp.co.intra_mart.foundation.master.authz.subjectresolver.ImDepartmentSubjectResolver

実装インタフェース
　DeclaredSubjectResolver

処理の内容

会社組織のサブジェクトを解決するリゾルバです。ユーザが組織に所属しており、かつその組織が認可設定画面でサブジェクトとして登録されている場合、その組織をサブジェクトとして解決します。

認可サブジェクトコンテキストではカレント所属会社についてのみ解決しますが、このリゾルバはカレント所属会社にこだわりません。

役職(ImCompanyPostSubjectResolver)

完全修飾クラス名
jp.co.intra_mart.foundation.master.authz.subjectresolver.ImCompanyPostSubjectResolver

実装インタフェース
DeclaredSubjectResolver

処理の内容
役職のサブジェクトを解決するリゾルバです。ユーザに役職が割り当てられており、かつその役職が認可設定画面でサブジェクトとして登録されている場合、サブジェクトとして解決します。

パブリックグループ(ImPublicGroupSubjectResolver)

完全修飾クラス名
jp.co.intra_mart.foundation.master.authz.subjectresolver.ImPublicGroupSubjectResolver

実装インタフェース
DeclaredSubjectResolver

処理の内容
パブリックグループのサブジェクトを解決するリゾルバです。ユーザがパブリックグループに所属しており、かつそのグループが認可設定画面でサブジェクトとして登録されている場合、サブジェクトとして解決します。

役割(ImPublicGroupRoleSubjectResolver)

完全修飾クラス名
jp.co.intra_mart.foundation.master.authz.subjectresolver.ImPublicGroupRoleSubjectResolver

実装インタフェース
DeclaredSubjectResolver

処理の内容
役割のサブジェクトを解決するリゾルバです。ユーザに役割が割り当てられており、かつその役割が認可設定画面でサブジェクトとして登録されている場合、サブジェクトとして解決します。

ユーザ(ImUserSubjectResolver)

完全修飾クラス名
jp.co.intra_mart.foundation.master.authz.subjectresolver.ImUserSubjectResolver

実装インタフェース
DeclaredSubjectResolver

処理の内容
ユーザをサブジェクトとして解決するリゾルバです。ユーザが認可設定画面でサブジェクトとして登録されている場合、サブジェクトとして解決します。

認可サブジェクトコンテキスト

項目

ここでは認可判断時に必要なユーザのサブジェクト情報を保持する認可サブジェクトコンテキストについて説明します。

> **コラム**
>
> 認可サブジェクトコンテキストはアクセスコンテキストの一実装です。

認可サブジェクトコンテキストの役割

認可判断処理を行う際、「誰が」「何を」「どうする」が明確になっていなければなりません。「何を」「どうする」は認可要求側が明確にできますが、「誰が」については、ユーザの情報を基に所属ロールや所属組織の情報を一通りそろえる必要があります。（この処理をサブジェクト解決と呼んでいます）

認可判断処理は頻繁に発生しますので、処理のたびにサブジェクト解決処理を行っていると非常にシステム負荷が高くなります。認可サブジェクトコンテキストは、ユーザに対してログイン時から自明なサブジェクトをログイン時に解決しコンテキスト情報として保持することで、サブジェクト解決処理の回数を低減するためのキャッシュとして働きます

認可機構では認可判断を行う際、可能な限り認可サブジェクトコンテキストの情報をそのまま使用して処理を行います。

> **コラム**
>
> 認可サブジェクトコンテキストは基本的に認可要求を行う時以外参照する必要はありません。
> また認可要求を行う場合でも、認可機構内で自動的に認可サブジェクトコンテキストを参照します（引数のサブジェクトグループIDを省略した場合）。
> このため、アプリケーション開発において認可サブジェクトコンテキストを参照する必要は通常ありません。

コンテキストインタフェース

完全修飾クラス名
`jp.co.intra_mart.foundation.authz.context.AuthzSubjectContext`

利用サンプル

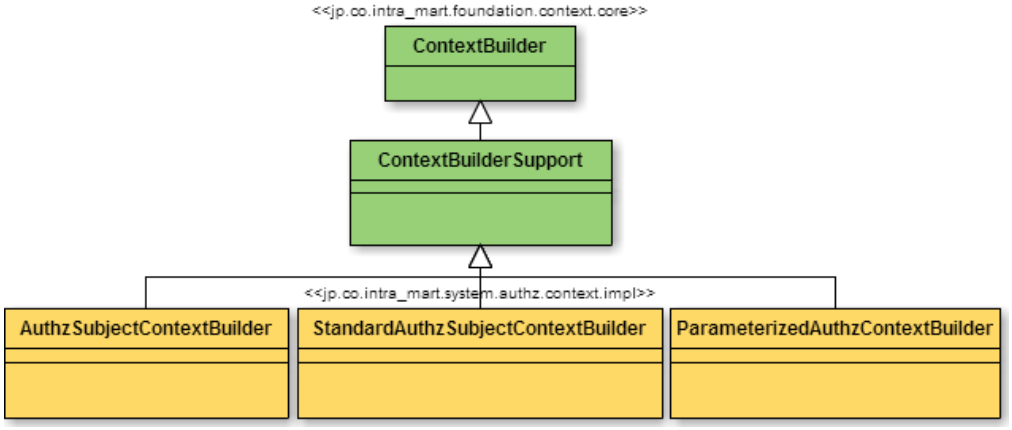
```
Set<String> subjectIds = new HashSet<String>();
AuthzSubjectContext authzContext = null;

try {
    authzContext = Contexts.get(AuthzSubjectContext.class); // コンテキストの取得
} catch (final ContextNotFoundException e) {
    LOGGER.debug("AuthzSubjectContext not defined", e);
}

if (authzContext != null ) {
    subjectIds.addAll(authzContext.getSubjectIds());
    LOGGER.debug("user {} can use authz subject context", authzContext.getUserIdentifier());
}

...
```

保持する属性

属性	型	説明
subjectIds	Set<String>	アカウントコンテキストの示すユーザに対するあらかじめ明示されるサブジェクトIDの一覧。
subjectGroupIds	Set<String>	上記 subjectIds から解決可能なサブジェクトグループIDの一覧。
userIdentifier	String	ユーザ識別子。現在のデフォルト実装ではユーザコードをそのまま格納しています。
revision	long	コンテキストリビジョン。このコンテキストの内容がアップデートされたり再構築されると値が変更されます。順序や連続性は保証されません。デフォルトの実装ではコンテキストビルダーがコンテキストを変更した時の System.currentTimeMillis() が入っています。

コンテキストビルダー

認可機構では認可サブジェクトコンテキストを構築するためのビルダーを3種類用意しています（下図中の黄色で示したクラス）。



AuthzSubjectContextBuilder

完全修飾クラス名

`jp.co.intra_mart.system.authz.context.impl.AuthzSubjectContextBuilder`

処理内容

認可サブジェクトコンテキストのインスタンスを作成して特に何もせず返すビルダーです。Web以外の環境などで後述の `StandardAuthzSubjectContextBuilder` が動作できる環境にない場合に使用します。

ほぼ空の状態のコンテキストが返されるので、ユーザのサブジェクトがない状態になります。

StandardAuthzSubjectContextBuilder

完全修飾クラス名

`jp.co.intra_mart.system.authz.context.impl.StandardAuthzSubjectContextBuilder`

処理内容

アカウントコンテキストやユーザコンテキストを参照して、それらのコンテキストが構築されるタイミングで同時に認可サブジェクトコンテキストを構築します。デコレータを実行することでサブジェクトIDをコンテキストに登録させ、そのあとでそれらのサブジェクトIDをもとにサブジェクトグループIDの解決を行います。

通常のWebでの利用を想定したコンテキストビルダです。

> **コラム**
>
> このコンテキストビルダはテナント管理機能のテナント環境セットアップが完了していなければデコレータを一切実行しません。
> これによりテナント環境セットアップが完了していない環境下ではユーザのサブジェクトが一切解決されなくなり、*OnDemandSubjectResolver* によって追加でサブジェクトが解決されない限りすべての認可判定が「禁止」になります。

ParameterizedAuthzContextBuilder

完全修飾クラス名

`jp.co.intra_mart.system.authz.context.impl.ParameterizedAuthzContextBuilder`

処理内容

おもに単体テスト目的で使用するビルダーですcontext-config.xmlに指定可能な `<init-param>` の値からコンテキストの内容を固定的に生成します。

