目次

改訂情報

変更年月日	変更内容
2014-09-01	初版
2014-12-01	第2版　下記を追加・変更しました ■ 「*SSO（シングルサインオン）*」の「*SSOユーザコードプロバイダの実装*」の記述を修正
2015-12-01	第3版　下記を追加しました ■ 「*一般ユーザのログイン処理フロー*」の「*認証API*」-「*ログイン*」に以下を追記 　■ ログイン不可となる条件 　■ ログイン不可と判定された場合に、実行されない認証プロバイダ・プラグインの種類 ■ 「*一般ユーザの強制ログイン*」の記述を追加
2016-04-01	第4版　下記を追加・変更しました ■ 「*一般ユーザの認証プロバイダ・プラグイン*」の「*認証プロバイダ・プラグインの詳細*」の以下のプラグインについて、パスワード保存方式を「ハッシュ化」を利用している場合の引数の仕様について追記 　■ 「*認証プロバイダ*」 　■ 「*認証条件判定プロバイダ*」 　■ 「*認証リスナ*」 ■ 「*一般ユーザの認証プロバイダ・プラグイン*」の「*認証プロバイダ・プラグインの詳細*」のサンプルについて、アカウントパスワードの照合を行う内容に変更
2018-08-01	第5版　下記を追加しました ■ 「*セッションタイムアウト*」-「*セッションタイムアウトの検出*」の「システム管理者フラグ」の用途について、 2018 Summer(Tiffany) 以降での動作を追記

はじめに

本書の目的

本書では、認証機能の仕様について説明します。

説明範囲は以下のとおりです。

- 認証機能の利用方法
- 設定の変更方法
- 認証機能を利用したアプリケーションの開発のための情報や注意点

対象読者

本書では次の開発者を対象としています。

- 認証機能の仕組みを理解したい。
- 認証機能を利用した実装をしたい。

次の内容を理解していることが必須です。

- JavaEE を利用したWebアプリケーションの基礎
- アクセスコンテキストについて

  ログイン・ログアウト処理はアクセスコンテキストの切り替え機能により実現されています。
  あわせて 「アクセスコンテキスト仕様書」 を参照してください。

対象開発モデル

本書では以下の開発モデルを対象としています。

- JavaEE開発モデル
- スクリプト開発モデル

  なお、スクリプト開発モデルでは、一部の認証機能の実装はできません。

本書の構成

- *概要*

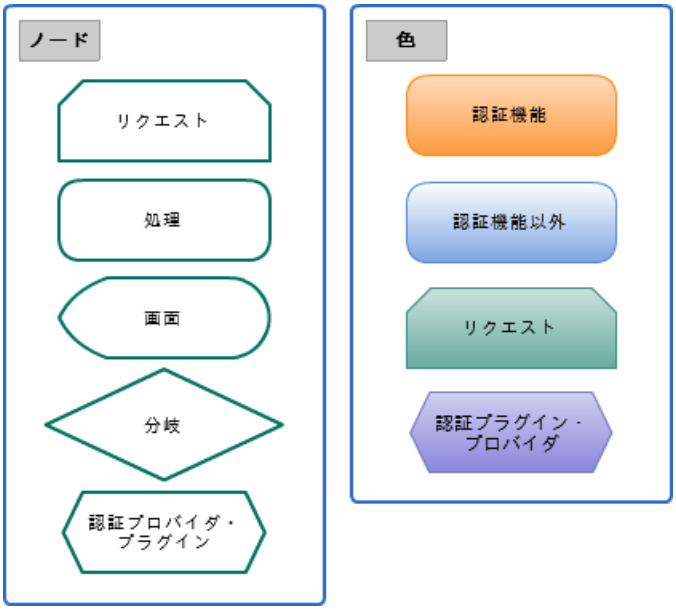
  認証機能の用途、および、提供する機能の概要について説明します。

- *各機能の詳細*

  認証機能が提供する各機能の仕組みについて説明します。

図の凡例

このドキュメントにおけるフローチャートの凡例は以下の通りです。


ノード
リクエスト
処理
画面
分岐
認証プロバイダ・プラグイン
色
認証機能
認証機能以外
リクエスト
認証プラグイン・プロバイダ

概要

intra-mart Accel Platform における認証機能の定義、および、認証機能が扱う範囲について説明します。

認証機能とは

intra-mart Accel Platform における認証機能とは、「認証」に関する様々な機能の総称です。

「認証」とは、ユーザの正当性を検証する処理です。
ユーザのみが知るパスワードなどのユーザを識別する情報（識別情報）を利用して、ユーザの正当性を確認します。

認証機能が提供する機能は、以下の通りです。

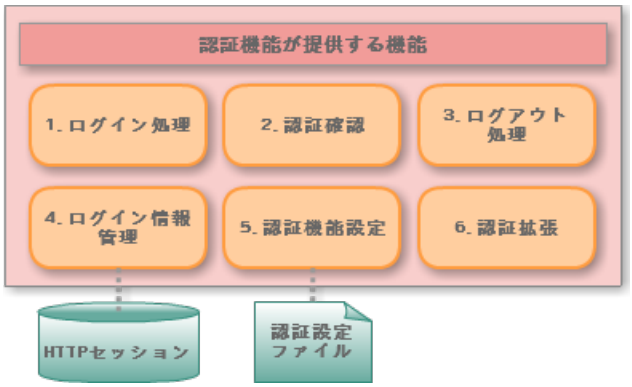


図 認証機能が提供する機能

1. **ログイン処理**
   ログイン画面の表示と認証の実行を行い、認証結果に基いてログインセッションを開始します。
   HTTPセッションを初期化し、認証したユーザに関する情報をHTTPセッションに保存し、認証したユーザとしてアクセス可能な状態にします。

2. **認証確認**
   指定した画面で「認証確認」画面を表示して、ログインユーザに認証を求めます。
   この機能を使うことで、決済処理などのような業務的に重要な処理において、セキュリティを高めることができます。

3. **ログアウト処理**
   ログインセッションを終了します。
   HTTPセッションを初期化し、ログイン中にHTTPセッション等に保持していたユーザに関する情報を破棄します。

4. **ログイン情報管理**
   ユーザのログイン状態の管理、および、検証を行います。
   HTTPセッションの有効性と Cookie に保存したログイン情報の有効性を検証し、無効となった場合はエラーとします。

5. **認証機能設定**
   認証機能に関する設定情報です。
   認証設定ファイル等で、認証方式の変更や認証機能における動作を設定します。

6. **認証拡張**
   ログイン・ログアウトなどの基本機能以外で、認証を補助する目的の機能群です。
   これらの機能が不要な場合は、セットアップ時に選択しないことで機能を無効にすることができます。
   以下のような機能が該当します。

   - パスワード履歴管理

     パスワードの履歴管理、パスワードの入力文字制限、および、パスワードを定期的に変更する機能を提供します。
     パスワード履歴管理機能については、「 設定ファイルリファレンス 」 - 「パスワード履歴管理設定 」 を参照してくださ

い。

- パスワードリマインダ

  パスワードを紛失した場合に、ユーザによりパスワードを再設定する機能を提供します。
  パスワードリマインダについては、「 一般ユーザ操作ガイド 」 - 「 パスワードを忘れた場合 」 を参照してください。
  パスワードリマインダの設定については、「 テナント管理者操作ガイド 」 - 「 パスワードリマインダを設定する 」 を参照してください。

- 二重ログイン防止

  既にログインしているユーザの、別のブラウザからのログインを防止する機能を提供します。
  二重ログイン防止機能については、「 認証プログラミングガイド 」 - 「 二重ログイン防止機能 」 を参照してください。

> **注意**
>
> 認証機能は、Webアクセス時の認証に関する機能を提供します。
> ジョブスケジューラなどのWebアクセス以外での認証はサポートしていません。

認証機能における処理の流れ

認証機能は、Web環境でのアクセス時に利用されます。
クライアントからのリクエスト時に認証機能がどのように利用されるかを、リクエストごとの処理の流れ（リクエストフロー）で説明します。
主なリクエストフローは、以下の通りです。

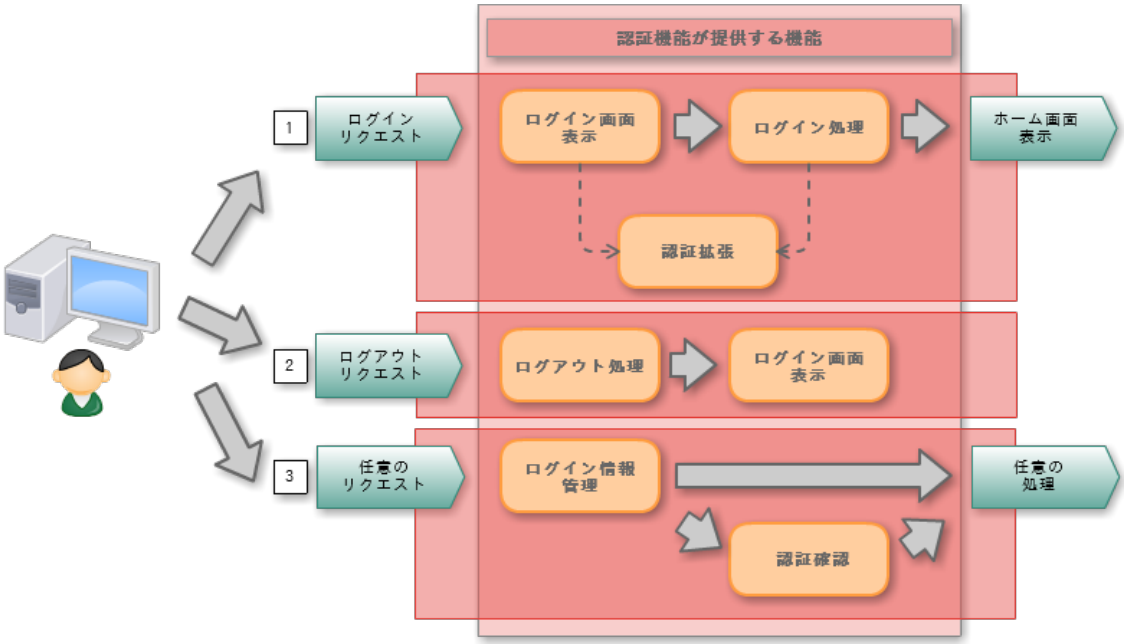


図 Webアクセスにおける認証機能のリクエストフロー

各リクエストフローの処理の概要は以下の通りです。

1. **ログインリクエスト**
   ログイン画面へのリクエストにより、「ログイン画面表示」を行います。
   ログイン画面では、ユーザコードとパスワードなどの識別情報を入力することで「ログイン処理」が実行され、ログイン状態となって、業務画面へ遷移します。
   ログインリクエストは、一般ユーザとシステム管理者に分けられます。
   「認証拡張」機能が有効な場合、必要に応じて「認証拡張」機能が利用されます。
   図では「ログイン画面表示処理」「ログイン処理」から参照されるイメージで記述されていますが、利用されるタイミングは「認証拡張」機能ごとに異なります。
   例えば、「パスワード履歴管理」機能は、「ログイン処理」の実行後に呼び出されます。

2. **ログアウトリクエスト**
   ログアウトリクエストでは、「ログアウト処理」が実行され、ログアウト後にログイン画面に遷移します。

3. **任意のリクエスト**
   全てのリクエストに対して、「ログイン情報管理」機能により、認証状況の確認が行われます。
   また、必要に応じて「認証確認」が実行されます。

それぞれのリクエストにおける認証機能の役割については、「 *認証機能のリクエストフロー* 」を参照してください。

制限事項

認証機能の制限事項については 「 リリースノート 」 - 「 制限事項 」 - 「 認証機能 」 を参照してください。

各機能の詳細

認証機能を構成する主な機能について説明します。

認証機能のリクエストフロー

Webアクセス時のリクエストフローにおける認証機能について説明します。
「 *概要* 」 - 「 *認証機能における処理の流れ* 」にて挙げた、認証機能に関連するリクエストフローの詳細を順に説明します。

項目

一般ユーザのログインリクエスト

一般ユーザがログインする場合は、「ログイン画面」にアクセスし、ログインを実行します。

一般ユーザのログイン処理におけるリクエストフローは、以下の通りです。

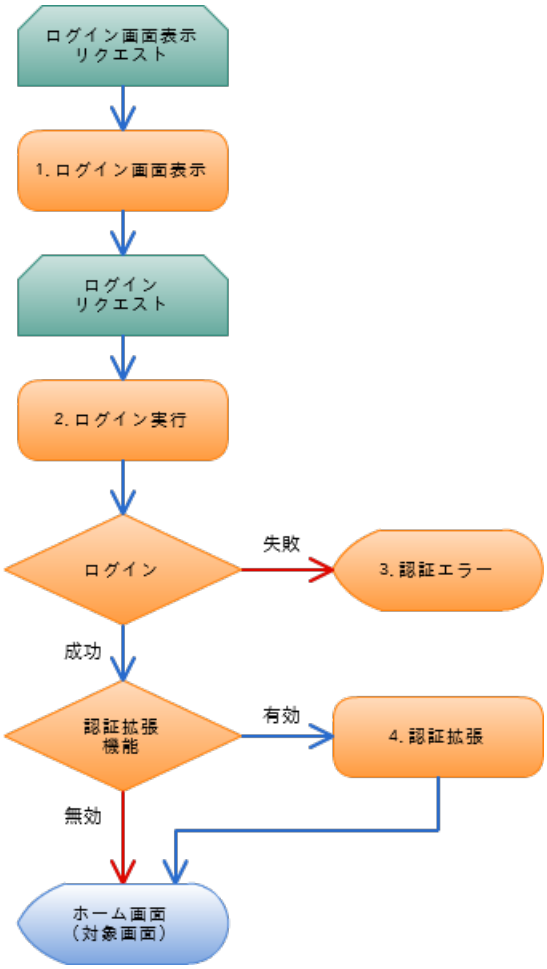


図 一般ユーザのログインリクエストフロー

1. ログイン画面表示
   「ログイン」画面を表示します。

2. ログイン実行
   ログインを実行します。

3. 認証エラー

認証に失敗した場合、エラーの内容に従って、「認証エラー」画面を表示します。

4. 認証拡張
   認証拡張機能が有効な場合、認証拡張機能ごとに処理を実行します。
   図では「ログイン実行」の後に実行されていますが、これは「パスワード履歴管理」機能のパスワード履歴チェックをイメージした記述を行っています。
   実行されるタイミングは「認証拡張」機能ごとに異なります。

このフローにおける処理の詳細は、「 *一般ユーザのログイン処理フロー* 」を参照してください。

システム管理者のログインリクエスト

システム管理者がログインする場合のリクエストフローは、一般ユーザとほぼ同じです。
ただし、システム管理者用の認証拡張機能はないため、認証拡張機能は実行されません。

システム管理者のログイン処理におけるリクエストフローは、以下の通りです。

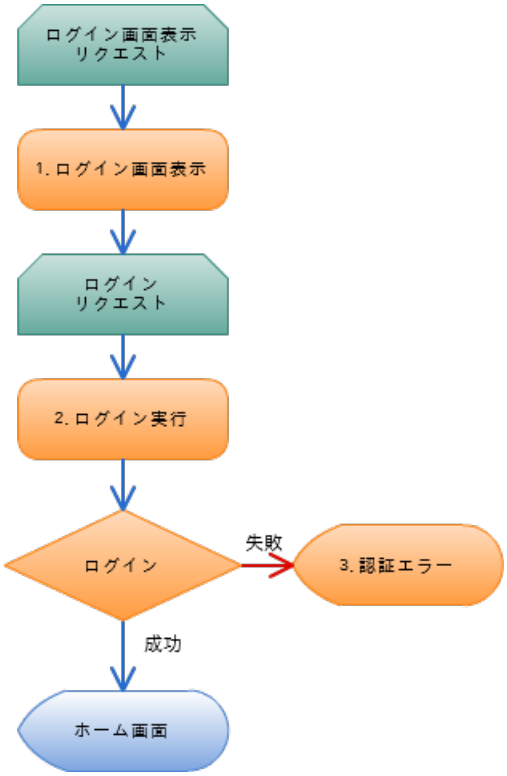


図 システム管理者のログインリクエストフロー

このフローにおける処理の詳細は、「 *システム管理者のログイン処理フロー* 」を参照してください。

ログアウトリクエスト

一般ユーザ、および、システム管理者が、ログアウトURLにアクセスし、ログアウトを実行します。

ログアウト実行時のリクエストフローは、以下の通りです。

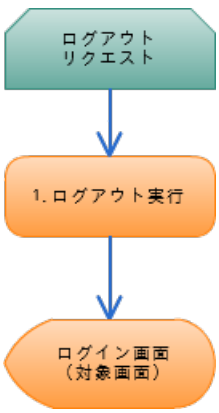


図 ログアウトリクエストフロー

1. ログアウト実行
   ログアウト処理を実行し、「ログイン画面」にリダイレクトします。
   ログアウト処理についての詳細は、「 *ログアウト* 」を参照してください。

任意のリクエスト

intra-mart Accel Platform の全てのリクエストに対して、「ログイン情報管理」機能により、認証状況の確認として、「セッションチェック」が行われます。
また、必要に応じて「認証確認」が実行されます。