`<init-param>` に以下のキーを指定可能です

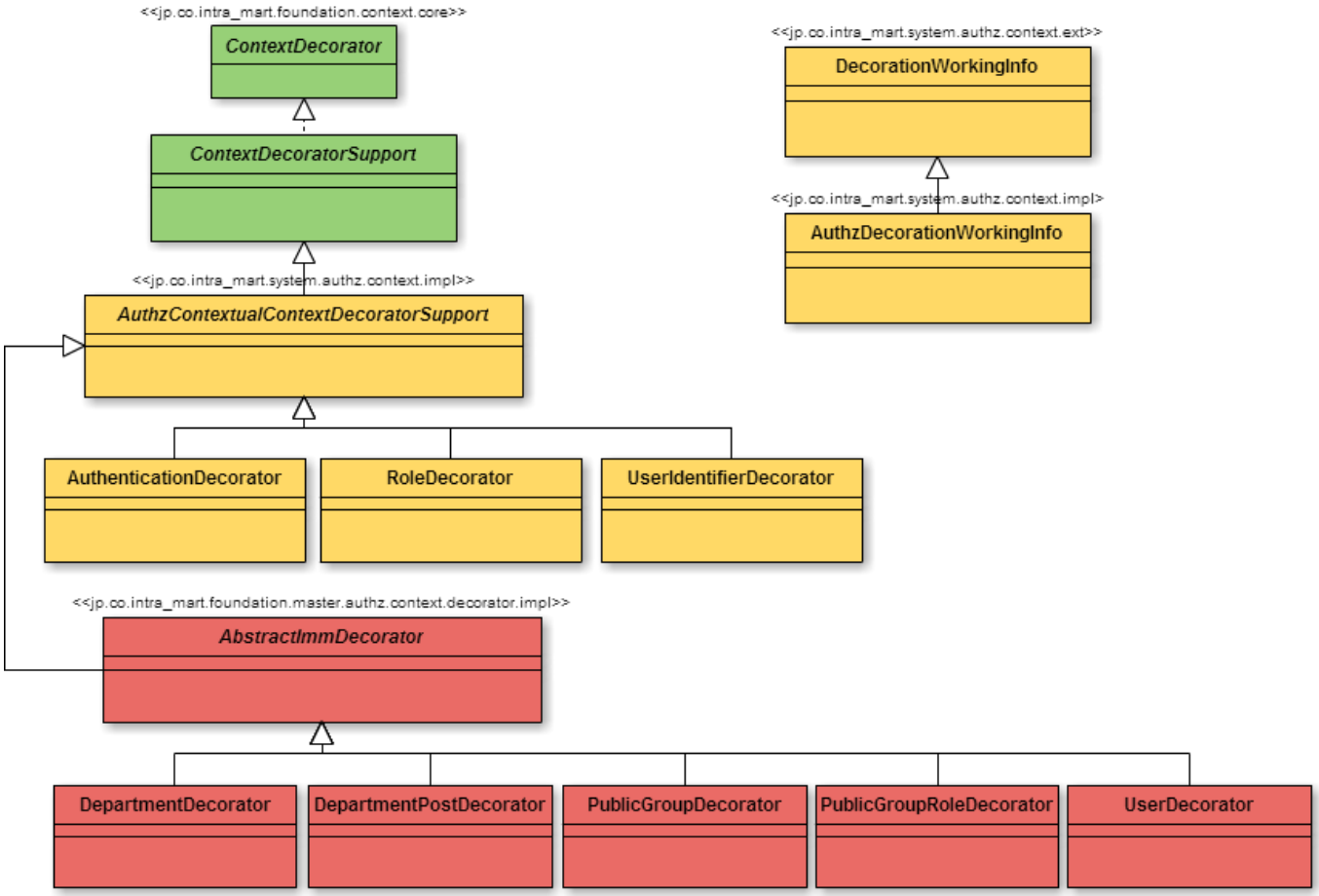
キー	説明
userCd	設定した値が userIdentifier に設定されます。

キー	説明
subject	サブジェクトIDに格納する値を指定します。サブジェクトタイプとビジネスキーで指定できます。ただし実際認可に使用されるのは後述のサブジェクトグループIDに指定する内容なので、内容の整合性が取れている必要があります。複数指定可能です。 書式： [サブジェクトタイプID]:[キー] ... この書式については「*式の文字列表現*」を参照してください。 例： b_m_role:menu_manager
subjectGroupId	サブジェクトグループIDに格納する値を指定します。実際認可に利用されるのはこの値になります。複数指定可能です。 書式： オペレータ(式, 式, 式, ...) この書式については「*式の文字列表現*」を参照してください。 例： AND(S(b_m_role:menu_manager), S(imm_user:aoyagi))

コンテキストデコレータ

コンテキストデコレータはコンテキストビルダーに依頼されてコンテキストに格納する情報を実際に収集する役割を持っています。ここでは認可サブジェクトコンテキストで用意しているコンテキストデコレータについて説明します。

デフォルトで存在するサブジェクトコンテキストデコレータ



基盤・テナント情報
UserIdentifierDecorator

完全修飾クラス名
jp.co.intra_mart.system.authz.context.decorator.impl.UserIdentifierDecorator

処理内容

アカウントコンテキストのユーザコードを認可サブジェクトコンテキストの userIdentifier に複写するデコレータです。この情報は認可要求時にセッションの情報を利用できるかどうかの判別のために利用されます。

AuthenticationDecorator

完全修飾クラス名

jp.co.intra_mart.system.authz.context.decorator.impl.AuthenticationDecorator

処理内容

アカウントコンテキストから認証済みユーザかそうでないかを判別して、ゲストユーザおよび認証済みユーザのサブジェクトを認可サブジェクトコンテキストに追加します。
また、招待機能と外部ユーザ モジュールを含めた環境では、認証済み外部ユーザかどうかの判別が追加されます。

RoleDecorator

完全修飾クラス名

jp.co.intra_mart.system.authz.context.decorator.impl.RoleDecorator

処理内容

アカウントコンテキストから所属ロール情報を取得して該当するサブジェクトを取得し、認可サブジェクトコンテキストに追加します。

IPv4AddressDecorator

完全修飾クラス名

jp.co.intra_mart.system.authz.context.decorator.impl.IPv4AddressDecorator

導入バージョン

intra-mart Accel Platform 2013 Spring

処理内容

HTTPコンテキストからリモートホストのIPアドレス(v4)を取得して該当するサブジェクトを取得し、認可サブジェクトコンテキストに追加します。

> **コラム**
>
> 該当するIPアドレス(v4)の条件指定は「xxx.xxx.xxx.xxx」形式で行います。 [m-n] での範囲指定や、* でのワイルドカード指定を行うことができます。
> 例) 192.168.[0-24].[0-254] 192.168.0.*

TermDecorator

完全修飾クラス名

jp.co.intra_mart.system.authz.context.decorator.impl.TermDecorator

導入バージョン

intra-mart Accel Platform 2013 Winter

処理内容

コンテキストのビルドを行う時間からサブジェクトの解決を行い、認可サブジェクトコンテキストに追加します。 期間の判定は利用者のタイムゾーンでの日付で行われます。

IM共通マスタ情報

以下はIM共通マスタ情報を解決するデコレータです。

> **注意**
>
> IM共通マスタ情報の組織情報を解決するデコレータ（DepartmentDecorator、DepartmentPostDecorator）では、デフォルト組織セットの情報のみを扱います。

> **コラム**
>
> IM共通マスタ情報のデコレータはIM共通マスタのテナント環境セットアップが完了していないと処理をスキップします。
> このため、IM共通マスタのテナント環境セットアップが完了していない状況下ではユーザは組織やパブリックグループといったサブジェクトとして解決されなくなります。

DepartmentDecorator

完全修飾クラス名

jp.co.intra_mart.foundation.master.authz.context.decorator.impl.DepartmentDecorator

処理内容

ユーザコンテキストのカレント組織情報からサブジェクトの解決を行い、認可サブジェクトコンテキストに追加します。

- カレント組織がサブジェクトとして登録されているかを検索します。
- カレント組織の上位組織が「下位」や「一致・下位」として登録されているかを検索します。
- カレント組織の下位組織が「上位」や「一致・上位」として登録されているかを検索します。

カレント組織以外の所属情報についてはサブジェクト解決しません。

DepartmentPostDecorator

完全修飾クラス名

jp.co.intra_mart.foundation.master.authz.context.decorator.impl.DepartmentPostDecorator

処理内容

ユーザコンテキストの役職情報からサブジェクトの解決を行い、認可サブジェクトコンテキストに追加します。

- 役職ランクから上位や下位の条件付けがされているサブジェクトも判定します。

PublicGroupDecorator

完全修飾クラス名

jp.co.intra_mart.foundation.master.authz.context.decorator.impl.PublicGroupDecorator

処理内容

ユーザコンテキストのパブリックグループ所属情報からサブジェクトの解決を行い、認可サブジェクトコンテキストに追加します。

- 所属グループがサブジェクトとして登録されているかを検索します。
- 所属グループの上位グループが「下位」や「一致・下位」として登録されているかを検索します。
- 所属グループの下位グループが「上位」や「一致・上位」として登録されているかを検索します。

PublicGroupRoleDecorator

完全修飾クラス名

jp.co.intra_mart.foundation.master.authz.context.decorator.impl.PublicGroupRoleDecorator

処理内容

ユーザコンテキストのパブリックグループ役割情報からサブジェクトの解決を行い、認可サブジェクトコンテキストに追加します。役割ランクから上位や下位の条件付けがされているサブジェクトも判定します。

UserDecorator

完全修飾クラス名

jp.co.intra_mart.foundation.master.authz.context.decorator.impl.UserDecorator

処理内容

アカウントコンテキストのユーザコードからサブジェクトの解決を行い、認可サブジェクトコンテキストに追加します。

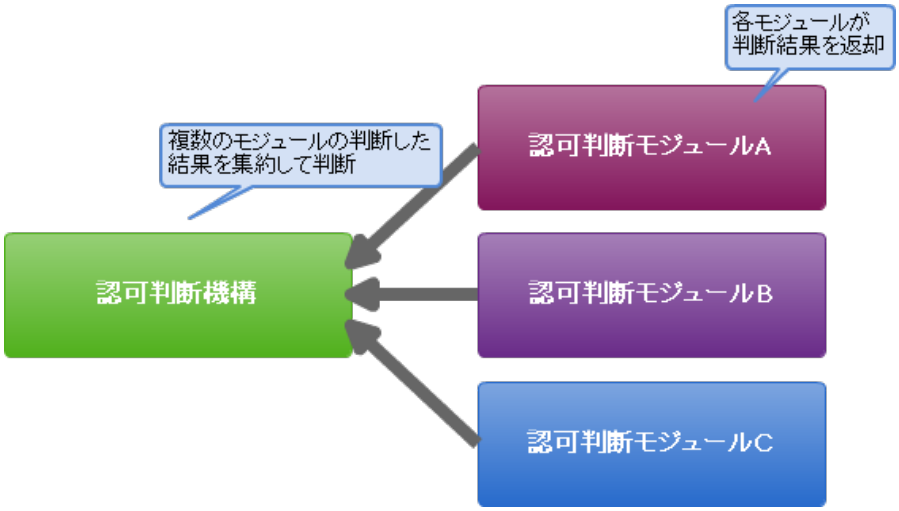
認可判断機能

ここでは認可判断をするための機能について解説します。

項目

認可判断機能の構造

認可判断機構は複数の認可判断モジュールを実行しその結果を集約して最終的な認可判断を行います。



認可判断機能は起動時に設定ファイル authz-decision-config.xml を読み込み、認可判断モジュールをロードします。

```
<?xml version="1.0" encoding="UTF-8"?>
<a:authz-decision-config
   xmlns:a="http://www.intra-mart.jp/authz/authz-decision-config"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.intra-mart.jp/authz/authz-decision-config ../schema/authz-decision-config.xsd">

 <a:decision-config name="default" combinator="permit-overrides">
  <a:module class="jp.co.intra_mart.foundation.authz.services.decision.impl.AdministratorByPassModule" />
  <a:module class="jp.co.intra_mart.foundation.authz.services.decision.impl.PlatformWorkerByPassModule" />
  <a:module class="jp.co.intra_mart.foundation.authz.services.decision.impl.StandardPolicyDecisionModule" />
 </a:decision-config>

</a:authz-decision-config>
```

- **decision-config 要素** 1 要素が設定の単位です。
  - **name 属性**：設定識別用の名称。将来の拡張の為に用意されています。現状では基本的に default しか受け付けません。他の値を設定しても参照しません。
  - **combinator 属性**：複数のモジュールの処理結果をどのように判断するかを指定します。以下の中から選択します。
    - permit-overrides
      - 上から順に評価して最初に Permit / Block を返したモジュールの結果で処理します。
    - deny-overrides
      - 上から順に評価して最初に Deny / Block を返したモジュールの結果で処理します。
    - first-applicable
      - 上から順に評価して最初に Permit / Deny / Block を返したモジュールの結果で処理します。
- **module 要素** 認可判断を行うモジュールクラスを定義します。このクラスは DecisionModule インタフェースを実装していなければなりません。
  - **class 属性**：完全修飾クラス名を指定します。

認可判断機能は上記の設定にしたがって、複数の認可判断モジュールを実行し、その結果から最終的な判断を行います。

認可判断モジュール

認可判断モジュールは認可要求の情報を受け取って、ユーザを認可してよいかどうかを判断する機構です。このモジュールは複数定義でき、複数の認可判定処理を実行することができるようになっています。認可判断モジュールの下した判断が最終的に採用されるかどうかは認可判断機能が決めます。

認可判断モジュールは次の4種類の値を返すことができます。

Permit
　要求を許可します。

Deny
　要求を禁止します。

Block
　要求されたリソースが閉塞状態であることを表します。

NotApplicable
　このモジュールでは判断しない・判断できないことを表します。

デフォルトで組み込まれるモジュール

デフォルトインストール状態では以下の3つのモジュールが組み込まれた状態になっています。

- StandardPolicyDecisionModule
- AdministratorByPassModule
- PlatformWorkerByPassModule

順に説明します。

StandardPolicyDecisionModule

完全修飾クラス名
　jp.co.intra_mart.foundation.authz.services.decision.impl.StandardPolicyDecisionModule

認可設定画面で設定したポリシーを参照して認可判断を行うモジュールです。認可設定画面の設定内容に基づいて、許可（Permit）・禁止（Deny）・閉塞（Block）を返します。

このモジュールを外すと認可設定画面の設定内容が認可判断に使用されなくなります。ほとんどの場合、通常の運用で必要になります。

ポリシーの設定内容を解釈するために内部で *ポリシー解釈器* を使用しています。

AdministratorByPassModule

完全修飾クラス名
　jp.co.intra_mart.foundation.authz.services.decision.impl.AdministratorByPassModule

アカウントコンテキストを参照し、システム管理者であればすべての認可要求に一律許可（Permit）を返すモジュールです。システム管理者でなければ NotApplicable を返します。このため、システム管理者はすべてのリソースに無制限にアクセスできます。特にテナント環境セットアップ時にすべての権限を超越するために使用されます。

PlatformWorkerByPassModule

完全修飾クラス名
　jp.co.intra_mart.foundation.authz.services.decision.impl.PlatformWorkerByPassModule

コンテキストを参照し、ユーザタイプがプラットフォームユーザの場合に認可要求に一律許可（Permit）を返すモジュールです。プラットフォームユーザでなければ NotApplicable を返します。これは主にバッチ処理時に一律認可を許可するために設定されています。

ポリシー解釈器

ポリシー解釈器は特定のリソースに対して実際に適用される認可設定を導出する役割を担う部分です。ポリシー解釈にあたっておもに以下のポイントを決定しています。

- リソースの階層構造に沿った権限の継承動作
- ポリシー設定が行われていない場合のデフォルト値(Permit / Deny)
- 複数のサブジェクトグループに対して異なるエフェクトが設定されている場合の判定方式

ポリシー解釈器は以下のインタフェースを持ちます。

完全修飾クラス名
　jp.co.intra_mart.foundation.authz.services.admin.impl.PolicyInterpreterModule

また、キャッシュの実装を想定したポリシー解釈器は以下のインタフェースです。

完全修飾クラス名
　jp.co.intra_mart.foundation.authz.services.admin.impl.cache.CachedPolicyInterpreterModule

ポリシー解釈は大きく分けて2つの場面で利用されます。

- 認可クライアントから認可要求を受け、実際に認可判断を行う際に指定されたリソース、ログインユーザのサブジェクトから、リソース構造や設定されたエフェクトを勘案し、実質的な認可判断処理を行います。(interpretEffect())
- 認可設定のメンテナンス機能において、リソースおよびサブジェクトに対して、リソース構造や設定されたエフェクトを勘案し実質的な設定を解釈します。(interpretPolicies())

このため、このインタフェースの持つ2つのメソッドは、双方とも同じ方式の解釈を行わなければなりません。

デフォルトで組み込まれるモジュール

認可機構はデフォルトではホワイトリスト方式の認可判断を行っています。

WhiteListPolicyInterpreterModule

完全修飾クラス名
jp.co.intra_mart.foundation.authz.services.admin.impl.WhiteListPolicyInterpreterModule

ポリシー設定をホワイトリストとして解釈します。

- 直接ポリシーが設定されていないリソースについて、リソース構造に従って設定の継承を行います。
- デフォルト値は Deny として扱います
- ユーザにマッチするサブジェクトグループについて一つでも Permit が設定されていれば Permit として判断します

内部でキャッシュを保持しています。 キャッシュを適切なタイミングで更新するために CachedPolicyInterpreterModule を実装してます。

キャッシュ設定

設定ファイルの変更によりキャッシュの利用方針の変更することが可能です。

1. ポリシーのキャッシュサイズの設定

   WhiteListPolicyInterpreterModule で使用するキャッシュの設定を行えます。キャッシュの有効・無効、キャッシングのサイズの設定が行えます。キャッシュの実装はim-cacheを利用しているため、im-cacheから提供している設定が行えます。

   設定ファイルの配置場所
   %CONTEXT_PATH%/WEB-INF/conf/im-ehcache-config/authz-policy.xml

2. ポリシーのキャッシュ対象となる条件の設定

   キャッシュを行うリソースのリソースタイプを設定します。この設定で指定したリソースタイプのポリシーのみキャッシュを行います。

   設定ファイルの配置場所
   %CONTEXT_PATH%/WEB-INF/conf/policy-cache-config/

   XMLスキーマファイル
   %CONTEXT_PATH%/WEB-INF/schema/policy-cache-config.xsd

   以下のサンプルをもとに設定内容について説明します。

```
<?xml version="1.0" encoding="UTF-8"?>
<policy-cache-config
   xmlns="http://www.intra-mart.jp/authz/policy-cache-config"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.intra-mart.jp/authz/policy-cache-config ../../schema/policy-cache-config.xsd">

 <cache-target>
  <resource-type>service</resource-type>
 </cache-target>

</policy-cache-config>
```

   - **policy-cache-config要素：**設定のルートになる要素です。
     - **cache-target要素：**キャッシュ対象の指定を行う要素です。

- **resource-type要素：**キャッシュ対象となるリソースタイプを指定する要素です。cache-target要素内に複数指定できます。

API

認可判断機構を呼び出すには下記のインタフェースを使用します。

認可判断機能API ファクトリ
PolicyDecisionServiceFactory (jp.co.intra_mart.foundation.authz.services.decision.PolicyDecisionServiceFactory)

認可判断機能API
PolicyDecisionService (jp.co.intra_mart.foundation.authz.services.decision.PolicyDecisionService)

> **注意**
> ただし、アプリケーションは原則的に認可判断機能のAPIを直接呼び出してはいけません。 認可判断を行いたい場合は、「*APIによる認可要求*」を参照してください。