全てのリクエストにおける認証に関連する処理の流れは、以下の通りです。

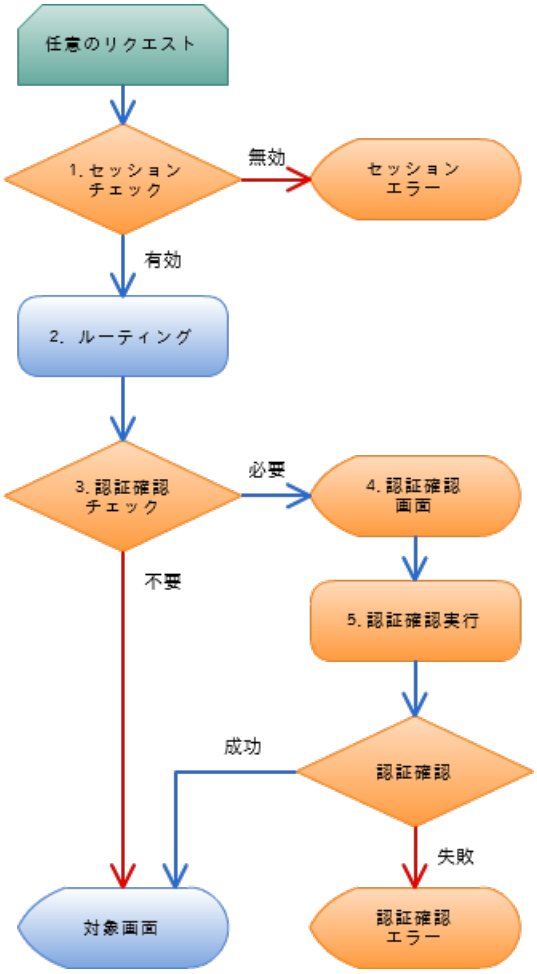


図 任意のリクエストのリクエストフロー

1. セッションチェック
   セッションチェックは、セッションの有効性を確認する処理です。
   セッションチェックで無効と判定された場合は、「セッションタイムアウトエラー」画面が表示されます。
   セッションチェックの処理については、「 *セッションタイムアウト* 」を参照してください。

2. ルーティング
   ルーティングでは、リクエストURLをもとにした処理の振り分けが行われます。
   ルーティング機能については、「スクリプト開発モデル プログラミングガイド」 - 「ルーティング」を参照してください。

3. 認証確認チェック
   認証確認が必要なリクエストかどうかを確認する処理です。

4. 認証確認画面
   認証確認が必要なリクエストの場合、「認証確認」画面を表示します。

5. 認証確認実行
   「認証確認画面」でパスワードなどの識別情報を入力することで、認証を実行し、 認証に成功した場合は、認証確認前に表示しようとした画面に遷移します。

認証確認に関する処理については、「 *認証確認* 」を参照してください。

一般ユーザのログイン処理

ここでは、一般ユーザのログイン処理について説明します。
一般ユーザのログイン処理とは、「ログイン」画面を表示し、ログインを実行してから「業務」画面を表示するまでの一連の処理を指します。

一般ユーザのログイン処理について、以下の順に説明します。

- *一般ユーザのログイン処理フロー*

  一般ユーザのログイン処理の流れを説明します。

- *一般ユーザの認証プロバイダ・プラグイン*

  一般ユーザのログイン処理で利用される認証プロバイダ・プラグインについて説明します。

- *一般ユーザのログイン画面*

  一般ユーザのログイン画面の機能について説明します。

一般ユーザのログイン処理フロー

一般ユーザのログイン処理の流れについて説明します。

一般ユーザのログイン処理フローの概要

ログイン処理では、ユーザが入力した情報などを利用した認証処理を行います。

認証処理は、認証APIを呼び出すことで実行されます。
認証APIでは、いくつかの認証プロバイダ・プラグインを呼び出すことで認証処理を実現しています。

認証APIと認証プロバイダ・プラグインについては、以下で説明します。

- *認証API*
- *一般ユーザの認証プロバイダ・プラグイン*

ログイン処理における、認証処理の流れは以下の通りです。

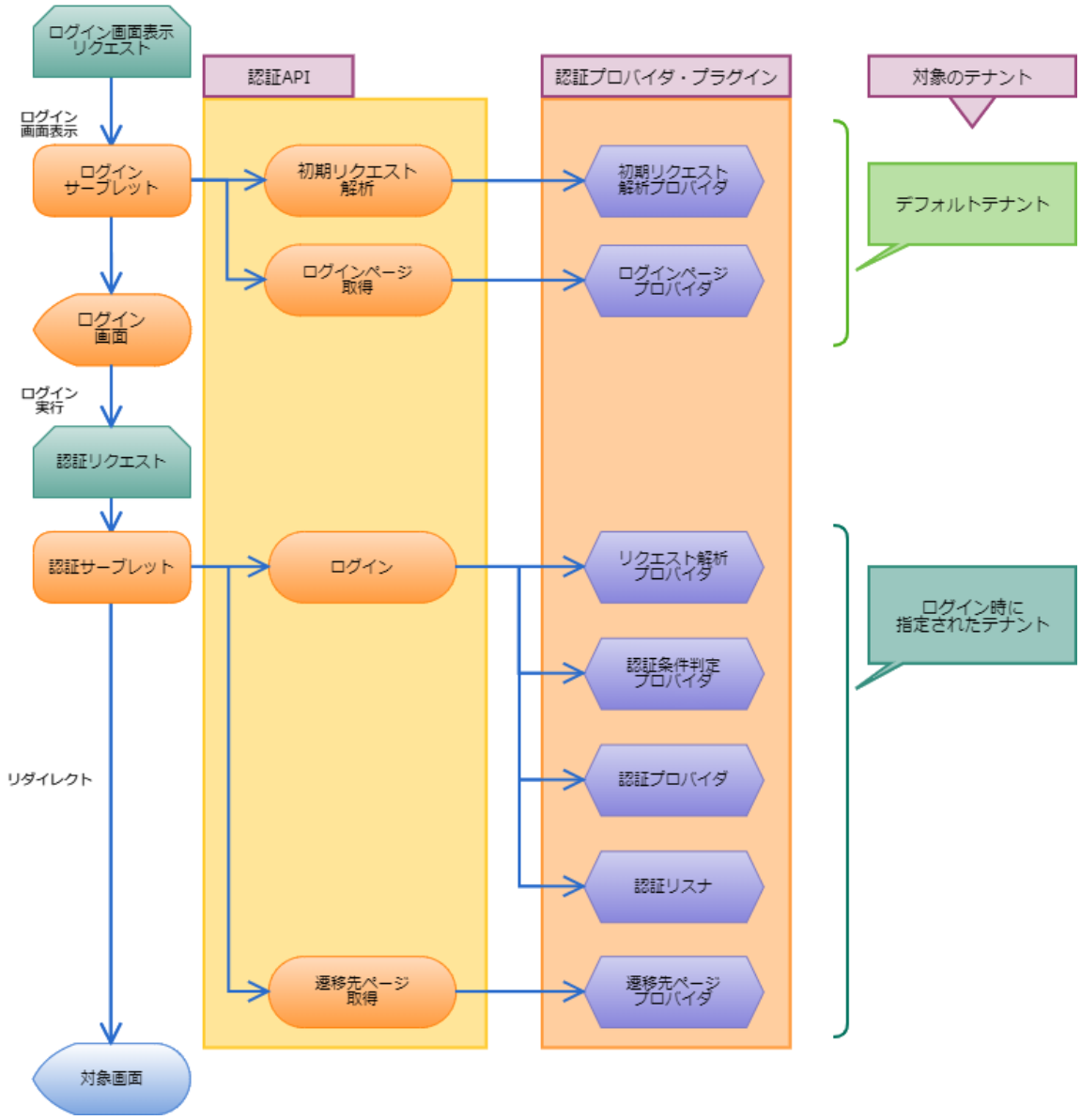


図 一般ユーザのログイン処理フロー

処理対象のテナントについて

バーチャルテナントによる複数テナントを利用している場合、処理する認証プロバイダ・プラグインによって、対象となるテナントが異なります。
図「一般ユーザのログイン処理フロー」の「対象のテナント」を参照してください。

ただし、リクエスト情報を利用したテナント自動解決機能を有効にした場合、常に自動解決されたテナントが対象です。
リクエスト情報を利用したテナント自動解決機能の詳細は「intra-mart Accel Platform セットアップガイド」 - 「テナント解決機能」 - 「リクエスト情報を利用したテナント自動解決機能について」 を参照してください。

認証API

ログイン処理で実行される認証APIは、以下のクラスです。

`jp.co.intra_mart.foundation.security.certification.UserCertificationManager`

この認証APIは、ログイン処理中に以下の処理を行います。

1. 初期リクエスト解析 （ `initialParseRequest()` ）
2. ログインページ取得 （ `getLoginPageUrl()` ）
3. ログイン （ `login()` ）
4. 遷移先ページ取得 （ `getTargetPageUrl()` ）

API の詳細については、「UserCertificationManager クラスの APIドキュメント 」を参照してください。

各処理の流れを説明します。

初期リクエスト解析

「 *初期リクエスト解析プロバイダ* 」を呼び出します。

ログインページ取得

「 *ログインページプロバイダ* 」を呼び出します。

ログイン

以下の処理を順に実行します。

1. 「 *リクエスト解析プロバイダ* 」を呼び出します。
2. アカウント情報を取得して、ログイン可否を確認します。
3. 「 *認証条件判定プロバイダ* 」を利用して、「 *認証プロバイダ* 」を決定します。
4. 「 *認証プロバイダ* 」を呼び出し、認証を行います。
5. アカウントコンテキストを切り替えてログイン状態にします。
6. 「 *認証リスナ* 」を呼び出します。

> **注意**
>
> 上記の「2. アカウント情報を取得して、ログイン可否を確認します」では、以下のいずれかの条件を満たす場合にログイン不可と判定します。
>
> - アカウントが存在しない
> - アカウントライセンスが登録されていない
> - アカウントのタイムゾーンにおける現在時刻が、アカウントの有効期間から外れている
> - アカウントがロックされている
>
> ログイン不可と判定された場合、「3.」以降の処理は実行されません。具体的には、以下の認証プロバイダ・プラグインは実行されません。
>
> - 「 *認証条件判定プロバイダ* 」
> - 「 *認証プロバイダ* 」
> - 「 *認証リスナ* 」

遷移先ページ取得

「 *遷移先ページプロバイダ* 」を呼び出します。
遷移先が取得できなかった場合、アカウントコンテキストのホームURLプロパティを取得します。

一般ユーザの認証プロバイダ・プラグイン

「 *一般ユーザのログイン処理フロー* 」で説明した、一般ユーザのログイン処理で利用する認証プロバイダ・プラグインと、 intra-mart Accel Platform の標準で提供されるプロバイダ（以下、標準プロバイダ）の動作について説明します。

項目

一般ユーザの認証プロバイダ・プラグインの概要

認証機能の主な処理を行う個別の拡張可能なプログラムを、「認証プロバイダ・プラグイン」と呼びます。

認証プロバイダ・プラグインはプラグイン化されており、新しいプラグインを追加することで、認証機能の動作を変更できます。
これにより、要件に合わせて認証方式を変更することが可能です。

認証プロバイダ・プラグインは一般ユーザ用とシステム管理者用が用意されており、別々の認証方式を利用できます。

> **注意**
>
> 認証プロバイダ・プラグインは、システムで共通で利用されます。
> そのため、バーチャルテナントによる複数テナントを利用する場合、全てのテナントで同じプロバイダが利用されます。
> もし、テナントごとに異なる認証方式を利用したい場合、プロバイダ内でテナントの判定を行って処理を行うプロバイダを作成する必要があります。

以下に、一般ユーザ用の認証プロバイダ・プラグインの詳細について説明します。

認証プロバイダ・プラグイン一覧

認証機能で提供される一般ユーザ用の認証プロバイダ・プラグインは、以下の通りです。
「対象のテナント」は、リクエスト情報を利用したテナント自動解決機能を利用しない場合の処理対象です。
リクエスト情報を利用したテナント自動解決機能の詳細は「intra-mart Accel Platform セットアップガイド」 - 「テナント解決機能」 - 「リクエスト情報を利用したテナント自動解決機能について」 を参照してください。

表 一般ユーザ用認証プロバイダ・プラグイン一覧

プロバイダ名	概要	拡張ポイント	対象のテナント
初期リクエスト解析プロバイダ	ログイン画面表示時に送信されたリクエスト情報を解析して、ログイン情報を設定します。	jp.co.intra_mart.security.user.initial_request_analyzer	デフォルトテナント
ログインページプロバイダ	ログイン画面用のページパス情報を取得します。	jp.co.intra_mart.security.user.login_page	デフォルトテナント
リクエスト解析プロバイダ	ログイン実行時に送信されたリクエスト情報を解析して、ログイン情報を設定します。	jp.co.intra_mart.security.user.login_request_analyzer	ログイン時に指定されたテナント

プロバイダ名	概要	拡張ポイント	対象のテナント
認証条件判定プロバイダ	認証プロバイダを利用するための条件判定を行います。	（認証プロバイダの追加情報として定義）	ログイン時に指定されたテナント
認証プロバイダ	ログイン情報などを利用して、実際の認証処理を実行します。	jp.co.intra_mart.security.user.certification	ログイン時に指定されたテナント
認証リスナ	認証処理の後処理を実行します。	（認証プロバイダの追加情報として定義）	ログイン時に指定されたテナント
遷移先ページプロバイダ	ログイン成功時に遷移する画面のページパス情報を取得します。	jp.co.intra_mart.security.user.target_page	ログイン時に指定されたテナント

認証プロバイダ・プラグインは、intra-mart Accel Platform のプラグイン機構として設定され、PluginManager により読み込まれます。
各プラグインには設定情報のキーとして、「拡張ポイント」が定義されています。
また、同じ拡張ポイントに複数のプラグインを設定でき、rank 属性を設定することにより、 プラグインの読み込み順序を制御することが可能です。

プラグインの仕様については、 「 PluginManagerのAPIドキュメント 」 を参照してください。

認証プロバイダ・プラグインは、それぞれ複数設定することが可能です。
複数設定されている場合の動作については、各認証プロバイダ・プラグインの項目を参照してください。
新しい認証プロバイダ・プラグインを作成して場合は、適切な順序で実行されるように設定を行ってください。

認証プロバイダ・プラグインの詳細

各プロバイダの詳細と実装方法について以下の順に説明します。

1. インタフェース定義
   プロバイダのインタフェース
2. 標準プロバイダ
   intra-mart Accel Platform が提供する標準プロバイダ
3. 実装サンプル
   プロバイダの実装サンプル

初期リクエスト解析プロバイダ

ログイン画面表示時に送信されたリクエスト情報を解析して、ログイン情報を設定します。

ログイン情報を変更することで、ログイン後のロケールやタイムゾーンを設定することが可能です。
ログイン情報の詳細については、 「 LoginRequestInfo クラスの APIドキュメント 」 を参照してください。

ここで設定されたログイン情報は、リクエストパラメータ 「im_login_info」 を利用してログイン処理に引き継がれます。

複数の初期リクエスト解析プロバイダが設定されている場合、プラグインとして設定された順番にプロバイダが実行されます。
順番を考慮した実装および設定を行ってください。

インタフェース定義

```
1  package jp.co.intra_mart.foundation.security.certification.provider;
2
3  public interface RequestAnalyzer {
4
5     LoginInfo parseRequest(LoginRequestInfo loginInfo, HttpServletRequest request);
6  }
```

第 1 引数の loginInfo の必要なプロパティを変更して、そのまま返却することを想定しています。
複数の初期リクエスト解析プロバイダが設定されている場合、返却した loginInfo は、次に実行されるプロバイダの第 1 引数として利用されます。

標準プロバイダ

標準プロバイダはありません。

実装サンプル

リクエストパラメータ「sample_home_url」からログイン情報にホームURLを設定するサンプルです。

- サンプル Java コード

```java
package sample;

public class SampleHomeUrlRequestAnalyzer implements RequestAnalyzer {

   @Override
   public LoginInfo parseRequest(LoginRequestInfo loginInfo, HttpServletRequest request) {

      // ログイン画面表示時にリクエストパラメータからホームURL情報を設定します。
      String homeUrl = request.getParameter("sample_home_url");

      if (homeUrl != null && !homeUrl.isEmpty()) {

         // ログイン情報に設定します。
         loginInfo.setHomeUrl(homeUrl);
      }

      return loginInfo;
   }

}
```

- サンプルプラグイン設定ファイル

```xml
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
 <extension point="jp.co.intra_mart.security.user.initial_request_analyzer">
  <initial_request
   name="Sample Initial RequestAnalyzer"
   id="sample.programming_guide.initial_request_analyzer"
   version="1.0"
   rank="200">

   <request-analyzer-class>sample.SampleHomeUrlRequestAnalyzer</request-analyzer-class>

  </initial_request>
 </extension>
</plugin>
```

引数のログイン情報を要件に合わせて更新し、更新したログイン情報を返却するようにします。
ここで設定した情報は、ログイン画面で保持されてログイン処理に引き継がれます。

リクエスト解析プロバイダ

ログイン実行時に送信されたリクエスト情報を解析して、ログイン情報を設定します。

ログイン情報を変更することで、ログイン後のロケールやタイムゾーンを設定することが可能です。
ログイン情報の詳細については、 「 LoginRequestInfo クラスの APIドキュメント 」 を参照してください。

プラグインとして設定された順番にプロバイダが実行されます。
順番を考慮した実装および設定を行ってください。

インタフェース定義

「 *初期リクエスト解析プロバイダ* 」 と同じインタフェースを利用します。
標準プロバイダ

実装クラス名（一般ユーザ用）

- jp.co.intra_mart.system.security.certification.provider.impl.StandardLoginRequestAnalyzer

処理概要

以下の情報をリクエストパラメータから取得してログイン情報に設定します。

取得情報	リクエストパラメータ名
ユーザコード	im_user
パスワード	im_password
遷移先ページパス	im_url

遷移先ページパスには、ログイン後に遷移するページのURLが指定できます、 外部サイトのURLの場合、許可されたサイトのみ遷移可能です。
許可されていないサイトの場合は、遷移先ページパスは設定されません。

外部サイトの許可設定については、「設定ファイルリファレンス」 - 「認証外部ページURL許可リスト設定」を参照してください。

遷移先ページパスによる遷移を無効にする場合は、「設定ファイルリファレンス」 - 「リクエストパラメータによる画面遷移サポートの有無（ログイン時）」を参照してください。

実装サンプル

特定のIPアドレスからアクセスされた場合、パスワードを要求せずに認証済みに設定するサンプルです。

- サンプル Java コード

```
1  package sample;
2
3  public class SampleCheckIPAddressRequestAnalyzer implements RequestAnalyzer {
4
5     private static final String TRUSTED_ADDRESS = "192.168.1.2";
6
7     @Override
8     public LoginInfo parseRequest(LoginRequestInfo loginInfo, HttpServletRequest request) {
9
10       // アクセスしたクライアントのIPアドレスを取得します。
11       String address = request.getRemoteAddr();
12
13       if (TRUSTED_ADDRESS.equals(address)) {
14
15          // 特定のIPアドレスからアクセスされた場合、認証済みに設定します。
16          // パスワード確認処理は行いません。
17          // ただし、ユーザコードがない場合、アカウント情報が取得できないためエラーになります。
18          loginInfo.setDoneCertification(true);
19       }
20
21       return loginInfo;
22    }
23
24 }
```

- サンプルプラグイン設定ファイル

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
 <extension point="jp.co.intra_mart.security.user.login_request_analyzer">
  <login_request
   name="Sample RequestAnalyzer"
   id="sample.programming_guide.request_analyzer" version="1.0" rank="50">

   <request-analyzer-class>sample.SampleCheckIPAddressRequestAnalyzer</request-analyzer-
class>

  </login_request>
 </extension>
</plugin>
```

引数のログイン情報を要件に合わせて更新し、更新したログイン情報を返却するようにします。
認証プロバイダや認証リスナで利用する情報は、このプロバイダでログイン情報に設定してください。
標準プロバイダでは、ログイン情報にユーザコード、パスワード、遷移先ページパスを設定しています。

また、サンプルのように認証済み設定（`loginInfo.setDoneCertification(true)`）を行うことで、認証処理をスキップすることが可能です。
その場合でも認証リスナは実行されます。

認証プロバイダ

ログイン情報などを利用して、実際の認証処理を実行します。
認証結果により、以下のコードを返却してください。

表 認証プロバイダの返却コード

返却コード	条件
`CertificationStatus.CR_OK`	認証に成功した場合
`CertificationStatus.CR_NG`	入力した情報が間違っているなどで、認証に失敗した場合
`CertificationStatus.CR_ERROR`	環境などの要因で、認証が実行できない場合

> **注意**
> ログイン不可と判定された場合、認証プロバイダは実行されません。詳しくは、「*一般ユーザのログイン処理フロー*」の「*認証API*」-「*ログイン*」を参照してください。

インタフェース定義

```
package jp.co.intra_mart.foundation.security.certification.provider;

public interface UserCertification {

   CertificationStatus certification(LoginInfo loginInfo, AccountInfo user, HttpServletRequest request,
HttpServletResponse response);
}
```

> **注意**
>
> パスワード保存方式に「ハッシュ化」を利用している場合、引数のAccountInfoからパスワード値を取得できません。
> 常に null が返ります。
> LoginInfo に指定されているパスワードに対して照合を行いたい場合は、「 AccountPasswordAdapter クラスの APIドキュメント 」を利用してください。

標準プロバイダ

実装クラス名（一般ユーザ用）

- jp.co.intra_mart.system.security.certification.provider.impl.StandardUserCertification

処理概要

ログイン情報とアカウント情報のユーザコード、パスワードを比較します。
ユーザコード、パスワードが未設定（null）の場合はエラーです。

実装サンプル

ユーザが入力したパスワードとアカウントのパスワードを比較するサンプルです。

- サンプル Java コード

```java
package sample;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import jp.co.intra_mart.foundation.admin.account.model.AccountInfo;
import jp.co.intra_mart.foundation.admin.account.password.AccountPasswordAdapter;
import jp.co.intra_mart.foundation.admin.exception.PasswordException;
import jp.co.intra_mart.foundation.security.certification.CertificationStatus;
import jp.co.intra_mart.foundation.security.certification.model.LoginInfo;
import jp.co.intra_mart.foundation.security.certification.provider.UserCertification;

public class SampleAccountPasswordUserCertification implements UserCertification {

   @Override
   public CertificationStatus certification(final LoginInfo loginInfo, final AccountInfo user, final
HttpServletRequest request, final HttpServletResponse response) {
      final String userCd = loginInfo.getUserCd();
      final String password = loginInfo.getPassword();

      final AccountPasswordAdapter adapter = new AccountPasswordAdapter();
      try {
         // アカウントのパスワード値の照合を行います。
         if (!adapter.collate(userCd, password)) {
            // パスワードが一致しないためNGを返却します。
            return CertificationStatus.CR_NG;
         }
      } catch (final PasswordException e) {
         e.printStackTrace();
         // 何らかの理由でパスワードの照合に失敗したためエラーを返却します。
         return CertificationStatus.CR_ERROR;
      }

      // パスワードの照合に成功したためOKを返却します。
      return CertificationStatus.CR_OK;
   }

}
```

- サンプルプラグイン設定ファイル

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
 <extension point="jp.co.intra_mart.security.user.certification">
  <certification
   name="Sample UserCertification / UserCertificationValidation / UserCertificationListener"
   id="sample.programming_guide.user_certification" version="1.0" rank="50">

   <!-- UserCertification -->
   <certification-class>sample.SampleAccountPasswordUserCertification</certification-class>

   <!-- UserCertificationValidation -->
   <certification-validation>
    <validation-class>sample.SampleAccountPasswordUserCertificationValidation</validation-class>
   </certification-validation>

   <!-- UserCertificationListener -->
   <certification-listener>
    <listener-class>sample.SampleMailAlertUserCertificationListener</listener-class>
   </certification-listener>

  </certification>
 </extension>
</plugin>
```

認証条件判定プロバイダ

「*認証プロバイダ*」 を利用するための条件判定を行います。

intra-mart Accel Platform では、プラグインを複数登録することで、複数の認証プロバイダを登録できます。
複数の認証プロバイダが登録されている場合に、認証の要求に対してどの認証プロバイダを利用するかを、この認証条件判定プロバイダを利用して判定します。

このプロバイダが設定されていない認証プロバイダは常に利用対象です。

認証条件判定プロバイダは、認証プロバイダと同じプラグイン設定に設定し、その認証プロバイダを条件判定の対象とします。
同じ認証プロバイダに対して、複数設定することが可能です。
複数設定されている場合は、設定された全ての認証条件判定プロバイダの処理結果が `true` である場合に、条件判定の対象となる認証プロバイダが利用されます。

> **注意**
>
> ログイン不可と判定された場合、認証条件判定プロバイダは実行されません。詳しくは、「*一般ユーザのログイン処理フロー*」の「*認証API*」-「*ログイン*」を参照してください。

インタフェース定義

```
package jp.co.intra_mart.foundation.security.certification.provider;

public interface UserCertificationValidation {

   boolean validate(LoginInfo loginInfo, AccountInfo user, HttpServletRequest request, HttpServletResponse
response);
}
```

> **注意**
>
> パスワード保存方式に「ハッシュ化」を利用している場合、引数のAccountInfoからパスワード値を取得できません。
> 常に `null` が返ります。

標準プロバイダ

標準プロバイダはありません。

実装サンプル

システムプロパティにて、サンプルの認証プロバイダを利用する設定が行われているかを確認するサンプルです。

- サンプル Java コード

```
package sample;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import jp.co.intra_mart.foundation.admin.account.model.AccountInfo;
import jp.co.intra_mart.foundation.security.certification.model.LoginInfo;
import jp.co.intra_mart.foundation.security.certification.provider.UserCertificationValidation;

public class SampleAccountPasswordUserCertificationValidation implements
UserCertificationValidation {

   @Override
   public boolean validate(final LoginInfo loginInfo, final AccountInfo user, final HttpServletRequest
request, final HttpServletResponse response) {

      // システムプロパティからサンプルのアカウントパスワードによる照合を行う認証プロバイダをサポートするかを判
定します。
      final boolean result = Boolean.getBoolean("sample.account.password.support");

      return result;
   }

}
```

- サンプルプラグイン設定ファイル

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
 <extension point="jp.co.intra_mart.security.user.certification">
  <certification
   name="Sample UserCertification / UserCertificationValidation / UserCertificationListener"
   id="sample.programming_guide.user_certification" version="1.0" rank="50">

   <!-- UserCertification -->
   <certification-class>sample.SampleAccountPasswordUserCertification</certification-class>

   <!-- UserCertificationValidation -->
   <certification-validation>
    <validation-class>sample.SampleAccountPasswordUserCertificationValidation</validation-class>
   </certification-validation>

   <!-- UserCertificationListener -->
   <certification-listener>
    <listener-class>sample.SampleMailAlertUserCertificationListener</listener-class>
   </certification-listener>

  </certification>
 </extension>
</plugin>
```

認証プロバイダを利用するための条件を指定します。
認証プロバイダの対応する認証方式がサポートされているかなどを確認します。

設定された認証条件判定プロバイダの処理結果が `true` である場合に、条件判定の対象となる認証プロバイダ（サンプルプラグイン設定ファイルの場合は、`sample.SampleAccountPasswordUserCertification` ）が利用されます。

認証リスナ

認証処理の後処理を実行します。

認証結果を基に、ログの出力や通知処理などを実行するために利用します。

このプロバイダは、認証が成功した場合も認証が失敗した場合も呼び出されます。
引数の認証結果を確認して必要な処理を行ってください。

> **注意**
>
> 認証が成功した場合、このプロバイダの実行時には、ログインが完了して認証済みのアクセスコンテキストが生成されています。
> また認証が失敗した場合は、アクセスコンテキストはログインしていない状態に切り替わります。
> アクセスコンテキストの情報を参照する場合は注意してください。

> **注意**
>
> 認証APIは、１つのトランザクションで処理されます。
> そのため、認証リスナでデータベースなどのトランザクション管理されている情報を更新しても、その後の処理によっては、ロールバックされる可能性があります。
> 別トランザクションとする場合は、独自にトランザクションを開始するようにしてください。

> **注意**
>
> ログイン不可と判定された場合、認証リスナは実行されません。詳しくは、「 *一般ユーザのログイン処理フロー* 」の「 *認証API* 」-「 *ログイン* 」を参照してください。

認証リスナは、複数の認証プロバイダが設定されている場合に、特定の認証プロバイダが実行されたときのみ実行されるように設定することが可能です。

- 特定の認証プロバイダのみ実行する場合

  プラグイン設定ファイルに、認証プロバイダ（<certification-class>）とセットで設定します。

  【設定例】

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
 <extension point="jp.co.intra_mart.security.user.certification">
  <certification
   name="Sample"
   id="sample.certification"
   version="8.0"
   rank="100">
   <certification-class>sample.certification.SampleUserCertification</certification-class>
   <certification-listener>
    <listener-class>sample.certification.SampleCertificationListener</listener-class>
   </certification-listener>
  </certification>
 </extension>
</plugin>
```

- 全ての認証プロバイダで実行する場合

  プラグイン設定ファイルに、認証プロバイダ（<certification-class>）を記述せずに設定します。

  【設定例】

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
 <extension point="jp.co.intra_mart.security.user.certification">
  <certification
   name="Sample"
   id="sample.certification"
   version="8.0"
   rank="100">
   <certification-listener>
    <listener-class>sample.certification.SampleCertificationListener</listener-class>
   </certification-listener>
  </certification>
 </extension>
</plugin>
```

コラム

アカウントコンテキストについては、「 アクセスコンテキスト仕様書 」-「 アカウントコンテキスト 」を参照してください。

コラム

一般ユーザの認証時の認証リスナを提供する場合は、一般ユーザの強制ログイン時の認証リスナの提供を行うことも検討してください。
詳細は、「 *強制ログイン用認証リスナ* 」 を参照してください。

インタフェース定義

```
package jp.co.intra_mart.foundation.security.certification.provider;

public interface UserCertificationListener {

   void doCertification(CertificationStatus status, LoginInfo loginInfo, AccountInfo user, HttpServletRequest request,
HttpServletResponse response);
}
```

注意

パスワード保存方式に「ハッシュ化」を利用している場合、引数のAccountInfoからパスワード値を取得できません。
常に `null` が返ります。

標準プロバイダ

標準プロバイダはありません。

実装サンプル

ログイン処理の失敗をメール通知するサンプルです。

- サンプル Java コード

```
package sample;

public class SampleMailAlertUserCertificationListener implements UserCertificationListener {

   @Override
   public void doCertification(CertificationStatus status, LoginInfo loginInfo, AccountInfo user,
HttpServletRequest request, HttpServletResponse response) {

      if (status != CertificationStatus.CR_OK) {

         // 通知メールを作成します。
         StandardMail mail = new StandardMail();
         mail.setFrom("iap@example.com");
         mail.addTo("admin@example.com");
         mail.setSubject("ログイン通知");
         mail.setText("ユーザ：" + loginInfo.getUserCd() + " のログインに失敗しました。");

         // 通知メールを送信します。
         MailSender sender = new JavaMailSender(mail);
         try {
            sender.send();
         } catch (MailSenderException e) {
            e.printStackTrace();
         }
      }
   }

}
```

- サンプルプラグイン設定ファイル

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
 <extension point="jp.co.intra_mart.security.user.certification">
  <certification
   name="Sample UserCertification / UserCertificationValidation / UserCertificationListener"
   id="sample.programming_guide.user_certification" version="1.0" rank="50">

   <!-- UserCertification -->
   <certification-class>sample.SampleAccountPasswordUserCertification</certification-class>

   <!-- UserCertificationValidation -->
   <certification-validation>
    <validation-class>sample.SampleAccountPasswordUserCertificationValidation</validation-class>
   </certification-validation>

   <!-- UserCertificationListener -->
   <certification-listener>
    <listener-class>sample.SampleMailAlertUserCertificationListener</listener-class>
   </certification-listener>

  </certification>
 </extension>
</plugin>
```

認証の後処理として実行したい、任意の処理を行います。

ログインページプロバイダ

ログイン画面用のページパス情報を取得します。
ここで取得されるパスは、 PageManager API の forward() メソッドで利用されるパス情報であり、URLではないので注意してください。

条件に従ってログインページ情報を、PageUrl オブジェクトに設定します。
設定されたページにはフォワードにより遷移されますが、PageUrl オブジェクトにリクエストパラメータを追加設定することが可能です。

条件に一致しない場合は、null を返却するようにします。
その場合、プラグインとして設定された次のプロバイダに処理が引き継がれます。

インタフェース定義

```
1  package jp.co.intra_mart.foundation.security.certification.provider;
2
3  public interface LoginPageProvider {
4
5      PageUrl getLoginPage(LoginInfo loginInfo, HttpServletRequest request, HttpServletResponse response);
6  }
```

標準プロバイダ

実装クラス名（一般ユーザ用）

- jp.co.intra_mart.system.security.certification.provider.impl.StandardUserLoginPageProvider

処理概要

PageManager.getPageUrl() API を利用して、ログインページのパスを取得します。
実際のページパスは、以下に設定されています。

- クライアントタイプ = PC の場合： <plugin/jp.co.intra_mart.certification.page.standard/plugin.xml>
- クライアントタイプ = SP の場合： <plugin/jp.co.intra_mart.security.user.login_page.sp/plugin.xml>

API の引数に指定するページコードは、jp.co.intra_mart.foundation.security.certification.CertificationPage.LOGIN_PAGE です。

実装サンプル

特定のリクエストパラメータ（custom）が設定されている場合に、プラグイン設定ファイルから、ログインページのページパスとリクエストパラメータを取得するサンプルです。

- サンプル Java コード

```
package sample;

public class SampleLoginPageProvider implements LoginPageProvider, XmlInitParamable {

   private String page;

   private Map<String, String> params;

   @Override
   public PageUrl getLoginPage(LoginInfo loginInfo, HttpServletRequest request, HttpServletResponse
response) {

      // ログイン画面表示時にリクエストパラメータからカスタムログインページ情報フラグを取得します。
      String custom = request.getParameter("custom");

      if (custom != null) {

         if (page != null && !page.isEmpty()) {
            // プラグイン設定ファイルの情報から、ログインページ情報を作成します。
            PageUrl pageUrl = new PageUrl(page, params);
            return pageUrl;
         }

      }

      // 対象外の場合は、その他のプロバイダに移譲します。
      return null;
   }

   @Override
   public void init(Node node) {
      // XmlInitParamable を実装することで、プラグイン設定ファイルからノード情報を取得できます。

      // プラグイン設定ファイルから、ページパスとリクエストパラメータを取得します。
      XmlNode[] nodes = new XmlNode(node).select("page");
      this.page = nodes[0].getValue();

      params = XmlNodeConverter.createInitParamMap(node);
   }
}
```

- サンプルプラグイン設定ファイル

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
 <extension point="jp.co.intra_mart.security.user.login_page">
  <page
   name="Sample LoginPageProvider" id="sample.programming_guide.login_page"
   version="1.0" rank="200">

   <page-provider-class>sample.SampleLoginPageProvider</page-provider-class>

   <page>/sample/login.jsp</page>
   <init-param>
    <param-name>foo</param-name>
    <param-value>bar</param-value>
   </init-param>

  </page>
 </extension>
</plugin>
```

遷移先ページプロバイダ

ログイン成功時に遷移する画面のページパス情報を取得します。
ここで取得されるパスは、 PageManager API の redirect() メソッドを利用してリダイレクトするURL情報です。

条件に従って遷移先情報を、PageUrl オブジェクトに設定します。
PageUrl オブジェクトは、リクエストパラメータを設定できます。
また、POSTによる遷移も可能です。

条件に一致しない場合は、null を返却するようにします。
その場合、プラグインとして設定された次のプロバイダに処理が引き継がれます。

全ての 「 *遷移先ページプロバイダ* 」 で情報が取得できなかった場合、認証APIはアカウントコンテキストのホームURLを取得します。

インタフェース定義

```
package jp.co.intra_mart.foundation.security.certification.provider;

public interface TargetPageProvider {

   PageUrl getTargetPage(LoginInfo loginInfo, HttpServletRequest request, HttpServletResponse response);
}
```

標準プロバイダ

実装クラス名（一般ユーザ用）

- jp.co.intra_mart.system.security.certification.provider.impl.StandardUserTargetPageProvider

処理概要

以下の順にページ情報を取得します。

- ログイン情報に遷移先ページパスが設定されている場合、そのページパスを取得します。
- HTTPセッションに遷移先情報が設定されている場合、そのページパスを取得します。
- 取得できなかった場合、null を返却します。

実装サンプル

初回ログインユーザに対してウェルカムページを表示するサンプルです。

- サンプル Java コード