使用例

```
String userCd   = "aoyagi"
String resource = "service://sample/service1"
String action   = "execute";

PolicyDecisionService policyDecisionService = PolicyDecisionServiceFactory.getInstance().getPolicyDecisionService();
Effect e = policyDecisionService.authorize(userCd, resource, action)

if (e.equals(Effect.PERMIT)) {
  // 許可されている場合に実行する処理

} else {
  // 許可されていない場合に実行する処理

}
```

閉塞

StandardPolicyDecisionModule では閉塞状態の判断を行います。閉塞とは、リソースとアクションに対して制限をかける機能です。
閉塞が行われているリソースとアクションに対して認可判断を行った場合、ポリシーの設定状態に関わらず、認可判断の結果は「閉塞（Block)」となります。
閉塞の設定は認可設定画面、および、「ResourceBlocker」で行えます。

> **コラム**
> ResourceBlocker は intra-mart Accel Platform 2013 Winter より利用可能です。

閉塞の仕様

閉塞の状態管理は「*リソースグループ汎用属性*」を利用しています。利用するリソースグループ汎用属性のキーは im_authz-blocked です。この値は「ResourceGroupAttributeKey.BLOCKED」で取得可能です。
閉塞状態別のリソースグループ汎用属性の値は以下の通りです。

- リソースグループ自体を閉塞する
  値には、文字列 ALL が代入されます。
- リソースグループに紐づく特定のアクションを閉塞する

  値には、閉塞するアクションが代入されます。複数のアクションを閉塞する場合は、カンマ区切りで複数のアクションが代入されます

  形式

  ```
  <リソースタイプ>:<アクション>,<リソースタイプ>:<アクション>,...
  ```

例：メニューグループのアクション、管理と参照を閉塞する場合

```
im-menu-group:admin,im-menu-group:read
```

- 閉塞しない

  リソースグループに紐づく汎用属性のキー `im_authz-blocked` を持つレコードがない状態です。

閉塞状態は、認可判断を行うリソースに紐づくリソースグループが閉塞されているかどうかで判別します。
特定のリソースグループ配下のリソースグループすべてに対して閉塞を行う場合、リソースグループ配下のリソースグループすべてに対して閉塞の設定を行う必要があります。

> **コラム**
> 認可設定画面、および、「ResourceBlocker」で閉塞状態の設定を行った場合、指定したリソースグループ配下のリソースグループすべてに対して閉塞状態の設定が反映されます。

過去バージョンにおける閉塞の仕様

intra-mart Accel Platform 2013 Autumn以前の閉塞

リソースグループ自体の閉塞のみ行えます。アクションを指定した閉塞は行えません。

intra-mart Accel Platform 2013 Spring以前の閉塞

閉塞機能は提供されていません。

付録

初期化仕様

認可機構では、システム起動時に複数のコンポーネントの初期化処理を行っています。

初期化処理では、以下のインタフェースを持つクラスを ServiceLoader を使って読み込み、以下の Level の順で実行します。

- Level0 (jp.co.intra_mart.foundation.authz.initialize.phases.Level0)

  XMLの設定情報をベースに動作可能なコンポーネントの初期化を行います。基本的にDBへのアクセスを必要としないものが対象になります。ResourceManagerやSubjectManager等のマネージャAPIはこのレベルでは使えません。

  - ルーティングテーブルに使用する認可リソースマッパー拡張の読み込み
  - 認可判断機能の認可判断モジュール拡張の読み込み
  - 式パーサーの初期化
  - リソースタイプ拡張の読み込み
  - サブジェクトタイプ拡張の読み込み
  - ポリシー解釈器設定の読み込み
- Level1 (jp.co.intra_mart.foundation.authz.initialize.phases.Level1)

  XMLの設定情報をベースに動作可能なコンポーネント（Level0で初期化されるコンポーネント）を使用するコンポーネントの初期化を行います。マネージャAPIのうち初期化が必要なものはこのレベルで初期化されます。ただしテナント環境セットアップが最後まで完了していない場合、このレベルの初期化が終わっても正常に動作できない可能性があります。

  - サブジェクト解決系の初期化
  - サブジェクトリゾルバ拡張の読み込み
- Level2 (jp.co.intra_mart.foundation.authz.initialize.phases.Level2)

  マネージャAPIを使用する必要のあるコンポーネントの初期化を行います。

プラグインや拡張などを追加する際に、起動時の初期化が必要であれば、適切なタイミングで初期化できるよう注意してください。

トランザクション管理方針

このドキュメントで紹介しているAPIは呼び出す側でトランザクションが張られていなければ、APIのメソッド呼び出しの先頭でトランザクションを張り、メソッドの終わりに Commit/Rollbackします。

呼び出す側でトランザクションが張られていれば、内部でトランザクションを操作することはありません。

intra-mart Accel Platform に含まれるリソースタイプ

intra-mart Accel Platform が標準的に提供しているリソースタイプです。リソースタイプIDはシステムで一意でなければなりません。アプリケーションのためにリソースタイプを追加する場合は重複しないよう注意してください。

項目
画面・処理(service)
テナント管理 / メニュー(im-menu-group)
ポータル / ポータル(im-portal-portal)
ポータル / ポートレット(im-portal-portlet)
ポータル / ポートレット編集モード(im-portal-portlet-editmode)
IM共通マスタ / 会社(im_master)
IM共通マスタ / ユーザプロファイル(im-master-user-profile)
IM共通マスタ / 個人プロファイル(im-master-user-self-profile)
IMBox / IMBox(imbox-auth)
IM-LogicDesigner / IM-LogicDesigner REST API(im-logic-rest)

画面・処理(service)

リソースタイプID
service

説明
おもにルーティングテーブルのURLやWebサービスに紐づけられるリソースです。実行プログラムのエントリーポイントになるリソースです。

定義アクション
実行(execute)

テナント管理 / メニュー(im-menu-group)

リソースタイプID
im-menu-group

説明
メニューグループを表すリソースです。メニューコンポーネントから利用されます。メニュー管理画面やAPIからメニューグループを作成すると自動的にリソースが追加され、メニューグループが削除されると同様に削除されます。

定義アクション
参照(read)
管理(admin)

ポータル / ポータル(im-portal-portal)

リソースタイプID
im-portal-portal

説明
ポータルを表すリソースです。

ポータル自体に紐づくリソースです。ポータルの使用権限にかかわります。ポータルを追加すると自動的にリソースが追加されます。

定義アクション
参照(execute)

ポータル / ポートレット(im-portal-portlet)

リソースタイプID
im-portal-portlet

説明
ポートレットを表すリソースです。

ポータルにポートレットを表示することができるかどうかの権限にかかわります。ポートレットを追加すると自動的にリソースが追加されます。

定義アクション
利用(execute)

ポータル / ポートレット編集モード(im-portal-portlet-editmode)

リソースタイプID
im-portal-portlet-editmode

説明
ポートレット編集モードを表すリソースです。

ポートレット毎に用意される設定変更画面に紐づけられます。ポートレットを追加すると自動的にリソースが追加されます。

定義アクション
実行(execute)

IM共通マスタ / 会社(im_master)

リソースタイプID
im_master

説明
会社を表すリソースです。会社の管理画面から会社を追加・削除すると連動して会社に対応するリソースが追加・削除されます。

定義アクション
参照(reader)
編集(writer)

IM共通マスタ / ユーザプロファイル(im-master-user-profile)

リソースタイプID
im-master-user-profile

説明
ユーザプロファイルを表すリソースです。他のユーザのプロファイル情報を参照できるかどうかの権限にかかわります。

定義アクション
参照(reader)

IM共通マスタ / 個人プロファイル(im-master-user-self-profile)

リソースタイプID
im-master-user-self-profile

説明
個人プロファイルを表すリソースです。自分自身のプロファイル情報を参照・編集できるかどうかの権限にかかわります。

定義アクション
参照(reader)
編集(writer)

IMBox / IMBox(imbox-auth)

リソースタイプID
imbox-auth

説明
IMBoxを表すリソースです。

定義アクション
利用(execute)

IM-LogicDesigner / IM-LogicDesigner REST API(im-logic-rest)

リソースタイプID
im-logic-rest

説明
フローをREST APIとして利用するためのルート情報に紐づくリソースです。 ユーザがREST APIを実行可能であるかを管理します。

定義アクション
実行(execute)

intra-mart Accel Platform に含まれるサブジェクトタイプ

intra-mart Accel Platform が標準的に提供しているサブジェクトタイプです。サブジェクトタイプIDはシステムで一意でなければなりません。アプリケーションのためにサブジェクトタイプを追加する場合は重複しないよう注意してください。

項目

IM共通マスタ / ユーザ（imm_user）

サブジェクトタイプID

`imm_user`

連携エンティティ

IM共通マスタ ユーザ（imm_user）

連携エンティティのキー

1. ユーザコード（user_cd）

ID生成に要求する値

1. ユーザコード（user_cd）

シリアライズされた文字列の書式

imm_user:[ユーザコード]

例

`imm_user:aoyagi`

IM共通マスタ / 会社組織（imm_department）

サブジェクトタイプID

`imm_department`

連携エンティティ

IM共通マスタ 会社組織（imm_department およびその関連テーブル）

連携エンティティのキー

1. 会社コード（company_cd）
2. 組織セットコード（department_set_cd）
3. 組織コード（department_cd）

ID生成に要求する値

1. 会社コード（company_cd）
2. 組織セットコード（department_set_cd）
3. 組織コード（department_cd）
4. 比較演算子