```java
package sample;

public class SampleWelcomeTargetPageProvider implements TargetPageProvider {

   private static final String WELCOME_PAGE = "/welcome";

   @Override
   public PageUrl getTargetPage(final LoginInfo loginInfo, final HttpServletRequest request, final
HttpServletResponse response) {

      // 初回ログインフラグを取得します。
      // 認証リスナ「SampleFirstLoginUserCertificationListener」であらかじめ、リクエストに設定しておきます。
      final String isFirstLoginFlag = (String) request.getAttribute("first_login");
      final boolean isFirstLogin = Boolean.valueOf(isFirstLoginFlag);

      if (isFirstLogin) {
         return new PageUrl(WELCOME_PAGE);
      }

      // 対象外の場合は、その他のプロバイダに移譲します。
      return null;
   }

}
```

- サンプルプラグイン設定ファイル

```xml
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
 <extension point="jp.co.intra_mart.security.user.target_page">
  <page
   name="Sample TargetPageProvider" id="sample.programming_guide.target_page"
   version="1.0" rank="200">

   <page-provider-class>sample.SampleWelcomeTargetPageProvider</page-provider-class>

  </page>
 </extension>
</plugin>
```

標準プロバイダの認証フロー

intra-mart Accel Platform の認証モジュールで提供されている一般ユーザ用標準プロバイダによる認証フローは以下の通りです。

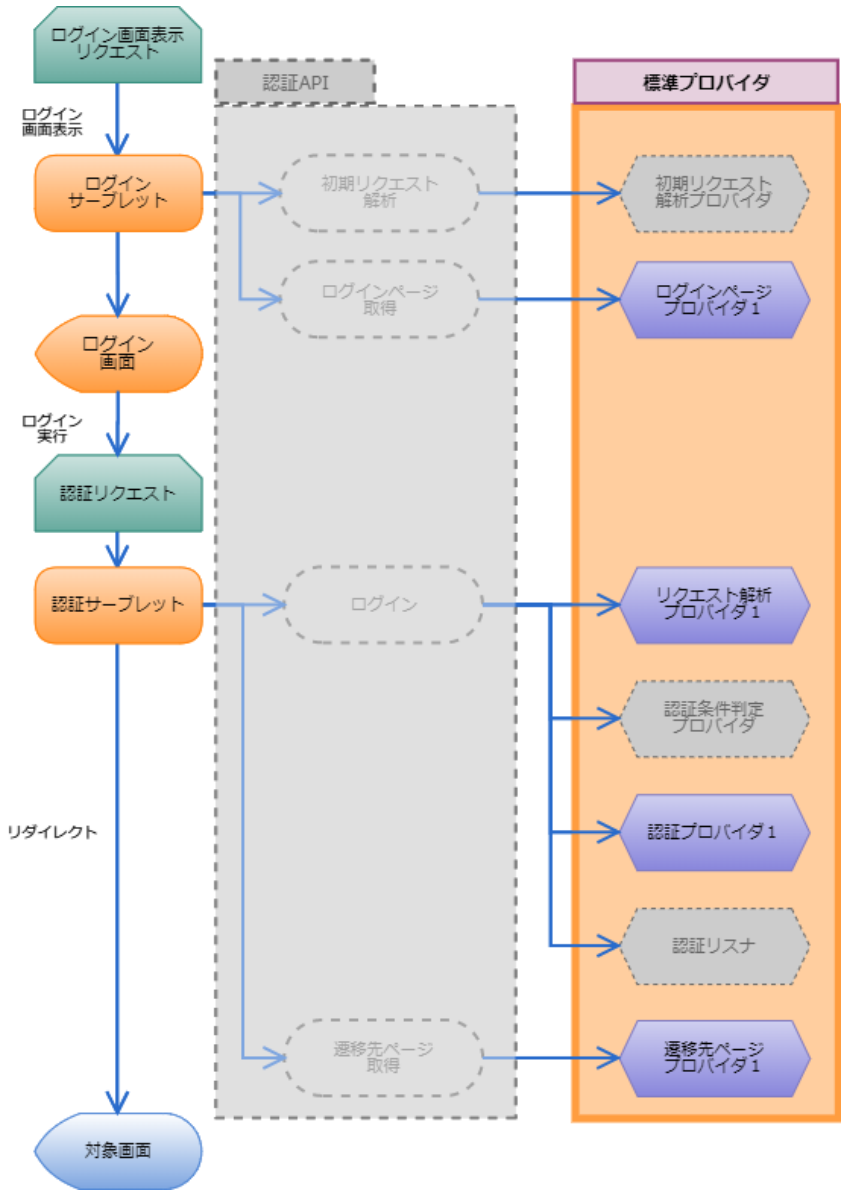


図 一般ユーザの標準プロバイダを利用した場合の認証フロー

図で示した標準プロバイダの実装クラスは、以下の通りです。

表 一般ユーザ用標準プロバイダ一覧

標準プロバイダ	実装クラス	rank
ログインページプロバイダ1	jp.co.intra_mart.system.security.certification.provider.impl.StandardUserLoginPageProvider	100
リクエスト解析プロバイダ1	jp.co.intra_mart.system.security.certification.provider.impl.StandardLoginRequestAnalyzer	100
認証プロバイダ1	jp.co.intra_mart.system.security.certification.provider.impl.StandardUserCertification	100
遷移先ページプロバイダ1	jp.co.intra_mart.system.security.certification.provider.impl.StandardUserTargetPageProvider	100

一般ユーザのログイン画面

一般ユーザのログイン画面の機能について説明します。

ログイン画面とは

ログイン画面とは、ログインに必要な情報を入力し、ログインを実行するための画面です。
多くの機能はログインすることで利用可能となるため、システムの入り口となる画面です。

ログイン画面の機能

intra-mart Accel Platform が標準で提供する一般ユーザ用のログイン画面の機能は以下の通りです。

- テナントID の制御
- ユーザコード入力補助
- 遷移先情報キー
- パスワードリマインダ
- セキュアトークン

ログイン画面項目

ログイン画面の入力項目について説明します。

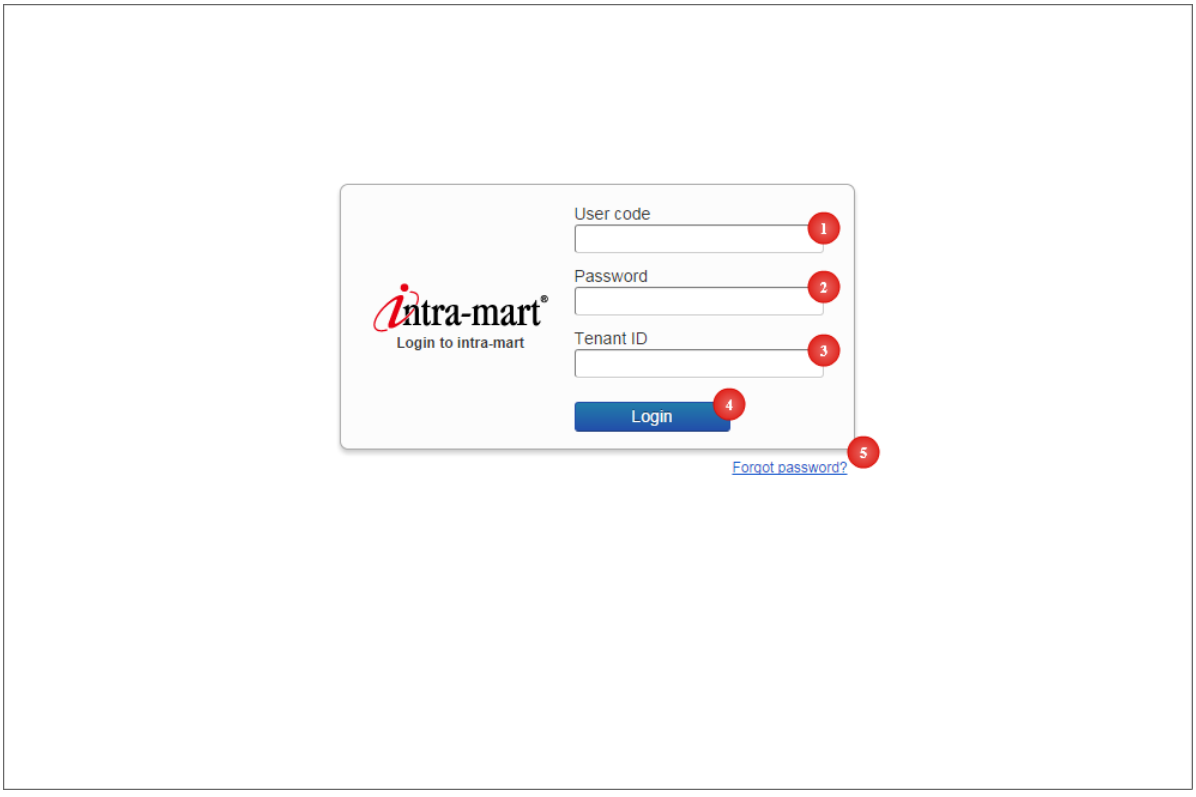


図 一般ユーザのログイン画面

1. ユーザコード
   ユーザコード入力欄です。
   Cookie を利用して、前回入力した情報が初期入力されます。
   ユーザコードは必須です。
   intra-mart Accel Platform の標準のログイン画面は、システムの入り口として、通常の画面と異なるデザインが利用されています。
   そのため、必須入力マークは表示されません。

2. パスワード
   パスワード入力欄です。
   パスワードの入力制限については、パスワード履歴管理設定ファイルで指定されます。
   「設定ファイルリファレンス」 - 「パスワード履歴管理設定」を参照してください。

3. テナントID
   テナントID 入力欄です。
   バーチャルテナントによる複数テナントを利用していて、ログイン対象のテナントを指定する必要がある場合に表示されます。
   「 *テナントID の制御* 」 を参照してください。

4. ログインボタン
   ログインを実行します。

5. 「パスワードを忘れた方はコチラ」リンク
   パスワードリマインダ画面に遷移します。
   「 *パスワードリマインダ* 」 を参照してください。

ログイン画面の実装

ログイン画面の実装について説明します。

ログイン画面のURL

intra-mart Accel Platform の標準の一般ユーザ用ログイン画面URL は以下の通りです。

`http://<HOST>:<PORT>/<CONTEXT_PATH>/login`

ログイン画面のURLは `web.xml` で、`UserLoginServlet` にマッピングされています。
そのため、ログイン画面のURLを変更するためには `web.xml` の修正が必要ですが、intra-mart Accel Platform では `web.xml` の変更を推奨していません。
ログイン画面を変更する場合は、URLの変更ではなく「 *ログインページプロバイダ* 」の利用を検討してください。

テナントID の制御

バーチャルテナントによる複数テナントを利用している場合、ログイン対象のテナントを指定するために、テナントID 入力欄が表示されます。
テナントID入力欄が表示される条件は、以下の通りです。

表 テナントID入力欄表示条件

テナント自動解決 [1]	テナント数	テナントID入力欄表示
なし	**1**	なし
なし	**2～**	あり
あり	**-**	なし

[1] 「 リクエスト情報を利用したテナント自動解決機能 」を利用しているかどうかです。
リクエスト情報を利用したテナント自動解決機能の詳細は、 「intra-mart Accel Platform セットアップガイド」 - 「テナント解決機能」 - 「リクエスト情報を利用したテナント自動解決機能について」 を参照してください。

ユーザコード入力補助

ユーザコードとテナントIDは、Cookie を利用して、前回入力した情報が初期入力されます。

項目	保存される **Cookie** 名
ユーザコード	im_user_id
テナントID	im_tenant_id

Cookie への保存は、画面の機能として行われます。
そのため、「ログイン」ボタンをクリックして、ログイン処理を実行した時点で保存され、ログイン処理に失敗した場合でも値が更新されます。

この機能が有効な場合、複数のユーザで同一のブラウザを利用している場合に別のユーザのユーザコードが初期表示されます。
この機能を無効にするためには、「設定ファイルリファレンス」 - 「認証設定（一般ユーザ用）」 - 「ユーザコードのCookie保存」を参照してください。

遷移先情報キー

ログイン後に遷移するための情報を保持するキーです。
ログイン画面では、以下のリクエストパラメータ名の隠し属性として保持します。

- リクエストパラメータ名 ： im_page_key

また、認証に関するエラー画面でも、このキー情報を持ちまわるように実装されています。

intra-mart Accel Platform では、以下の場合に遷移先情報をセッションに保存します。

- 未認証で任意の画面のURLに直接アクセスし、未認証エラーとなった場合
- ログイン画面に、リクエストパラメータ im_url を利用してアクセスした場合

intra-mart Accel Platform 標準の 「*遷移先ページプロバイダ*」 では、リクエストパラメータ im_page_key から、セッションに保存された遷移先情報を参照して、ログイン後の遷移先を決定します。
これにより、未認証エラーなどのエラーが発生した場合でも、ログイン後に適切な画面に遷移することが可能です。

新しく、ログイン画面、または、エラー画面を作成する場合は、遷移先情報キーを持ちまわるように対応してください。

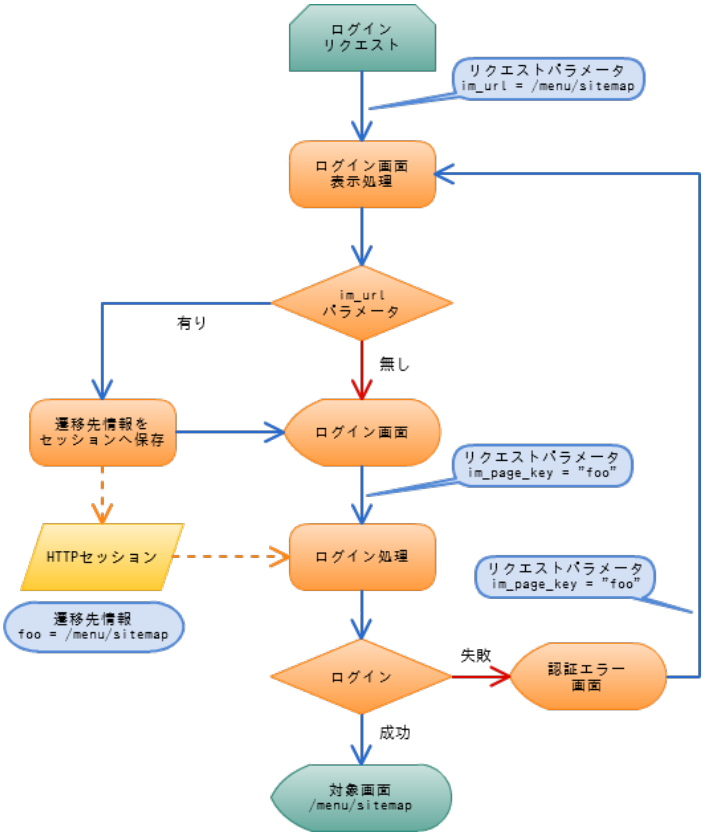


図 遷移先情報の持ちまわり

スクリプト開発モデルで、遷移先情報キーを持ちまわる場合の実装方法は、以下の通りです。

- ファンクションコンテナ

```
var pageKey = request.im_page_key;

function init(request) {

    :

  // 遷移先情報キーをリクエストパラメータから取得
  var pageKey = request.im_page_key;

  if (!pageKey) {
    // 遷移先情報キーをリクエスト属性から取得
    pageKey = request.getAttribute("im_page_key");
  }

    :

}
```

- プレゼンテーションページ

```
  :

<!-- hidden タグに遷移先情報キーを設定する -->
<imart type="hidden" im_page_key=page_key />

  :
```

パスワードリマインダ

パスワードリマインダ機能が有効な場合、「パスワードリマインダ」画面に遷移するためのリンクが表示されます。

パスワードリマインダ機能を無効にするためには、以下のいずれかの設定を行います。

- パスワードリマインダモジュールを除外する。

  IM-Juggling で、プロジェクトのモジュール構成を開き、以下のモジュールのチェックを外してください。

  「標準機能」 - 「基盤機能」 - 「パスワードリマインダモジュール」

  WAR ファイルを作成して、アプリケーションサーバに再デプロイしてください。

- パスワードリマインダ設定ファイルを変更する。

  「設定ファイルリファレンス」 - 「パスワードリマインダ設定」 - 「パスワードリマインダの利用」を参照してください。

セキュアトークン

ログイン処理では、悪意あるユーザからのアクセスを防ぐため、セキュアトークンチェックを行っています。
セキュアトークンチェックを行うために、ログイン画面ではセキュアトークンチェックのためのトークン情報を保持します。
このため、ログイン画面を経由せずにログイン処理を実行できないようにしています。

> **注意**
>
> セキュアトークンチェックは、HTTPセッションに保存したトークン情報を基にチェックを行います。
> このため、ログイン画面表示後しばらくアクセスしなかった場合、セッションタイムアウトが発生してトークン情報が破棄されます。
> トークン情報が破棄されたままログイン画面でログインを実行した場合、ユーザコード、パスワードが正しくても、ログインできずに HTTP403 権限エラーが表示されます。
>
> この場合は、再度ログイン画面にアクセスしてからログインしなおしてください。

システム管理者のログイン処理

ここでは、システム管理者のログイン処理について、一般ユーザの場合と同様に、以下の順に説明します。
処理の流れは一般ユーザの場合と同様のため、システム管理者に関連する情報のみ説明します。

- *システム管理者のログイン処理フロー*

  システム管理のログイン処理の流れを説明します。

- *システム管理者の認証プロバイダ・プラグイン*

  システム管理のログイン処理で利用される認証プロバイダ・プラグインについて説明します。

- *システム管理者のログイン画面*

  システム管理のログイン画面の機能について説明します。

- *IPアドレスによるアクセス制限*

  システム管理で利用可能なIPアドレスによるアクセス制限について説明します。

システム管理者のログイン処理フロー

システム管理者のログイン処理の流れについて説明します。

システム管理者のログイン処理フローの概要

ログイン処理では、ユーザが入力したシステム管理者の情報などを利用した認証処理を行います。

システム管理者の認証処理は、システム管理者の認証APIを呼び出すことで実行されます。

システム管理者の認証APIと認証プロバイダ・プラグインについては、以下で説明します。

- *認証API*
- *システム管理者の認証プロバイダ・プラグイン*

システム管理者のログイン処理における、認証処理の流れは以下の通りです。

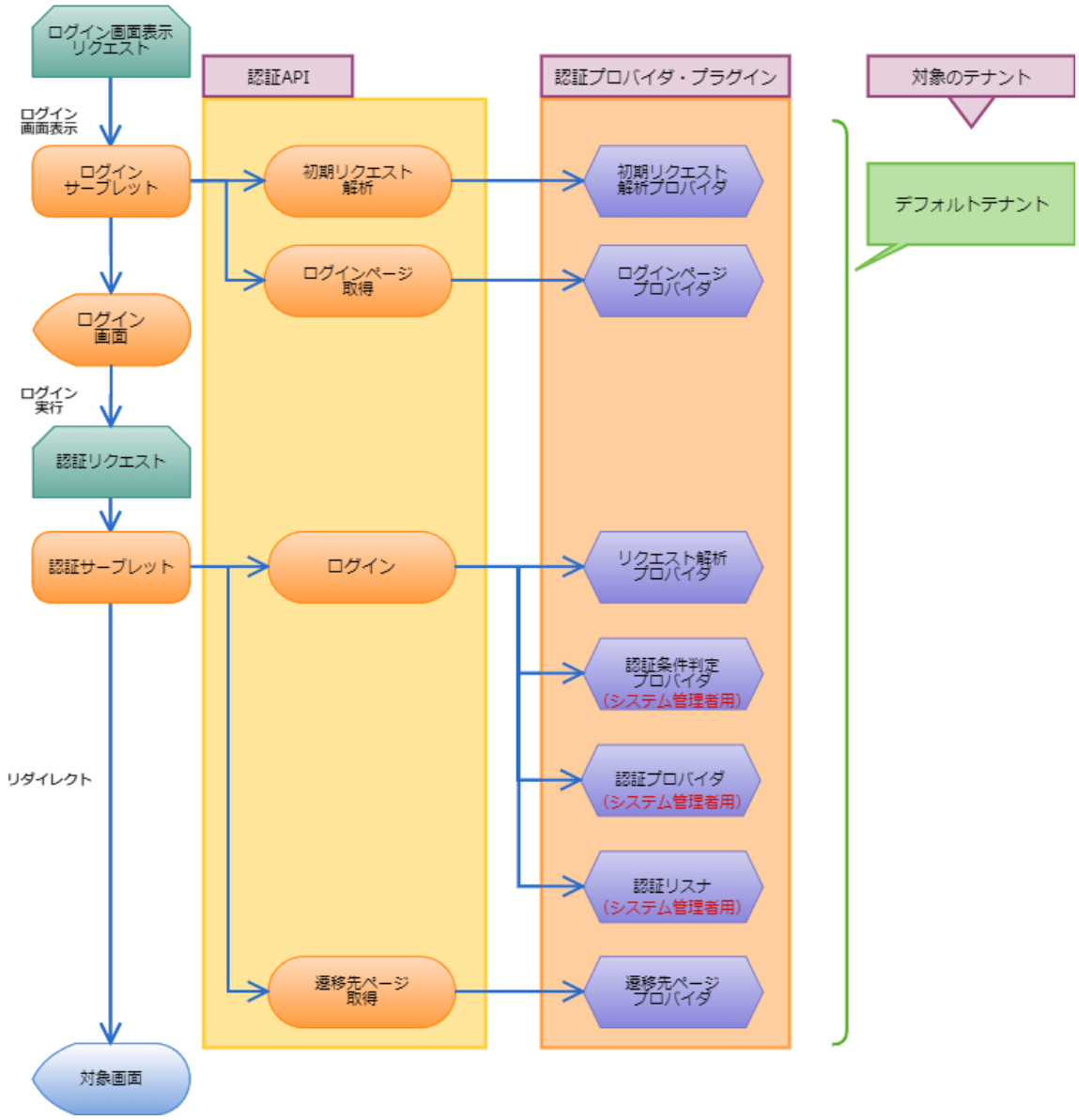


図 システム管理者のログイン処理フロー

処理対象のテナントについて

システム管理者の認証プロバイダ・プラグインでの処理対象テナントは、バーチャルテナントによる複数テナントを利用している場合でも、常にデフォルトテナントです。

認証API

システム管理者のログイン処理で実行される認証APIは、以下のクラスです。

`jp.co.intra_mart.foundation.security.certification.AdministratorCertificationManager#login()`

この認証APIは、ログイン処理中に以下の処理を行います。

1. 初期リクエスト解析 （ `initialParseRequest()` ）
2. ログインページ取得 （ `getLoginPageUrl()` ）
3. ログイン （ `login()` ）
4. 遷移先ページ取得 （ `getTargetPageUrl()` ）

API の詳細については、「 AdministratorCertificationManager クラスの APIドキュメント 」を参照してください。

各処理の流れを説明します。

初期リクエスト解析

「 *初期リクエスト解析プロバイダ* 」を呼び出します。

ログインページ取得

「 *ログインページプロバイダ* 」を呼び出します。

ログイン

以下の処理を順に実行します。

1. 「 *リクエスト解析プロバイダ* 」を呼び出します。
2. 「 *認証条件判定プロバイダ* 」を利用して、「 *認証プロバイダ* 」を決定します。
3. 「 *認証プロバイダ* 」を呼び出し、認証を行います。
4. アカウントコンテキストを切り替えてログイン状態にします。
5. 「 *認証リスナ* 」を呼び出します。

遷移先ページ取得

「 *遷移先ページプロバイダ* 」を呼び出します。
遷移先が取得できなかった場合、アカウントコンテキストのホームURLプロパティを取得します。

システム管理者の認証プロバイダ・プラグイン

「 *システム管理者のログイン処理フロー* 」で説明した、システム管理者のログイン処理で利用する認証プロバイダ・プラグインと、intra-mart Accel Platform の標準で提供されるプロバイダ（以下、標準プロバイダ）の動作について説明します。

システム管理者の認証プロバイダ・プラグインの概要

認証プロバイダ・プラグインについては、一般ユーザの説明 「 *一般ユーザの認証プロバイダ・プラグインの概要* 」 を参照してください。

以下に、システム管理者用の認証プロバイダ・プラグインの詳細について説明します。

認証プロバイダ・プラグイン一覧

認証機能で提供されるシステム管理者用の認証プロバイダ・プラグインは、以下の通りです。
「対象のテナント」は、リクエスト情報を利用したテナント自動解決機能を利用しない場合の処理対象です。
リクエスト情報を利用したテナント自動解決機能の詳細は「intra-mart Accel Platform セットアップガイド」 - 「テナント解決機能」 - 「リクエスト情報を利用したテナント自動解決機能について」 を参照してください。

表 システム管理者用認証プロバイダ・プラグイン一覧

プロバイダ名	概要	拡張ポイント	対象のテナント
初期リクエスト解析プロバイダ	ログイン画面表示時に送信されたリクエスト情報を解析して、ログイン情報を設定します。	jp.co.intra_mart.security.administrator.initial_request_analyzer	デフォルトテナント
ログインページプロバイダ	ログイン画面用のページパス情報を取得します。	jp.co.intra_mart.security.administrator.login_page	デフォルトテナント
リクエスト解析プロバイダ	ログイン実行時に送信されたリクエスト情報を解析して、ログイン情報を設定します。	jp.co.intra_mart.security.administrator.login_request_analyzer	デフォルトテナント
認証条件判定プロバイダ	認証プロバイダを利用するための条件判定を行います。	（認証プロバイダの追加情報として定義）	デフォルトテナント
認証プロバイダ	ログイン情報などを利用して、実際の認証処理を実行します。	jp.co.intra_mart.security.administrator.certification	デフォルトテナント
認証リスナ	認証処理の後処理を実行します。	（認証プロバイダの追加情報として定義）	デフォルトテナント
遷移先ページプロバイダ	ログイン成功時に遷移する画面のページパス情報を取得します。	jp.co.intra_mart.security.administrator.target_page	デフォルトテナント

認証プロバイダ・プラグインの詳細

各認証プロバイダ・プラグインの役割は、一般ユーザ用と同じです。
一般ユーザ用の認証プロバイダ・プラグインの説明 「 *認証プロバイダ・プラグインの詳細* 」 を参照してください。
ここでは、システム管理者用インタフェース一覧と標準プロバイダの動作のみ説明します。

システム管理者用インタフェース一覧

表 システム管理者用インタフェース一覧

プロバイダ名	インタフェース
初期リクエスト解析プロバイダ	jp.co.intra_mart.foundation.security.certification.provider.RequestAnalyzer （一般ユーザ用と共用）
リクエスト解析プロバイダ	jp.co.intra_mart.foundation.security.certification.provider.RequestAnalyzer （一般ユーザ用と共用）
認証プロバイダ	jp.co.intra_mart.foundation.security.certification.provider.AdministratorCertification
認証条件判定プロバイダ	jp.co.intra_mart.foundation.security.certification.provider.AdministratorCertificationValidation
認証リスナ	jp.co.intra_mart.foundation.security.certification.provider.AdministratorCertificationListener

プロバイダ名	インタフェース
ログインページプロバイダ	jp.co.intra_mart.foundation.security.certification.provider.LoginPageProvider （一般ユーザ用と共用）
遷移先ページプロバイダ	jp.co.intra_mart.foundation.security.certification.provider.TargetPageProvider （一般ユーザ用と共用）

初期リクエスト解析プロバイダ

intra-mart Accel Platform の初期状態では、標準プロバイダが設定されています。
標準プロバイダは、アクセス元の IP アドレスを取得して、アクセスが許可されている IP アドレスの範囲内かどうかを確認します。
このプロバイダの詳細については、 「 *IPアドレスによるアクセス制限* 」 を参照してください。

リクエスト解析プロバイダ

intra-mart Accel Platform の初期状態では、標準プロバイダが２つ設定されています。

１つ目の標準プロバイダは、初期リクエスト解析プロバイダと共通で、 「 *IPアドレスによるアクセス制限* 」 のためのプロバイダです。
２つ目の標準プロバイダは、一般ユーザ用と共通で、ユーザコード、パスワード、遷移先ページパスをログイン情報に設定します。

認証プロバイダ

intra-mart Accel Platform の初期状態では、標準プロバイダが設定されています。
標準プロバイダは、ログイン情報に設定されたパスワードとシステム管理者情報のパスワードの比較を行います。

認証条件判定プロバイダ

intra-mart Accel Platform の初期状態では、このプロバイダは設定されていません。
このプロバイダが設定されていない場合、該当の認証プロバイダは常に利用対象です。

認証リスナ

intra-mart Accel Platform の初期状態では、このプロバイダは設定されていません。

ログインページプロバイダ

intra-mart Accel Platform の初期状態では、標準プロバイダが設定されています。
標準プロバイダは、PageManager API の getPageUrl() メソッドを利用して、標準のシステム管理者用ログイン画面のページパスを取得します。

遷移先ページプロバイダ

intra-mart Accel Platform の初期状態では、標準プロバイダが設定されています。
標準プロバイダは、ログイン情報に設定された遷移先ページパスまたはセッションに設定された遷移先情報からページパス情報を取得します。

ただし、これらに遷移先情報を設定する処理はないため、標準プロバイダが返す値は null です。
そのため、認証APIは常にアカウントコンテキストのホームURL（システム管理者用ホーム画面URL）を取得します。

標準プロバイダの認証フロー

intra-mart Accel Platform の認証モジュールで提供されているシステム管理者用標準プロバイダによる認証フローは以下の通りです。

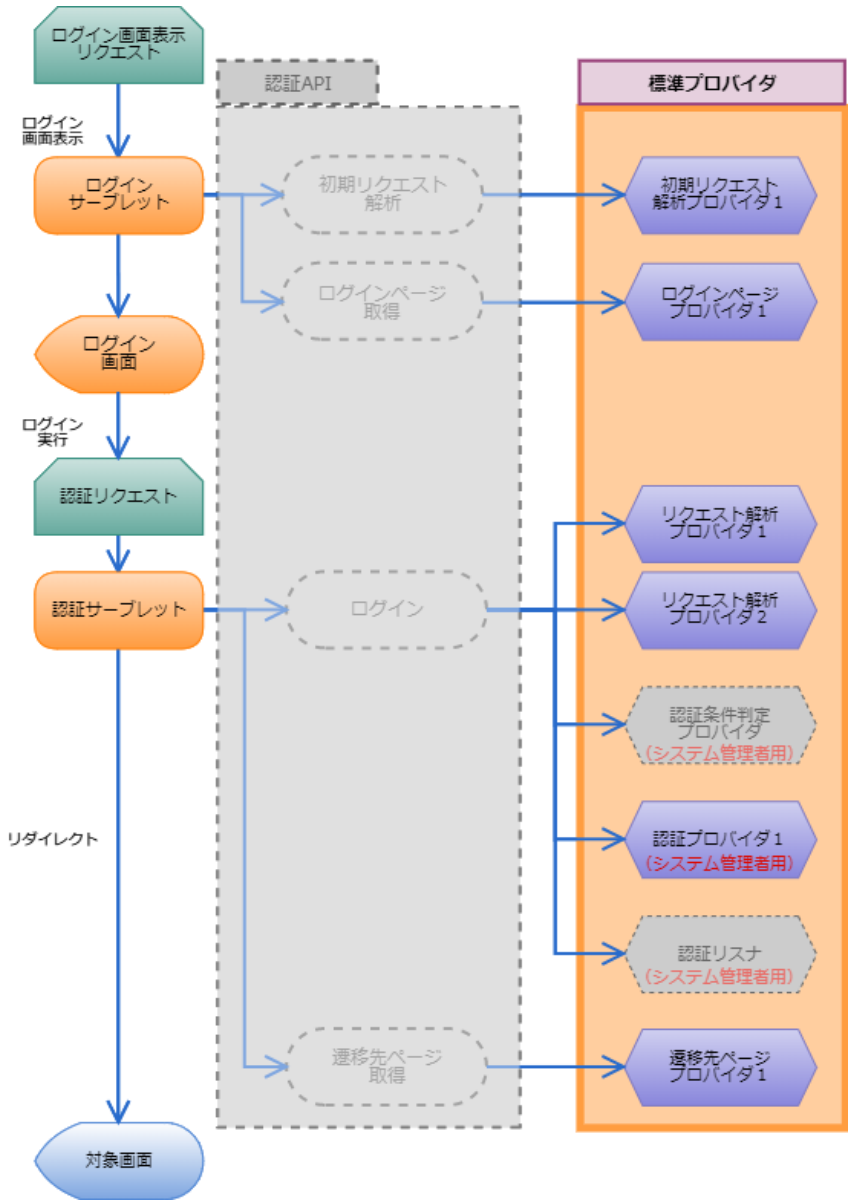


図 システム管理者の標準プロバイダを利用した場合の認証フロー

図で示した標準プロバイダの実装クラスは、以下の通りです。

表 システム管理者用標準プロバイダ一覧

標準プロバイダ	実装クラス	rank
初期リクエスト解析プロバイダ1	jp.co.intra_mart.system.security.certification.provider.impl.AdministratorIPAddressLoginRequestAnalyzer （リクエスト解析プロバイダ1と共通）	10

標準プロバイダ	実装クラス	rank
ログインページプロバイダ1	jp.co.intra_mart.system.security.certification.provider.impl.StandardAdministratorLoginPageProvider	100
リクエスト解析プロバイダ1	jp.co.intra_mart.system.security.certification.provider.impl.AdministratorIPAddressLoginRequestAnalyzer （初期リクエスト解析プロバイダ 1 と共通）	10
リクエスト解析プロバイダ2	jp.co.intra_mart.system.security.certification.provider.impl.StandardLoginRequestAnalyzer （一般ユーザのリクエスト解析プロバイダ 1 と共通）	100
認証プロバイダ1	jp.co.intra_mart.system.security.certification.provider.impl.StandardAdministratorCertification	100
遷移先ページプロバイダ1	jp.co.intra_mart.system.security.certification.provider.impl.StandardAdministratorTargetPageProvider	100

システム管理者のログイン画面

システム管理者のログイン画面の機能について説明します。

ログイン画面の機能

intra-mart Accel Platform が標準で提供するシステム管理者用のログイン画面の機能は以下の通りです。

- ユーザコード入力補助

- 遷移先情報キー
- セキュアトークン

ログイン画面項目

ログイン画面の入力項目について説明します。

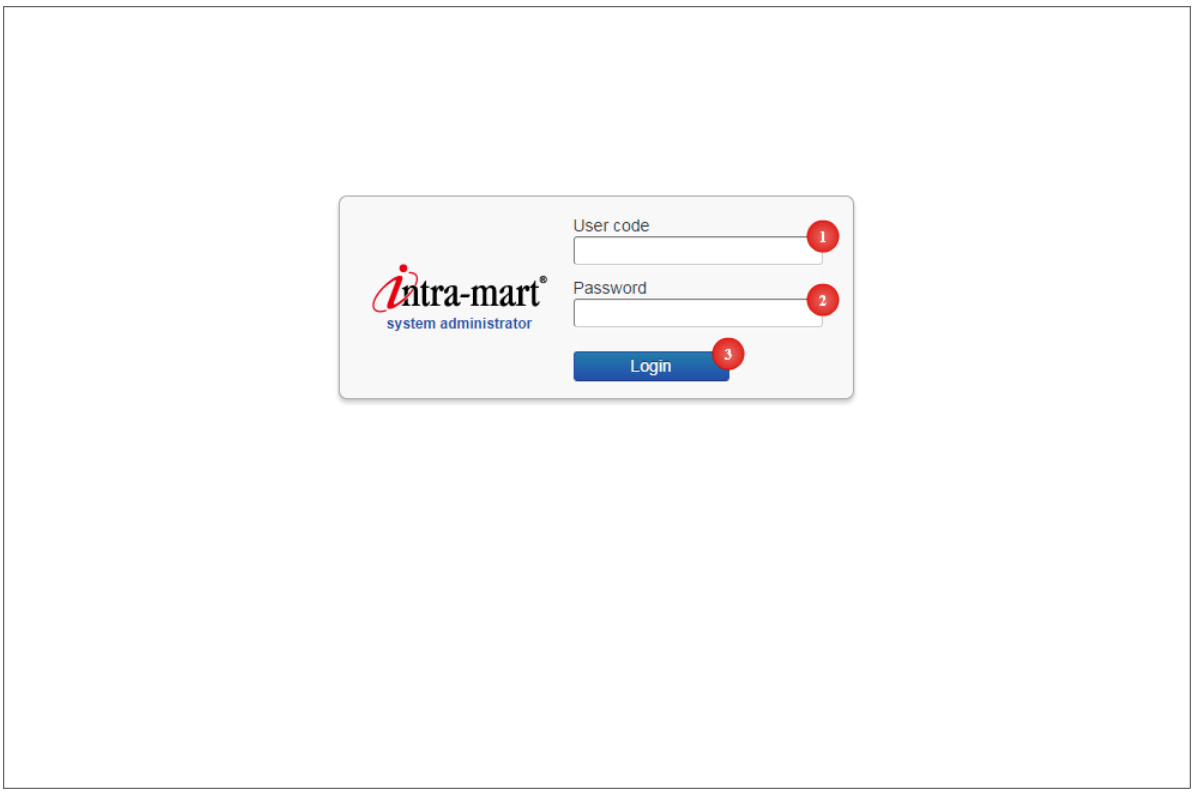


図 システム管理者のログイン画面

1. ユーザコード
   ユーザコード入力欄です。
   Cookie を利用して、前回入力した情報が初期入力されます。
   ユーザコードは必須です。
   intra-mart Accel Platform の標準のログイン画面は、システムの入り口として、通常の画面と異なるデザインが利用されています。
   そのため、必須入力マークは表示されません。
2. パスワード
   パスワード入力欄です。
3. ログインボタン
   ログインを実行します。

ログイン画面の実装

システム管理者用のログイン画面の実装について説明します。

ログイン画面のURL

intra-mart Accel Platform の標準のシステム管理者用ログイン画面URL は以下の通りです。

`http://<HOST>:<PORT>/<CONTEXT_PATH>/system/login`

ログイン画面のURLは `web.xml` で、`AdministratorLoginServlet` にマッピングされています。
そのため、ログイン画面のURLを変更するためには `web.xml` の修正が必要ですが、intra-mart Accel Platform では `web.xml` の変更を推奨していません。
ログイン画面を変更する場合は、URLの変更ではなく「*ログインページプロバイダ*」の利用を検討してください。

ユーザコード入力補助

ユーザコードは、Cookie を利用して、前回入力した情報が初期入力されます。

項目	保存される Cookie 名
ユーザコード	im_user_id

Cookie への保存は、画面の機能として行われます。
そのため、「ログイン」ボタンをクリックして、ログイン処理を実行した時点で保存され、ログイン処理に失敗した場合でも値が更新されます。

この機能を無効にするためには、「設定ファイルリファレンス」 - 「認証設定（システム管理者用）」 - 「ユーザコードのCookie保存」を参照してください。

遷移先情報キー

ログイン後に遷移するための情報を保持するキーです。
ログイン画面では、以下のリクエストパラメータ名の隠し属性として保持します。

- リクエストパラメータ名 ： im_page_key

また、認証に関するエラー画面でも、このキー情報を持ちまわるように実装されています。

ただし、システム管理者の場合は遷移先情報を保存していないため、この情報は利用していません。

セキュアトークン

ログイン処理では、悪意あるユーザからのアクセスを防ぐため、セキュアトークンチェックを行っています。
セキュアトークンチェックを行うために、ログイン画面ではチェックのためのトークン情報を保持します。
このため、ログイン画面を経由せずにログイン処理を実行できないようにしています。

> **注意**
>
> セキュアトークンチェックは、HTTPセッションに保存したトークン情報を基にチェックを行います。
> このため、ログイン画面表示後しばらくアクセスしなかった場合、セッションタイムアウトが発生してトークン情報が破棄されます。
> トークン情報が破棄されたままログイン画面でログインを実行した場合、ユーザコード、パスワードが正しくても、ログインできずに HTTP403 権限エラーが表示されます。
>
> この場合は、再度ログイン画面にアクセスしてからログインしなおしてください。

IPアドレスによるアクセス制限

システム管理者のIPアドレスによるアクセス制限について説明します。

IPアドレスによるアクセス制限の概要

特定のIPアドレスからアクセスした場合のみ、システム管理者のログインを許可するための機能です。

ログイン画面表示リクエスト、および、ログインリクエストに対して、許可されたIPアドレスかどうかのチェックが実行されます。

許可されたIPアドレス以外からアクセスされた場合は、HTTPステータスコード 404 の「エラー」画面が表示されます。

IPアドレスによるアクセス制限の設定

設定ファイルに、アクセス可能なIPアドレスを指定します。
IPアドレスは範囲指定や、ワイルドカードによる指定が可能です。
設定方法は、「設定ファイルリファレンス」 - 「認証IPアドレス制限設定（システム管理者用）」 を参照してください。
また、IPアドレスの取得に関しては「設定ファイルリファレンス」 - 「IPアドレス取得元設定」を参照してください。

セッションタイムアウト

セッションタイムアウトについて説明します。

項目

セッションタイムアウトの概要

ログインユーザが操作中にセッションタイムアウトが発生した場合、継続して利用すると不整合が発生する可能性があります。
そのため、システムでセッションタイムアウトを検知し、「エラー」画面を表示します。

セッションタイムアウトの検知は、Cookie に保存したログイン情報の比較によって行います。
また、ログイン署名の検証により、セッション情報が不正であるかどうかも検証し、セッションタイムアウトと判定します。

対象の画面が未認証ユーザにもアクセス可能な画面だった場合でも、セッション状態が変更されたことを通知するため、同様に「エラー」画面が表示されます。

セッションタイムアウトの機能

セッションタイムアウトの機能について説明します。

セッションタイムアウトの検出

intra-mart Accel Platform では以下のフローに従いセッション情報を検証し、セッションタイムアウトの検出を行います。

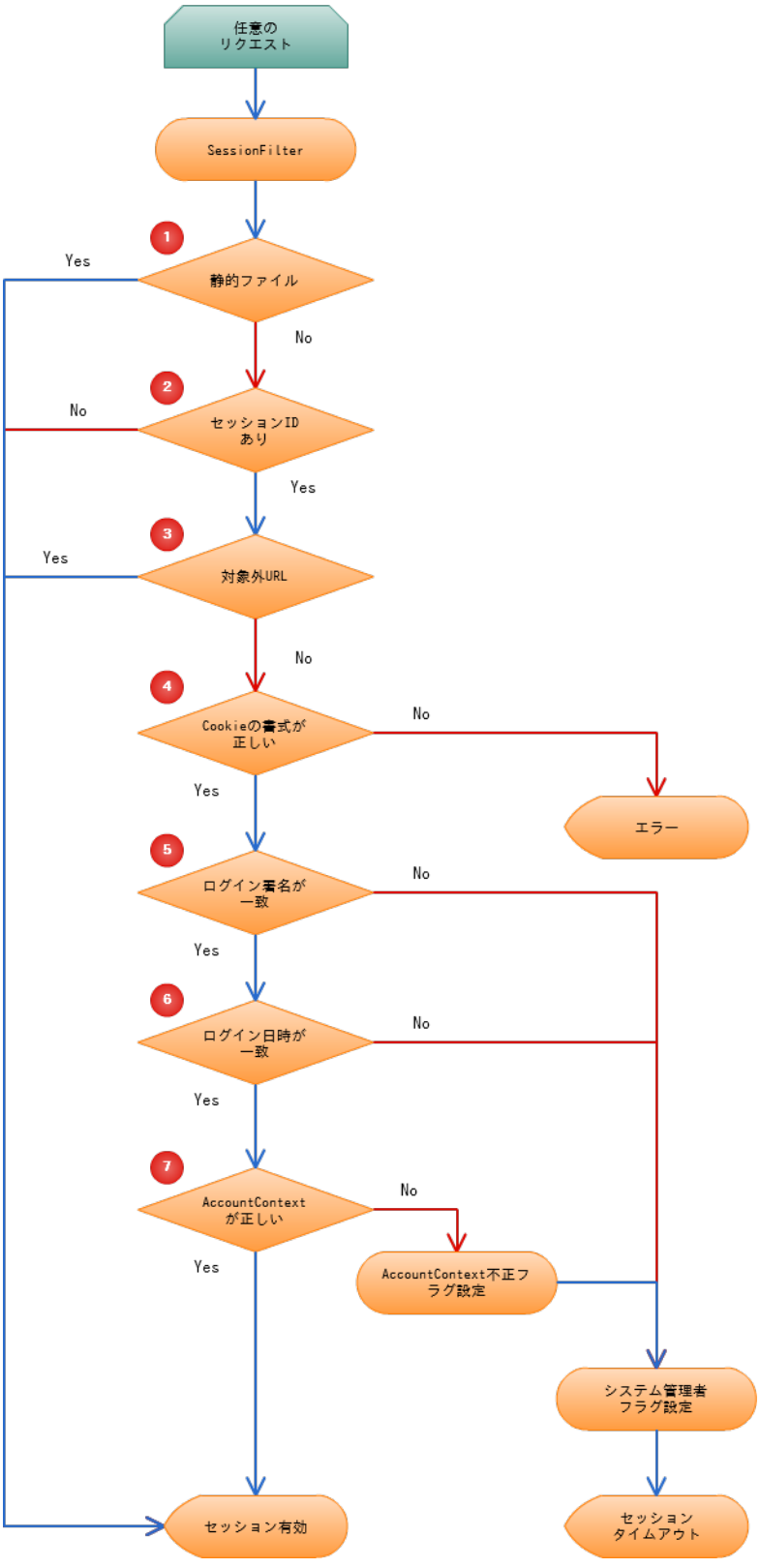


図 セッションタイムアウト検出フロー

No.	チェック内容	詳細
1	静的ファイルチェック	Webサーバで扱う静的ファイルは対象外のため、セッションが有効であると判定します。
2	セッションIDチェック	Cookie にセッションIDがない場合、初回アクセス（ログインしていない状態）のため、セッションが有効であると判定します。
3	対象外URLチェック	システムで扱う一部のURLはセッションタイムアウトを検出する必要がないため、セッションが有効であると判定します。 （対象外とするURLは、設定ファイルにより追加することが可能です。詳細は「*セッションタイムアウト対象外URLの指定*」を参照してください）

No.	チェック内容	詳細
4	Cookie のログイン情報書式チェック	Cookie にログイン情報が設定されているが、正しい書式で設定されていない場合、システムエラーとします。
5	ログイン署名チェック	Cookie のログイン情報とアカウントコンテキストのログイン署名が異なる場合、セッションタイムアウトとして判定します。
6	ログイン日時チェック	Cookie のログイン情報とアカウントコンテキストのログイン日時が異なる場合、セッションタイムアウトとして判定します。
7	AccountContext チェック	アカウントコンテキストのユーザを特定する情報とログイン署名が不整合な場合、セッションタイムアウトとして判定します。

検出フローを元にセッションタイムアウトと判定された場合、以下のフラグが設定されます。

AccountContext 不正フラグ

- 説明
  - 「7. AccountContext チェック」にてログイン署名が不正であると判定された場合に設定されます。
  - このフラグは将来的に利用する目的の情報です。
- フラグの設定先
  - リクエスト属性
- キー名
  - `jp.co.intra_mart.session.context.invalid`

システム管理者フラグ

- 説明
  - アクセス前のユーザが、システム管理者でログイン中だった場合に設定されます。
  - 2018 Summer(Tiffany) 以降、セッションタイムアウト後の遷移先（一般ユーザのログイン画面、または、システム管理者のログイン画面）の判断に利用します。
- フラグの設定先
  - リクエスト属性
- キー名
  - `jp.co.intra_mart.session.cookie.administrator`

セッションタイムアウトエラー画面の表示

セッションタイムアウトが発生した場合、以下の例外が発生します。

`jp.co.intra_mart.foundation.security.filter.exception.SessionTimeoutException`

`SessionTimeoutException` 発生時、`web.xml` のエラーハンドリング設定に従って、以下のエラー画面に遷移します。

`%CONTEXT_PATH%/alert/session_timeout_error.jsp`

> **注意**
>
> この画面では、システム管理者でログイン中にセッションタイムアウトが発生した場合も、一般ユーザの「ログイン」画面へ遷移するボタンが表示されます。

なお、セッションタイムアウトを検出しても、セッションの初期化は行われません。
アプリケーションサーバでセッションタイムアウトが発生した場合、セッションの初期化が行われていますが、セッション情報の不整合であった場合、そのまま別の画面にアクセスしても再度「セッションタイムアウトエラー」画面が表示されます。
ログイン画面へ一度遷移することでセッションの初期化が行われるため、ログインし直す必要があります。

セッションタイムアウト対象外URLの指定

セッションタイムアウト対象外とするURLは、「セッション情報チェック設定」に定義することで追加することが出来ます。
設定方法の詳細は、「 設定ファイルリファレンス 」 -「 認証機能 」 -「 セッション情報チェック設定 」を参照してください。

コラム

以下のURLはシステムデフォルトのURLとして、セッションタイムアウトの対象外です。

名称	URL
システム管理者ログイン画面アクセスURL	/system/login
システム管理者ログイン実行URL	/system/certification
一般ユーザログイン画面アクセスURL	/login
一般ユーザログイン実行URL	/certification
ログアウト実行URL	/logout

コラム

URLの検証には、 `HttpServletRequest` クラスの `getServletPath()` を利用して取得したパスに対して実行します。

ログアウト

ログアウト処理について説明します。

項目

ログアウトの概要

ログアウトすることで、ログインセッションを終了します。
ユーザの状態としては、ログイン状態から未認証状態に遷移します。

ログアウト処理では、HTTPセッションを初期化し、ログイン中にHTTPセッション等に保持していたユーザに関する情報を破棄します。

ログアウトの機能

ログアウトの主な機能について説明します。

ログアウト処理

ログアウト処理では、HTTPセッションが破棄され、新しいHTTPセッションが作成されます。
アクセスコンテキストが切り替えられて、未認証ユーザの状態に遷移します。

ログアウト後の遷移先の指定

一般ユーザの場合、ログアウト時にリクエストパラメータ 「`im_url`」を指定することで、指定した画面に遷移することが可能です。

以下のように指定した場合、ログアウト後にサイトマップに遷移します。

`http://iap.example.com/imart/logout?im_url=/menu/sitemap`

`im_url` に外部サイトのURLを指定した場合、許可されたサイトのみ遷移可能です。
許可されていないサイトの場合は指定は無視されて、「ログイン画面」に遷移します。

外部サイトの許可設定については、「設定ファイルリファレンス」 - 「認証外部ページURL許可リスト設定」を参照してください。

im_url による遷移を無効にする場合は、「設定ファイルリファレンス」 - 「リクエストパラメータによる画面遷移サポートの有無（ログアウト時）」を参照してください。

ログアウトの実装

ログアウトのURL

intra-mart Accel Platform の標準のログアウトURL は以下の通りです。