比較演算子には次のいずれかを指定してください。

演算子	説明
lt	下位組織 を示します。
le	一致するかまたは下位の組織 を示します。
eq	一致する組織 を示します。
ge	一致するかまたは上位の組織 を示します。
gt	上位組織 を示します。

シリアライズされた文字列の書式

imm_department:[会社コード] [組織セットコード] [組織コード] [比較演算子]

各コードの間は空白区切りです。

例

`imm_department:comp_sample_01 comp_sample_01 comp_sample_01 eq`

IM共通マスタ / 役職（imm_company_post）

サブジェクトタイプID
　`imm_company_post`

連携エンティティ
　IM共通マスタ 役職（imm_company_post およびその関連テーブル）

連携エンティティのキー

1. 会社コード （company_cd）
2. 組織セットコード （department_set_cd）
3. 役職コード （post_cd）

ID生成に要求する値

1. 会社コード （company_cd）
2. 組織セットコード （department_set_cd）
3. 役職コード （post_cd）
4. 比較演算子

比較演算子には次のいずれかを指定してください。

演算子	説明
lt	ランクが下位の役職 を示します。
le	ランクが同じか下位の役職 を示します。
eq	指定の役職 を示します。
ge	ランクが同じか上位の役職 を示します。
gt	ランクが上位の役職 を示します。

ランクの数値が小さいほうが上位と判定されます。

シリアライズされた文字列の書式

imm_company_post:[会社コード] [組織セットコード] [役職コード] [比較演算子]

各コードの間は空白区切りです。

例

`imm_company_post:comp_sample_01 comp_sample_01 ps001 lt`

IM共通マスタ / パブリックグループ（imm_public_grp）

サブジェクトタイプID

`imm_public_grp`

連携エンティティ

IM共通マスタ パブリックグループ（imm_public_grp およびその関連テーブル）

連携エンティティのキー

1. パブリックグループセットコード（public_group_set_cd）
2. パブリックグループコード（public_group_cd）

ID生成に要求する値

1. パブリックグループセットコード（public_group_set_cd）
2. パブリックグループコード（public_group_cd）
3. 比較演算子

比較演算子には次のいずれかを指定してください。

演算子	説明
`lt`	下位グループ を示します。
`le`	一致するかまたは下位のグループ を示します。
`eq`	一致するグループ を示します。
`ge`	一致するかまたは上位のグループ を示します。
`gt`	上位グループ を示します。

シリアライズされた文字列の書式

imm_public_grp:[パブリックグループセットコード] [パブリックグループコード] [比較演算子]

例

`imm_public_grp:sample_public public_group_a ge`

IM共通マスタ / パブリックグループ役割（imm_public_grp_role）

サブジェクトタイプID

`imm_public_grp_role`

連携エンティティ

IM共通マスタ パブリックグループ役割（imm_public_grp_role およびその関連テーブル）

連携エンティティのキー

1. パブリックグループセットコード（public_group_set_cd）
2. 役割コード（public_group_role_cd）

ID生成に要求する値

1. パブリックグループセットコード（public_group_set_cd）
2. 役割コード（public_group_role_cd）
3. 比較演算子

比較演算子には次のいずれかを指定してください。

演算子	説明
lt	ランクが下位の役割 を示します。
le	ランクが同じか下位の役割 を示します。
eq	指定の役割 を示します。
ge	ランクが同じか上位の役割 を示します。
gt	ランクが上位の役割 を示します。

ランクの数値が小さいほうが上位と判定されます。

シリアライズされた文字列の書式

imm_public_grp_role:[パブリックグループセットコード] [役割コード] [比較演算子]

例

`imm_public_grp_role:sample_public role1 lt`

テナント管理 / ロール（b_m_role）

サブジェクトタイプID
　`b_m_role`

連携エンティティ
　ロール（b_m_role_b およびその関連テーブル）

連携エンティティのキー

1. ロールID（role_id）

ID生成に要求する値

1. ロールID（role_id）

シリアライズされた文字列の書式

b_m_role:[ロールID]

例

`b_m_role:authz_manager`

テナント管理 / IPv4 アドレス（im_authz_ipv4）

サブジェクトタイプID
　`im_authz_ipv4`

導入バージョン
　intra-mart Accel Platform 2013 Spring

連携エンティティ
　IPv4アドレス（imaz_ipv4_addr_patterns）

連携エンティティのキー

1. アドレスパターン（addr_pattern）

ID生成に要求する値

1. アドレスパターン（addr_pattern）

シリアライズされた文字列の書式

im_authz_ipv4:[IPv4アドレス]

例

im_authz_ipv4:192.168.0.1

テナント管理 / 認証 （im_authz_meta_subject）

サブジェクトタイプID
　im_authz_meta_subject

連携エンティティ
　なし

ID生成に要求する値

1. メタサブジェクトID（meta_subject_id）

メタサブジェクトIDには ゲストユーザを表す anonymous または 認証済みユーザを表す authenticated のみが使用できます。

シリアライズされた文字列の書式

im_authz_meta_subject:[メタサブジェクトID]

例

im_authz_meta_subject:anonymous

テナント管理 / 期間 （im_authz_term）

利用者のタイムゾーンにて期間内であるかを判定するサブジェクトです。 対象期間であるかの判断は以下の式で行われます。

- 開始日 ≦ 日付 ＜ 終了日

期間の終了日は認可設定画面上で確認した場合、前日での日付で表示されます。

サブジェクトタイプID
　im_authz_term

導入バージョン
　intra-mart Accel Platform 2013 Winter

連携エンティティ
　期間（imaz_term_sid）

連携エンティティのキー

1. 期間の開始日（yyyy-MM-dd の形式）
2. 期間の終了日（yyyy-MM-dd の形式）

ID生成に要求する値

1. 期間の開始日（yyyy-MM-dd の形式）
2. 期間の終了日（yyyy-MM-dd の形式）

シリアライズされた文字列の書式

im_authz_term:[期間の開始日] [期間の終了日]

例

im_authz_term:2010-01-01 2020-01-01

プロジェクトチーム機能 / プロジェクト（imprj_project）

サブジェクトタイプID
　imprj_project

連携エンティティ
　プロジェクトチーム（imprj_project およびその関連テーブル）
　IM共通マスタ 役職（imm_company_post およびその関連テーブル）

連携エンティティのキー

1. プロジェクトコード （project_cd）
2. 役職コード （post_cd）

ID生成に要求する値

1. プロジェクトコード （project_cd）
2. 役職コード （post_cd）
3. 比較演算子

比較演算子には次を指定してください。

演算子	説明
eq	指定の役職 を示します。

シリアライズされた文字列の書式

imprj_project:[プロジェクトコード] [役職コード] [比較演算子]

各コードの間は空白区切りです。
役職コードと比較演算子は省略可能です。

例

`imprj_project:sample_project leader eq`

予約されているリソースグループセット一覧

ここに定義されているリソースグループセットは intra-mart Accel Platform で予約されています。アプリケーション用にリソースグループセットを追加する場合はこれらと重複しないよう注意してください。

画面・処理

リソースグループセットID
　http-services

名前
　画面・処理

定義モジュール
　テナント管理機能（ルータ）

説明
　ルータで利用するURLに対する認可設定を定義するグループです。新しくアプリケーションを作成する際などに、新たなURLをルーティングテーブルに追加する際に、このリソースグループ配下にもリソースとして登録する必要があります。

HTTPサービス（設定不可）

リソースグループセットID
　http-services-unmanaged

名前
　HTTPサービス（設定不可）

定義モジュール
　テナント管理機能（ルータ）

説明
　このリソースグループセットに対してはポリシー部分編集定義がされていません。このため、ユーザはこのリソースグループセットに格納されるリソースの権限を編集できません。権限の変更をする必要のないリソースがここに格納されます。

Webサービス

リソースグループセットID
　web-services

名前
　Webサービス

定義モジュール
　WebService認証・認可モジュール

説明
　「画面・処理」同様に、Webサービスで使用するためのリソースはこのリソースグループセットに格納します。

メニューグループ

リソースグループセットID
　im-menu-group

名前
　メニューグループ

定義モジュール
　テナント管理機能（メニュー）

説明
　メニューグループに紐づくリソースを格納します。

IM共通マスタ 会社リソース

リソースグループセットID
　im-master

名前
　共通マスタ

定義モジュール
　IM共通マスタ

説明
　IM共通マスタの管理するリソースを格納します。

IM共通マスタ ユーザプロファイル

リソースグループセットID
　im-master-user-profile

名前
　ユーザプロファイル

定義モジュール

IM共通マスタ

説明

IM共通マスタのユーザプロファイルを管理するリソースを格納します。

IM共通マスタ 個人プロファイル

リソースグループセットID

im-master-user-self-profile

名前

自分のプロファイル

定義モジュール

IM共通マスタ

説明

IM共通マスタの個人プロファイルを管理するリソースを格納します。

ポータル

リソースグループセットID

im-portal-portal

名前

ポータル

定義モジュール

ポータル

説明

ポータルに紐づくリソースを格納します。

ポートレット

リソースグループセットID

im-portal-portet

名前

ポートレット

定義モジュール

ポータル

説明

ポートレットに紐づくリソースを格納します。

ポートレット編集モード

リソースグループセットID

im-portal-portet-editmode

名前

ポートレット編集モード

定義モジュール

ポータル

説明

ポートレット編集機能を表すリソースを格納します。

IMBox

リソースグループセットID

imbox-auth

名前

IMBox

定義モジュール
IMBox

説明
IMBoxの管理するリソースを格納します。

IM-LogicDesigner REST API

リソースグループセットID
im-logic-rest-api

名前
IM-LogicDesigner REST API

定義モジュール
IM-LogicDesigner

説明
フローをREST APIとして実行するためのルート情報を表すリソースを格納します。

設定ファイル一覧

ここでは認可機構が使用する設定ファイルの一覧を掲載しています。各設定の詳細についてはそれぞれ仕様書の該当箇所で説明していますので、そちらを参照してください。

項目

サブジェクトタイプ拡張

サブジェクトタイプの追加を行う際に記述する設定ファイルです。複数の設定を読み込みます。

詳細は「*サブジェクトタイプ*」を参照してください。

リソースタイプ拡張

リソースタイプの追加を行う際に記述する設定ファイルです。複数の設定を読み込みます。

詳細は「*リソースタイプとアクション*」を参照してください。

認可判断モジュール設定

使用する認可判断モジュールと、モジュールの結果をどう扱うかを設定ファイルに記載します。複数の設定を読み込みません。

詳細は「*認可判断機能の構造*」を参照してください。

ポリシー部分編集定義

認可設定画面の部品化や編集可能領域の定義を記述する設定ファイルです。既定のディレクトリ配下の複数のファイルを読み込みます。

詳細は「*部品化*」を参照してください。

サブジェクトリゾルバ（OnDemandSubjectResolver）拡張

リソース要求時にサブジェクト解決するサブジェクトリゾルバを追加するための設定ファイルです。複数の設定を読み込みます。

詳細は「*サブジェクトリゾルバ*」の該当項目を参照してください。

サブジェクトリゾルバ（DeclaredSubjectResolver）拡張

ログインユーザに対して明示されているサブジェクトを解決するためのサブジェクトリゾルバを追加する設定ファイルです。複数の設定を読み込みます。

詳細は「*サブジェクトリゾルバ*」の該当項目を参照してください。

ポリシーのキャッシュサイズの設定

ポリシー解釈器の実装クラスが提供するポリシーのキャッシュのルールを決定するための設定ファイルです。

詳細は「*キャッシュ設定*」の該当項目を参照してください。

ポリシーのキャッシュ対象となる条件の設定

ポリシー解釈機の実装クラスが提供するキャッシュ対象とするポリシーを決定するための設定ファイルです。複数の設定を読み込みます。

詳細は「*キャッシュ設定*」の該当項目を参照してください。

認可設定画面の設定

認可設定画面での挙動を変更するための設定ファイルです。

詳細は「*認可設定画面の設定*」の該当項目を参照してください。

認可のキャッシュ設定

認可機能は頻繁に使用されるため、キャッシュを使用して処理を高速化しています。ここでは、認可機能で使用しているキャッシュの箇所と、キャッシュするオブジェクトの単位、キャッシュサイズの計算式についての情報を記載しています。

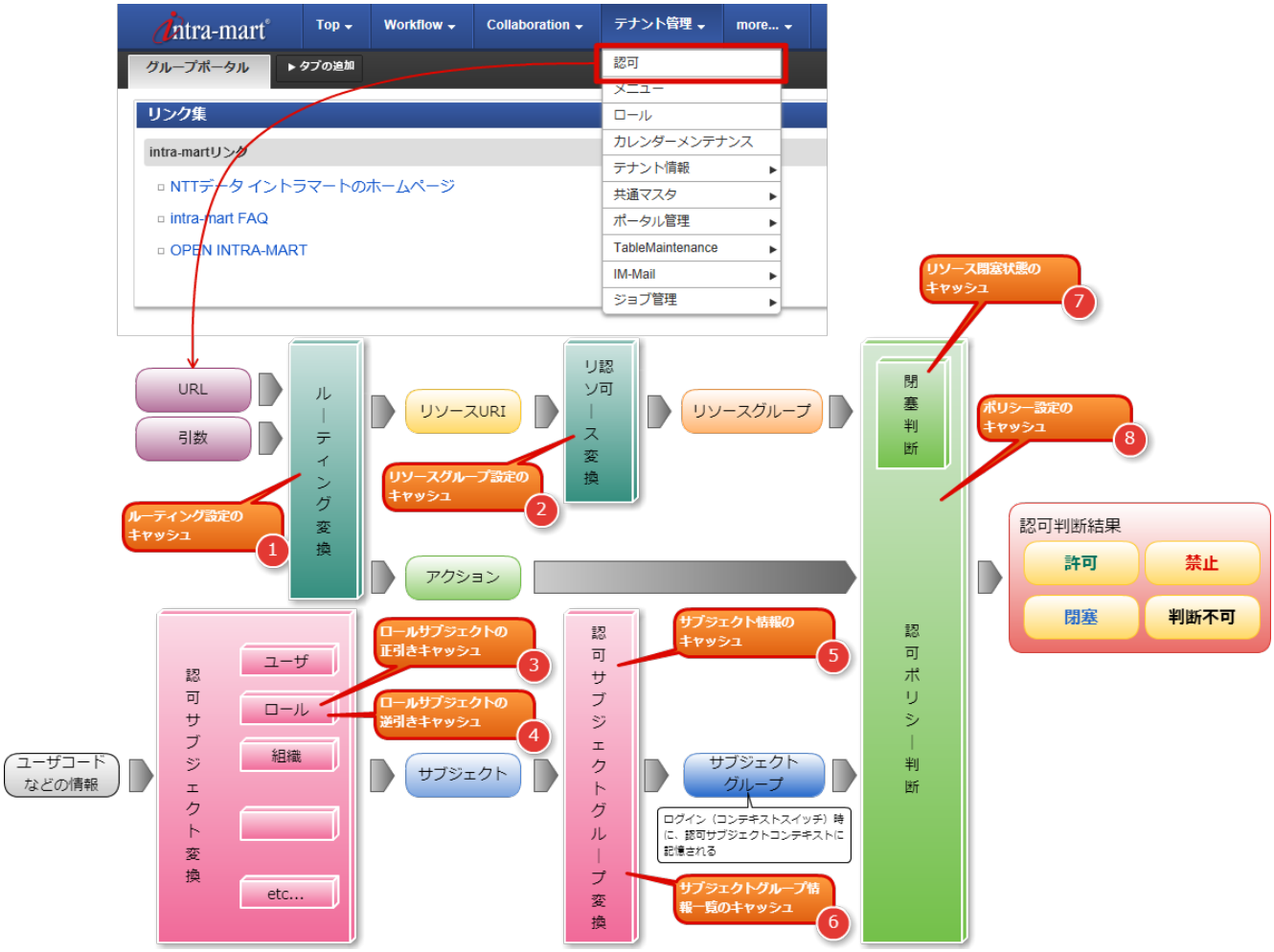
インストールするアプリケーションとサーバ環境に応じて、キャッシュを設定してください。

項目

キャッシュの概要

intra-mart Accel Platform の機能のうち、頻繁に利用される機能のひとつに認可があります。認可は画面やデータ、各サービスにおける権限チェックを統括して行うため、あらゆる機能から使用される機能です。

例えば、intra-mart Accel Platform にログインした後に表示されるグローバルナビや、各機能へのリンクを表示するサイトマップでは、アクセス権限のあるリンクのみを表示しているため、各リンクごとに認可の判断処理が行われます。メニューを例として、認可が行われるまでの処理の流れは下図の通りです。



このように、認可の判断結果が確定するまでには多くの処理があり、各処理においてファイルやデータベースへのアクセスが発生します。そこで認可機能では、各処理のうちデータの更新頻度が低く、参照頻度が高いものについて、キャッシュを利用して高速化しています。

これらの処理を高速化するために、以下についてキャッシュしています。

項番	名称	導入バージョン	備考
1	*ルーティング設定のキャッシュ*	2013 Spring(Climbing)	※
2	*リソースグループ設定のキャッシュ*	2013 Spring(Climbing)	
3	*ロールサブジェクトの正引きキャッシュ*	2015 Spring(Juno)	
4	*ロールサブジェクトの逆引きキャッシュ*	2015 Spring(Juno)	
5	*サブジェクト情報のキャッシュ*	2015 Spring(Juno)	
6	*サブジェクトグループ情報一覧のキャッシュ*	2015 Spring(Juno)	
7	*リソース閉塞状態のキャッシュ*	2013 Summer(Damask)	
8	*ポリシー設定のキャッシュ*	2012 Winter(Bourbon)	

※ メニュー側のキャッシュですが、便宜上この資料で説明します。

キャッシュされるデータのサイズは、認可に登録されているリソースグループの数や、サブジェクトグループの数に比例します。ユーザモジュールを追加でインストールしたり、取り扱う認可リソース数やサブジェクトグループの数が想定数より多くなったりする場合は、キャッシュサイズを見直してください。

キャッシュのサイズは intra-mart Accel Platform と代表的なアプリケーションをインストールした環境のためにサイジングした値をデフォルト値として設定しています。デフォルト値の説明については各キャッシュの計算式を参照してください。

> **コラム**
>
> 設定可能な値については「設定ファイルリファレンス キャッシュ設定」を参照してください。

> **コラム**
>
> キャッシュするオブジェクトの上限値に関して、以下の設定方法があります。
> 運用環境に応じて、どちらの設定を採用するか決めてください。
>
> - オブジェクトを格納する際の最大サイズ（max-bytes-XXXX）
> - キャッシュするオブジェクトの最大数（max-elements-on-XXXX）