```
http://<HOST>:<PORT>/<CONTEXT_PATH>/logout
```

このURLは、一般ユーザとシステム管理者で共通です。

ログアウト処理の流れ

ログアウト処理の流れは、以下の通りです。

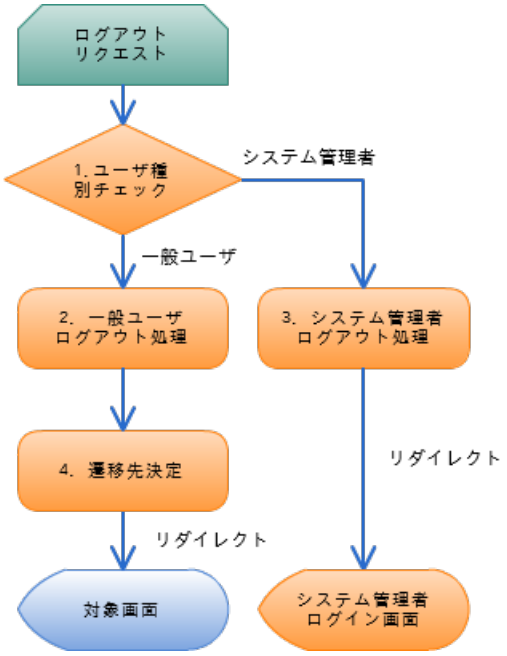


図 ログアウト処理フロー

1. ユーザ種別チェック
   一般ユーザかシステム管理者かで処理を振り分けます。
   ログインしていなかった場合にログアウト処理が呼び出された場合は、一般ユーザとして処理されます。

2. 一般ユーザログアウト処理
   一般ユーザのログアウト処理を実行します。

3. システム管理者ログアウト処理
   システム管理者のログアウト処理を実行します。
   ログアウト処理の実行後は、「システム管理者のログイン画面」にリダイレクトします。

4. 遷移先決定処理
   遷移先を決定し、決定された画面にリダイレクトします。
   一般ユーザの場合、リクエストパラメータにより遷移先の変更が可能です。
   「 *ログアウト後の遷移先の指定* 」 を参照してください。
   リクエストパラメータが指定されていない場合は、「一般ユーザのログイン画面」が遷移先です。

> **注意**
>
> ログアウト処理では、ユーザ種別を確認して処理を振り分けています。
> ユーザ種別は、HTTPセッションに保存されているアカウントコンテキストの情報であるため、セッションタイムアウトが発生した場合、ユーザ種別が判定できません。
> その場合、一般ユーザとして判定されるため、システム管理者の画面でログアウトを実行しても、「一般ユーザのログイン画面」に遷移します。

認証確認

認証確認機能について説明します。

項目

認証確認の概要

intra-mart Accel Platform では、任意の画面で認証を行うために、認証確認機能が提供されています。
認証確認機能とは、ログイン、または、認証確認を実行してから一定時間が経過した場合に、「認証確認」画面を表示してログインユーザに認証を求める機能です。
この機能を使うことで、決済処理などのような業務的に重要な処理において、セキュリティを高めることができます。

認証確認の利用方法

認証確認機能を利用するためには、あらかじめ認証確認の設定を行います。
認証確認の設定には、「認証確認設定」と「認証確認対象ページ設定」があります。
詳細は、「 *認証確認の設定* 」を参照してください。

> **注意**
>
> 認証確認の設定は、システムで共通の設定です。
> バーチャルテナントによる複数テナントを利用している場合、テナントごとに設定を行うことはできません。

認証確認の実装

認証確認機能の実装についての情報を記述します。

一般ユーザの認証確認リクエストフロー

一般ユーザの認証確認におけるリクエストフローは、以下の通りです。

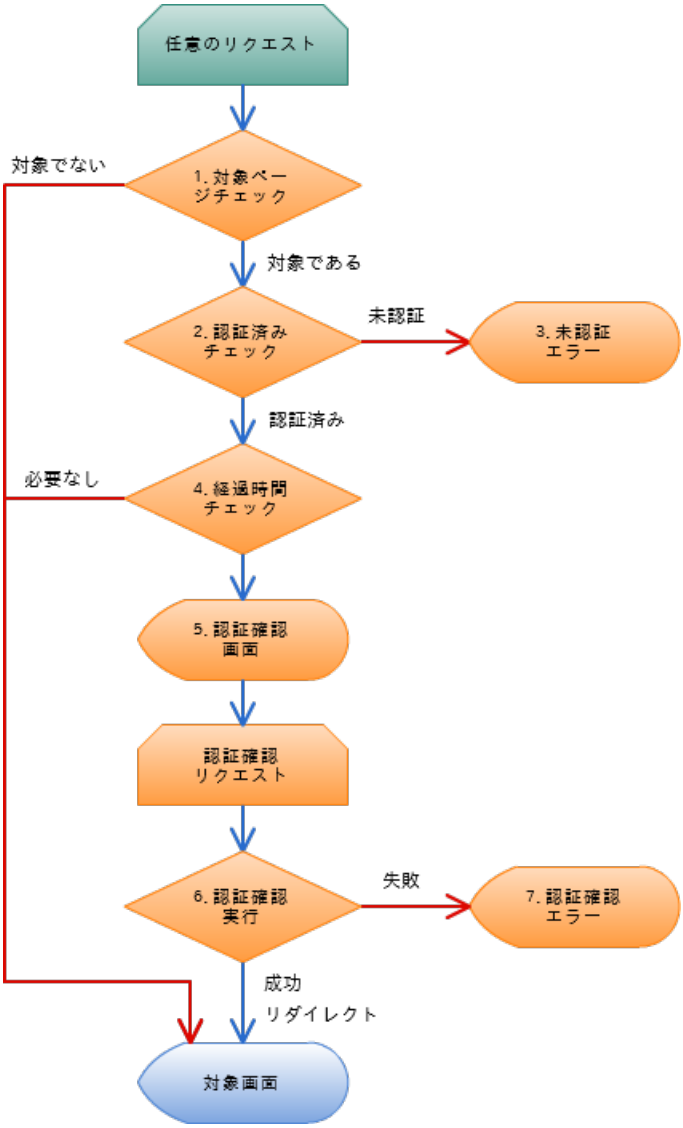


図 認証確認機能のリクエストフロー

1. 対象ページチェック
   対象ページチェックは、リクエストされた画面が認証確認の対象ページかどうかを確認する処理です。
   認証確認の対象でないと判定された場合は、リクエストされた画面が表示されます。

2. 認証済みチェック
   ログイン済みかどうかを確認します。

3. 未認証エラー
   ログインしていない場合、「未認証エラー」画面を表示します。

4. 経過時間チェック
   ログイン、または、前回認証確認実行時からの経過時間を設定値と比較して、認証確認が必要かどうかを確認する処理です。
   認証確認の必要がないと判定された場合は、リクエストされた画面が表示されます。

5. 認証確認画面
   認証確認が必要なリクエストの場合、「認証確認」画面を表示します。

6. 認証確認実行
   「認証確認画面」でパスワードなどの識別情報を入力することで、認証を実行し、 認証に成功した場合は、認証確認前に表示しようとした画面に遷移します。

7. 認証確認エラー
   認証に失敗した場合、「認証確認エラー」画面を表示します。

> **注意**
>
> 認証確認の経過時間チェックは、画面単位では行われません。
> ある画面で認証確認を実行し、設定された時間が経過した後に認証確認対象の別の画面を開いた場合、「認証確認」画面が表示されます。

システム管理者の認証確認リクエストフロー

システム管理者の認証確認のリクエストフローは、一般ユーザとほぼ同じです。

認証確認の設定

一般ユーザの認証確認設定

一般ユーザの画面で認証確認を利用する手順について説明します。

認証確認の有効化

一般ユーザの認証確認は、初期状態では無効です。

認証確認設定の「認証確認の間隔」を設定することで、認証確認機能を有効にします。
「認証確認の間隔」には、ログイン、または、認証確認を実行から、再度「認証確認」画面を表示するまでの時間（分）を設定します。
設定された時間が経過するまでは、「認証確認画面」を表示せずにリクエストされた画面が表示されます。

初期状態では、認証確認設定ファイルの「認証確認の期間」設定がコメントアウトされているので、次の【設定例】を参考にして有効にしてください。
詳細は、「 設定ファイルリファレンス 」-「 認証確認設定（一般ユーザ用） 」を参照してください。

【設定例】