ルーティング設定のキャッシュ

キャッシュ内容

メニューアイテムに登録されている「URL＋引数」を「リソースURI」「アクション」に変換するマッピングをキャッシュします。
このキャッシュは、概要図の 1 の部分に該当します。

キャッシュサイズは、「認可リソースマッピング情報キャッシュ設定 (im-ehcache-config/authz-mapped-entry-url.xml)」で設定されています。

> **コラム**
>
> このキャッシュは intra-mart Accel Platform 2013 Spring(Climbing) 以降で利用可能です。

キャッシュするオブジェクトの単位

ルーティング設定は、メニューアイテム単位でキャッシュされます。
そのため、キャッシュするオブジェクト数は、登録されているメニューアイテムの数によって見積もります。

キャッシュサイズの計算式

キャッシュが行われる対象データのおおまかなサイズは以下の計算式で求めることができます。

キャッシュサイズ ＝ ((1) ＋ (2)) × (3)

(1) ... 各メニューアイテムに設定されている内容のサイズ (平均 50byte)
　(1) ＝ (1 a) ＋ (1 b) × (1 c)
　(1 a) ... URLのバイト数
　(1 b) ... 引数(キー＋値) のバイト数
　(1 c) ... 引数の数
(2) ... リソースグループIDとアクションを格納したモデルのサイズ (平均 180byte)
　(2) ＝ (2 a) ＋ (2 b)
　(2 a) ... リソースグループIDのバイト数
　(2 b) ... アクションのバイト数
(3) ... 登録されているメニューアイテムの数

intra-mart Accel Platform のデフォルト値は以下の計算をもとに設定されています。

(50 ＋ 180) × 500 ＝ 115,000 (約 115KB)
⇒ 128KB

リソースグループ設定のキャッシュ

キャッシュ内容

「リソースURI」を「認可リソースグループ」に変換するマッピングをキャッシュします。
このキャッシュは、概要図の 2 の部分に該当します。

デフォルト状態でキャッシュ対象となっているリソースタイプは以下の通りです。
キャッシュ設定ファイル（%CONTEXT_PATH%/WEB-INF/conf/ からの相対パス）、および、導入バージョンもあわせて記載します。

リソースタイプ（リソースタイプID）	キャッシュ設定ファイル	導入バージョン

画面・処理（service）	im-ehcache-config/authz-resourcetype-service.xml	2013 Spring(Climbing)
会社一覧（im_master）	im-ehcache-config/authz-resourcetype-im_master.xml	2015 Winter(Lydia)
IMBox（imbox-auth）	im-ehcache-config/authz-resourcetype-imbox.xml	2013 Autumn(Eden)

> **コラム**
>
> このキャッシュは intra-mart Accel Platform 2013 Spring(Climbing) 以降で利用可能です。

キャッシュするオブジェクトの単位

リソースグループ設定は、リソースURI 単位でキャッシュされます。
キャッシュするオブジェクト数は、登録されているリソースURI の数によって見積もります。

キャッシュサイズの計算式

キャッシュが行われる対象データのおおまかなサイズは以下の計算式で求めることができます。

キャッシュサイズ ＝ ((1) ＋ (2)) × (3)

(1) ... リソースURI のバイト数 (平均 44byte)
(2) ... リソースグループID のバイト数 (平均 40byte)
(3) ... 登録されているリソースURI の数

intra-mart Accel Platform のデフォルト値は以下の計算をもとに設定されています。

画面・処理 (service) のサイズ（画面のリソースの数を1000個と想定しています。）
(44 ＋ 40) × 1,000 ＝ 84,000 (約 84KB)
⇒ 128KB
会社一覧 (im_master) のサイズ（IM-共通マスタの会社数を100個と想定しています。）
(44 ＋ 40) × 100 ＝ 8,4000 (約 8.4KB)
⇒ 16KB
IMBox（imbox-auth) のサイズ（IMBox実行権限の数。5権限で固定です。）
(44 ＋ 40) × 5 ＝ 420 (約 0.042KB)
⇒ 1KB

ロールサブジェクトの正引きキャッシュ

キャッシュ内容

サブジェクトIDをキーとしてロールサブジェクト情報をキャッシュします。
ロールサブジェクトIDから高速にロールサブジェクトを取得することが可能になります。
このキャッシュは、概要図の 3 の部分に該当します。

キャッシュサイズは、「認可サブジェクト元マスタ情報キャッシュ設定 (im-ehcache-config/authz-subject-sid-im-authz.xml)」で設定されています。

> **コラム**
>
> このキャッシュは intra-mart Accel Platform 2015 Spring(Juno) 以降で利用可能です。

キャッシュするオブジェクトの単位

対象者条件に指定した対象者のうち、ロール 1 件あたり 1 つとなります。
対象者は対象者条件を作成時に指定した対象者の数だけ作成されます。

キャッシュサイズの計算式

キャッシュが行われる対象データのおおまかなサイズは以下の計算式で求めることができます。

キャッシュサイズ ＝ (1) × (2)

(1) ... ロールサブジェクト情報のサイズ (平均 800byte)
(2) ... ロールサブジェクト数（対象者数）

intra-mart Accel Platform のデフォルト値は以下の計算をもとに設定されています。

800 × 100 ＝ 80,000 (約 78KB)
⇒ 128KB

ロールサブジェクトの逆引きキャッシュ

キャッシュ内容

ロールIDをキーとしてロールサブジェクト情報をキャッシュします。
ロールIDから高速にロールサブジェクトを取得することが可能になります。
このキャッシュは、概要図の 4 の部分に該当します。

キャッシュサイズは、「認可サブジェクト元マスタ情報キャッシュ設定 (im-ehcache-config/authz-subject-sid-im-authz.xml)」で設定されています。

> **コラム**
>
> このキャッシュは intra-mart Accel Platform 2015 Spring(Juno) 以降で利用可能です。

キャッシュするオブジェクトの単位

対象者条件に指定した対象者のうち、ロール 1 件あたり 1 つとなります。
対象者は対象者条件を作成時に指定した対象者の数だけ作成されます。

キャッシュサイズの計算式

キャッシュが行われる対象データのおおまかなサイズは以下の計算式で求めることができます。

キャッシュサイズ ＝ (1) × (2)

(1) ... ロールサブジェクト情報のサイズ (平均 800byte)
(2) ... ロール数

intra-mart Accel Platform のデフォルト値は以下の計算をもとに設定されています。

800 × 100 ＝ 80,000 (約 78KB)
⇒ 128KB

サブジェクト情報のキャッシュ

キャッシュ内容

認可の対象者に当たるサブジェクト情報をキャッシュします。
このキャッシュは、概要図の 5 の部分に該当します。

キャッシュサイズは、「認可サブジェクト情報キャッシュ設定 (im-ehcache-config/authz-subject.xml)」で設定されています。

> **コラム**
>
> このキャッシュは intra-mart Accel Platform 2015 Spring(Juno) 以降で利用可能です。

キャッシュするオブジェクトの単位

対象者につき 1 つとなります。
対象者は対象者条件を作成時に指定した対象者の数だけ作成されます。

キャッシュサイズの計算式

キャッシュが行われる対象データのおおまかなサイズは以下の計算式で求めることができます。

キャッシュサイズ ＝ (1) × (2)

(1) ... サブジェクト情報のサイズ (平均 800byte)
(2) ... サブジェクト数（対象者数）

intra-mart Accel Platform のデフォルト値は以下の計算をもとに設定されています。

800 × 1000 ＝ 800,000 (約 781KB)
⇒ 1MB

サブジェクトグループ情報一覧のキャッシュ

キャッシュ内容

認可の対象者 (サブジェクト) の集合から、条件に一致する対象者条件 (サブジェクトグループ) の集合をキャッシュします。
これにより、ログインユーザが持つサブジェクトグループ群がキャッシュされます。
このキャッシュは、概要図の 6 の部分に該当します。

キャッシュサイズは、「認可サブジェクトグループ情報キャッシュ設定 (im-ehcache-config/authz-subject-group.xml)」で設定されています。

> **コラム**
>
> このキャッシュは intra-mart Accel Platform 2015 Spring(Juno) 以降で利用可能です。

キャッシュするオブジェクトの単位

サブジェクトの集合ごとにキャッシュが生成されます。
ユーザがサブジェクト解決を行う際に、ユーザが持つ対象者の集合から割り出されてた対象者条件の集合情報がキャッシュされるため、おおよそ 1 ユーザごとにキャッシュが生成されます。
ただし、以下のように 1 ユーザごとに 1 つのキャッシュが生成されない場合もあります。

- 異なるユーザでありながら同一の集合を持つ場合は、同じキャッシュを利用します。
- 同じユーザであってもユーザに依存しないサブジェクト（例：IPv4 アドレスサブジェクト等）が利用される場合、状況・環境毎に対象者の集合が異なるため、複数のキャッシュが作成されます。

キャッシュサイズの計算式

キャッシュが行われる対象データのおおまかなサイズは以下の計算式で求めることができます。

キャッシュサイズ ＝ ((1) × (2) + (3)) × (4)

(1) ... サブジェクトグループIDのサイズ (平均200byte)
(2) ... 1 ユーザあたりの平均サブジェクトグループ数
(3) ... 集合情報 (平均500byte)
(4) ... サブジェクト集合の数（おおよそ、運用ユーザ数）

intra-mart Accel Platform のデフォルト値は以下の計算をもとに設定されています。

(200 × 20 ＋ 500) × 2000 ＝ 9000000 (約 8.6MB)
⇒ 9MB

リソース閉塞状態のキャッシュ

キャッシュ内容

「認可リソースグループ」1 件あたりの閉塞状態をキャッシュします。
このキャッシュは、概要図の 7 の部分に該当します。

このキャッシュはすべてのリソースグループに対して行われます。

キャッシュサイズは、「認可リソース閉塞情報キャッシュ設定 (im-ehcache-config/authz-resource-block.xml)」で設定されています。

> **コラム**
>
> このキャッシュは intra-mart Accel Platform 2013 Summer(Damask) 以降で利用可能です。

キャッシュするオブジェクトの単位

リソース閉塞状態は、リソースグループ単位でキャッシュされます。
そのため、キャッシュするオブジェクト数は、登録されているリソースグループの数によって見積もります。

キャッシュサイズの計算式

キャッシュが行われる対象データのおおまかなサイズは以下の計算式で求めることができます。

キャッシュサイズ ＝ (1) × (2)

(1) ... リソースグループID、閉塞状態のセット (平均 320byte)
(2) ... 登録されているリソースグループの数

intra-mart Accel Platform のデフォルト値は以下の計算をもとに設定されています。

320 × 1,000 ＝ 320,000 (約 312.5KB)
⇒ 320KB

ポリシー設定のキャッシュ

キャッシュ内容

「認可リソースグループ」「認可サブジェクトグループ」「アクション」から取得した「ポリシー」の状態をキャッシュします。
このキャッシュは、概要図の 8 の部分に該当します。

デフォルト状態でキャッシュ対象となっているリソースタイプは以下の通りです。
導入バージョンもあわせて記載します。

リソースタイプ（リソースタイプID）	導入バージョン
画面・処理（service）	2012 Winter(Bourbon)
会社一覧（im_master）	2015 Winter(Lydia)

キャッシュサイズは、「認可ポリシー情報キャッシュ設定 (im-ehcache-config/authz-policy.xml)」で設定されています。

他のリソースタイプでキャッシュを行う場合は、別途設定が必要です。詳しくは、「設定ファイルリファレンス」の「認可ポリシーキャッシュ対象設定」を参照してください。

コラム

このキャッシュは intra-mart Accel Platform 2012 Winter(Bourbon) 以降で利用可能です。

intra-mart Accel Platform 2013 Autumn(Eden) 以降から、キャッシュするオブジェクトの単位が変更されています。
これに伴って、キャッシュサイズの計算式も変更されています。

intra-mart Accel Platform 2015 Winter(Lydia) 以降から、インストールした直後の状態で 会社一覧（im_master）もキャッシュされるよう変更されています。
これに伴って、キャッシュサイズのデフォルト値が変更されています。

キャッシュするオブジェクトの単位

intra-mart Accel Platform 2013 Autumn(Eden) 以降の場合

ポリシー設定は、単一の「サブジェクトグループ」に対しての「リソースグループ、リソースタイプ、リソースタイプが指すアクション」毎のポリシー設定をキャッシュします。キャッシュするオブジェクト数は、以下の計算式で求めることができます。

キャッシュするオブジェクト数 ＝ (１) × (２) × (３) × (４)

(１) ... 登録されているリソースグループの数
(２) ... リソースグループがリソースとして持つリソースタイプの数
(３) ... リソースタイプが定義するアクションの数
(４) ... 登録されているサブジェクトグループの数

intra-mart Accel Platform 2013 Autumn(Eden) より前の場合

ポリシー設定は、サブジェクトグループ単位でキャッシュされます。
キャッシュするオブジェクト数は、登録されているサブジェクトグループの数によって見積もります。

キャッシュサイズの計算式

キャッシュが行われる対象データのおおまかなサイズは、リソースタイプ毎に以下の計算式で求めることができます。

intra-mart Accel Platform 2013 Autumn(Eden) 以降の場合

キャッシュサイズ ＝ (１) × (２) × (３) × (４)

(１) ... 登録されているサブジェクトグループの数
(２) ... サブジェクトグループID、リソースグループID、リソースタイプ、アクションのセット(約 170byte)
(３) ... 登録されているリソースグループの数
(４) ... ポリシー設定数の係数 (認可設定マトリクス全体のうち、運用中キャッシュされる割合)

intra-mart Accel Platform 2013 Autumn(Eden) より前の場合

キャッシュサイズ ＝ (１) × ((２) × (３) ＋ (４)) × (５)

(１) ... 登録されているサブジェクトグループの数
(２) ... リソースグループID、リソースタイプ、アクションのセット (平均 600byte)
(３) ... 登録されているリソースグループの数
(４) ... (２)(３)を保持するためのマップの初期サイズ (約 2000byte)
(５) ... ポリシー設定数の係数 (認可設定マトリクス全体のうち、運用中キャッシュされる割合)

デフォルト値は以下の計算をもとに設定されています。

intra-mart Accel Platform 2015 Winter(Lydia) 以降の場合

画面・処理 (service) のサイズ

2,200 × 170 × 1,000 × 0.5 = 187,000,000
会社一覧 (im_master) のサイズ
2,200 × 170 × 100 × 1 = 37,400,000
ポリシー設定のサイズ
187,000,000 + 37,400,000 = 224,400,000 (214MB)
⇒ 220MB

intra-mart Accel Platform 2013 Autumn(Eden) ～ 2015 Summer(Karen) の場合

ポリシー設定のサイズ
2,200 × 170 × 1,000 × 0.5 = 187,000,000 (約 178.3MB)
⇒ 180MB

intra-mart Accel Platform 2013 Autumn(Eden) より前の場合

ポリシー設定のサイズ
2,200 × (150 × 1,000 + 2,000) × 0.5 = 167,200,000 (約 159.5MB)
⇒ 160MB

コラム
ポリシー設定数の係数について
リソースタイプ「メニュー (im-menu-group)」に対して権限が無い場合には、その配下のメニュー項目へ権限確認を行う必要が無いため、リソースタイプ「画面・処理 (service)」への認可設定がすべてキャッシュされるわけではありません。
この係数は、このようなキャッシュが利用されないポリシー設定を考慮するために導入されています。

この係数は、リソースタイプ「会社一覧 (im_master)」のように、すべてに対して権限確認を行う可能性がある場合は 1 になります。

運用時のTips

項目

任意のタイミングで機能へのアクションを制限する

目的

一定期間の間、任意の機能（リソース）に対してのユーザからのアクセス（アクション）を遮断します。

例えば、以下のようなケースを想定しています。

- 対象となる機能の計画メンテナンス
- 対象となる機能内で利用する連携先のサービスが、特定の期間の間、停止することが判明している
- 対象となる機能の利用を特定の時間で終了させる

実現方法

「*閉塞*」とジョブスケジューラ機能を利用して実現します。
特定のリソースの閉塞を行うジョブ実行プログラムを作成し、ジョブスケジューラ機能を使用して、作成したジョブ実行プログラムをスケジューリングすることで、特定の期間から機能を閉塞することが可能です。閉塞の解除も同様に行います。

プログラムによる閉塞の実装方法については、「*閉塞の管理* 」を参照してください。
ジョブスケジューラ機能については、「ジョブスケジューラ仕様書 」を参照してください。

任意のタイミングでポリシーの設定状態を更新する

目的

あらかじめ認可ポリシーの権限設定を決めておき、ある時刻で権限設定を適用します。

例えば、以下のようなケースを想定しています。

- 権限設定の変更を、業務時間外（例：夜間）に自動で行う
- 権限設定の簡易的な期間化を行う

実現方法

「IM-Authz（認可）Excelインポート・エクスポート」モジュールとジョブスケジューラ機能を利用して実現します。
あらかじめ、IM-Jugglingで「追加機能」 - 「IM-Authz（認可）Excelインポート・エクスポート」を追加したwarをアプリケーションサーバにデプロイします（Excelインポート・エクスポートが使用できるようになります）。

1. 認可設定画面、または、ジョブスケジューラを使用して、あらかじめ現時点での認可設定を、Excel（xlsx）形式でエクスポートします。
2. Excel（xlsx）ファイルを表計算ソフトで開き、権限設定を変更して内容を確認します。
   （Excel（xlsx）ファイル上であれば、インポートを行わない限り、権限設定が即時反映されることはありません。）
3. 変更内容を保存したExcel（xlsx）ファイルをパブリックストレージ上に保存しておき、ジョブスケジューラで指定した時間に「認可インポート（Excel）」ジョブネットを実行するよう設定します。

ジョブパラメータがデフォルト状態の場合、インポートファイルを作成してから実際にインポートされるまでの間に、インポートファイルに定義しているリソースグループ・リソースが削除された場合は、インポート時にエラーとなり処理が中断されます。
サブジェクトグループが削除された場合は、削除分に対するポリシー設定はインポートされません（無視されます）。

これらの動作はジョブパラメータ「リソースグループ・リソース存在検証フラグ」・「サブジェクト存在検証フラグ」によって変更できます。

- データ系のリソースなど、運用中に削除されることが想定されているリソースがインポートファイルに含まれることがわかっている場合、「リソースグループ・リソース存在検証フラグ」を `false` に設定することで、インポート時のエラー発生を回避できます。
- インポートファイルに必ず権限を設定したい対象者条件が含まれている場合、「サブジェクト存在検証フラグ」を `true` に設定することで、想定していないポリシーがインポートされることを防ぎます。

ジョブパラメータの詳細は、「IM-Authz（認可）インポート・エクスポート仕様書 - ファイルフォーマット 」を参照してください。
ジョブスケジューラ機能については、「ジョブスケジューラ仕様書 」を参照してください。