```
 :

<!-- 認証確認ページへの遷移で使用する設定 -->
<category name="im_login_confirm">

  :

 <!-- 認証確認の期間 -->
 <param>
   <param-name>validity_period</param-name>
   <param-value>5</param-value>
 </param>
</category>

 :
```

認証確認対象ページの設定

認証確認の対象ページを設定します。
対象ページは、コンテキストパスからの相対パスによって指定します。
ユーザの「パスワード設定画面」を対象ページとする場合、次の【設定例】のようにします。
詳細は、「 設定ファイルリファレンス 」-「 認証確認対象ページ設定（一般ユーザ用） 」を参照してください。

【設定例】

```
:

<target>/user/settings/password</target>

:
```

認証確認画面の実行

intra-mart Accel Platform を起動し、認証確認を実行します。
ログイン、または、前回認証確認を実行してから、「認証確認の間隔」に設定された時間が経過した後、認証確認対象ページにアクセスすることで、「認証確認」画面が表示されます。

前項の【設定例】の設定を行った場合、以下の手順で「認証確認」画面が表示されます。

- 一般ユーザでログインする。
- 5 分経過後に「ユーティリティメニュー」 - 「個人設定」 - 「パスワード設定画面」にアクセスする。

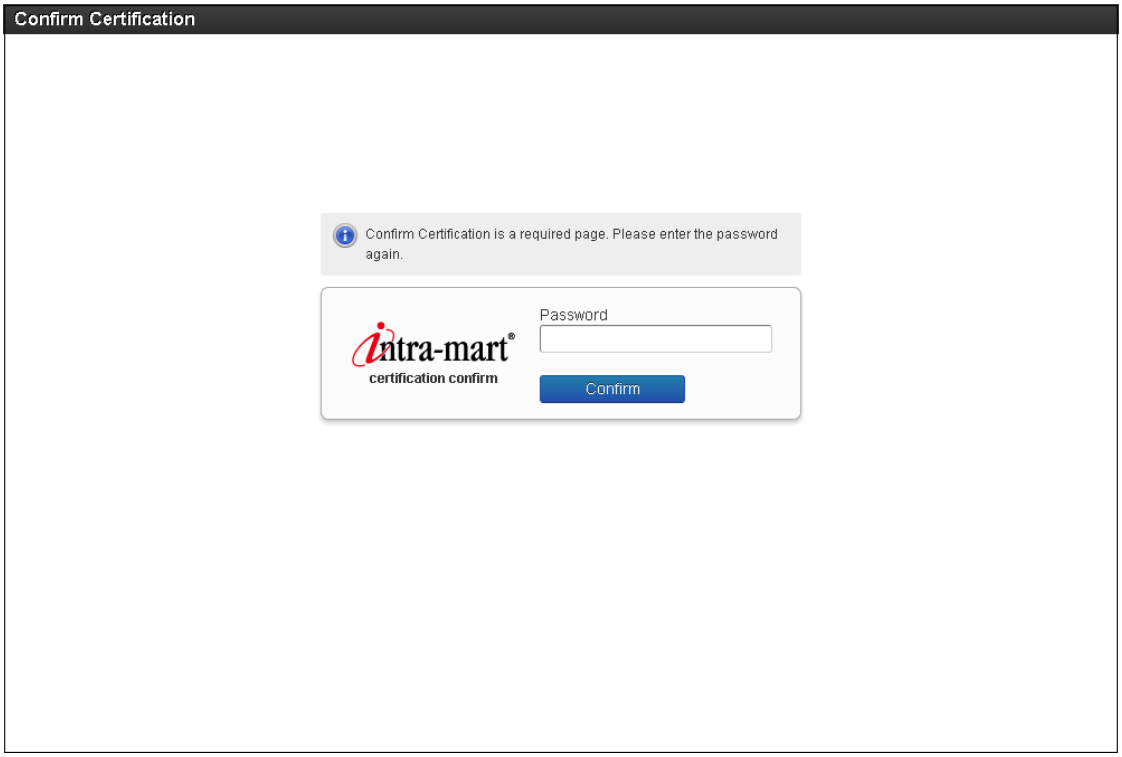


図 認証確認画面

表示された「認証確認画面」で、パスワードを入力して認証を実行します。
認証確認に成功すると対象ページへ遷移します。

パスワードを間違えた場合、以下の「認証確認エラー」画面が表示されます。
「認証確認画面」へ戻り、再度認証確認を行ってください。

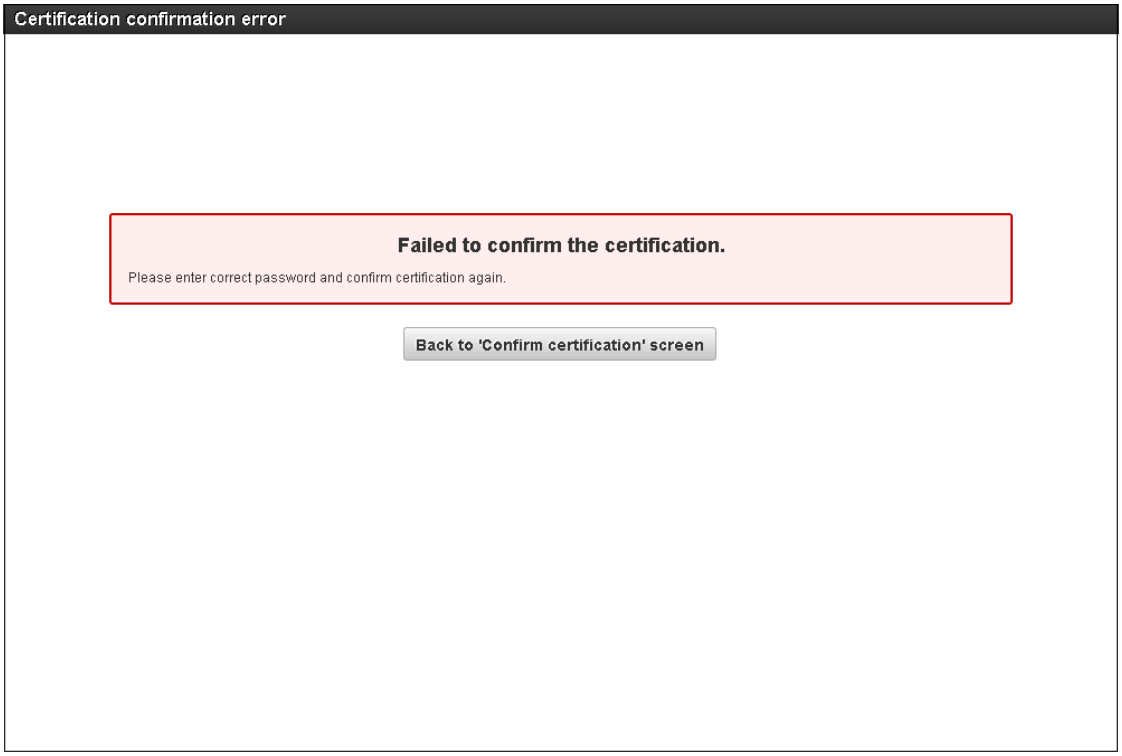


図 認証確認エラー画面

システム管理者の認証確認設定

システム管理者の認証確認の設定は、一般ユーザとほぼ同じです。
ただし、システム管理者の認証確認は、初期設定では有効です。
また、システム管理者のいくつかの画面は、初期設定で認証確認の対象ページに設定されています。
システム管理者の認証確認の実行方法は、一般ユーザと同じです。

システム管理者の認証確認設定については、「 設定ファイルリファレンス 」 - 「 認証確認設定（システム管理者用） 」を参照してください。
システム管理者の認証確認対象ページ設定については、「 設定ファイルリファレンス 」 - 「 認証確認対象ページ設定（システム管理者用） 」を参照してください。

認証エラー

認証機能で発生するエラーについて説明します。

項目

認証エラーの概要

認証機能では、処理の結果に応じた認証エラーが定義されています。
認証エラーが発生した場合は、認証エラーの種別に応じた「エラー」画面が表示されます。
エラー種別とエラー画面は設定ファイルで定義され、エラー時の処理を変更することが可能です。

認証エラーの一覧は、「*認証エラー一覧*」を参照してください。
エラー画面を変更する設定方法は、「*認証エラー画面の設定*」を参照してください。

> **注意**
>
> ここでは、認証機能が提供する認証エラーの説明と、エラー画面の変更方法を説明します。
> 独自に認証エラーを追加することはできません。

認証エラー一覧

認証エラーの一覧は、以下の通りです。

一般ユーザの認証エラー

ログインエラー

エラー種別コード	CERTIFICATION_ERROR
エラーページパス	user/certification/error/certification_error_page
戻り先	ログイン画面
説明	エラーが発生する原因は、以下の通りです。 ・ユーザが存在しない場合。 ・認証モジュールが認証失敗を返却した場合。

ライセンスエラー

エラー種別コード	LICENSE_ERROR
エラーページパス	user/certification/error/certification_error_page

戻り先	ログイン画面
説明	エラーが発生する原因は、以下の通りです。 ▪ アカウントライセンスが付与されていない場合。 ▪ アカウントの有効期間が期間外の場合。

アカウントロックエラー

エラー種別コード	LOCKED_ERROR
エラーページパス	user/certification/error/certification_error_page
戻り先	ログイン画面
説明	エラーが発生する原因は、以下の通りです。 ▪ アカウントロックされていて、アカウントロック期間を超えていない場合。

認証処理エラー

エラー種別コード	SYSTEM_ERROR
エラーページパス	user/certification/error/system_error_page
戻り先	なし
説明	エラーが発生する原因は、以下の通りです。 ▪ 認証モジュールが解決できない場合。 ▪ 認証モジュール、または、認証リスナーがエラーを返却した場合。 ▪ ユーザコードが取得できない場合。

未認証エラー（認証確認）

エラー種別コード	CONFIRM_UNAUTHORIZED_ERROR
エラーページパス	初期設定なし。（HTTP401エラー画面を利用します）
戻り先	ログイン画面
説明	エラーが発生する原因は、以下の通りです。 ▪ 認証確認対象画面アクセス時に、未認証の場合。

認証確認エラー

エラー種別コード	CERTIFICATION_CONFIRM_ERROR
エラーページパス	user/certification/error/certification_confirm_error_page
戻り先	認証確認画面
説明	エラーが発生する原因は、以下の通りです。 ▪ 認証モジュールが認証失敗を返却した場合。

テナントID無効エラー

エラー種別コード	TENANT_NOT_FOUND_ERROR
エラーページパス	user/certification/error/certification_error_page

戻り先	ログイン画面
説明	エラーが発生する原因は、以下の通りです。 ▪ 指定されたテナントIDが存在しない場合。
導入バージョン	intra-mart Accel Platform 2014 Spring(Granada)

システム管理者の認証エラー

ログインエラー

エラー種別コード	ADMIN_CERTIFICATION_ERROR
エラーページパス	system/certification/error/admin_certification_error_page
戻り先	ログイン画面
説明	エラーが発生する原因は、以下の通りです。 ▪ システム管理者が存在しない場合。 ▪ 認証モジュールが認証失敗を返却した場合。

認証処理エラー

エラー種別コード	ADMIN_SYSTEM_ERROR
エラーページパス	system/certification/error/admin_system_error_page
戻り先	なし
説明	エラーが発生する原因は、以下の通りです。 ▪ 認証モジュールが解決できなかった場合。 ▪ 認証モジュール、または、認証リスナーがエラーを返却した場合。 ▪ ユーザコードが取得できない場合。

未認証エラー（認証確認）

エラー種別コード	ADMIN_CONFIRM_UNAUTHORIZED_ERROR
エラーページパス	初期設定なし。（HTTP401エラー画面を利用します）
戻り先	ログイン画面
説明	エラーが発生する原因は、以下の通りです。 ▪ 認証確認対象画面アクセス時に、未認証の場合。

認証確認エラー

エラー種別コード	ADMIN_CERTIFICATION_CONFIRM_ERROR
エラーページパス	system/certification/error/admin_certification_confirm_error_page
戻り先	認証確認画面
説明	エラーが発生する原因は、以下の通りです。 ▪ 認証モジュールが認証失敗を返却した場合。

認証エラー画面の設定

認証エラー画面は、PageManager API を利用して、以下の plugin.xml から取得しています。

一般ユーザ用設定

ファイル	<plugin/jp.co.intra_mart.certification.page.standard/plugin.xml>
プラグインID	jp.co.intra_mart.certification.page.standard.error

一般ユーザ用設定（テナントID無効エラー）

ファイル	<plugin/jp.co.intra_mart.certification.page.vt/plugin.xml>
プラグインID	jp.co.intra_mart.certification.page.vt.error
導入バージョン	intra-mart Accel Platform 2014 Spring(Granada)

システム管理者用設定

ファイル	<plugin/jp.co.intra_mart.certification.page.administrator/plugin.xml>
プラグインID	jp.co.intra_mart.certification.page.administrator.error

認証エラー画面を変更する場合、上記プラグインの情報を参照して、新しい設定ファイルを作成してください。

ログインエラー画面の変更方法は以下の通りです。

1. 新しいエラー画面を作成する。
   任意のエラー画面を作成してください。
   エラー画面は、認証機能から FORWARD されて呼びだされます。
   エラー画面からログイン画面に遷移する場合、遷移先情報を持ちまわる必要があります。
   遷移先情報については、「 *遷移先情報キー* 」を参照してください。

2. plugin.xml ファイルを作成する。
   plugin.xml ファイルを作成し、plugin ディレクトリ配下の任意のディレクトリに配置してください。
   ユニークなプラグインIDを作成し、rank 属性を標準プラグインの「100」より小さく設定して、プラグインの設定を行ってください。
   以下の設定を行う、plugin.xml の例を示します。
   - プラグインID： sample.certification.error
   - rank： 90
   - エラー種別： ログインエラー（ CERTIFICATION_ERROR ）
   - エラー画面： /sample/certification/error/certification_error_page

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>

  <extension point="jp.co.intra_mart.page">
    <page-config
      name="Sample"
      id="sample.certification.error"
      version="8.0"
      rank="90">
      <!-- スクリプト開発モデルの場合のページプロバイダ -->
      <page-provider-class>jp.co.intra_mart.system.page.provider.script.StandardJsspPageProvider</page-provider-class>
      <init-param>
        <!-- ログインエラーのエラーページID -->
        <param-name>jp.co.intra_mart.foundation.security.certification.CertificationStatus.CERTIFICATION_ERROR</param-name>
        <!-- 変更するエラーページのパス -->
        <param-value>/sample/certification/error/certification_error_page</param-value>
      </init-param>
    </page-config>
  </extension>

</plugin>
```

plugin.xml のエラー種別（<param-name>）には、以下を設定してください。

jp.co.intra_mart.foundation.security.certification.CertificationStatus. ＋ 「エラー種別コード」

> **コラム**
>
> PageManager API については、APIドキュメントを参照してください。
>
> - PageManager クラスの APIドキュメント （JavaEE開発モデル）
> - PageManager オブジェクトの APIドキュメント （スクリプト開発モデル）

一般ユーザの強制ログイン

一般ユーザの強制ログイン処理について説明します。

項目

一般ユーザの強制ログインの概要

強制ログインは、パスワードによる検証を省略して intra-mart Accel Platform にログインする機能です。
「統合Windows認証モジュール」や「IM-SecureSignOn for Accel Platform」などの機能で利用しています。

一般ユーザのログイン処理 と同様に認証APIを呼び出すことで実行されます。
上記のログイン処理とは、呼び出す認証プロバイダ・プラグインが異なります。

強制ログイン時のフローは、強制ログインを実装している機能に準じます。

認証API

一般ユーザの強制ログインの実行には、以下のAPIを利用します。

`jp.co.intra_mart.foundation.security.certification.UserCertificationManager#forceLogin()`

API の詳細については、「 UserCertificationManager クラスの APIドキュメント 」を参照してください。

一般ユーザの強制ログインAPIの動作

以下の処理を順に実行します。

1. アカウントコンテキストを切り替えてログイン状態にします。
2. 必要に応じて「 *強制ログイン用認証リスナ* 」を呼び出します。
   詳細は後述します。

強制ログイン用認証リスナ

強制ログイン成功時の後処理を実行します。
強制ログイン用認証リスナは、APIから呼び出す際に認証リスナを動作させる指定を行った場合のみ、実行されます。
強制ログインの利用用途により、認証リスナの実行有無が決定します。

以下のような用途で強制ログインを実行する場合は、認証リスナが動作します。

- 強制ログイン実施後に、ユーザがブラウザ操作が行う場合

  例：シングルサインオン、ログイン認証が必要なショートカットURLによるアクセス、など

以下のような用途で強制ログインを実行する場合は、認証リスナは動作しません。

- 特定の処理を動かすために、一時的にユーザを切り替える場合

  例：Webサービス、外部ソフトウェア連携、など

コラム

一般ユーザのログイン処理用のプロバイダである「認証リスナ」は動作しません。
強制ログイン時は、以下の設定を行ったリスナのみ動作します。

注意

強制ログイン用認証リスナは、2015 Winter(Lydia) より利用可能です。
2015 Summer(Karen) 以前のバージョンでは、強制ログイン時に認証リスナは実行されません。

注意

認証APIは、 1 つのトランザクションで処理されます。
そのため、認証リスナでデータベースなどのトランザクション管理されている情報を更新しても、その後の処理によっては、ロールバックされる可能性があります。
別トランザクションとする場合は、独自にトランザクションを開始するようにしてください。

強制ログイン用リスナ設定

拡張ポイント `jp.co.intra_mart.security.user.force_login` を指定します。

リスナクラスの指定方法については、以下の設定例の通りです。

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin>
 <extension point="jp.co.intra_mart.security.user.force_login">
  <certification
   name="Sample"
   id="sample.certification"
   version="8.0"
   rank="100">
   <certification-listener>
    <listener-class>sample.certification.SampleCertificationListener</listener-class>
   </certification-listener>
  </certification>
 </extension>
</plugin>
```

強制ログイン用認証リスナインタフェース

リスナクラスのインタフェースは、一般ユーザのログイン処理に動作する認証リスナと同様です。
「 *認証リスナ* 」 - 「 *インタフェース定義* 」を参照してください。

認証拡張機能

SSO（シングルサインオン）

intra-mart Accel Platform における シングルサインオン（以下、SSO と記述します）の仕様について説明します。

項目

SSO の概要

SSO とは、特定のシステムにログインすることで、連携する複数のシステムを利用可能とするための仕組みです。

SSO を実現するためには、連携するシステムがそれぞれ共通の SSO の方式に対応している必要があります。
SSO の方式としては、汎用的な方式を利用する場合もあれば、そのシステム固有の方式を利用する場合もあります。
汎用的な方式として、以下のような規格があります。

- SAML
- OAuth
- OpenID

SSO を構成する要素としては、「認証を行うシステム」と「サービスを提供するシステム」があります。
これらの名称は規格ごとに異なりますが、 intra-mart Accel Platform では以下のように定義します。

- SSO認証プロバイダ

  認証を行うシステム。

- SSOサービスプロバイダ

  サービスを提供するシステム。認証が必要な場合、SSO認証プロバイダに認証要求を行います。

SSO の構成図は以下の通りです。

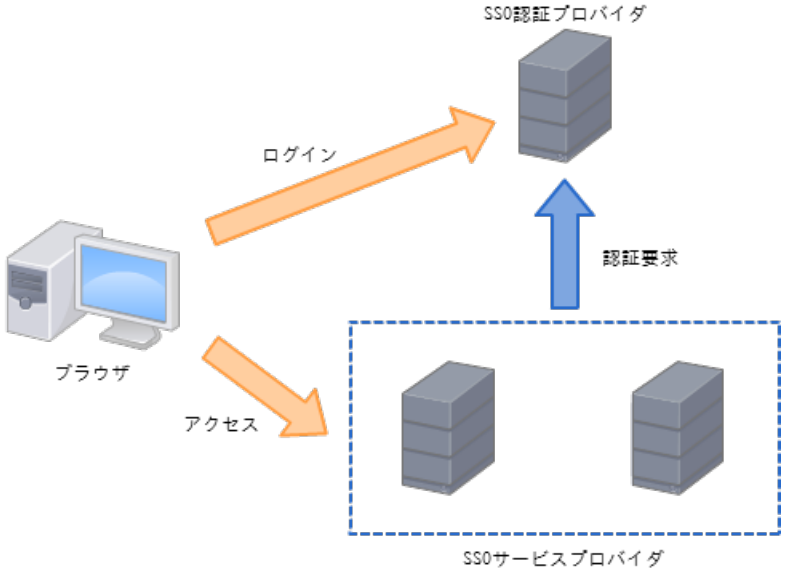


図 SSOの構成

SSOサービスプロバイダでは、アクセスがあった場合にSSO認証プロバイダと連携して、 自動的にログイン済みの状態にします。
これを「SSO自動ログイン処理」と呼びます。

認証機能では、SSOサービスプロバイダでSSO自動ログイン処理を実装するためのインタフェースを提供しています。
ここでは、認証機能を利用してSSO自動ログイン処理を実装するための説明をします。

SSOサービスプロバイダの認証フロー

SSO自動ログイン処理は、アクセスコンテキストの生成処理時に実行されます。

> **注意**
>
> バーチャルテナントによる複数テナントを利用する場合、SSO自動ログイン処理を行うには SSO 対象のテナントを解決する必要があります。
> 具体的には、リクエスト情報を利用したテナント自動解決機能の実装が必要です。
>
> リクエスト情報を利用したテナント自動解決機能の詳細は「intra-mart Accel Platform セットアップガイド」 - 「テナント解決機能」 - 「リクエスト情報を利用したテナント自動解決機能について」 を参照してください。
>
> なお、リクエスト情報を利用したテナント自動解決機能の実装は、バーチャルテナントによる複数テナントが利用可能となった intra-mart Accel Platform 2014 Spring(Granada) 以降にて必要になりました。
> これより前のバージョンで、SSOユーザコードプロバイダを実装し、 2014 Spring(Granada) 以降へアップデートする場合は、リクエスト情報を利用したテナント自動解決機能の実装を行ってください。

SSOサービスプロバイダの認証フローの概要は、以下の通りです。

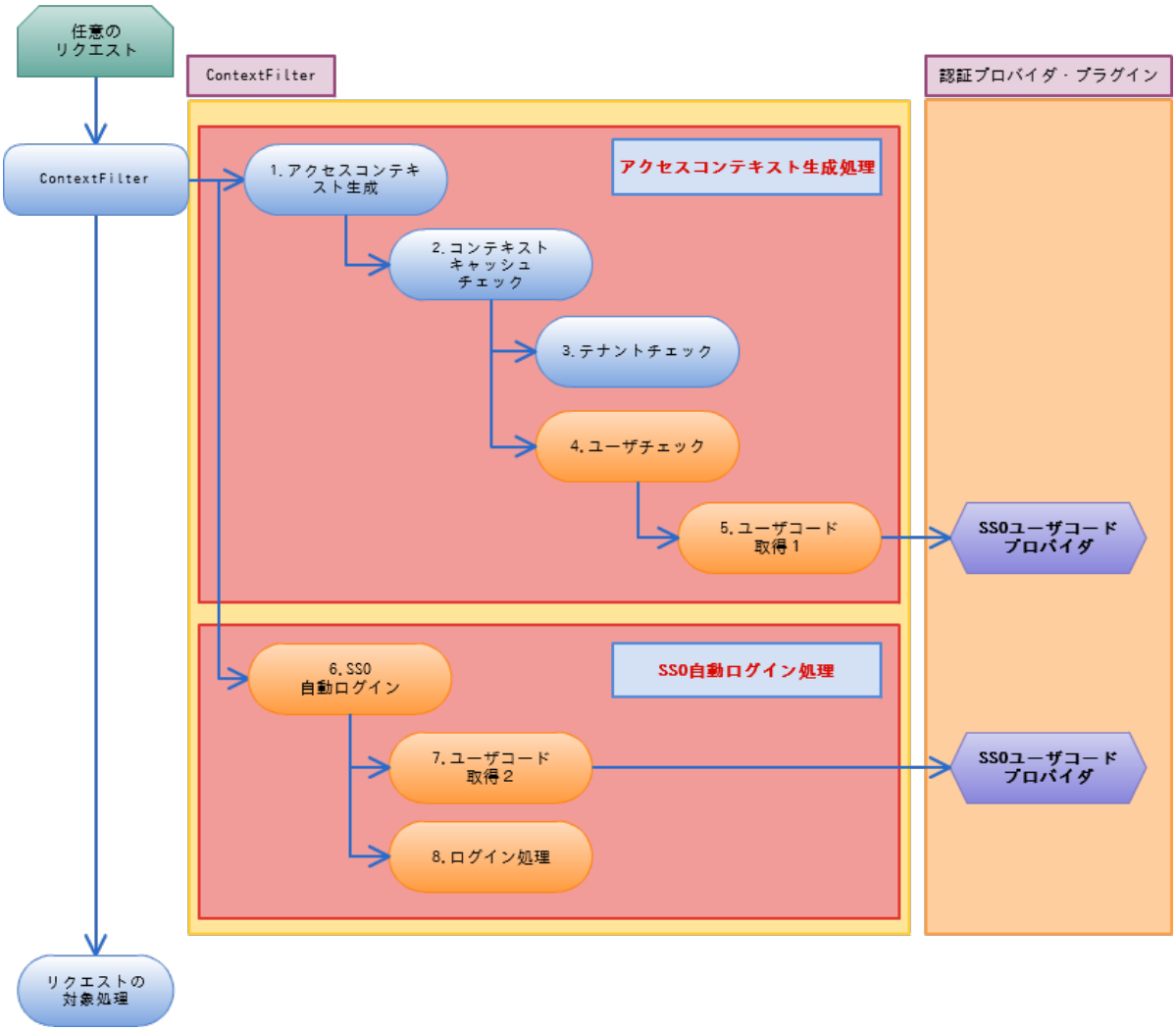


図 SSOサービスプロバイダの認証フロー概要

1. アクセスコンテキスト生成
   アクセスコンテキストを生成します。
   次の「コンテキストキャッシュチェック」を実行した結果キャッシュが有効である場合、コンテキストキャッシュを利用します。
   そうでない場合、新しいアクセスコンテキストを生成します。

2. コンテキストキャッシュチェック
   HTTPセッションに保存したアクセスコンテキストが有効かどうかを確認します。
   この処理は、全てのリクエストで実行されます。
   コンテキストキャッシュはブラウザからの初回アクセス時は存在しません。
   コンテキストキャッシュが存在し、以下のチェックが全て有効と判定された場合は、有効と判断されます。
   以下のチェックを順に実行し、いずれかのチェックで無効と判定された場合は以降のチェックは実行されません。

   1. アクセスコンテキストごとの有効期間の検証
   2. 「テナントチェック」
   3. 「ユーザチェック」

3. テナントチェック
   コンテキストキャッシュのテナントID と リクエスト情報を利用したテナント自動解決機能 により、解決されたテナントID が一致しているかどうかを確認します。

4. ユーザチェック
   コンテキストキャッシュのユーザコードと次の「ユーザコード取得 1 」処理により、取得したユーザコードが一致しているかどうかを確認します。

5. ユーザコード取得 1
   「 *SSOユーザコードプロバイダ* 」を呼び出して、SSO のためのユーザコードを取得します。

6. SSO自動ログイン
   SSO のために自動的にログイン状態とします。
   この処理は、基本的に初回アクセス時に実行されます。
   ログイン中でコンテキストキャッシュが有効である間は、実行されません。
   SSO認証プロバイダに別のユーザでログインした場合は再度実行されます。
   以下のいずれかの条件を満たす場合に実行されます。
   - アクセスコンテキストキャッシュが存在しなかった場合
   - 「テナントチェック」「ユーザチェック」のいずれかで、アクセスコンテキストキャッシュ無効と判断された場合

   以下の「ユーザコード取得２」でユーザコードが取得できた場合、「ログイン処理」を実行します。

7. ユーザコード取得２
   SSO のためのユーザコードを取得します。
   「ユーザコード取得１」が実行済みの場合、そのユーザコードを利用します。
   「ユーザコード取得１」が実行されなかった場合、「*SSOユーザコードプロバイダ*」を呼び出して、ユーザコードを取得します。

8. ログイン処理
   ユーザコードが取得できた場合に、以下の処理を実行します。
   - ユーザの有効チェック
   - ログイン

SSOユーザコードプロバイダとは

SSOユーザコードプロバイダとは、SSO認証プロバイダへのログイン時に保存されたユーザに関する情報から、ユーザコードを解析するためのプラグインです。

SSO の方式によって、ユーザに関する情報が保存される場所や形式が異なります。
例えば IM-SecureSignOn for Accel Platform では、ユーザに関する情報は Cookie に保存されています。
SSOユーザコードプロバイダでは、利用する SSO の方式に従って実装する必要があります。

> **注意**
>
> SSOユーザコードプロバイダは、SSO認証プロバイダの情報を取得するためのプラグインです。
> 必要な情報は HTTPリクエストから取得するようにしてください。
> SSOサービスプロバイダの情報を参照することは想定していません。
>
> アクセスコンテキストの生成処理では、テナントの解決が実行されます。
> SSOユーザコードプロバイダはアクセスコンテキスト生成中に実行されるため、まだ、テナントの情報にはアクセスできません。
> そのため、データベースやパブリックストレージなどのテナントに紐づく情報は参照できませんので、注意してください。

SSOユーザコードプロバイダの実装

SSOユーザコードプロバイダは、以下のインタフェースを実装します。

`jp.co.intra_mart.foundation.security.certification.sso.SSOUserProvider`

作成した SSOユーザコードプロバイダは、以下の拡張ポイントのプラグインとして登録します。

`jp.co.intra_mart.foundation.security.certification.sso.user.providers`

SSOユーザコードプロバイダでは、SSO認証プロバイダから取得したユーザコードを返却してください。
取得できない場合は、 `null` を返却します。

作成した SSOユーザコードプロバイダは、プラグインとして複数設定することが可能です。
その場合、プラグインの優先度に従って順番に実行され、最初に `null` でない値が取得できた場合に、解決されたユーザコードが返却されます。
全てのSSOユーザコードプロバイダが `null` を返却した場合、認証処理を行いません。

SSOユーザコードプロバイダが存在しないユーザコードを返却した場合、SSO自動ログインできないため、HTTPステータスコード 500 の「エラー」画面が表示されます。

SSOユーザコードプロバイダの具体的な実装方法は、「 認証プログラミングガイド 」 -「 SSO対応 」を参照してください。